Logic in Computer Science

Priv.-Doz. Dr. Iosif Petrakis

Ludwig-Maximilians-Universität, Institut für Informatik
Wintersemester 21/22

Contents

Chapter 1

Introduction

1.1 Intuitionistic logic and constructive mathematics

Mathematical logic (ML), or simply logic, is concerned with the study of formal systems related to the foundations and practice of mathematics. ML is a very broad field encompassing various theories, like the following. *Proof theory*, the main object of study of which is the concept of (formal) derivation, or (formal) proof (see e.g., [33]). *Model theory* studies interpretations, or models, of formal theories (see e.g., [9]). Axiomatic *set theory* is the formal theory of sets that underlies most of the standard mathematical practice (see e.g., [22]). It is also called Zermelo-Fraenkel set theory (ZF). The theory ZFC is ZF together with the axiom of choice. ML has strong connections to *category theory*, a theory developed first by Eilenberg and Mac Lane within homology and homotopy theory (see e.g. [2]). *Categorical logic* is that part of category theory connected to logic (see [24]). *Computability theory* is the theory of computable functions, or in general of algorithmic objects (see e.g., [31]).

ML lies also at the core of theoretical computer science. Its fundamental notion of computation was first approached mathematically by logicians like Church, Turing and Gödel. Church's λ-calculus was a formal system designed to express computation and able to simulate a Turing machine. Martin-Löf's type theory (MLTT), which has its origins to Russell's so called ramified theory of types, is a predicative modification of λ-calculus with many applications in the theory of programming languages (see [25], [26]). Recently, the late Fields medalist V. Voevodsky revealed unexpected connections between homotopy theory and logic, developing *Homotopy Type Theory* (HoTT), an extension of MLTT with his *axiom of univalence* and *higher inductive types* (see [36]). MLTT, HoTT and their connections to category theory are currently among the most actively studied mathematical frameworks for the logical foundations of mathematics and theoretical computer science.

One of the most important breakthroughs in the foundations of mathematics in the 20th century that also influenced crucially the foundations of theoretical computer science was the introduction and use of intuitionistic logic in mathematics by the great topologist Brouwer. Roughly speaking, in intuitionistic logic the principle of the excluded middle (PEM) and (or) the double negation elimination (DNE) are not accepted. If A is a certain mathematical formula, the corresponding instances of these principles are the following formulas:

$$(\mathrm{PEM}_A) \qquad A \vee \neg A,$$

$$(\mathrm{DNE}_A) \qquad \neg\neg A \to A.$$

Proposition 1.1.1. *There are irrational numbers a, b such that a^b is rational.*

Proof. (with PEM): Let the following instance of PEM:

$$\sqrt{2}^{\sqrt{2}} \in \mathbb{Q} \ \vee \ \sqrt{2}^{\sqrt{2}} \notin \mathbb{Q}.$$

In the first case take $a = b = \sqrt{2}$. In the second, take $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$, as

$$a^b = \left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\sqrt{2}} = \sqrt{2}^2 = 2. \qquad \square$$

The above "proof" is accepted by a standard mathematician, although he would also prefer to have a method to decide whether $\sqrt{2}^{\sqrt{2}} \in \mathbb{Q}$ or not. If one wants to write a "program" based on this proof, this seems to be impossible. DNE can be equally problematic.

Proposition 1.1.2 (Fundamental theorem of algebra)**.** *If $p\colon \mathbb{C} \to \mathbb{C}$ is a non-constant polynomial in the set of complex numbers $\mathbb{C}$ i.e., there are $a_0, \dots a_n \in \mathbb{C}$ with $a_n \neq 0$ such that*

$$p(z) = a_0 + a_1 z + a_2 z^2 + \ldots + a_n z^n,$$

for every $z \in \mathbb{C}$, then there is $z_0 \in \mathbb{C}$ such that $p(z_0) = 0$.

Proof. (with PEM and DNE) Since p is non-constant, we have that $|p(z)| \longrightarrow +\infty$, as $|z| \longrightarrow +\infty$, since we can write

$$p(z) = z^n \left(a_n + \frac{a_{n-1}}{z} + \ldots + \frac{a_0}{z}\right).$$

Hence we can find $r > 0$ such that $|p(z)| > |p(0)| = |a_0|$, for every $z \in \mathbb{C}$ with $|z| \geq r$. If not, and here we use PEM, then for every $r > 0$ there is $z \in \mathbb{C}$ with $|z| \geq r$ and $|p(z)| \leq |p(0)|$, which contradicts the above implication $(|z| \longrightarrow +\infty) \Rightarrow (|p(z)| \longrightarrow +\infty)$. Suppose that p has no root i.e., $|p(z)| > 0$, for every $z \in \mathbb{C}$, or

$$\neg\big[\exists_{z\in\mathbb{C}}\big(p(z) = 0\big)\big].$$

The closed disc

$$D(r) = \{z \in \mathbb{C} \mid |z| \leq r\}$$

is compact and, as p is continuous, there is $z_0 \in D$ such that

$$0 < |p(z_0)| = \min\{|p(z)| \mid z \in D\} = \min\{|p(z)| \mid z \in \mathbb{C}\},$$

as $0 \in D(r)$. Hence for every $z \in \mathbb{C}$ we have that $\left|\frac{1}{p(z)}\right| = \frac{1}{|p(z)|} \leq \frac{1}{|p(z_0)|}$. Consequently, the function $\frac{1}{p}$ is bounded, and as holomorphic, it is constant (by the Liouville theorem). Hence p is constant, which is a contradiction. I.e., we showed

$$\neg\neg\big[\exists_{z\in\mathbb{C}}\big(p(z) = 0\big)\big].$$

Hence by the corresponding instance of DNE:

$$\neg\neg\big[\exists_{z\in\mathbb{C}}\big(p(z) = 0\big)\big] \Rightarrow \exists_{z\in\mathbb{C}}\big(p(z) = 0\big)$$

we get $\exists_{z\in\mathbb{C}}\big(p(z) = 0\big)$. $\square$

From the computational, or constructive, or "programs out of proofs" point of view, what we showed in the previous proof, relying already on PEM, is that it is impossible not to find a root of p. But we didn't really find it. In *classical logic* the principles PEM and DNE are accepted and, as we showed, they are used very often in standard mathematical practice. *Classical mathematics*, is, roughly speaking, mathematics done with classical logic i.e. with principles, like PEM and DNE (we shall describe classical logic explicitly later in the course). Mathematics done within constructive logic is called *constructive mathematics*. We shall see later that *intuitionistic logic is more general than classical logic*. It is a surprising fact that intuitionisitic logic is, in general, the logic of a *topos*, a certain type of category introduced by Grothendieck, one of the most important mathematicians of the second half of the 20th century, in relation to its work on algebraic geometry!

After Brouwer, it was Bishop who managed to develop a large part of mathematics using intuitionistic logic and, in contrast to Brouwer, without contradicting classical mathematics (see [4] and [5]). In this way, *constructive mathematics can also be considered more general than classical mathematics*.

Notice that we cannot constructively prove that PEM is false i.e., $\neg(A \vee \neg A)$, as there is a constructive proof of $\neg\neg(A \vee \neg A)$.

1.2 Inductive definitions

In order to define the fundamental concepts of first-order logic, we need a so-called *metatheory* $\mathcal{M}$ that permits such definitions. This metatheory $\mathcal{M}$ is, in principle, a formal theory the exact description of which is left here open. What we ask from $\mathcal{M}$ is to include some theory of natural numbers, of sets and functions and of rather simple inductively defined sets. For example, one could take $\mathcal{M}$ to be the whole Zermelo-Fraenkel set theory (ZF), described in section 1.5, but smaller parts of ZF would also suffice. One could use a constructive theory of sets as a metatheory $\mathcal{M}$. Next we explain the kind of inductive definitions that must be possible in $\mathcal{M}$.

An *inductively defined set*, or an *inductive set*, X is determined by two kinds of rules (or axioms); the *introduction rules*, which determine the way the elements of X are formed, or introduced, and the *induction principle* Ind_X for X (or *elimination rule* for X) which guarantees that X is the *least* set satisfying its introduction rules.

Example 1.2.1. The most fundamental example of an inductive set is that of the set of *natural numbers* $\mathbb{N}$. Its introduction rules are:

$$\frac{}{0 \in \mathbb{N}}\ , \quad \frac{n \in \mathbb{N}}{\mathrm{Succ}(n) \in \mathbb{N}}\ .$$

According to these rules, the elements of $\mathbb{N}$ are formed by the element 0 and by the primitive, or given successor-function $\mathrm{Succ}\colon \mathbb{N} \to \mathbb{N}$. These rules alone do not determine a unique set; for example the rationals $\mathbb{Q}$ and the reals $\mathbb{R}$ satisfy the same rules. We determine $\mathbb{N}$ by postulating that $\mathbb{N}$ is the least set satisfying the above rules. This we do in a " bottom-up" way[1] with the induction principle for $\mathbb{N}$.

[1] If we work inside the set of real numbers $\mathbb{R}$, we can also do this in a "top-down" way by defining $\mathbb{N}$ to be the intersection of all sets satisfying these introduction rules. Notice that also in this way $\mathbb{N}$ is the least subset of $\mathbb{R}$ satisfying its introduction rules.

The induction principle $\mathrm{Ind}_{\mathbb{N}}$ for $\mathbb{N}$ is the following formula (in $\mathcal{M}$):

$$\big[A(0)\ \&\ \forall_{n\in\mathbb{N}}(A(n)\Rightarrow A(\mathrm{Succ}(n)))\big]\Rightarrow\forall_{n\in\mathbb{N}}(A(n)),$$

where $A(n)$ is an arbitrary formula on $\mathbb{N}$ in $\mathcal{M}$. The interpretation of $\mathrm{Ind}_{\mathbb{N}}$ is the following: the hypotheses of $\mathrm{Ind}_{\mathbb{N}}$ say that A satisfies the two formation rules for $\mathbb{N}$ i.e., $A(0)$ and $\forall_{n\in\mathbb{N}}(A(n)\rightarrow A(\mathrm{Succ}(n)))$. In this case A is a "competitor" predicate to $\mathbb{N}$. Then, if we view A as the set of all objects such that $A(n)$ holds, the conclusion of $\mathrm{Ind}_{\mathbb{N}}$ guarantees that $\mathbb{N}\subseteq A$, i.e., $\forall_{n\in\mathbb{N}}(A(n))$. In other words, $\mathbb{N}$ is "smaller" than A, and this is the case for any such A. Notice that we use the following conventions in $\mathcal{M}$:

$$\forall_{x\in X}A(x):\Leftrightarrow\forall_x\big(x\in X\Rightarrow A(x)\big),$$

$$\exists_{x\in X}A(x):\Leftrightarrow\exists_x\big(x\in X\ \&\ A(x)\big).$$

The induction principle in an inductive definition is the main tool for proving properties of the defined set. In the case of $\mathbb{N}$, one can prove (exercise) its corresponding *recursion theorem* $\mathrm{Rec}_{\mathbb{N}}$, which determines the way one defines functions on $\mathbb{N}$. According to a simplified version of it, if X is a set, $x_0\in X$ and $g:X\rightarrow X$, there exists a unique function $f:\mathbb{N}\rightarrow X$ such that

$$f(0)=x_0,$$

$$f(\mathrm{Succ}(n))=g(f(n));\qquad n\in\mathbb{N},$$

To show e.g., the uniqueness of f with the above properties, let $h:\mathbb{N}\rightarrow X$ such that $h(0)=x_0$ and $h(\mathrm{Succ}(n))=g(h(n))$, for every $n\in\mathbb{N}$. Using $\mathrm{Ind}_{\mathbb{N}}$ on $A(n):\Leftrightarrow(f(n)=h(n))$, we get $\forall_n(A(n))$. As an example of a function defined through $\mathrm{Rec}_{\mathbb{N}}$, let $\mathtt{Double}:\mathbb{N}\rightarrow\mathbb{N}$ defined by

$$\mathtt{Double}(0)=0,$$

$$\mathtt{Double}(\mathrm{Succ}(n))=\mathrm{Succ}(\mathrm{Succ}(\mathtt{Double}(n)))$$

i.e., $X=\mathbb{N}, x_0=0$ and $g=\mathrm{Succ}\circ\mathrm{Succ}$.

Example 1.2.2. Let A be a non-empty set that we call *alphabet*. The set A^* of *words* over A is introduced by the following rules

$$\frac{}{\mathrm{nil}_{A^*}\in A^*}\ ,\qquad\frac{w\in A^*, a\in A}{w\star a\in A^*}\ .$$

The symbol nil_{A^*} denotes the empty word, while the word $w\star s$ denotes the concatenation of the word w and the letter $a\in A$. The induction principle Ind_{A^*} for A^* is the following: if $P(w)$ is any formula on A^* in $\mathcal{M}$, then

$$\big[P(\mathrm{nil}_{A^*})\ \&\ \forall_{w\in A^*}\forall_{a\in A}\big(P(w)\Rightarrow P(w\star a)\big)\big]\Rightarrow\forall_{w\in A^*}(P(w)).$$

A simplified version of the corresponding recursion theorem Rec_{A^*} is the following: If X is a set, $x_0\in X$, and if $g_a:X\rightarrow X$, for every $a\in A$, there is a unique function $f:A^*\rightarrow X$ such that

$$f(\mathrm{nil}_{A^*})=x_0,$$

$$f(w\star a)=g_a(f(w));\qquad w\in A^*,\ a\in A.$$

As an example of a function defined through Rec_{A^*}, if $X = A^*$, $w_0 \in A^*$ and if $g_a(w) = w \star a$, for every $a \in A$, let the function $f_{w_0} : A^* \to A^*$ defined by

$$f_{w_0}(\mathrm{nil}_{A^*}) = w_0,$$

$$f_{w_0}(w \star a) = g_a(f_{w_0}(w))$$

i.e., $f_{w_0}(w) = w_0 \star w$ is the concatenation of the words w_0 and w (we use the same symbol for the concatenation of a word and a symbol and for the concatenation of two words).

If ZF is our metatheory $\mathcal{M}$, then the proof of the recursion theorem that corresponds to an inductive definition can be complicated. If as metatheory we use a theory like Martin-Löf's type theory MLTT, we will see that there is a completely mechanical, hence trivial, way to recover the corresponding recursion rule from the induction rule of an inductive definition.

1.3 First-order logic in a nutshell

A *first-order language* $\mathcal{L}$ comprises a set $\mathtt{Var}$ of *variables* x, y, z, a set of logical symbols $\{\to, \wedge, \vee, \forall, \exists, (,), , \}$, a set $\mathtt{Rel}$ of *relation symbols* and a set $\mathtt{Fun}$ of function symbols. A constant symbol is a function symbol of arity 0. The symbol $\bot$ (read falsum) is a fixed relation symbol of arity 0, contained always for us in $\mathtt{Rel}$. Let $\mathtt{Fun}^{(n)}$ be the set of function symbols of arity n and $\mathtt{Rel}^{(n)}$ the set of relation symbols of arity n (these can also be empty). Usually, we use only the sets $\mathtt{Rel}$ and $\mathtt{Fun}$ to describe a first-order language. The set $\mathtt{Term}$ of *terms* of $\mathcal{L}$ is defined inductively by the rules

$$\frac{x \in \mathtt{Var}}{x \in \mathtt{Term}}, \qquad \frac{c \in \mathrm{Const}}{c \in \mathtt{Term}},$$

$$\frac{n \in \mathbb{N}^+, \quad t_1, \ldots, t_n \in \mathtt{Term}, \quad f \in \mathtt{Fun}^{(n)}}{f(t_1, \ldots, t_n) \in \mathtt{Term}},$$

and the set $\mathtt{Form}$ of *formulas* of $\mathcal{L}$ by the rules

$$\frac{n \in \mathbb{N}, \quad t_1, \ldots, t_n \in \mathtt{Term}, \quad R \in \mathtt{Rel}^{(n)}}{R(t_1, \ldots, t_n) \in \mathtt{Form}},$$

$$\frac{A, B \in \mathtt{Form}}{A \to B,\ A \wedge B,\ A \vee B \in \mathtt{Form}},$$

$$\frac{A \in \mathtt{Form}, \quad x \in \mathtt{Var}}{\forall_x A,\ \exists_x A \in \mathtt{Form}}.$$

The formulas of the form $R(t_1, \ldots, t_n)$ are called *prime formulas*, or *atomic formulas*. If $r, s \in \mathtt{Term}$ and $\sim \in \mathtt{Rel}^{(2)}$, we also write $r \sim s$ for the prime formula $\sim (r, s)$. Since $\bot \in \mathtt{Rel}^{(0)}$, we get $\bot \in \mathtt{Form}$. We call $A \to B$ the *implication* from A to B, $A \wedge B$ the *conjunction* of A, B, and $A \vee B$ the *disjunction* of A, B. The *negation* $\neg A$ of a formula A is defined as the formula

$$\neg A = A \to \bot.$$

We use the notational convention

$$A \to B \to C = A \to (B \to C).$$

The formulas generated by the prime formulas are called *complex*, or *non-atomic* formulas. A formula $\forall_x A$ is called a *universal* formula, and a formula $\exists_x A$ is called an *existential* formula.

If $R, S \in \mathtt{Rel}^{(1)}$, we have the following first-order formulas:

$$\bot \to \bot,$$

$$\forall_x(\bot \to \bot),$$

$$\exists_x(R(x) \vee S(x)).$$

We have added two parentheses (left and right) in both last examples, in order to make them clear to read. A first-order language may have its own equality.

Definition 1.3.1. *Let $\mathcal{L}$ be a first-order language and let $\approx \in \mathtt{Rel}^{(2)}$. The set $\mathrm{Eq}_{\mathcal{L}}$ of $\mathcal{L}$-equality axioms consists of (the universal closures of) the following $\mathcal{L}$-formulas:*

(Eq_1) $x \approx x$,

(Eq_2) $x \approx y \to y \approx x$,

(Eq_3) $x \approx y \wedge y \approx z \to x \approx z$,

(Eq_4) $x_1 \approx y_1 \ \wedge \ \ldots \ \wedge \ x_n \approx y_n \to f(x_1, \ldots, x_n) \approx f(y_1, \ldots, y_n)$,

(Eq_5) $x_1 \approx y_1 \ \wedge \ \ldots \ \wedge \ x_n \approx y_n \ \wedge \ R(x_1, \ldots, x_n) \to R(y_1, \ldots, y_n)$,

for all n-ary function symbols f, for all relation symbols R of $\mathcal{L}$, and for all $n \in \mathbb{N}$.

If f is a 0-ary function symbol, then Eq_4 has as special case the axiom $c \approx c$. Notice that this equality is the given "internal" equality of $\mathcal{L}$ and must not be confused with the "external" equality $x = y$, which is the metatheoretical equality of the set $\mathtt{Var}$. Consequently, if $t, s \in \mathtt{Term}$, we get the following formula of $\mathcal{L}$ with equality:

$$t \approx s.$$

If we have another set of variables X, Y, Z, for a different *sort*, we construct similarly a *second-order* language. As prime formulas are defined by an n-relation symbol and n-number of terms, it is very often the case that in order to define a function on $\mathtt{Form}$, we need first to define a corresponding function on $\mathtt{Term}$. These functions are defined by the *recursion* (more on this issue later). If X is a set, $\mathcal{P}^{\mathrm{fin}}(X)$ denotes the set of finite subsets of X. If $Y, Z \subseteq X$, then $Y \setminus Z = \{x \in X \mid x \in Y \ \& \ x \notin Z\}$.

Definition 1.3.2. *Let the function $\mathrm{FV}_{\mathtt{Term}} : \mathtt{Term} \to \mathcal{P}^{fin}(\mathtt{Var})$ defined by*

$$\mathrm{FV}_{\mathtt{Term}}(x) = \{x\},$$

$$\mathrm{FV}_{\mathtt{Term}}(c) = \emptyset,$$

$$\mathrm{FV}_{\mathtt{Term}}(f(t_1, \ldots, t_n)) = \bigcup_{i=1}^{n} \mathrm{FV}_{\mathtt{Term}}(t_i).$$

The function $\mathrm{FV}_{\mathtt{Form}} : \mathtt{Form} \to \mathcal{P}^{fin}(\mathtt{Var})$ is defined by

$$\mathrm{FV}_{\mathtt{Form}}(R) = \emptyset, \quad R \in \mathtt{Rel}^{(0)},$$

$$\mathrm{FV}_{\mathtt{Form}}(R(t_1, \ldots, t_n)) = \bigcup_{i=1}^{n} \mathrm{FV}_{\mathtt{Term}}(t_i), \quad R \in \mathtt{Rel}^{(n)}, n \in \mathbb{N}^{+},$$

$$\mathrm{FV}_{\mathtt{Form}}(A \square B) = \mathrm{FV}_{\mathtt{Form}}(A) \cup \mathrm{FV}_{\mathtt{Form}}(B),$$

$$\mathrm{FV}_{\mathtt{Form}}(\triangle_x A) = \mathrm{FV}_{\mathtt{Form}}(A) \setminus \{x\}.$$

If $\mathrm{FV}(A) = \emptyset$*, then* A *is called a sentence, or a closed formula.*

According to Definition 1.3.2, a variable y is free in a prime formula A, if just occurs in A, it is free in $A \square B$, if it is free in A or free in B, and it is free in $\triangle_x A$, if it is free in A and $y \neq x$. E.g., the formulas

$$\forall_y(R(y) \to S(y)), \quad \forall_y(R(y) \to \forall_z S(z))$$

are sentences, while y is free in the formula

$$(\forall_y(R(y)) \to S(y).$$

If a variable x is free in A, then x can be *substituted*[2] by some term $t \in \mathtt{Term}$. If $\mathrm{FV}(A) \subseteq \{x\}$, we write $A(x)$, while $A(t)$ is the formula generated by the substitution of x by t.

Although mathematicians for centuries were proving mathematical statements, which in many cases these can be written as first-order formulas, it was only until recently that the following question was systematically studied in mathematical terms:

"what does it mean to prove some $A \in \mathtt{Form}$**?"**

In order to give an answer to this question, a set of (logical) rules, or (logical) axioms, is added that governs the derivation, or the proof, of some $A \in \mathtt{Form}$. E.g., PEM or DNE are such logical axioms. For example, the *introduction* and *elimination rules* for implication are:

$$\frac{\begin{array}{c}[u\colon A]\\ \vert\ M\\ B\end{array}}{A \to B} \to^{+} u \qquad \frac{\vert\ M \qquad \vert\ N}{B} \to^{-}$$

These rules are represented by *derivation trees*, where in the left tree the formula $A \to B$ is "introduced", and in the second it is "eliminated" (a complete presentation of Gentzen's derivation rules of *natural deduction* rules will be given later). The $\mathcal{L}$-equality axioms are also included, if there is an (internal) equality in $\mathcal{L}$. If some extra, *non-logical axioms* are added, then we get an $\mathcal{L}$*-theory* T, or a *first-order theory* T. If T is a first-order theory over $\mathcal{L}$, and $A \in \mathtt{Form}$, we say that A is *derivable* or *provable* by T, in symbols

$$T \vdash A,$$

if there is a derivation, or proof, of A by formulas $A_1, \ldots, A_n$ in T. If

$$T \nvdash A \ \&\ T \nvdash \neg A,$$

then A is called *independent* of T.

[2]These concepts will be defined properly later in the course.

Example 1.3.3. The first-order language of *arithmetic* is the pair $\mathcal{N} := (\{\bot, \approx\}, \{0, S, +, \cdot\})$, or simpler $\mathcal{N} := (\bot, \approx, 0, S, +, \cdot)$, where $0 \in \mathrm{Const}, S \in \mathtt{Fun}^{(1)}$, and $+, \cdot \in \mathtt{Fun}^{(2)}$. The non-logical axioms of *Peano arithmetic* $\mathfrak{N}$ are the universal closures of the following formulas in $\mathcal{N}$:

(PA_1) $\neg S(x) \approx 0$ (or $S(x) \not\approx 0$).

(PA_2) $S(x) \approx S(y) \to x \approx y$.

(PA_3) $x + 0 \approx 0$.

(PA_4) $x + S(y) \approx S(x + y)$.

(PA_5) $x \cdot 0 \approx 0$.

(PA_6) $x \cdot S(y) \approx x \cdot y + x$,

and the axiom-scheme (PA_{7A}) of induction :

$$A(0) \wedge \forall_x (A(x) \to A(S(x)) \to \forall_x A(x),$$

where $A(x)$ is an arbitrary formula of $\mathcal{N}$.

1.4 The Brouwer-Heyting-Kolmogorov-interpretation

A first, informal, answer to the question mentioned above was given by Brouwer, his student Heyting, and, independently, from Kolmogorov. The combination of the proof-interpretation of formlulas given by these mathematicians is called the Brouwer-Heyting-Kolmogorov-interpretation, or the BHK-interpretation. Notice that this is interpretation presupposes an informal, or primitive, or unexplained, notion of proof. Moreover, the interpretation of a proof of a prime formula, other than $\bot$, is not addressed in the BHK-interpretation.

Definition 1.4.1 (BHK-interpretation)**.** *Let $A, B \in \mathtt{Form}$, such that it is understood what it means "q is a proof (or witness, or evidence) of A" and "r is a proof of B".*

($\wedge$) *A proof of $A \wedge B$ is a pair (p_0, p_1) such that p_0 is a proof of A and p_1 is a proof of B.*

($\to$) *A proof of $A \to B$ is a rule r that associates to any proof p of A a proof $r(p)$ of B.*

($\vee$) *A proof of $A \vee B$ is a pair (i, p_i), where if $i = 0$, then p_0 is a proof of A, and if $i = 1$, then p_1 is a proof of B.*

($\bot$) *There is no proof of $\bot$.*

For the next two rules let $P(x)$ be a formula, such that it is understood what it means "q is a proof of $P(a)$, where $a \in A$".

($\forall$) *A proof of $\forall_{x \in A} P(x)$ is a rule R that associates to any given element a of A a proof R_a of $P(a)$.*

($\exists$) *A proof of $\exists_{x \in A} P(x)$ is a pair (a, q), where a is an element of A and q is a proof of $P(a)$.*

We write $p : B$ to denote that p is a proof of the formula B.

The notions of rule in the clauses ($\to$) and ($\forall$) are unclear, and are taken as primitive. As we have already said, the nature of a proof, or a witness, is also left unexplained. These will be clarified in MLTT, where proof-terms will be first class citizens. A formal version of the BHK-interpretation of $\mathtt{Form}$ is a so-called *realisability interpretation* (see [35]). Within Gentzen's natural deduction the BHK-interpretation is reflected through the so-called *Curry-Howard correspondence*. According to it, a derivation tree is represented by some typed *derivation term*, where the derived formula is the *type* of the term (and displayed as a superscript). E.g.,

Derivation	Term
$\dfrac{\begin{array}{c}[u\colon A]\\ \mid M\\ B\end{array}}{A \to B} \to^{+} u$	$(\lambda_{u^A} M^B)^{A \to B}$
$\dfrac{\begin{array}{cc}\mid M & \mid N\\ A \to B & A\end{array}}{B} \to^{-}$	$(M^{A \to B} N^A)^B$

Table 1.1: Derivation terms for $\to$

Despite these problems, the BHK-interpretation captures essential elements of the mathematical process of proof. Especially, it captures, informally, the notion of a *constructive proof*, as the clauses for $(\vee)$ and $(\exists)$ indicate.

Example 1.4.2. Let the formula $D = (A \to B \to C) \to (A \to B) \to A \to C$, which, according to our notational convention, is the formula

$$(A \to (B \to C)) \to ((A \to B) \to (A \to C)).$$

According to BHK, a proof

$$p : (A \to B \to C) \to ((A \to B) \to (A \to C))$$

is a rule that sends a supposed proof $q : A \to (B \to C)$ to a proof

$$p(q) : (A \to B) \to (A \to C),$$

which, in turn, is a rule that sends a proof $r : A \to B$ to a proof $[p(q)](r) = [p(q)](r) : A \to C$. This proof is a rule that sends a proof $s : A$ to a proof $\big[[p(q)](r)\big](s) : C$. Hence we need to define the later proof through our supposed proofs. By definition $q(s) : B \to C$, and hence $[q(s)](r(s)) : C$. Thus we define

$$\big[[p(q)](r)\big](s) = [q(s)](r(s)).$$

Example 1.4.3. Let the formula

$$P = \forall_{x \in A}(\bot \to Q) \to \bot \to \forall_{x \in A} Q.$$

According to BHK, a proof $p : \forall_{x \in A}(\bot \to Q) \to \bot \to \forall_{x \in A} Q$ is a rule that sends a supposed proof $q : \forall_{x \in A}(\bot \to Q)$ to a proof

$$p(q) : \bot \to \forall_{x \in A} Q,$$

which is a rule that sends a proof $r : \bot$ to some proof

$$[p(q)](r) : \forall_{x \in A} Q.$$

The proof $q : \forall_{x\in A}(\bot \to Q)$ is understood as a family of proofs

$$q = \big(q_x : \bot \to Q\big)_{x\in A},$$

and, similarly, the required proof $[p(q)](r) : \forall_{x\in A} Q$ is a family of proofs

$$[p(q)](r) = \Big(\big[[p(q)](r)\big]_{x\in A} : Q\Big)_{x\in A}.$$

We define this family of proofs by the rule

$$\big[[p(q)](r)\big]_{x\in A} = q_x(r).$$

Example 1.4.4. A BHK-proof $p : A \to A$ is a rule that associates to every $q : A$ a proof of A. Clearly the identity rule $p(q) = q$ is such a proof.

Example 1.4.5. A BHK-proof $p^* : A \to \neg\neg A$ is a rule that associates to every $q : A$ a proof $p^*(q) : (A \to \bot) \to \bot$. If $r : A \to \bot$, we need to get a proof $[p^*(q)](r) : \bot$. For that we define $[p^*(q)](r) = r(q)$.

It is easy to see that there is no straightforward method to find a BHK-proof of the converse implication DNE_A. There are some instances of DNE though, that can be shown constructively. E.g., it is possible to find (exercise) a BHK-proof of

$$(\mathrm{DNE}_{\neg A}) \qquad \neg\neg\neg A \to \neg A.$$

1.5 Cantor sets I and Zermelo-Fraenkel sets

The theory of sets is a very recent enterprise in the history of mathematics, which was introduced by Cantor. At the beginning Cantor used the *Full Comprehension Scheme* (FCS):

$$\exists_u(u = \{x \mid \phi(x)\}),$$

which guarantees the existence of a set generated by *any* formula ϕ that is formed by the symbol of elementhood $\in$. Russell, and independently Zermelo, found that Cantor's principle is contradictory: if $\phi(x)$ is the formula $x \notin x :\Leftrightarrow \neg(x \in x)$, then by FCS we have that

$$R = \{x \mid x \notin x\}$$

is a set. The contradiction

$$R \in R \Leftrightarrow R \notin R$$

is known as *Russell's paradox*. Zermelo's *Restricted Comprehension Scheme* (RCS), also known as *Separation Scheme*,

$$\exists_u(u = \{x \in v \mid \phi(x)\})$$

replaced the problematic principle FCS, and Russell's paradox is avoided. If V is the *universe* of all sets i.e.,

$$V = \{x \mid x = x\},$$

RCS implies that $V \notin V$: if $V \in V$, then by RCS we get $u = \{x \in V \mid x \notin x\} \in V$ and then $u \in u \Leftrightarrow u \notin u$. If FCS was not contradictory, we wouldn't need so many axioms to describe our intuition about sets. E.g., the union of two sets would be defined as $u \cup v = \{x \mid x \in u \vee x \in v\}$.

Example 1.5.1. The first-order language of set theory is the pair $\mathcal{S} = (\{\perp, \approx, \in\}, \emptyset)\})$, which is written for simplicity as $\mathcal{S} = (\perp, \approx, \in)$, where $\in$ is in $\mathtt{Rel}^{(2)}$. The first-order non-logical axioms of the first-order *Zermelo-Fraenkel set theory* ZF in the first-order language $\mathcal{S}$ are the following[3]:

Extensionality: $\forall_{x,y}\big(\forall_z(z \in x \leftrightarrow z \in y) \rightarrow x \approx y\big)$.

Empty set: $\exists_x \forall_y (y \notin x)$.

Unordered pair: $\forall_{x,y}\exists_z\forall_w(w \in z \leftrightarrow w \approx x \vee w \approx y)$.

Union: $\forall_x\exists_y\forall_z\big(z \in y \leftrightarrow \exists_w(w \in x \ \&\ z \in w)\big)$.

Replacement Scheme: If $\phi(x, y, \vec{w})$ is a function formula[4], then

$$\forall_x\exists_v\forall_y\big(y \in v \leftrightarrow \exists_z(z \in x \ \wedge \ \phi(z, y, \vec{w}))\big).$$

Power-set: $\forall_x\exists_y\forall_z\big(z \in y \leftrightarrow \forall_w(w \in z \rightarrow w \in x)\big)$.

Foundation: $\forall_x\Big(x \neq \emptyset \rightarrow \exists_z\big(z \in x \ \wedge \ \neg\exists_w(w \in z \wedge w \in x)\big)\Big)$.

Infinity: $\exists_x\big(\emptyset \in x \ \wedge \ \forall_y(y \in x \rightarrow y \cup \{y\} \in x)\big)$.

Unlike the axioms for vector spaces for example (first came the examples, or *models*, of vector spaces, and then came the axioms for a linear space) the axioms for sets were given first and their models were studied only afterwords! The axioms of ZF are generally "accepted" by the classical mathematicians, and ZF is considered the standard foundation of classical mathematics. The are also constructive theories of sets i.e., set theories within intuitionistic logic (see [1]).

In ZF *functions are not basic objects*, since they are certain sets. A function $f : x \rightarrow y$, where x, y are sets in ZF, is an appropriate subset of $x \times y = \{(a, b) \mid a \in x \wedge b \in y\}$, where the notion of an ordered pair (a, b) is defined through the notion of an unordered pair as follows:

$$(a, b) := \big\{\{a\}, \{a, b\}\big\}.$$

Clearly,

$$\big\{\{a\}, \{a, b\}\big\} \approx \big\{\{c\}, \{c, d\}\big\} \Leftrightarrow a \approx c \ \&\ b \approx d.$$

If to ZF we add the axiom of choice (AC), we get the system ZFC. One formulation of it is the following [5] :

Axiom of choice (AC): If $(A_j)_{j\in J}$ is a family of non-empty sets indexed by some set J, there is a choice function

$$f : J \rightarrow \bigcup_{j\in J} A_j,$$

such that $f(j) \in A_j$, for every $j \in J$.

With the use of Cohen's method of *forcing* it is shown that

$$\mathrm{ZF} \nvdash \mathrm{AC},$$

[3] An introduction to the axiomatic set theory can be found in [12]. More advanced books are [21] and [22].
[4] I.e., if $\phi(x, y, \vec{w})$ and $\phi(x, z, \vec{w})$, then $y = z$.
[5] Another formulation of AC is: $\forall_v\exists_{f:v\rightarrow V}\forall_{x\in v}\big(x \neq \emptyset \Rightarrow f(x) \in x\big)$.

while with the use of the method of *inner models* Gódel showed that

$$\mathrm{ZF} \nvdash \neg\mathrm{AC}$$

i.e., AC is independent from ZF. AC is not that "innocent". When a proof uses it, usually this is noted. It has many important consequences in standard mathematics, for example every ideal in a ring is contained in a maximal ideal, every vector space has a basis, and every product of compact spaces is compact. For many equivalent formulations of AC see [32]. But AC has also some counter-intuitive consequences like the *Banach-Tarski paradox*: it is possible to decompose the 3-dimensional solid unit ball into finitely many pieces and, using only rotations and translations, reassemble the pieces into two solid balls each with the same volume as the original. Another unexpected consequence of the AC is the following.

Theorem 1.5.2 (Diaconescu 1975)**.** *The* AC *together with some very small part of* ZF *implies constructively i.e., without the use of* PEM *or* DNE*, the principle of the excluded middle* PEM*.*

Proof. Let A be any formula, $\mathbf{2} = \{0, 1\}$, and let the sets

$$A_0 = \{x \in \mathbf{2} \mid x = 0 \vee A\},$$

$$A_1 = \{x \in \mathbf{2} \mid x = 1 \vee A\}.$$

Since $0 \in A_0$ and $1 \in A_1$, the sets A_0 and A_1 are non-empty. By AC there is

$$f : \mathbf{2} \to \bigcup_{j \in \mathbf{2}} A_j = A_0 \cup A_1 \subseteq \mathbf{2}$$

such that

$$f(0) \in A_0 \Leftrightarrow f(0) = 0 \vee A \quad \text{and}$$

$$f(1) \in A_1 \Leftrightarrow f(1) = 1 \vee A.$$

Since f takes values in $\mathbf{2}$, we consider the following cases. If $f(0) = 1$, then, since $f(0) \in A_0$, we get A. If $f(0) = 0$, we consider the two possible cases for $f(1)$. If $f(1) = 0$, then, since $f(1) \in A_1$, we get A. If $f(1) = 1$, we show $\neg A$ i.e., we reach a contradiction by supposing A. Suppose A. In this case $A_0 = A_1 = \mathbf{2}$. Let the set $\{A_0, A_1\}$ and let the function

$$f^* : \{A_0, A_1\} \to \mathbf{2},$$

$$f^*(A_0) = f(0) = 0 \quad \& \quad f^*(A_1) = f(1) = 1.$$

Since $A_0 = A_1$ though, we get $f^*(A_0) = f^*(A_1)$ i.e., $0 = 1$, which is a contradiction. Hence, we showed $A \vee \neg A$. □

In the previous proof we used the (full) Separation Scheme, and the axioms for the unordered pair, the empty set to define 0 and 1, and the extensionality axiom. One could argue that the axiom most crucial to the previous proof is the axiom of choice. As from the constructive point of view the other axioms are quite "safe" to use, except maybe the full separation axiom, it is AC the source of the non-constructivity.

Definition 1.5.3. *A partially ordered set, or a poset, is a pair $(I, \preceq)$, where I is a set, and $\preceq \subseteq I \times I$ satisfying the following conditions:*
(i) $\forall_{i\in I}\big(i \preceq i\big)$.
(ii) $\forall_{i,j\in I}\big(i \preceq j \;\&\; j \preceq i \Rightarrow i = j\big)$.
(iii) $\forall_{i,j,k\in I}\big(i \preceq j \;\&\; j \preceq k \Rightarrow i \preceq k\big)$.
A subset C of a poset $(I, \preceq)$ is called a chain in I, or a totally ordered subset of I, if

$$\forall_{c,c'\in C}\big(c \preceq c' \;\vee\; c' \preceq c\big).$$

A subset J of I is bounded in I, if there is $i_0 \in I$ such that $\forall_{j\in J}\big(j \preceq i_0\big)$. In this case i_0 is called a bound of J. An element i_0 of I is called maximal in I, if

$$\forall_{i\in I}\big(i_0 \preceq i \Rightarrow i = i_0\big).$$

A bound i_0 of I itself is called the maximum element[6] *of I. If the poset $(I, \preceq)$ is clear from the context, we just say C is a chain, J is bounded, and i_0 is a maximal element. For simplicity we say that I is a poset, and we do not write $(I, \preceq)$, when $\preceq$ is clear from the context.*

If $\mathcal{P}(I)$ denotes the *powerset* of the set I i.e., the set of all subsets of I, then $\mathcal{P}(I)$ is *partially ordered* by the relation $A \subseteq B$, "the subset A is included in the subset B" i.e.,

$$A \subseteq B :\Leftrightarrow \forall_{i\in I}\big(i \in A \Rightarrow i \in B\big).$$

An infinite, countable chain C in $\mathcal{P}(X)$ can take the form of a sequence

$$A_1 \subseteq A_2 \subseteq \ldots \subseteq A_n \subseteq \ldots,$$

and

$$\bigcup_{n=1}^{\infty} A_n = \big\{i \in I \mid \exists_{n\geq 1}\big(i \in A_n\big)\big\}$$

is a bound of C. Clearly, X is the maximum element of $\mathcal{P}(X)$, and $\emptyset$ is the minimum of $\mathcal{P}(X)$.

Theorem 1.5.4. AC *implies classically i.e., with the use of* PEM*, Zorn's lemma: if I is a non-empty poset, such that every chain in I is bounded, then I has a maximal element*[7].

Proof. **(Sketch)** Let $(I, \preceq)$ be a non-empty poset, such that every chain in I is bounded. Let C be a fixed chain in I, where by hypothesis

$$B(C) = \{i \in I \mid i \text{ is a bound of } C\} \neq \emptyset.$$

By PEM either C contains a maximal element of I, or not. In the first case, the existence of a maximal element follows immediately. Suppose that C does not contain a maximal element of I (Hyp_1). In this case, we show that

$$B^*(C) = \{i \in B(C) \mid i \notin C\} \neq \emptyset.$$

[6] A maximum element i_0 is uniquely determined i.e., if j_0 is also bound of I, then $j_0 = i_0$. The maximum element is also a maximal element, while the converse is not generally the case.

[7] If $I = \emptyset$, then it is easy to show that the ex falsum quodlibet scheme $\mathrm{Efq}_A\colon \bot \to A$, for every formula A, implies that if every chain of $\emptyset$ is bounded, then $\emptyset$ has a maximal element.

Suppose that $B^*(C) = \emptyset$ i.e.,

$$\forall_{i \in B(C)}(i \in C) \qquad (\text{Hyp}_2).$$

Let $i_0 \in B(C)$. By Hyp_2 we get $i_0 \in C$. If $i \in I$, such that $i_0 \preceq i$, then $i \in B(C)$ too, hence by Hyp_2 we have that $i \in C$. Since $i_0 \in B(C)$, we get $i \preceq i_0$, hence by the transitivity of $\preceq$ we conclude that $i_0 = i$. Since i is an arbitrary element of I, we have that i_0 is a maximal element in I that it is also in C, which contradicts the hypothesis Hyp_1. Let

$$\mathcal{C} = \big\{C \subseteq I \mid C \text{ is a chain in } I \text{ that contains no maximal element of } I\big\}.$$

By the previous remark the family $\big(B^*(C)\big)_{C \in \mathcal{C}}$ is a family of non-empty sets indexed by $\mathcal{C}$, hence by AC there is a function

$$f : \mathcal{C} \to \bigcup_{C \in \mathcal{C}} B^*(C)$$

such that

$$f(C) \in B^*(C), \quad \text{for every } C \in \mathcal{C}.$$

The idea of the rest of the proof is the following. Let $i_0 \in I$. By PEM, either i_0 is maximal in I, and we are done, or it is not. In the latter case, $\{i_0\}$ is a chain in I that contains no maximal element of I i.e., $\{i_0\} \in \mathcal{C}$. Hence,

$$f\big(\{i_0\}\big) \in B^*\big(\{i_0\}\big) = \big\{i \in B\big(\{i_0\}\big) \mid i \notin \{i_0\}\big\}$$

i.e., $f\big(\{i_0\}\big) = i_1$, such that $i_0 \prec i_1 :\Leftrightarrow i_0 \preceq i_1 \;\&\; i_0 \neq i_1$. Repeating the same argument, either i_1 is maximal in I, or not. In the latter case, $\{i_0, i_1\}$ is a chain in I that contains no maximal element of I i.e., $\{i_0, i_1\} \in \mathcal{C}$. Consequently,

$$i_0 \prec i_1 \prec f\big(\{i_0, i_1\}\big).$$

Proceeding similarly, and repeating these steps at most as many times as the *cardinality* of I, the procedure will terminate, something which turns out to be equivalent to the existence of a maximal element in I. □

Theorem 1.5.5 (ZL)**.** *A non-trivial vector space X has a basis.*

Proof. If we define the set

$$I(X) = \{Y \subseteq X \mid Y \text{ is linearly independent}\},$$

then $\emptyset \in I(X)$, hence $I(X)$ is non-empty. If $Y, Z \in I(X)$, we define $Y \preceq Z :\Leftrightarrow Y \subseteq Z$, which is a partial order on $I(X)$. If $C \subseteq I(X)$ is a chain in $I(X)$, then

$$\bigcup C = \{x \in X \mid \exists_{A \in C}(x \in A)\}$$

is a bound of C in $\mathcal{P}(X)$, and it is also a bound in $I(X)$ i.e., $\bigcup C \in I(X)$. Show this, and complete the proof using Zorn's lemma. □

Notice that the previous proof of existence of a basis is very "indirect", as it provides no algorithm, to find, or construct a basis. One can show[8] that the existence of a basis of a non-trivial vector space implies ZL, hence "the existence of a base of a linear space" and ZL are equivalent (over ZF). One can show similarly the following stronger version of Theorem 1.5.5.

[8]For that see [6]. In [32] many statements from classical mathematics are shown to be equivalent to the axiom of choice.

Theorem 1.5.6 (ZL). *If Y is a linearly independent subset of a non-trivial linear space X, there is a basis B of X, such that $Y \subseteq B$.*

Proof. Exercise. □

1.6 Cantor sets II and Bishop sets

In [8], pp. 114-5, Cantor described a set as follows:

> A manifold (a sum, a set) of elements belonging to some conceptual sphere is called well-defined if, on the basis of its definition and in accordance with the logical principle of the excluded third, it must be regarded as internally determined, both whether any object of that conceptual sphere belongs as an element to the mentioned set, and also whether two objects belonging to the set, in spite of formal differences in the mode of givenness, are equal to each other or not.

Errett Bishop (1928-1983) was an outstanding mathematician with contributions in the theory of Banach spaces, like the Bishop-Phelps theorem, in the theory of complex manifolds, like the embedding theorem for an n-dimensional Stein manifold, in the theory of integral representation of points in compact convex sets, like the the Bishop-de Leeuw theorem, and in many other areas of analysis. Moreover, with his work [4] and [5] he revolutionized constructive analysis and the foundations of mathematics. He and Brouwer are the most important constructive mathematicians of the previous century. Bishop developed the informal system of constructive mathematics BISH, a common territory between classical mathematics, intuitionism and recursive mathematics. This means that if p is a proof of a proposition Q in BISH, then p is a proof of Q interpreted in classical mathematics, and at the same time p is a proof of Q interpreted in other intuitionistic systems of mathematics like Brouwer's intuitionistic mathematics INT, or Markov's recursive mathematics RUSS. All these pairwise incompatible disciplines can be seen then as special varieties of Bishop's constructive mathematics. In Bishop's book [4], and in many publications after 1967, a large part of classical mathematics has found its constructive counterpart in BISH.

Bishop's intuitive notion of set is similar to Cantor's, except that he does not invoke PEM. In MLTT a Bishop set corresponds to the type-theoretic notion of a *setoid*. His informal system of constructive mathematics BISH, in which large parts of mathematics can be developed (see [4], [5]), has the following main features (see also [29]):

- The logic of proofs in BISH is intuitionistic.
- There is a primitive equality between terms, denoted by $s := t$, and it is understood as a *definitional*, or *logical*, equality.
- There is one only basic, or primitive set, the set of natural numbers $\mathbb{N}$ for which the Peano axioms hold.
- A (non-inductive)*defined totality* X is defined by a membership condition $x \in X :\Leftrightarrow \mathcal{M}_X(x)$, where $\mathcal{M}_X$ is a formula with x as a free variable.
- A *defined totality X with equality*, or simply, a totality X with equality is a defined totality X equipped with an equality condition $x =_X y :\Leftrightarrow \mathcal{E}_X(x, y)$, where $\mathcal{E}_X(x, y)$ is a formula with free variables x and y that satisfies the conditions of an equivalence relation i.e., $\mathcal{E}_X(x, x)$ and $\mathcal{E}_X(x, y) \Rightarrow \mathcal{E}_X(y, x)$, and $[\mathcal{E}_X(x, y)\ \&\ \mathcal{E}_X(y, z)] \Rightarrow \mathcal{E}_X(x, y)$.

- A *defined set* is a preset with a given equality.
- A *set* is either a primitive set, or a defined set. Clearly, each set X has its own equality $=_X$. This is in contrast to the "global" equality of sets in ZF.
- Sets are inductively defined in the extension BISH* of BISH with inductive definitions with rules of countably many premisses.
- An undefined notion of mathematical *construction*, or *algorithm*, or of finite *routine* is considered as primitive i.e., unexplained.
- *Assignment routines* are basic objects and functions are certain assignment routines. Moreover, *function extensionality* holds by definition i.e., if $\mathbb{F}(X,Y)$ is the set of functions from X to Y its equality is defined by

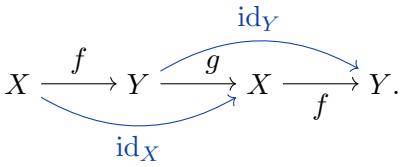
$$f =_{\mathbb{F}(X,Y)} g :\Leftrightarrow \forall_{x\in X}\big(f(x) =_Y f(x)\big).$$

- A global operation $(\cdot,\cdot)$ of pairing is also considered primitive. I.e., if s,t are terms, their pair (s,t) is a new term. The corresponding equality axiom is

$$(s,t) := (s',t') :\Leftrightarrow s := s' \;\&\; t := t'.$$

  The global projection routines $\mathbf{pr}_1(s,t) := s$ and $\mathbf{pr}_2(s,t) := t$ are also primitive.
- *Dependent assignment routines* are basic objects and dependent functions are certain dependent assignment routines.
- There is one open-ended universe $\mathbb{V}_0$ of sets. $\mathbb{V}_0$ is a not a set, but a *proper class*. Moreover, $\mathbb{V}_0$ is by definition *univalent* i.e., its equality is given by the following condition

$$X =_{\mathbb{V}_0} Y :\Leftrightarrow \exists_{f\in\mathbb{F}(X,Y)}\exists_{g\in\mathbb{F}(Y,X)}\big(g\circ f = \mathrm{id}_X \;\&\; f\circ g = \mathrm{id}_Y\big)$$



  In this case we write $(f,g) : X =_{\mathbb{V}_0} Y$.
- *Proof-relevance* can be added to some degree. E.g., in the definition of the canonical equality of $\mathbb{V}_0$ above we write $(f,g) : X =_{\mathbb{V}_0} Y$ and we define the set

$$\mathrm{Eq}(X,Y) := \big\{(f,g) \in \mathbb{F}(X,Y)\times\mathbb{F}(Y,X) \mid (f,g) : X =_{\mathbb{V}_0} Y\big\}$$

  of all objects that "witness", or "realise", or prove the equality $X =_{\mathbb{V}_0} Y$. The equality of $\mathrm{Eq}(X,Y)$ is the canonical one i.e., $(f,g) =_{\mathrm{Eq}(X,Y)} (f',g') :\Leftrightarrow f =_{\mathbb{F}(X,Y)} f' \;\&\; g =_{\mathbb{F}(Y,X)} g'$.
- Negation is avoided, as much as possible. Positively defined objects are used, as counterparts to negatively defined concepts. E.g., inequality of elements x,x' of a set X is not defined by $\neg(x =_X x')$, but through a positively defined inequality condition $x \neq_X x'$, for which we show that the axioms of an *apartness* relation are satisfied.

- A very weak form of choice, like the *principle of countable choice* (CC) is used

  (CC) $$\forall_{m\in\mathbb{N}}\exists_{y\in Y}R(m,y)\Rightarrow\exists_{f\in\mathbb{F}(\mathbb{N},Y)}\forall_{n\in\mathbb{N}}R(n,f(n)),$$

  where $R(m,y)$ is a relation on $\mathbb{N}\times Y$. The principle (CC) follows (exercise) from the stronger *principle of dependent choice* (DC) that is also used. According to it:

  (DC) $$\forall_{x\in X}\exists_{y\in X}(Q(x,y))\Rightarrow\exists_{f\in\mathbb{F}(\mathbb{N},X)}\big(f(0)=x_0\ \&\ \forall_{n\in\mathbb{N}}(Q(f(n),f(n{+}1)))\big),$$

  where $Q(x,x')$ is a relation on $X\times X$ and $x_0\in X$. It is useful to avoid CC and DC.

The proof-terms in $\mathrm{Eq}(X,Y)$ are compatible with the properties of the equivalence relation $X=_{\mathbb{V}_0}Y$. This means that we can define a distinguished proof-term

$$\mathtt{refl}(X)\in\mathrm{Eq}(X,X)$$

that proves the reflexivity of $X=_{\mathbb{V}_0}Y$, an operation $^{-1}$, such that

$$(f,g):X=_{\mathbb{V}_0}Y\Rightarrow(f,g)^{-1}:Y=_{\mathbb{V}_0}X,$$

and an operation of "composition" $*$ of proof-terms, such that

$$(f,g):X=_{\mathbb{V}_0}Y\ \&\ (h,k):Y=_{\mathbb{V}_0}Z\Rightarrow(f,g)*(h,k):X=_{\mathbb{V}_0}Z.$$

If $h\in\mathbb{F}(Y,W)$ and $k\in\mathbb{F}(W,Y)$, let

$$\mathtt{refl}(X):=\big(\mathrm{id}_X,\mathrm{id}_X\big)\ \ \&\ \ (f,g)^{-1}:=(g,f)\ \ \&\ \ (f,g)*(h,k):=(h\circ f,g\circ k).$$

It is immediate to see that these operations satisfy the *groupoid laws*:
(i) $\mathtt{refl}(X)*(f,g)=_{\mathrm{Eq}(X,Y)}(f,g)$ and $(f,g)*\mathtt{refl}(Y)=_{\mathrm{Eq}(X,Y)}(f,g)$.
(ii) $(f,g)*(f,g)^{-1}=_{\mathrm{Eq}(X,X)}\mathtt{refl}(X)$ and $(f,g)^{-1}*(f,g)=_{\mathrm{Eq}(Y,Y)}\mathtt{refl}(Y)$.
(iii) $\big((f,g)*(h,k)\big)*(s,t)=_{\mathrm{Eq}(X,W)}(f,g)*\big((h,k)*(s,t)\big)$.
Moreover, the following *compatibility condition* is satisfied:
(iv) If $(f,g),(f',g')\in\mathrm{Eq}(X,Y)$ and $(h,k),(h',k')\in\mathrm{Eq}(Y,Z)$, then

$$(f,g)=_{\mathrm{Eq}(X,Y)}(f',g')\ \&\ (h,k)=_{\mathrm{Eq}(Y,Z)}(h',k')\Rightarrow(f,g)*(h,k)=_{\mathrm{Eq}(X,Z)}(f',g')*(h',k').$$

Definition 1.6.1. *If X,Y are sets, their product $X\times Y$ is the defined totality with equality*

$$(x,y)\in X\times Y:\Leftrightarrow x\in A\ \&\ y\in B,$$

$$z\in X\times Y:\Leftrightarrow\exists_{x\in A}\exists_{y\in B}\big(z:=(x,y)\big).$$

$X\times Y$ is considered to be a set, and its membership condition is written simpler as follows:

$$(x,y)=_{X\times Y}(x',y'):\Leftrightarrow x=_X x'\ \&\ y=_Y y'.$$

Definition 1.6.2. *A bounded formula on a set X is called an extensional property on X, if*

$$\forall_{x,y\in X}\big([x =_X y \ \&\ P(x)] \Rightarrow P(y)\big).$$

The totality X_P generated by $P(x)$ is defined by $x \in X_P :\Leftrightarrow x \in X \ \&\ P(x)$,

$$x \in X_P :\Leftrightarrow x \in X \ \&\ P(x),$$

and the equality of X_P is inherited from the equality of X. We also write

$$X_P := \{x \in X \mid P(x)\}.$$

The totality X_P is a set, and it is called the extensional subset of X generated by P.

Using the properties of an equivalence relation, it is immediate to show that an equality condition $\mathcal{E}_X(x, y)$ on a totality X is an extensional property on the product $X \times X$ i.e., $[(x, y) =_{X\times Y} (x', y') \ \&\ x =_X y] \Rightarrow x' =_X y'$. Let the following extensional subsets of $\mathbb{N}$:

$$\mathbb{1} := \{x \in \mathbb{N} \mid x =_{\mathbb{N}} 0\} := \{0\},$$

$$\mathbb{2} := \{x \in \mathbb{N} \mid x =_{\mathbb{N}} 0 \ \vee x =_{\mathbb{N}} 1\} := \{0, 1\}.$$

Since $n =_{\mathbb{N}} m :\Leftrightarrow n := m$, the property $P(x) :\Leftrightarrow x =_{\mathbb{N}} 0 \ \vee x =_{\mathbb{N}} 1$ is extensional. In general, not all elements of $\mathrm{Eq}(X, Y)$ are equal. One can find (exercise) sets X, Y and realizers

$$(f, g) : X =_{\mathbb{V}_0} Y \ \&\ (f', g') : X =_{\mathbb{V}_0} Y \ \&\ \neg[(f, g) = (f', g')].$$

Definition 1.6.3. *If $(X, =_X)$ is a set, its diagonal is the extensional subset of $X \times X$*

$$D(X, =_X) := \{(x, y) \in X \times X \mid x =_X y\}.$$

If $=_X$ is clear from the context, we just write $D(X)$.

Definition 1.6.4. *Let X be a set. An inequality on X, or an apartness relation on X, is a relation $x \neq_X y$ such that the following conditions are satisfied:*
(Ap_1) $\forall_{x,y\in X}\big(x =_X y \ \&\ x \neq_X y \Rightarrow \bot\big)$.
(Ap_2) $\forall_{x,y\in X}\big(x \neq_X y \Rightarrow y \neq_X x\big)$.
(Ap_3) $\forall_{x,y\in X}\big(x \neq_X y \Rightarrow \forall_{z\in X}(z \neq_X x \ \vee \ z \neq_X y)\big)$.
We write $(X, =_X, \neq_X)$ to denote the equality-inequality structure of a set X, and for simplicity we refer the set $(X, =_X, \neq_X)$. The set $(X, =_X, \neq_X)$ is called discrete, if

$$\forall_{x,y\in X}\big(x =_X y \ \vee \ x \neq_X y\big).$$

An inequality $\neq_X$ on X is called tight, if $\neg(x \neq_X y) \Rightarrow x =_X y$, for every $x, y \in X$.

Remark 1.6.5. *An inequality relation $x \neq_X y$ is extensional on $X \times X$.*

Proof. Exercise. □

For example, if $(X, =_X, \neq_X)$ and $(Y, =_Y, \neq_Y)$ are given, the canonical inequality on $X \times Y$ induced by $\neq_X$ and $\neq_Y$, is defined by

$$(x, y) \neq_{X\times Y} (x', y') :\Leftrightarrow x \neq_X x' \ \vee \ y \neq_Y y',$$

for every (x, y) and $(x', y') \in X \times Y$.

Definition 1.6.6. *Let X, Y be totalities. A non-dependent assignment routine f from X to Y, in symbols $f\colon X \rightsquigarrow Y$, is a finite routine that assigns an element y of Y to each given element x of X. In this case we write $f(x) := y$. If $g\colon X \rightsquigarrow Y$, let*

$$f := g :\Leftrightarrow \forall_{x\in X}\big(f(x) := g(x)\big).$$

If $f := g$, we say that f and g are definitionally equal. If $(X, =_X)$ and $(Y, =_Y)$ are sets, an operation from X to Y is a non-dependent assignment routine from X to Y, while a function from X to Y, in symbols $f\colon X \to Y$, is an operation from X to Y that respects equality i.e.,

$$\forall_{x,x'\in X}\big(x =_X x' \Rightarrow f(x) =_Y f(x')\big).$$

If $f\colon X \rightsquigarrow Y$ is a function from X to Y, we say that f is a function, without mentioning the expression "from X to Y". A function $f\colon X \to Y$ is an embedding, in symbols $f\colon X \hookrightarrow Y$, if

$$\forall_{x,x'\in X}\big(f(x) =_Y f(x') \Rightarrow x =_X x'\big).$$

Let the sets $(X, =_X, \neq_X)$ and $(Y, =_Y, \neq_Y)$. A function $f\colon X \to Y$ is strongly extensional, if

$$\forall_{x,x'\in X}\big(f(x) \neq_Y f(x') \Rightarrow x \neq_X x'\big).$$

If X is a set, the *identity map* id_X on X is the operation $\mathrm{id}_X\colon X \rightsquigarrow X$, defined by $\mathrm{id}_X(x) := x$, for every $x \in X$. Clearly, id_X is an embedding, which is strongly extensional, if $\neq_X$ is a given inequality on X. If Y is also a set, the projection maps $\mathtt{pr}_X$ and $\mathtt{pr}_Y$ on X and Y, respectively, are the operations $\mathtt{pr}_X\colon X \times Y \rightsquigarrow X$ and $\mathtt{pr}_Y\colon X \times Y \rightsquigarrow Y$, where

$$\mathtt{pr}_X(x, y) := \mathbf{pr}_1(x, y) := x \;\;\&\;\; \mathtt{pr}_Y(x, y) := \mathbf{pr}_2(x, y) := y; \qquad (x, y) \in X \times Y.$$

Definition 1.6.7. *Let I be a set and $\lambda_0\colon I \rightsquigarrow \mathbb{V}_0$ a non-dependent assignment routine from I to $\mathbb{V}_0$. A dependent operation Φ over λ_0, in symbols*

$$\Phi\colon \bigwedge_{i\in I} \lambda_0(i),$$

is an assignment routine that assigns to each element i in I an element $\Phi(i)$ in the set $\lambda_0(i)$. If $i \in I$, we call $\Phi(i)$ the i-component of Φ, and we also use the notation $\Phi_i := \Phi(i)$. An assignment routine is either a non-dependent assignment routine, or a dependent operation over some non-dependent assignment routine from a set to the universe. Let the non-dependent assignment routines $\lambda_0\colon I \rightsquigarrow \mathbb{V}_0, \mu_0\colon I \rightsquigarrow \mathbb{V}_0, \nu_0\colon I \rightsquigarrow \mathbb{V}_0$ and $\kappa_0\colon I \rightsquigarrow \mathbb{V}_0$. Let $\mathbb{F}(\lambda_0, \mu_0)\colon I \rightsquigarrow \mathbb{V}_0$ be defined by $\mathbb{F}(\lambda_0, \mu_0)(i) := \mathbb{F}(\lambda_0(i), \mu_0(i))$, for every $i \in I$. The identity operation Id_{λ_0} over λ_0 is the dependent operation

$$\mathrm{Id}_{\lambda_0}\colon \bigwedge_{i\in I} \mathbb{F}(\lambda_0(i), \lambda_0(i)) \qquad \mathrm{Id}_{\lambda_0}(i) := \mathrm{id}_{\lambda_0(i)}; \qquad i \in I.$$

Let $\Psi\colon \bigwedge_{i\in I} \mathbb{F}(\mu_0(i), \nu_0(i))$ and $\Phi\colon \bigwedge_{i\in I} \mathbb{F}(\lambda_0(i), \mu_0(i))$. Their composition $\Psi \circ \Phi$ is defined by

$$\Psi \circ \Phi\colon \bigwedge_{i\in I} \mathbb{F}(\lambda_0(i), \nu_0(i)) \qquad (\Psi \circ \Phi)_i := \Psi_i \circ \Phi_i; \qquad i \in I.$$

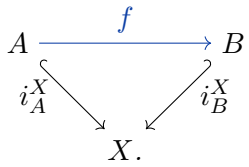
If I is a set and $\lambda_0 : I \rightsquigarrow \mathbb{V}_0$, let $\mathbb{A}(I, \lambda_0)$ be the totality of dependent operations over λ_0, equipped with the canonical equality:

$$\Phi =_{\mathbb{A}(I,\lambda_0)} \Psi :\Leftrightarrow \forall_{i\in I}\big(\Phi_i =_{\lambda_0(i)} \Psi_i\big).$$

The totality $\mathbb{A}(I, \lambda_0)$ is considered to be a set. If $\neq_{\lambda_0(i)}$ is an inequality on $\lambda_0(i)$, for every $i \in I$, the canonical inequality $\neq_{\mathbb{A}(I,\lambda_0)}$ on $\mathbb{A}(I, \lambda_0)$ is defined by $\Phi \neq_{\mathbb{A}(I,\lambda_0)} \Psi :\Leftrightarrow \exists_{i\in I}\big(\Phi_i \neq_{\lambda_0(i)} \Psi_i\big)$.

1.7 Bishop subsets

Definition 1.7.1. *Let X be a set. A subset of X is a pair (A, i_A^X), where A is a set and $\imath_A^X \colon A \hookrightarrow X$ is an embedding of A into X. If (A, i_A^X) and (B, i_B^X) are subsets of X, then A is a subset of B, in symbols $(A, i_A^X) \subseteq (B, i_B^X)$, or simpler $A \subseteq B$, if there is $f \colon A \to B$ such that the following diagram commutes*



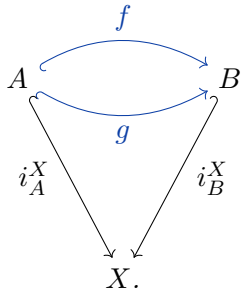
In this case we use the notation $f \colon A \subseteq B$. Usually we write A instead of (A, i_A^X). The totality of the subsets of X is the powerset $\mathcal{P}(X)$ of X, and it is equipped with the equality

$$(A, i_A^X) =_{\mathcal{P}(X)} (B, i_B^X) :\Leftrightarrow A \subseteq B \;\&\; B \subseteq A.$$

If $f \colon A \subseteq B$ and $g \colon B \subseteq A$, we write $(f, g) \colon A =_{\mathcal{P}(X)} B$.

Since the membership condition for $\mathcal{P}(X)$ requires quantification over $\mathbb{V}_0$, the totality $\mathcal{P}(X)$ is a class. Clearly, $(X, \mathrm{id}_X) \subseteq X$. If X_P is an extensional subset of X (see Definition 1.6.2), then $(X_P, i_P^X) \subseteq X$, where $i_P^X \colon X_P \rightsquigarrow X$ is defined by $i_P^X(x) := x$, for every $x \in X_P$.

Proposition 1.7.2. *If $A, B \subseteq X$, and $f, g : A \subseteq B$, then f is an embedding, and $f =_{\mathbb{F}(A,B)} g$*



Proof. Exercise. □

The "internal" equality of subsets implies (exercise) their "external" equality as sets i.e., $(f, g) : A =_{\mathcal{P}(X)} B \Rightarrow (f, g) : A =_{\mathbb{V}_0} B$. Let the set

$$\mathrm{Eq}(A, B) := \big\{(f, g) \in \mathbb{F}(A, B) \times \mathbb{F}(B, A) \mid f \colon A \subseteq B \;\&\; g \colon B \subseteq A\big\},$$

equipped with the canonical equality of pairs as in the case of $\mathrm{Eq}(X, Y)$. Because of Proposition 1.7.2, the set $\mathrm{Eq}(A, B)$ is a *subsingleton* i.e.,

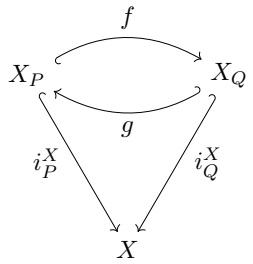
$$(f, g) \colon A =_{\mathcal{P}(X)} B \;\&\; (f', g') \colon A =_{\mathcal{P}(X)} B \Rightarrow (f, g) = (f', g').$$

If $f \in \mathbb{F}(A, B), g \in \mathbb{F}(B, A), h \in \mathbb{F}(B, C)$, and $k \in \mathbb{F}(C, B)$, let $\mathtt{refl}(A) := \big(\mathrm{id}_A, \mathrm{id}_A\big)$ and $(f, g)^{-1} := (g, f)$, and $(f, g) * (h, k) := (h \circ f, g \circ k)$.

Remark 1.7.3. *If P, Q are extensional properties on the set X, then*

$$X_P =_{\mathcal{P}(X)} X_Q \Leftrightarrow \forall_{x\in X}\big(P(x) \Leftrightarrow Q(x)\big).$$

Proof. The implication $(\Leftarrow)$ is immediate to show, since the corresponding identity maps witness the equality $X_P =_{\mathcal{P}(X)} X_Q$. For the converse implication, let $(f, g) : X_P =_{\mathcal{P}(X)} X_Q$. Let $x \in X$ such that $P(x)$. By the commutativity of the following outer diagram



we get $f(x) := i_Q^X(f(x)) =_X i_P^X(x) := x$, and by the extensionality of Q and the fact that $Q(f(x))$ holds we get $Q(x)$. By the commutativity of the above inner diagram and the extensionality of P we get similarly the inverse implication. □

Definition 1.7.4. *If $(A, i_A^X), (B, i_B^X) \subseteq X$, their union $A \cup B$ is the totality defined by*

$$z \in A \cup B :\Leftrightarrow z \in A \ \vee \ z \in B,$$

equipped with the non-dependent assignment routine[9] *$i_{A\cup B}^X : A \cup B \rightsquigarrow X$, defined by*

$$i_{A\cup B}^X(z) := \begin{cases} i_A^X(z) & , z \in A \\ \\ i_B^X(z) & , z \in B. \end{cases}$$

If $z, w \in A \cup B$, we define $z =_{A\cup B} w :\Leftrightarrow i_{A\cup B}^X(z) =_X i_{A\cup B}^X(w)$.

One can show (exercise) that $=_{A\cup B}$ is an equality on $A \cup B$, $i_{A\cup B}^X$ is an embedding of $A \cup B$ into X, and hence the pair $\big(A \cup B, i_{A\cup B}^X\big)$ is a subset of X.

Definition 1.7.5. *If $(A, i_A^X), (B, i_B^X) \subseteq X$, their intersection $A \cap B$ is the totality defined by separation on $A \times B$ as follows:*

$$A \cap B := \{(a, b) \in A \times B \mid i_A^X(a) =_X i_B^X(b)\}.$$

Let the non-dependent assignment routine $i_{A\cap B}^X : A \cap B \rightsquigarrow X$, defined by $i_{A\cap B}^X(a, b) := i_A^X(a)$, for every $(a, b) \in A \cap B$. If (a, b) and (a', b') are in $A \cap B$, let

$$(a, b) =_{A\cap B} (a', b') :\Leftrightarrow i_{A\cap B}^X(a, b) =_X i_{A\cap B}^X(a', b') :\Leftrightarrow i_A^X(a) =_X i_A^X(a').$$

We write $A \between B$ to denote that the intersection $A \cap B$ is inhabited.

One can show (exercise) that $=_{A\cap B}$ is an equality on $A \cap B$, $i_{A\cap B}^X$ is an embedding of $A \cap B$ into X, and hence the pair $\big(A \cap B, i_{A\cap B}^X\big)$ is a subset of X.

[9] Here we define a non-dependent assignment routine on the totality $A \cup B$, without knowing beforehand that $A \cup B$ is a set. It turns out that $A \cup B$ is set, but for that we need to define $i_{A\cup B}^X$ first.

Definition 1.7.6. *Let X, Y be sets, $(A, i_A^X)(C, i_C^X) \subseteq X$, $e\colon (A, i_A^X) \subseteq (C, i_C^X)$, $f\colon C \to Y$, and $(B, i_B^Y) \subseteq Y$. The restriction $f_{|_A}$ of f to A is the function $f_A := f \circ e$*

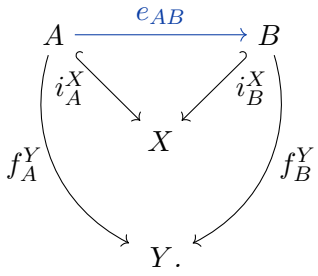
$$A \xhookrightarrow{e} C \xrightarrow{f} Y. \qquad f_{|_A}$$

The image $f(A)$ of A under f is the pair $f(A) := (A, f_A)$, where A is equipped with the equality $a =_{f(A)} a' :\Leftrightarrow f_{|_A}(a) =_Y f_{|_A}(a')$, for every $a, a' \in A$. We denote $\{f(a) \mid a \in A\} := f(A)$. The pre-image $f^{-1}(B)$ of B under f is the set

$$f^{-1}(B) := \{(c, b) \in C \times B \mid f(c) =_Y i_B^Y(b)\}.$$

Let $i^C_{f^{-1}(B)}\colon f^{-1}(B) \hookrightarrow C$, defined by $i^C_{f^{-1}(B)}(c, b) := c$, for every $(c, b) \in f^{-1}(B)$. The equality of the extensional subset $f^{-1}(B)$ of $C \times B$ is inherited from the equality of $C \times B$.

Definition 1.7.7. *Let X, Y be sets. A partial function from X to Y is a triplet (A, i_A^X, f_A^Y), where $(A, i_A^X) \subseteq X$, and $f_A^Y \in \mathbb{F}(A, Y)$. Often, we use only the symbol f_A^Y instead of the triplet (A, i_A^X, f_A^Y), and we also write $f_A^Y\colon X \rightharpoonup Y$. If (A, i_A^X, f_A^Y) and (B, i_B^X, f_B^Y) are partial functions from X to Y, we call f_A^Y a subfunction of f_B^Y, in symbols $(A, i_A^X, f_A^Y) \leq (B, i_B^X, f_B^Y)$, or simpler $f_A^Y \leq f_B^Y$, if there is $e_{AB}\colon A \to B$ such that the following inner diagrams commute*



In this case we use the notation $e_{AB}\colon f_A^Y \leq f_B^Y$. The totality of partial functions from X to Y is the partial function space $\mathfrak{F}(X, Y)$, and it is equipped with the equality

$$(A, i_A^X, f_A^Y) =_{\mathfrak{F}(X,Y)} (B, i_B^X, f_B^Y) :\Leftrightarrow f_A^Y \leq f_B^Y \;\&\; f_B^Y \leq f_A^Y.$$

If $e_{AB}\colon f_A^Y \leq f_B^Y$ and $e_{BA}\colon f_B^Y \leq f_A^Y$, we write $(e_{AB}, e_{BA}) : f_A^Y =_{\mathfrak{F}(X,Y)} f_B^Y$.

Since the membership condition for $\mathfrak{F}(X, Y)$ requires quantification over $\mathbb{V}_0$, the totality $\mathfrak{F}(X, Y)$ is a class. Clearly, if $f\colon X \to Y$, then $(X, \mathrm{id}_X, f) \in \mathfrak{F}(X, Y)$. If $(e_{AB}, e_{BA}) : f_A^Y =_{\mathfrak{F}(X,Y)} f_B^Y$, then $(e_{AB}, e_{BA}) : A =_{\mathcal{P}(X)} B$, and $(e_{AB}, e_{BA}) : A =_{\mathbb{V}_0} B$. Let the set

$$\mathrm{Eq}(f_A^Y, f_B^Y) := \big\{(f, g) \in \mathbb{F}(A, B) \times \mathbb{F}(B, A) \mid f\colon f_A^Y \leq f_B^Y \;\&\; g\colon f_B^Y \leq f_A^Y\big\},$$

equipped with the canonical equality of the product. All the elements of $\mathrm{Eq}(f_A^Y, f_B^Y)$ are equal to each other. If $f \in \mathbb{F}(A, B), g \in \mathbb{F}(B, A), h \in \mathbb{F}(B, C)$, and $k \in \mathbb{F}(C, B)$, let

$$\mathtt{refl}(f_A^Y) := \big(\mathrm{id}_A, \mathrm{id}_A\big) \;\;\&\;\; (f, g)^{-1} := (g, f) \;\;\&\;\; (f, g) * (h, k) := (h \circ f, g \circ k).$$

If $(A, i_A^X, f_A^Y) \in \mathfrak{F}(X, Y)$ and $(B, i_B^Y, g_B^Z) \in \mathfrak{F}(Y, Z)$, their composition can be defined (exercise), which is a partial function in $\mathfrak{F}(X, Z)$.

1.8 Families of Bishop sets

Roughly speaking, a family of sets indexed by some set I is an assignment routine $\lambda_0 : I \rightsquigarrow \mathbb{V}_0$ that behaves like a function i.e., if $i =_I j$, then $\lambda_0(i) =_{\mathbb{V}_0} \lambda_0(j)$. Next follows an exact formulation of this description that reveals the witnesses of the equality $\lambda_0(i) =_{\mathbb{V}_0} \lambda_0(j)$.

Definition 1.8.1. *If I is a set, a family of sets indexed by I, or an I-family of sets, is a pair $\Lambda := (\lambda_0, \lambda_1)$, where $\lambda_0 \colon I \rightsquigarrow \mathbb{V}_0$, and λ_1, a modulus of function-likeness for λ_0, is given by*

$$\lambda_1 \colon \bigdownarrow_{(i,j)\in D(I)} \mathbb{F}\big(\lambda_0(i), \lambda_0(j)\big), \quad \lambda_1(i,j) := \lambda_{ij}, \quad (i,j) \in D(I),$$

such that the transport maps λ_{ij} of Λ satisfy the following conditions:

(a) *For every $i \in I$, we have that $\lambda_{ii} := \mathrm{id}_{\lambda_0(i)}$.*

(b) *If $i =_I j$ and $j =_I k$, the following diagram commutes*

$$\begin{array}{ccc} \lambda_0(i) & & \\ {\scriptstyle \lambda_{ij}}\downarrow & \searrow^{\lambda_{ik}} & \\ \lambda_0(j) & \xrightarrow[\lambda_{jk}]{} & \lambda_0(k). \end{array}$$

I is the index-set of the family Λ. If X is a set, the constant I-family of sets X is the pair $C^X := (\lambda_0^X, \lambda_1^X)$, where $\lambda_0(i) := X$, for every $i \in I$, and $\lambda_1(i,j) := \mathrm{id}_X$, for every $(i,j) \in D(I)$ (see the left diagram in Definition 1.8.2).

The dependent operation λ_1 should have been written as follows

$$\lambda_1 \colon \bigdownarrow_{z\in D(I)} \mathbb{F}\big(\lambda_0(\mathbf{pr}_1(z)), \lambda_0(\mathbf{pr}_2(z))\big),$$

but, for simplicity, we avoid the use of the primitive projections $\mathbf{pr}_1, \mathbf{pr}_2$. Condition (a) of Definition 1.8.1 could have been written as $\lambda_{ii} =_{\mathbb{F}(\lambda_0(i),\lambda_0(i))} \mathrm{id}_{\lambda_0(i)}$. If $i =_I j$, then by conditions (b) and (a) of Definition 1.8.1 we get $id_{\lambda_0(i)} := \lambda_{ii} = \lambda_{ji} \circ \lambda_{ij}$ and $\mathrm{id}_{\lambda_0(j)} := \lambda_{jj} = \lambda_{ij} \circ \lambda_{ji}$ i.e., $(\lambda_{ij}, \lambda_{ji}) \colon \lambda_0(i) =_{\mathbb{V}_0} \lambda_0(j)$. In this sense λ_1 is a modulus of function-likeness for λ_0.

Definition 1.8.2. *The pair $\Lambda^{\mathbb{2}} := (\lambda_0^{\mathbb{2}}, \lambda_1^{\mathbb{2}})$, where $\lambda_0^{\mathbb{2}} \colon \mathbb{2} \rightsquigarrow \mathbb{V}_0$ with $\lambda_0^{\mathbb{2}}(0) := X$, $\lambda_0^{\mathbb{2}}(1) := Y$, and $\lambda_1^{\mathbb{2}}(0,0) := \mathrm{id}_X$ and $\lambda_1^{\mathbb{2}}(1,1) := \mathrm{id}_Y$, is the $\mathbb{2}$-family of X and Y*

$$\begin{array}{ccc} X & & \\ {\scriptstyle \mathrm{id}_X}\downarrow & \searrow^{\mathrm{id}_X} & \\ X & \xrightarrow[\mathrm{id}_X]{} & X \end{array} \qquad \begin{array}{ccc} Y & & \\ {\scriptstyle \mathrm{id}_Y}\downarrow & \searrow^{\mathrm{id}_Y} & \\ Y & \xrightarrow[\mathrm{id}_Y]{} & Y. \end{array}$$

The $\mathbb{n}$-family $\Lambda^{\mathbb{n}}$ of the sets $X_1, \ldots X_n$, where $n \geq 1$, and the $\mathbb{N}$-family $\Lambda^{\mathbb{N}} := (\lambda_0^{\mathbb{N}}, \lambda_1^{\mathbb{N}})$ of the sets $(X_n)_{n\in\mathbb{N}}$ are defined similarly[10].

[10] It is immediate to show that $\Lambda^{\mathbb{n}}$ is an $\mathbb{n}$-family, and $\Lambda^{\mathbb{N}}$ is an $\mathbb{N}$-family.

Definition 1.8.3. *Let $\Lambda := (\lambda_0, \lambda_1)$ be an I-family of sets. The exterior union, or disjoint union, or the $\sum$-set $\sum_{i\in I} \lambda_0(i)$ of Λ, and its canonical equality are defined by*

$$w \in \sum_{i\in I} \lambda_0(i) :\Leftrightarrow \exists_{i\in I}\exists_{x\in\lambda_0(i)}\big(w := (i,x)\big),$$

$$(i,x) =_{\sum_{i\in I}\lambda_0(i)} (j,y) :\Leftrightarrow i =_I j \;\&\; \lambda_{ij}(x) =_{\lambda_0(j)} y.$$

The $\sum$-set of the 2-family $\Lambda^{\mathbb{2}}$ of the sets X and Y is the coproduct of X and Y, and we write

$$X + Y := \sum_{i\in\mathbb{2}} \lambda_0^{\mathbb{2}}(i).$$

Proposition 1.8.4. *The equality on $\sum_{i\in I}\lambda_0(i)$ satisfies the conditions of an equivalence relation.*

Proof. Exercise. □

Definition 1.8.5. *Let $\Lambda := (\lambda_0, \lambda_1), M := (\mu_0, \mu_1)$ be I-families of sets. The coproduct family of Λ and M is the pair $\Lambda + M := (\lambda_0 + \mu_0, \lambda_1 + \mu_1)$, where $(\lambda_0 + \mu_0)(i) := \lambda_0(i) + \mu_0(i)$, for every $i \in I$, and the map $\big(\lambda_1 + \mu_1\big)_{ij}\colon \lambda_0(i) + \mu_0(i) \to \lambda_0(j) + \mu_0(j)$ is defined by*

$$\big(\lambda_1 + \mu_1\big)_{ij}(w) := \begin{cases} \big(0, \lambda_{ij}(x)\big) & , w := (0,x) \\ \big(1, \mu_{ij}(y)\big) & , w := (1,y) \end{cases} ; \qquad w \in \lambda_0(i) + \mu_0(i).$$

It is straightforward to show that $\Lambda + M$ is an I-family of sets.

Proposition 1.8.6. *If $\Lambda := (\lambda_0, \lambda_1)$, $M := (\mu_0, \mu_1) \in \mathtt{Fam}(I)$, then*

$$\sum_{i\in I}\big(\lambda_0(i) + \mu_0(i)\big) =_{\mathbb{V}_0} \Big(\sum_{i\in I}\lambda_0(i)\Big) + \Big(\sum_{i\in I}\mu_0(i)\Big).$$

Proof. Exercise. □

Proposition 1.8.7. *Let $\Lambda := (\lambda_0, \lambda_1)$, $M := (\mu_0, \mu_1)$ be I-families of sets, and*

$$\Psi\colon \bigwedge_{i\in I} \mathbb{F}\big(\lambda_0(i), \mu_0(i)\big)$$

such that for every $(i,j) \in D(I)$ the following diagram commutes

$$\begin{array}{ccc} \lambda_0(i) & \xrightarrow{\lambda_{ij}} & \lambda_0(j) \\ {\scriptstyle\Psi_i}\downarrow & & \downarrow{\scriptstyle\Psi_j} \\ \mu_0(i) & \xrightarrow[\mu_{ij}]{} & \mu_0(j). \end{array}$$

(i) *For every $i \in I$ the operation $e_i^\Lambda : \lambda_0(i) \rightsquigarrow \sum_{i\in I} \lambda_0(i)$, defined by $e_i^\Lambda(x) := (i, x)$, for every $x \in \lambda_0(i)$, is an embedding.*
(ii) *The operation $\Sigma\Psi : \sum_{i\in I} \lambda_0(i) \rightsquigarrow \sum_{i\in I} \mu_0(i)$, defined by*

$$\Sigma\Psi(i, x) := (i, \Psi_i(x)); \qquad (i, x) \in \sum_{i\in I} \lambda_0(i),$$

is a function, such that for every $i \in I$ the following diagram commutes

$$\begin{array}{ccc} \lambda_0(i) & \xrightarrow{\Psi_i} & \mu_0(i) \\ {\scriptstyle e_i^\Lambda}\downarrow & & \downarrow{\scriptstyle e_i^M} \\ \sum_{i\in I} \lambda_0(i) & \xrightarrow[\Sigma\Psi]{} & \sum_{i\in I} \mu_0(i). \end{array}$$

(iii) *If Ψ_i is an embedding, for every $i \in I$, then $\Sigma\Psi$ is an embedding.*

Proof. Exercise. □

Definition 1.8.8. *Let $\Lambda := (\lambda_0, \lambda_1)$ be an I-family of sets. The first projection on $\sum_{i\in I} \lambda_0(i)$ is the operation $\mathtt{pr}_1^\Lambda \colon \sum_{i\in I} \lambda_0(i) \rightsquigarrow I$, defined by $pr_1^\Lambda(i, x) := \mathbf{pr}_1(i, x) := i$, for every $(i, x) \in \sum_{i\in I} \lambda_0(i)$. We may only write $\mathtt{pr}_1$, if Λ is clearly understood from the context.*

By the definition of the canonical equality on $\sum_{i\in I} \lambda_0(i)$ we get that $\mathtt{pr}_1^\Lambda$ is a function.

Definition 1.8.9. *Let $\Lambda := (\lambda_0, \lambda_1)$ be an I-family of sets. The second projection on $\sum_{i\in I} \lambda_0(i)$ is the dependent operation $\mathtt{pr}_2^\Lambda \colon \bigwedge_{(i,x)\in\sum_{i\in I} \lambda_0(i)} \lambda_0(i)$, defined by $\mathtt{pr}_2^\Lambda(i, x) := \mathbf{pr}_2(i, x) := x$, for every $(i, x) \in \sum_{i\in I} \lambda_0(i)$. We may only write $\mathtt{pr}_2$, when the family of sets Λ is clearly understood from the context.*

Definition 1.8.10. *Let $\Lambda := (\lambda_0, \lambda_1)$ be an I-family of sets. The totality $\prod_{i\in I} \lambda_0(i)$ of dependent functions over Λ, or the $\prod$-set of Λ, is defined by*

$$\Theta \in \prod_{i\in I} \lambda_0(i) :\Leftrightarrow \Theta \in \mathbb{A}(I, \lambda_0) \;\&\; \forall_{(i,j)\in D(I)}\big(\Theta_j =_{\lambda_0(j)} \lambda_{ij}(\Theta_i)\big),$$

and it is equipped with the canonical equality and the canonical inequality of the set $\mathbb{A}(I, \lambda_0)$. If X is a set and Λ^X is the constant I-family X (see Definition 1.8.1), we use the notation

$$X^I := \prod_{i\in I} X.$$

Clearly, the property $P(\Phi) :\Leftrightarrow \forall_{(i,j)\in D(I)}\big(\Theta_j =_{\lambda_0(j)} \lambda_{ij}(\Theta_i)\big)$ is extensional on $\mathbb{A}(I, \lambda_0)$, the equality on $\prod_{i\in I} \lambda_0(i)$ is an equivalence relation. $\prod_{i\in I} \lambda_0(i)$ is considered to be a set.

Definition 1.8.11. *Let $\Lambda := (\lambda_0, \lambda_1)$ be an I-family of sets. The $\sum$-indexing of Λ is the pair $\Sigma^\Lambda := (\sigma_0^\Lambda, \sigma_1^\Lambda)$, where $\sigma_0^\Lambda \colon \sum_{i\in I} \lambda_0(i) \rightsquigarrow \mathbb{V}_0$ is defined by $\sigma_0^\Lambda(i, x) := \lambda_0(i)$, for every $(i, x) \in \sum_{i\in I} \lambda_0(i)$, and $\sigma_1^\Lambda\big((i, x), (j, y)\big) := \lambda_{ij}$, for every $\big((i, x), (j, y)\big) \in D\big(\sum_{i\in I} \lambda_0(i)\big)$.*

Remark 1.8.12. *If $\Lambda := (\lambda_0, \lambda_1)$ is an I-family of sets and $\Sigma^\Lambda := (\sigma_0^\Lambda, \sigma_1^\Lambda)$ is the $\sum$-indexing of Λ, then $\mathtt{pr}_2^\Lambda$ is a dependent function over Σ^Λ.*

Proof. Exercise. □

Remark 1.8.13. (i) *If $\Lambda^{\mathbb{2}}$ is the $\mathbb{2}$-family of the sets X and Y, then $\prod_{i\in\mathbb{2}}\lambda_0^{\mathbb{2}}(i) =_{\mathbb{V}_0} X\times Y$.*
(ii) *If I, A are sets, and $\Lambda := (\lambda_0^A, \lambda_1)$ is the constant I-family A, then $A^I =_{\mathbb{V}_0} \mathbb{F}(I,A)$.*

Proof. Exercise. □

In general, we may want to have *more than one transport maps* from $\lambda_0(i)$ to $\lambda_0(j)$, if $i =_I j$. In this case, to each $(i,j)\in D(I)$ we associate a set of transport maps.

Definition 1.8.14. *If I is a set, a set-relevant family of sets indexed by I, is a triplet $\Lambda^* := \big(\lambda_0, \varepsilon_0^\lambda, \lambda_2\big)$, where $\lambda_0 : I \rightsquigarrow \mathbb{V}_0$, $\varepsilon_0^\lambda : D(I) \rightsquigarrow \mathbb{V}_0$, and*

$$\lambda_2 : \bigsmallint_{(i,j)\in D(I)} \bigsmallint_{p\in\varepsilon_0^\lambda(i,j)} \mathbb{F}\big(\lambda_0(i),\lambda_0(j)\big), \quad \lambda_2\big((i,j),p\big) := \lambda_{ij}^p, \quad (i,j)\in D(I), \quad p\in\varepsilon_0^\lambda(i,j),$$

such that the following conditions hold:
(i) *For every $i\in I$ there is $p\in\varepsilon_0^\lambda(i,i)$ such that $\lambda_{ii}^p =_{\mathbb{F}(\lambda_0(i),\lambda_0(i))} \mathrm{id}_{\lambda_0(i)}$.*
(ii) *For every $(i,j)\in D(I)$ and every $p\in\varepsilon_0^\lambda(i,j)$ there is some $q\in\varepsilon_0^\lambda(j,i)$ such that such that the following left diagram commutes*

$$\begin{array}{ccc} \lambda_0(i) & & \\ {\scriptstyle\lambda_{ij}^p}\downarrow & \searrow^{\mathrm{id}_{\lambda_0(i)}} & \\ \lambda_0(j) & \xrightarrow[\lambda_{ji}^q]{} & \lambda_0(i) \end{array} \qquad \begin{array}{ccc} \lambda_0(i) & & \\ {\scriptstyle\lambda_{ij}^p}\downarrow & \searrow^{\lambda_{ik}^r} & \\ \lambda_0(j) & \xrightarrow[\lambda_{jk}^q]{} & \lambda_0(k). \end{array}$$

(iii) *If $i =_I j =_I k$, then for every $p\in\varepsilon_0^\lambda(i,j)$ and every $q\in\varepsilon_0^\lambda(j,k)$ there is $r\in\varepsilon_0^\lambda(i,k)$ such that the above right diagram commutes.*
We call Λ^* function-like, if $\forall_{(i,j)\in D(I)}\forall_{p,p'\in\varepsilon_0^\lambda(i,j)}\big(p =_{\varepsilon_0^\lambda(i,j)} p' \Rightarrow \lambda_{ij}^p =_{\mathbb{F}(\lambda_0(i),\lambda_0(j))} \lambda_{ij}^{p'}\big)$.

It is immediate to show that if $\Lambda := (\lambda_0,\lambda_1)\in\mathtt{Fam}(I)$, then Λ generates a set-relevant family over I, where $\varepsilon_0^\lambda(i,j) := \mathbb{1}$, and $\lambda_2\big((i,j),p)\big) := \lambda_{ij}$, for every $(i,j)\in D(I)$.

Definition 1.8.15. *Let $\Lambda^* := \big(\lambda_0,\varepsilon_0^\lambda,\lambda_2\big)\in\mathtt{Fam}^*(I)$. The exterior union $\sum_{i\in I}^*\lambda_0(i)$ of Λ^* is the totality $\sum_{i\in I}\lambda_0(i)$, equipped with the following equality*

$$(i,x) =_{\sum_{i\in I}^*\lambda_0(i)} (j,y) :\Leftrightarrow i =_I j \;\&\; \exists_{p\in\varepsilon_0^\lambda(i,j)}\big(\lambda_{ij}^p(x) =_{\lambda_0(j)} y\big).$$

The totality $\prod_{i\in I}^*\lambda_0(i)$ of dependent functions over Λ^* is defined by

$$\Theta\in\prod_{i\in I}^*\lambda_0(i) :\Leftrightarrow \Theta\in\mathbb{A}(I,\lambda_0)\;\&\;\forall_{(i,j)\in D(I)}\forall_{p\in\varepsilon_0^\lambda(i,j)}\big(\Theta_j =_{\lambda_0(j)} \lambda_{ij}^p(\Theta_i)\big),$$

and it is equipped with the pointwise equality.

One can show (exercise) that the equalities on $\sum_{i\in I}^*\lambda_0(i)$ and $\prod_{i\in I}^*\lambda_0(i)$ satisfy the conditions of an equivalence relation.

Chapter 2

Basic types of Martin-Löf Type Theory

In this chapter we present intensional MLTT in a naive, or informal manner following [**?**]. Initially, MLTT was designed as a formal system for BISH. It turned out to be a fundamental functional programming language. Some basic features of MLTT are the following:

- The "logic" of MLTT is intuitionistic. This is why MLTT is also called *intuitionisitic type theory*.
- Moreover, logic is built in (Curry-Howard identification, instead of Curry-Howard correspondence).
- The BHK-interpretation is "satisfied".
- Every term (closed or not) is typed i.e., we always have $a\colon A$, a is a term of type A.
- There is a primitive equality between terms, denoted by $a \equiv b\colon A$, or simpler $a \equiv b$, and it is understood as a *definitional* equality between the terms a and b of type A.
- There is a hierarchy of open-ended universes universes (Martin-Löf's option), which are types themselves..
- The main objects are the types, which are defined inductively. The universes are also types, but they are not defined inductively in this open-ended approach towards them.
- It consists of rules. No axioms are used.
- Proof-terms are first-class citizens.
- Functions (non-dependent and dependent) are basic objects.
- Each type A has its own equality $=_A$, which is an inductively defined type. The the behavior of this equality type though, is treated in a uniform way for all types.
- The distinction between *intensional* and *extensional* MLTT lies on the treatment of equality. Namely, in extensional MLTT there is no distinction between $a \equiv b$ and $a =_A b$ for objects a, b of type A. In intensional MLTT both equalities are used.
- The type-theoretic axiom of choice is provable. This is a result of the proof-relevant character of MLTT.

2.1 Propositions and Judgments

A *proposition* A is a mathematical expression that can be proved, and it is going to be a type. A *judgment* is a mathematical expression that can be used in the ambient space of a proposition. We can use a judgment as a hypothesis in a proof (*hypothetical judgment*), we can derive a judgment (*derivation judgment*), but we cannot treat a judgment as a proposition e.g., we cannot negate a judgment.

Definition 2.1.1. *The basic judgments of* MLTT *are the following:*

(i) $a\colon A$ *i.e., a is an object (or a term) of type A. The judgment $A\colon \mathcal{U}$ is special case, and it is understood intuitively as "the type A is in the universe $\mathcal{U}$".*

(ii) $a \equiv b\colon A$, *or* $a \equiv_A b$, *or simpler* $a \equiv b$ *i.e., a, b are definitionally equal objects (or terms) of type A. The judgment $A \equiv B\colon \mathcal{U}$ is special case, and it is understood as "the types A, B are by definition equal in the universe $\mathcal{U}$".*

Formula/Proposition/Type	Proof
A	$a\colon A$

Table 2.1: A closed term of type A is a "proof" of A

Notice that in ZF the mathematical expressions $a \in A$, or $a = b$, are formulas, while in MLTT the corresponding mathematical expressions $a\colon A$, or $a \equiv b$, are judgments. For simplicity we write $a, b\colon A$, instead of $a\colon A, b\colon A$.

2.2 Universes of types

As we cannot negate judgments in MLTT, the mathematical expression $\neg(A\colon A)$ has no meaning, and hence the Russell paradox cannot be even formulated in MLTT. If we consider one only universe $\mathcal{U}$ though, the set-theoretic Burali-Forti paradox[1] was formulated in a type-teoretic way by Girard. Based on an informal notion of natural number, a hierarchy of universes of types is used, and the paradox of Girard in a version of MLTT with one universe is avoided. This hierarchy of universes in MLTT is motivated by Grothendieck's hierarchy of set-universes. Grothendieck introduced the notion of a Grothendieck universe, in order to avoid the use of proper classes in algebraic geometry.

Definition 2.2.1. *A Grothendieck universe (of small sets) is a set $\mathcal{U}$ that satisfies the following properties:*

(GU1) *If $x \in a \in \mathcal{U}$, then $x \in \mathcal{U}$.*

(GU2) *If $a, b \in \mathcal{U}$, then $\{a, b\}$ and $a \times b$ are in $\mathcal{U}$.*

(GU3) *If $a \in \mathcal{U}$, then $\bigcup a$ and $\mathcal{P}(a)$ are in $\mathcal{U}$.*

[1] According to it, the *set* of all ordinals On leads to the contradiction On $<$ On and $\neg$(On $<$ On). Hence On has to be a proper class.

(GU4) *If* $f\colon a \to b$ *is a surjection,* $a \in \mathcal{U}$ *and* $b \subseteq \mathcal{U}$*, then* $b \in \mathcal{U}$*.*
(GU5) $\mathbb{N} \in \mathcal{U}$*.*

One can show that $\left(\mathcal{U}, \in_{|\mathcal{U}\times\mathcal{U}}\right)$ satisfies the axioms of ZFC i.e., it is a small (inner) *model* of ZFC. By Gödel's second incompleteness theorem, ZFC cannot prove the existence of a Grothendieck universe (notice that the empty set satisfies conditions (GU1)-(GU4), but not (GU5). To ZFC one can add the Grothendieck Universe-axiom :

(GUA) For every set a there is a Grothendieck universe $\mathcal{U}$ such that $a \in \mathcal{U}$.

As $\mathcal{U}$ is a set itself, by this axiom there is another Grothendieck universe $\mathcal{U}'$ such that

$$\mathcal{U} \in \mathcal{U}'.$$

By (GU1) we have that

$$a \in \mathcal{U} \in \mathcal{U}' \Rightarrow a \in \mathcal{U}' \quad \text{i.e., } \mathcal{U} \subseteq \mathcal{U}'.$$

This hierearchy of Grothendieck universes is incorporated into MLTT.

Definition 2.2.2. *There is a hierarchy of universes of (small) types*

$$\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \dots,$$

such that for every $n \in \mathbb{N}$ *we have that*

$$\frac{A : \mathcal{U}_n}{A : \mathcal{U}_{n+1}}\ .$$

Usually we work with a single universe $\mathcal{U}$ *the index of which is omitted i.e., we write* $A\colon \mathcal{U}$*, instead of* $A\colon \mathcal{U}_n$*. In this case, the universe* $\mathcal{U}'$ *denotes the immediate next universe* $\mathcal{U}_{n+1}$ *in the given hierarchy of universes. The conversion rule for types in a universe is the following:*

$$\frac{A : \mathcal{U}, B\colon \mathcal{U}, A \equiv B\colon \mathcal{U}, a\colon A}{a : B}\ .$$

We use capital letters for variables of a universe-type i.e., we write $X_n, Y_n, Z_n\colon \mathcal{U}_n$*, or simpler* $X, Y, Z\colon \mathcal{U}$ *and* $X', Y', Z'\colon \mathcal{U}'$*.*

In MLTT the axiom (GUA) is incorporated, rather than satisfied, in the sense that every (small) type A is understood, or given, always as a term of some type $\mathcal{U}_n$, where $n \in \mathbb{N}$. As $A\colon \mathcal{U}_{n+1}$ and $A\colon \mathcal{U}_{n+2}$ and so on, there is no unique typing for types A. No $\mathcal{U}_n : \mathcal{U}_n$ is ever used. As already said, we do not accommodate universes with some induction principle, as we want the the notion of type to be open-ended i.e., to add new types, if necessary. We can give examples of types $A, B\colon \mathcal{U}$ such that $A \equiv B\colon \mathcal{U}$ after introducing type families over a type (see next section). Universes are essential in order to define type families over a type and to prove inequalities such as $0_{\mathbf{2}} \neq_{\mathbf{2}} 1_{\mathbf{2}}$, $0_{\mathbf{N}} \neq_{\mathbf{N}} 1_{\mathbf{N}}$, where $\mathbf{2}$ is the type of booleans and $\mathbf{N}$ is the type of natural numbers within MLTT.

2.3 Function type

An inductive definition of a type in MLTT is determined by three main rules: its formation rule, its introduction rule, and the elimination or induction rule. The computation and uniqueness rules are optional.

Definition 2.3.1. $\mathrm{Form}_{\to}$:

$$\frac{A, B : \mathcal{U}}{A \to B \in \mathcal{U}} .$$

I.e., given $A, B : \mathcal{U}$ the function type $A \to B$ of (non-dependent) functions with domain A and codomain B is in $\mathcal{U}$. We write $f : A \to B$ to denote that f is a term of type $A \to B$ and $\phi, \psi, \theta : A \to B$ are variables of type $A \to B$.

$\mathrm{Intro}_{\to}$:

$$\frac{[x : A, b(x) : B]}{(\lambda_{x:A}.b(x) : B) : A \to B} .$$

I.e., if x is a variable of type A and $b(x)$ is a term of type B, where the free variables of b are included in $\{x\}$, then the λ-term $\lambda_{x:A}.b(x) : B$, or simpler $\lambda_{x:A}.b(x)$ is a term of type $A \to B$. This rule is also called λ-abstraction and it introduces the "canonical" elements of the type $A \to B$.

$\mathrm{Elim}_{\to}$, *or* $\mathrm{Ind}_{\to}$:

$$\frac{f : A \to B,\ a : A}{f(a) : B} .$$

Hence the application of f on the term $a : A$ is the term $f(a) \equiv f[x/a]$ resulting from the substitution of $x : A$ by $a : A$ in f.

$\mathrm{Comp}_{\to}$:

$$\frac{[\lambda_{x:A}.b(x) : B] : A \to B,\ a : A}{[\lambda_{x:A}.b(x) : B](a) \equiv b(a) : B} ,$$

where $b(a) \equiv b[a/x]$ is the term resulting after the substitution of $x :$ by $a : A$ in b. This computation rule, which explains the application of the elimination rule to the introduction rule, is also known as β-conversion.

$\mathrm{Uniq}_{\to}$:

$$\frac{f : A \to B}{f \equiv [\lambda_{x:A}.f(x) : B] : A \to B} .$$

This optional uniqueness principle, which expresses that every element of the type $A \to B$ is uniquely determined by is elimination and abstraction, is called η-conversion.

Example 2.3.2 (The constant function)**.** By the introduction rule for $\to$, if

$$\frac{x : A, b : B}{[\lambda_{x:A}.b] : A \to B} ,$$

Formula / Type	Proof-term
$A \Rightarrow B \;\; / \;\; A \to B$	$f\colon A \to B$

Table 2.2: A function $f\colon A \to B$ is a "proof" of the type-theoretic implication

we set $\mathrm{Const}(A, B, b) \equiv \lambda_{x\colon A}.b$ for the constant function on $A\colon \mathcal{U}$ with value $b\colon B$, where $B\colon \mathcal{U}$ too. By the computation rule for $\to$, if $a\colon A$, we get the expected definitional equality

$$[\lambda_{x\colon A}.b](a) \equiv b.$$

Example 2.3.3 (The identity function)**.** By the introduction rule for $\to$, if

$$\frac{x\colon A, x\colon A}{[\lambda_{x\colon A}.x]\colon A \to A}\ ,$$

we set $\mathrm{id}_A \equiv \lambda_{x\colon A}.x$ for the identity function on $A\colon \mathcal{U}$. By the computation rule for $\to$, if $a\colon A$, we get the expected definitional equality

$$[\lambda_{x\colon A}.x](a) \equiv a.$$

Example 2.3.4 (The composite function)**.** By the introduction rule for $\to$, if $f\colon A \to B$ and $g\colon B \to C$

$$\frac{x\colon A, g(f(x))\colon C}{[\lambda_{x\colon A}.g(f(x))\colon C]\colon A \to B}\ ,$$

we set $g \circ f \equiv \lambda_{x\colon A}.g(f(x))$ for the composite function on $A\colon \mathcal{U}$ with values in C, where $B, C\colon \mathcal{U}$ too. By the computation rule for $\to$, if $a\colon A$, we get the expected definitional equality

$$[\lambda_{x\colon A}.(g(f(x))](a) \equiv g(f(a)).$$

We use the following notation conventions:

$$A \to B \to C \equiv A \to (B \to C),$$

and if $g : A \to B \to C$, then

$$g(x, y) \equiv g(x)(y),$$

for every $x\colon A$ and $y\colon B$.

Definition 2.3.5. *If $A\colon \mathcal{U}$, a function $P : A \to \mathcal{U}$ is called a family of types over A in $\mathcal{U}$, or a type family over A in $\mathcal{U}$. Notice that from the rules for the hierarchy of universes we have that*

$$A\colon \mathcal{U},\ \textit{hence}\ A\colon \mathcal{U}',\ \textit{and}\ \mathcal{U}\colon \mathcal{U}',$$

where $\mathcal{U}'$ is the immediate next universe to $\mathcal{U}$. By the rule $\mathrm{Form}_{\to}$ *we get*

$$\frac{A, \mathcal{U}\colon \mathcal{U}'}{A \to \mathcal{U} \in \mathcal{U}'}\ .$$

We write $\Pi, R\colon A \to \mathcal{U}$ for variables of type $A \to \mathcal{U}$.

Example 2.3.6 (The constant type family)**.** The term

$$(\lambda_{x:\,A}.B) : A \to \mathcal{U}$$

is the *constant* type family $B : \mathcal{U}$ in $\mathcal{U}$ over $A : \mathcal{U}$.

Example 2.3.7 (A type family over a universe)**.** The term

$$(\lambda_{X:\,\mathcal{U}}.X \to X) : \mathcal{U} \to \mathcal{U}$$

is a type family in $\mathcal{U}$ over $\mathcal{U}$. We have the following definitional equality between types $A : \mathcal{U}$

$$P(A) \equiv A \to A.$$

2.4 Dependent function type

Next follows the dependent version of the function type.

Definition 2.4.1. $\mathrm{Form}_{\prod}$:

$$\frac{A : \mathcal{U},\ P : A \to \mathcal{U}}{\prod_{x:\,A} P(x) : \mathcal{U}}\ .$$

I.e., given $A : \mathcal{U}$ and a type family over A the type of dependent functions with respect to A and P is in $\mathcal{U}$. A term

$$F : \prod_{x:\,A} P(x)$$

is called a dependent function or a section of the type family P over A. We use capital letters $F, G, H : \prod_{x:\,A} P(x)$ for closed terms and Greek capital letters $\Phi, \Psi, \Theta : \prod_{x:\,A} P(x)$ for variables of this type.

$\mathrm{Intro}_{\prod}$:

$$\frac{A : \mathcal{U},\ P : A \to \mathcal{U},\ \left[x : A, b(x) : P(x)\right]}{[\lambda_{x:\,A}.b(x) : P(x)] : \prod_{x:\,A} P(x)}\ .$$

I.e., if x is a variable of type A and $b(x)$ is a term of type $P(x)$, where the free variables of b are included in $\{x\}$, then the dependent λ-term $\lambda_{x:\,A}.b(x) : P(x)$, or simpler $\lambda_{x:\,A}.b(x)$ is a term of type $\prod_{x:\,A} P(x)$. This rule is also called the dependent λ-abstraction and it introduces the "canonical" elements of the type $\prod_{x:\,A} P(x)$.

$\mathrm{Elim}_{\prod}$, *or* $\mathrm{Ind}_{\prod}$:

$$\frac{F : \prod_{x:\,A} P(x),\ a : A}{F(a) : P(a)}\ .$$

Hence the application of F on the term $a : A$ is the term $F(a) \equiv F[x/a]$ resulting from the substitution of x by a in F.

$\mathrm{Comp}_{\prod}$:

$$\frac{[\lambda_{x:\,A}.b(x) : P(x)] : \prod_{x:\,A} P(x),\ a : A}{[\lambda_{x:\,A}.b(x) : P(x)](a) \equiv b(a) : P(a)}\ ,$$

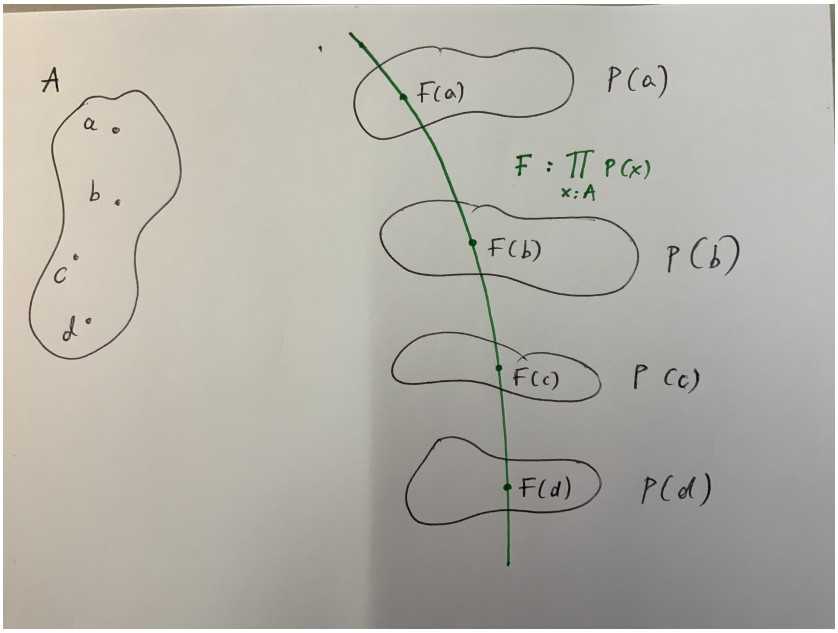


Figure 2.1: A dependent function $F\colon \prod_{x\colon A} P(x)$ is a section of the type family $P\colon A \to \mathcal{U}$

where $b(a) \equiv b[a/x]$ is the term resulting after the substitution of x by a in b. This computation rule, which explains the application of the elimination rule to the introduction rule, is the dependent version of β-conversion.

$\mathrm{Uniq}_{\prod}$*:*

$$\frac{F\colon \prod_{x\colon A} P(x)}{F \equiv [\lambda_{x\colon A}.F(x)\colon P(x)]\colon \prod_{x\colon A} P(x)}\ .$$

This optional uniqueness principle, which expresses that every element of the type $\prod_{x\colon A} P(x)$ is uniquely determined by is elimination and abstraction, is the dependent version of η-conversion.

Formula / Type	Proof-term
$\forall_{x\in A} P(x)$ / $\prod_{x\colon A} P(x)$	$F\colon \prod_{x\colon A} P(x)$

Table 2.3: A dependent function $F\colon \prod_{x\colon A} P(x)$ is a "proof" of the type-theoretic universal formula

Example 2.4.2 (The dependent functions over the constant type family)**.** If $P \equiv \lambda_{x\colon A}.B\colon A \to \mathcal{U}$ is the *constant* type family $B\colon \mathcal{U}$ in $\mathcal{U}$ over $A\colon \mathcal{U}$, then

$$\prod_{x\colon A} P(x) \equiv \prod_{x\colon A} B \equiv A \to B.$$

Example 2.4.3 (The dependent identity function)**.** If $P \equiv (\lambda_{X:\mathcal{U}}.X \to X) : \mathcal{U} \to \mathcal{U}$, the dependent dependent function

$$\mathrm{Id}\colon \prod_{X:\mathcal{U}} (X \to X)$$

is defined by

$$\mathrm{Id} \equiv \lambda_{X:\mathcal{U}}.\mathrm{id}_X,$$

hence $\mathrm{Id}(A) \equiv \mathrm{id}_A$, for every $A\colon \mathcal{U}$.

Example 2.4.4 (Composition of a dependent function with a function)**.** If $A, B\colon \mathcal{U}$, $f\colon A \to B$ $Q\colon B \to \mathcal{U}$ and $G\colon \prod_{y:B} Q(y)$, then

$$(Q \circ f)\colon A \to \mathcal{U},$$

and we define the dependent function

$$(G \circ f)\colon \prod_{x:A} (Q \circ f)(x) \equiv \prod_{x:A} (Q(f(x))$$

by

$$G \circ f \equiv \lambda_{x:A}.G(f(x)),$$

hence $(G \circ f)(a) \equiv G(f(a))$, for every $a\colon A$.

We use the following notational convention

$$\prod_{x,y:A} \equiv \prod_{x:A} \prod_{y:A}.$$

2.5 Product type

Definition 2.5.1. $\mathrm{Form}_\times$:

$$\frac{A, B\colon \mathcal{U}}{A \times B\colon \mathcal{U}}.$$

I.e., given $A, B\colon \mathcal{U}$ the cartesian product, or simply the product $A \times B$ of A, B is in $\mathcal{U}$.
$\mathrm{Intro}_\times$:

$$\frac{a\colon A,\ b\colon B}{(a, b)\colon A \times B}.$$

$\mathrm{Rec}_\times$:

$$\frac{A, B, C\colon \mathcal{U},\ g\colon A \to B \to C}{f\colon A \times B \to C,\ f((a, b)) \equiv g(a, b) \equiv g(a)(b),\ \textit{for every } a\colon A, b\colon B}.$$

$\mathrm{Ind}_\times$:

$$\frac{A, B\colon \mathcal{U},\ P\colon A \times B \to \mathcal{U},\ G\colon \prod_{x:A} \prod_{y:B} P((x, y))}{F\colon \prod_{z:A\times B} P(z),\ F((a, b)) \equiv G(a, b) \equiv G(a)(b),\ \textit{for every } a\colon A, b\colon B}.$$

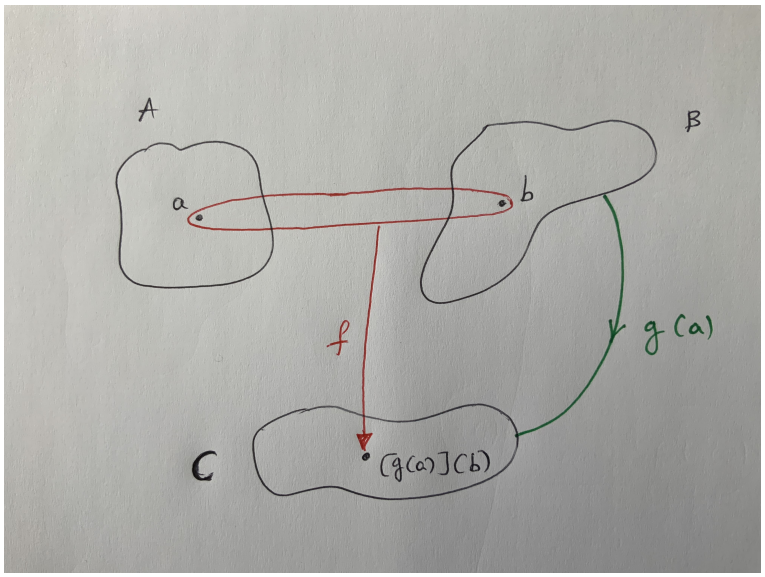


Figure 2.2: The recursion rule for the product $A \times B$

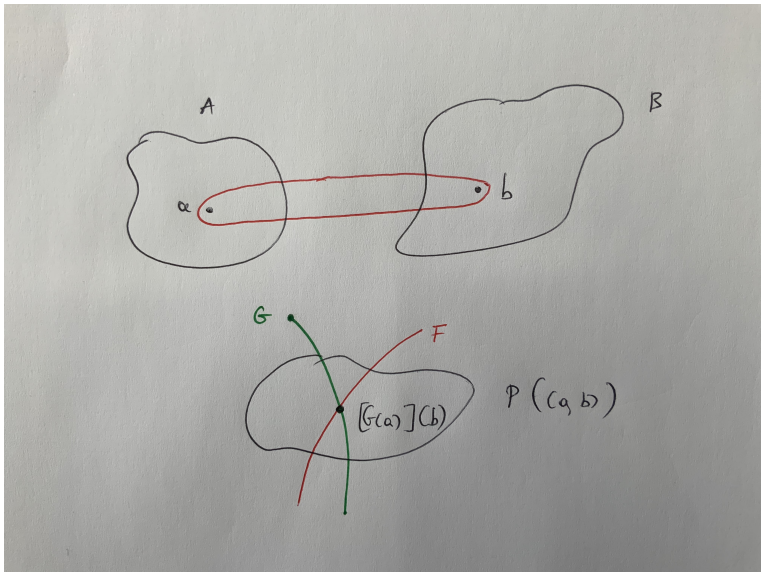


Figure 2.3: The induction rule for the product $A \times B$

Formula / Type	Proof-term
$A \wedge B \;\; / \;\; A \times B$	$(a, b) \colon A \times B$

Table 2.4: A pair $(a, b) \colon A \times B$ is a "proof" of the type-theoretic conjunction

The *recursion principle* $\mathrm{Rec}_\times$ presents the data necessary to the definition of a function f on the defined type, here $A \times B$, and the way this function f operates on the canonical elements of the type, here on the pairs of the form (a, b) with $a \colon A$ and $b \colon B$. Namely, if $C \colon \mathcal{U}$ and $g \colon A \to B \to C$ are given, then there is a function

$$f \colon A \times B \to C,$$

defined by

$$f((a, b)) \equiv g(a, b) \equiv g(a)(b),$$

for every $a \colon A$ and $b \colon B$. We can write this recursion rule as the existence of a term

$$\mathrm{Rec}_\times \colon \prod_{X,Y,Z \colon \mathcal{U}} (X \to Y \to Z) \to (X \times Y) \to Z$$

satisfying the computation rule

$$\mathrm{Rec}_\times(A, B, C, g, (a, b)) \equiv \mathrm{Rec}_{A \times B}(C, g, (a, b)) \equiv g(a)(b)$$

i.e.,

$$f \equiv \mathrm{Rec}_\times(A, B, C, g) \equiv \mathrm{Rec}_{A \times B}(C, g).$$

The induction principle $\mathrm{Ind}_\times$ presents the data necessary to the definition of a *dependent* function F on the defined type, here $A \times B$, and the way this dependent function F operates on the canonical elements of the type, here on the pairs of the form (a, b) with $a \colon A$ and $b \colon B$. Namely, If $P \colon A \times B \to \mathcal{U}$ and

$$G \colon \prod_{x \colon A} \prod_{y \colon B} P((x, y))$$

are given, there is a dependent function

$$F \colon \prod_{z : A \times B} P(z)$$

such that

$$F((a, b)) \equiv G(a, b),$$

for every $a \colon A$ and $b \colon B$. We can write this induction rule as the existence of a term

$$\mathrm{Ind}_\times \colon \prod_{X,Y \colon \mathcal{U}} \prod_{\Pi \colon X \times Y \to \mathcal{U}} \prod_{\Phi \colon \prod_{x \colon X} \prod_{y \colon Y} \Pi((x,y))} \prod_{z : X \times Y} \Pi(z)$$

satisfying the computation rule

$$\mathrm{Ind}_{\times}(A, B, P, G, (a, b)) \equiv \mathrm{Ind}_{A\times B}(P, G, (a, b)) \equiv G(a)(b)$$

i.e.,

$$F \equiv \mathrm{Ind}_{\times}(A, B, P, G) \equiv \mathrm{Ind}_{A\times B}(P, G).$$

The induction rule of an inductively defined type implies the corresponding recursion rule by considering the constant type family. This is also the case here.

Proposition 2.5.2. *The induction rule* $\mathrm{Ind}_{A\times B}$ *implies the recursion rule* $\mathrm{Rec}_{A\times B}$ *i.e., if there is a term*

$$\mathrm{Ind}_{A\times B}\colon \prod_{\Pi\colon A\times B\to\mathcal{U}} \prod_{\Phi\colon \prod_{x\colon A}\prod_{y\colon B}\Pi((x,y))} \prod_{z:A\times B} \Pi(z),$$

satisfying the computation rule $\mathrm{Ind}_{A\times B}(P, G, (a, b)) \equiv G(a)(b)$*, then there is a term*

$$\mathrm{Rec}_{A\times B}\colon \prod_{Z\colon \mathcal{U}} (A \to B \to Z) \to (A \times B) \to Z,$$

satisfying the computation rule $\mathrm{Rec}_{A\times B}(C, g, (a, b)) \equiv g(a)(b)$.

Proof. Exercise. □

In other words, we find a term of type

$$\left(\prod_{X,Y\colon \mathcal{U}} \prod_{\Pi\colon X\times Y\to\mathcal{U}} \prod_{\Phi\colon \prod_{x\colon X}\prod_{y\colon Y}\Pi((x,y))} \prod_{z:X\times Y} \Pi(z)\right) \to \left(\prod_{X,Y,Z\colon \mathcal{U}} (X \to Y \to Z) \to (X\times Y) \to Z\right).$$

Proposition 2.5.3. *If* $A, B\colon \mathcal{U}$*, the first and second projection functions*

$$\mathtt{pr}_{1,A\times B}\colon A \times B \to A, \quad \mathtt{pr}_{1,A\times B}((a, b)) \equiv a, \quad a\colon A, b\colon B$$

$$\mathtt{pr}_{2,A\times B}\colon A \times B \to B, \quad \mathtt{pr}_{2,A\times B}((a, b)) \equiv b, \quad a\colon A, b\colon B,$$

are defined through the recursion principle for $A \times B$*.*

Proof. Exercise. □

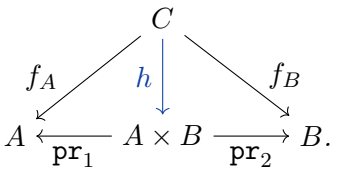
We also write $\mathtt{pr}_1$, instead of $\mathtt{pr}_{1,A\times B}$, and $\mathtt{pr}_2$, instead of $\mathtt{pr}_{2,A\times B}$. The converse to Proposition 2.5.3 holds. Namely, given the projection functions, the recursion term is definable.

Proposition 2.5.4. *If the projection functions* $\mathtt{pr}_{1,A\times B}$ *and* $\mathtt{pr}_{2,A\times B}$*, together with their computation rules* $\mathtt{pr}_{1,A\times B}((a, b)) \equiv a$ *and* $\mathtt{pr}_{2,A\times B}((a, b)) \equiv b$*, for every* $a\colon A$ *and* $b\colon B$*, are given, then the recursion term* $\mathrm{Rec}_{A\times B}$ *is derivable.*

Proof. Exercise. □

Definition 2.5.5. *If* $A, B\colon \mathcal{U}$*, let the new type* $A \leftrightarrow B \equiv (A \to B) \times (B \to A)$*.*

Proposition 2.5.6. *If* $A, B, C\colon \mathcal{U}$*,* $f_A\colon C \to A$ *and* $f_B\colon C \to B$*, there is a function* $h\colon C \to A \times B$*, such that*

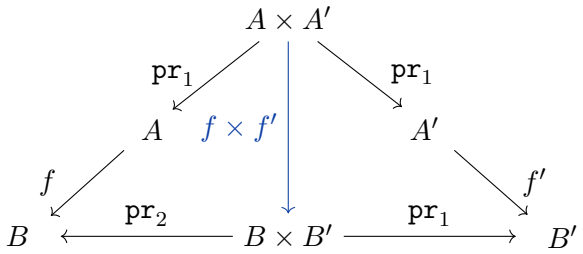


$$(\mathtt{pr}_1 \circ h)(c) \equiv f_A(c) \quad \& \quad (\mathtt{pr}_2 \circ h)(c) \equiv f_B(c),$$

for every $c\colon C$.

Proof. Let $h \equiv \lambda_{x\colon C}.(f_A(x), f_B(x))$, and use β-conversion. □

Corollary 2.5.7. *If* $A, B, A', B'\colon \mathcal{U}$, $f\colon A \to B$ *and* $f'\colon A' \to B'$, *there is a function* $f \times f'\colon A \times A' \to B \times B'$, *such that for every* $a\colon A$ *and* $a'\colon A'$



$$(\mathtt{pr}_1 \circ (f \times f'))((a, a')) \equiv f(a) \quad \& \quad (\mathtt{pr}_2 \circ (f \times f'))((a, a')) \equiv f'(a').$$

Proof. Exercise. □

In summary, we have a *universal-property term* for $A \times B$ of the following type:

$$\mathtt{Univ}_{A\times B}\colon \prod_{X\colon \mathcal{U}} [X \to (A \times B)] \leftrightarrow (X \to A) \times (X \to B).$$

2.6 Coproduct type

Next we define the type that corresponds to the disjoint union of two sets.

Definition 2.6.1. Form_+:

$$\frac{A, B\colon \mathcal{U}}{A + B\colon \mathcal{U}}\ .$$

I.e., given $A, B\colon \mathcal{U}$ *the coproduct, or the disjoint union* $A + B$ *of* A, B *is in* $\mathcal{U}$.
Intro_+:

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

I.e., every term $a\colon A$ *is left-injected into* $A + B$ *and every term* $b\colon B$ *is right-injected into* $A + B$. *Notice that for simplicity we write* $\mathrm{inl}(a)$, *instead of* $\mathrm{inl}_{A+B}(a)$, *and* $\mathrm{inr}(b)$, *instead of* $\mathrm{inr}_{A+B}(b)$.

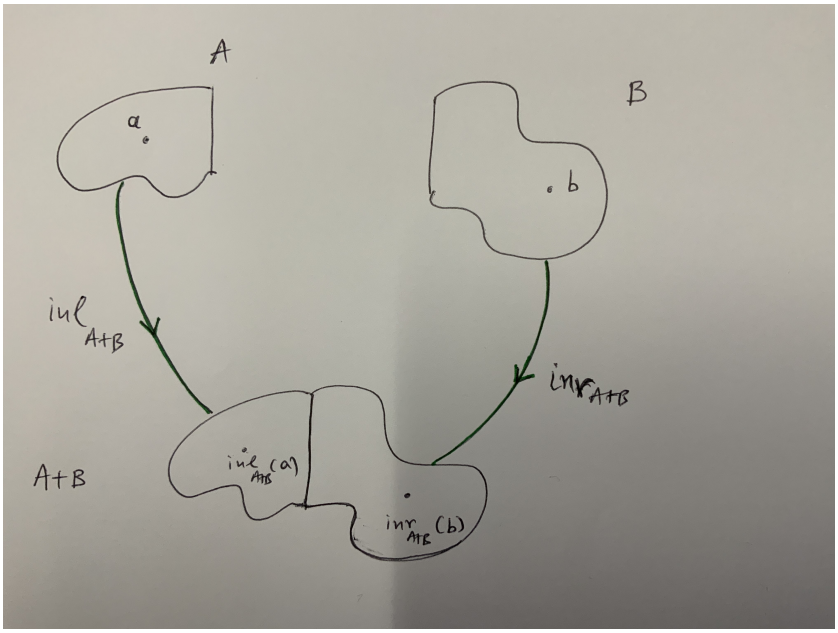


Figure 2.4: Canonical terms of the type $A + B$

Rec_+:

$$\frac{A, B, C\colon \mathcal{U},\ g_l\colon A \to C,\ g_r\colon B \to C}{f\colon A + B \to C,\ f(\mathrm{inl}(a)) \equiv g_l(a),\ f(\mathrm{inr}(b)) \equiv g_r(b),\ \textit{for every } a\colon A, b\colon B}\ .$$

Ind_+:

$$\frac{A, B\colon \mathcal{U},\ P\colon A + B \to \mathcal{U},\ G_l\colon \prod_{x\colon A} P(\mathrm{inl}(x)),\ G_r\colon \prod_{y\colon B} P(\mathrm{inr}(y))}{F\colon \prod_{z\colon A+B} P(z),\ F(\mathrm{inl}(a)) \equiv G_l(a),\ F(\mathrm{inr}(b)) \equiv G_r(b),\ \textit{for every } a\colon A, b\colon B}\ .$$

Formula / Type	Proof-term
$A \vee B \quad / \quad A + B$	$\mathrm{inl}(a), \mathrm{inr}(b)\colon A + B$

Table 2.5: A term $\mathrm{inl}(a)$, or $\mathrm{inr}(b)$ is a "proof" of the type-theoretic disjunction

The *recursion principle* Rec_+ presents the data necessary to the definition of a function f on the defined type, here $A + B$, and the way this function f operates on the canonical elements of the type, here on the terms of the form $\mathrm{inl}(a)$, or $\mathrm{inr}(b)$, with $a\colon A$ and $b\colon B$. Namely, if $C\colon \mathcal{U}$ and if functions $g_l\colon A \to C$ and $g_r\colon B \to C$ are given, then there is a function

$$f\colon A + B \to C,$$

defined by

$$f(\mathrm{inl}(a)) \equiv g_l(a),\ f(\mathrm{inr}(b)) \equiv g_r(b),$$

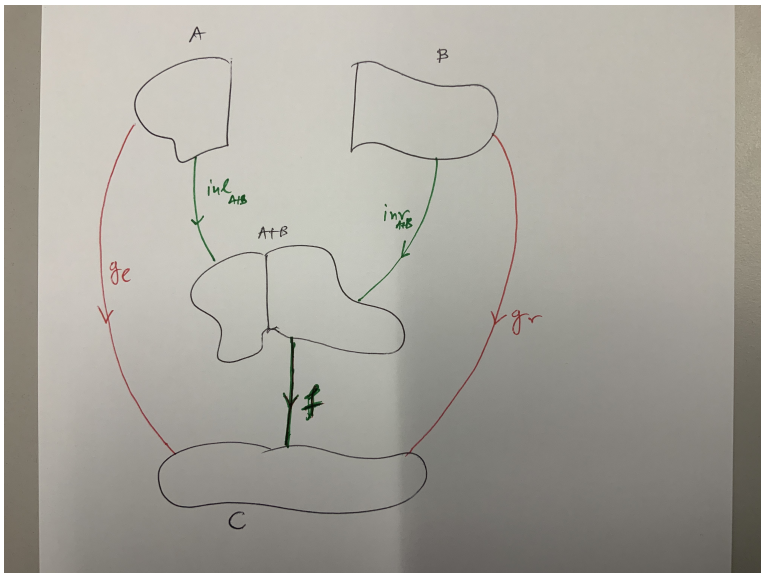


Figure 2.5: The recursion rule for the coproduct $A + B$

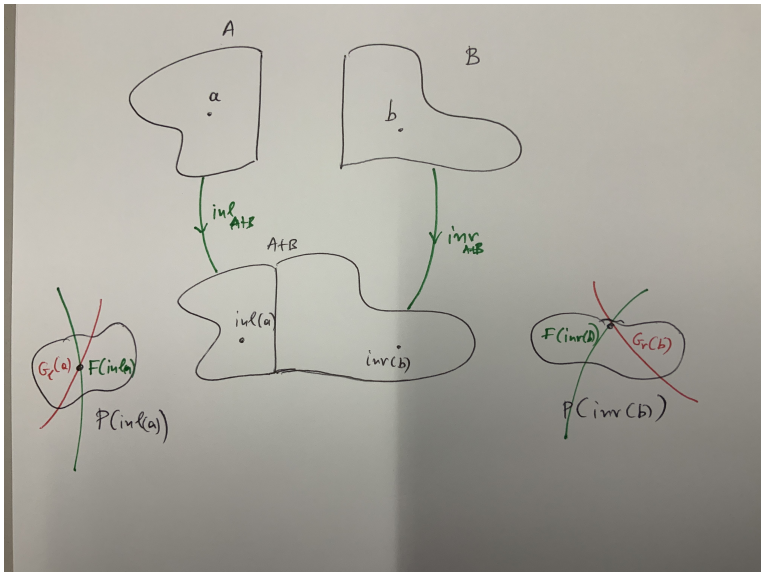


Figure 2.6: The induction rule for the coproduct $A + B$

for every $a\colon A$ and $b\colon B$. We can write this recursion rule as the existence of a term

$$\mathrm{Rec}_+\colon \prod_{X,Y,Z\colon \mathcal{U}} (X \to Z) \to (Y \to Z) \to (X + Y) \to Z$$

satisfying the computation rules

$$\mathrm{Rec}_+(A, B, C, g_l, g_r, \mathrm{inl}(a)) \equiv \mathrm{Rec}_{A+B}(C, g_l, g_r, \mathrm{inl}(a)) \equiv g_l(a),$$

$$\mathrm{Rec}_+(A, B, C, g_l, g_r, \mathrm{inr}(b)) \equiv \mathrm{Rec}_{A+B}(C, g_l, g_r, \mathrm{inr}(b)) \equiv g_r(b)$$

i.e.,

$$f \equiv \mathrm{Rec}_+(A, B, C, g_l, g_r) \equiv \mathrm{Rec}_{A+B}(C, g_l, g_r).$$

The induction principle Ind_+ presents the data necessary to the definition of a *dependent* function F on $A \times B$ and the way this dependent function F operates on the canonical elements $\mathrm{inl}(a)$, or $\mathrm{inr}(b)$ of $A + B$, with $a\colon A$ and $b\colon B$. Namely, If $P : A + B \to \mathcal{U}$ and dependent functions

$$G_l\colon \prod_{x\colon A} P(\mathrm{inl}(x)),\ G_r\colon \prod_{y\colon B} P(\mathrm{inr}(y))$$

are given, there is a dependent function

$$F\colon \prod_{z:A+B} P(z)$$

such that

$$F(\mathrm{inl}(a)) \equiv G_l(a),\ F(\mathrm{inr}(b)) \equiv G_r(b),$$

for every $a\colon A$ and $b\colon B$. We can write this induction rule as the existence of a term

$$\mathrm{Ind}_+\colon \prod_{X,Y\colon \mathcal{U}} \prod_{\Pi\colon X+Y\to\mathcal{U}} \prod_{\Phi_l\colon \prod_{x\colon A}\Pi(\mathrm{inl}(x))} \prod_{\Phi_r\colon \prod_{y\colon B}\Pi(\mathrm{inr}(y))} \prod_{z:X+Y} \Pi(z)$$

satisfying the computation rules

$$\mathrm{Ind}_+(A, B, P, G_l, G_r, \mathrm{inl}(a)) \equiv \mathrm{Ind}_{A+B}(P, G_l, G_r, \mathrm{inl}(a)) \equiv G_l(a),$$

$$\mathrm{Ind}_+(A, B, P, G_l, G_r, \mathrm{inr}(b) \equiv \mathrm{Ind}_{A+B}(P, G_l, G_r, \mathrm{inr}(b)) \equiv G_r(b)$$

i.e.,

$$F \equiv \mathrm{Ind}_+(A, B, P, G_l, G_r) \equiv \mathrm{Ind}_{A+B}(P, G_l, G_r).$$

The above induction rule implies the above recursion rule.

Proposition 2.6.2. *The induction rule* Ind_{A+B} *implies the recursion rule* Rec_{A+B} *i.e., if there is a term*

$$\mathrm{Ind}_{A+B}\colon \prod_{\Pi\colon X+Y\to\mathcal{U}} \prod_{\Phi_l\colon \prod_{x\colon A}\Pi(\mathrm{inl}(x))} \prod_{\Phi_r\colon \prod_{y\colon B}\Pi(\mathrm{inr}(y))} \prod_{z:X+Y} \Pi(z),$$

satisfying the above computation rules, then there is a term

$$\mathrm{Rec}_{A+B}\colon \prod_{Z\colon \mathcal{U}} (A \to Z) \to (B \to Z) \to (A + B) \to Z,$$

satisfying the above computation rules.

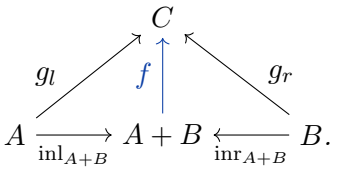
Proof. Exercise. □

Proposition 2.6.3. *If $A, B : \mathcal{U}$, let the left and right injections*

$$\mathrm{inl}_{A+B} : A \to A + B, \quad \mathrm{inl}_{A+B} \equiv \lambda_{x:\,A}.\mathrm{inl}(x),$$

$$\mathrm{inr}_{A+B} : A \to A + B, \quad \mathrm{inr}_{A+B} \equiv \lambda_{y:\,B}.\mathrm{inr}(y).$$

If $C : \mathcal{U}$ and $g_l : A \to C, g_r : B \to C$, there is $f : A + B \to \mathcal{C}$ such that for every $a : A$ and $b : B$

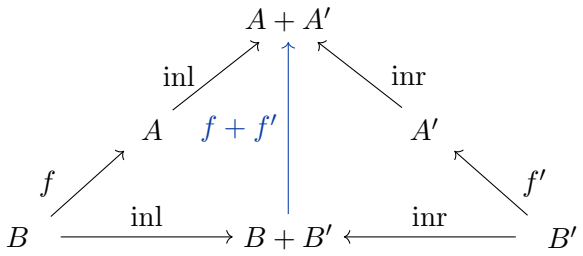


$$(f \circ \mathrm{inl}_{A+B})(a) \equiv g_l(a), \quad \& \quad (f \circ \mathrm{inr}_{A+B})(b) \equiv g_r(b),$$

Proof. Exercise. □

For simplicity, we write inl, instead of inl_{A+B}, and inr, instead of inr_{A+B}. In contrast to what happens in the product type, the converse to Proposition 2.6.3 cannot be shown (why?).

Corollary 2.6.4. *If $A, B, A', B' : \mathcal{U}$, $f : B \to A$ and $f' : B' \to A'$, there is a function $f + f' : B + B' \to A + A'$, such that*



$$\big(f + f'\big)(\mathrm{inl}(b)) \equiv \mathrm{inl}(f(b)) \quad \& \quad \big(f + f'\big)(\mathrm{inr}(b')) \equiv \mathrm{inr}(f'(b')),$$

for every $b : B$ and $b' : B'$.

Proof. Exercise. □

In summary, we have a *universal-property term* for $A + B$ of the following type:

$$\mathtt{Univ}_{A+B} : \prod_{X\,:\,\mathcal{U}} [(A + B) \to X] \leftrightarrow (A \to X) \times (B \to X).$$

Notice that we cannot prove now that the injection of A under inl into $A + B$ is "disjoint" to the injection of B under inr into $A + B$.

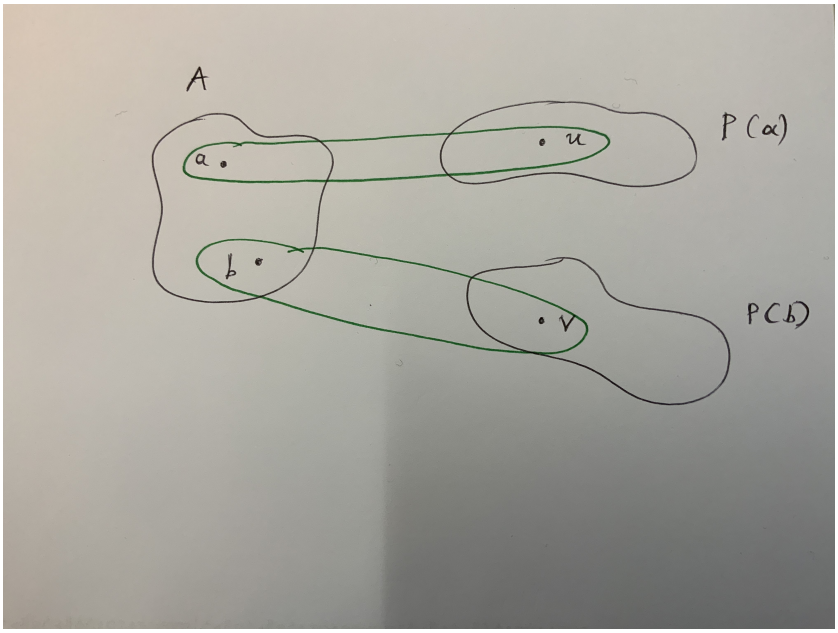


Figure 2.7: Canonical terms of the type $\sum_{x:\,A} P(x)$

2.7 Dependent pair type

Next we define the dependent version of the product type.

Definition 2.7.1. Form$_{\sum}$:

$$\frac{A\colon \mathcal{U}, P\colon A \to \mathcal{U}}{\sum_{x:\,A} P(x)\colon \mathcal{U}}\ .$$

I.e., given $A\colon \mathcal{U}$ *and a type family* $P\colon A \to \mathcal{U}$*, the dependent pair type* $\sum_{x:\,A} P(x)$ *is in* $\mathcal{U}$*.*
Intro$_{\sum}$:

$$\frac{a\colon A,\ b\colon P(a)}{(a,b)\colon \sum_{x:\,A} P(x)}\ .$$

I.e., a canonical term of $\sum_{x:\,A} P(x)$ *is a pair* (a,b)*, where* $a\colon A$ *and* $b\colon P(a)$*.*
Rec$_{\sum}$:

$$\frac{A, C\colon \mathcal{U},\ P\colon A \to \mathcal{U},\ G\colon \prod_{x:\,A} \big(P(x) \to C\big)}{f\colon \big(\sum_{x:\,A} P(x)\big) \to C,\quad f((a,b)) \equiv G((a,b)) \equiv G(a)(b),\ \textit{for every } a\colon A, b\colon P(a)}\ .$$

Ind$_{\sum}$:

$$\frac{A\colon \mathcal{U},\ P\colon A \to \mathcal{U},\ Q\colon \big(\sum_{x:\,A} P(x)\big) \to \mathcal{U},\ G\colon \prod_{x:\,A} \prod_{y:\,P(x)} Q((x,y))}{F\colon \prod_{z:\,\sum_{x:\,A} P(x)} Q(z),\ F((a,b)) \equiv G(a,b) \equiv G(a)(b),\ \textit{for every } a\colon A, b\colon P(a)}\ .$$

Formula / Type	Proof-term
$\exists_{x\in A} P(x)$ / $\sum_{x:\,A} P(x)$	$(a,b):\ \sum_{x:\,A} P(x)$

Table 2.6: A dependent pair $(a,b):\ \sum_{x:\,A} P(x)$ is a "proof" of the type-theoretic existential formula

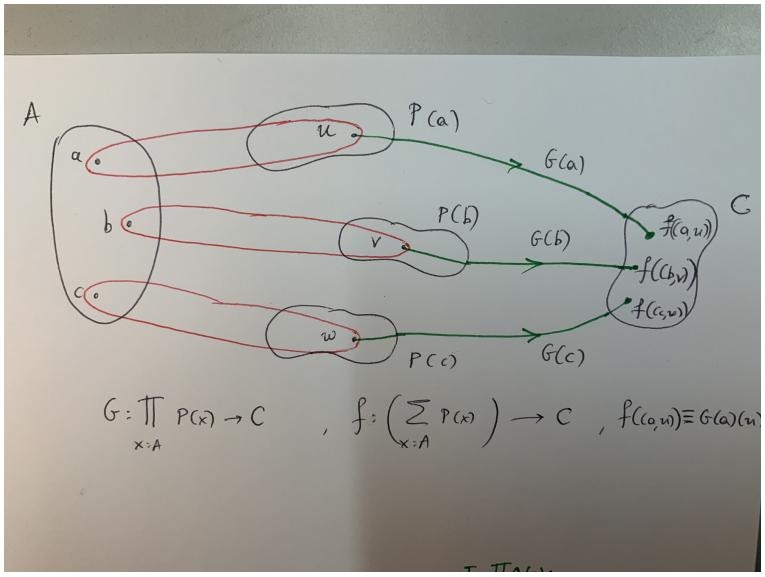


Figure 2.8: Recursion rule for the Sigma-type

If we interpret a type family $P\colon A \to \mathcal{U}$ over A as a *predicate* on A, then we can correspond the type $\sum_{x:\,A} P(x)$ to the set

$$X_P = \{x \in A \mid P(x)\}.$$

The rule $\mathrm{Rec}_{\sum}$ presents the data necessary to the definition of a function f on $\sum_{x:\,A} P(x)$ and the way this function f operates on the canonical elements of it i..e, on the pairs of the form (a,b) with $a\colon A$ and $b\colon P(a)$. Namely, if $C\colon \mathcal{U}$ and $G : \prod_{x:\,A}(P(x) \to C)$, there exists

$$f\colon \left(\sum_{x:\,A} P(x)\right) \to C$$

such that

$$f((a,b)) \equiv G(a,b) \equiv G(a)(b),$$

for every $a\colon A$ and $b\colon P(a)$. We can write this recursion rule as the existence of a term

$$\mathrm{Rec}_{\sum}\colon \prod_{X,Z:\,\mathcal{U}} \prod_{\Pi:\,X\to\mathcal{U}} \prod_{\Phi:\,\prod_{x:\,X}(\Pi(x)\to Z)} \left[\left(\sum_{x:\,X} \Pi(x)\right) \to Z\right]$$

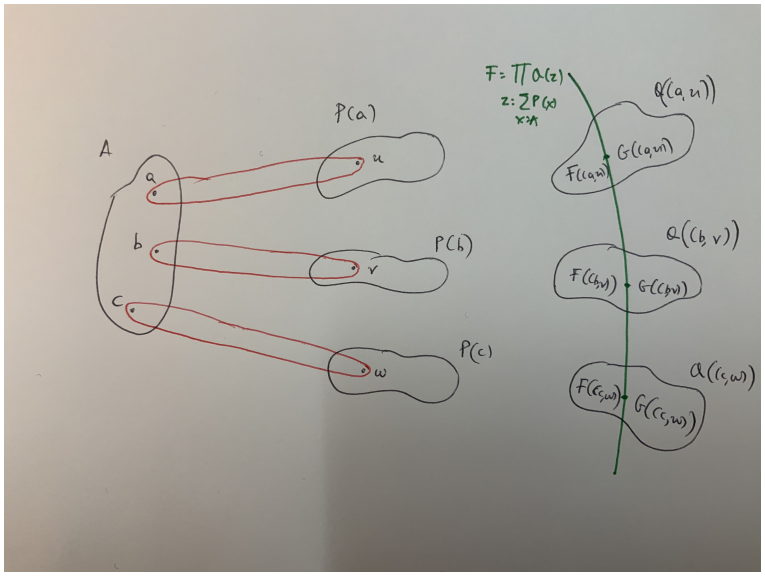


Figure 2.9: Induction rule for the Sigma-type

satisfying the computation rule

$$\mathrm{Rec}_{\sum}(A, C, P, G, (a, b)) \equiv \mathrm{Rec}_{\sum_{x:\,A} P(x)}(C, G, (a, b)) \equiv G(a)(b)$$

i.e.,

$$f \equiv \mathrm{Rec}_{\sum}(A, C, P, G) \equiv \mathrm{Rec}_{\sum_{x:\,A} P(x)}(C, G).$$

The induction principle $\mathrm{Ind}_{\sum}$ presents the data necessary to the definition of a *dependent* function F on $\sum_{x:\,A} P(x)$ and the way this dependent function F operates on the canonical elements of $\sum_{x:\,A} P(x)$. Namely, if $Q : \left(\sum_{x:\,A} P(x)\right) \to \mathcal{U}$, and

$$G : \prod_{x:\,A} \prod_{y:\,P(x)} Q((x, y)),$$

there is a dependent function

$$F : \prod_{z:\,\sum_{x:\,A} P(x)} Q(z),$$

such that

$$F((a, b)) \equiv G(a, b) \equiv G(a)(b),$$

for every $a : A$ and $b : P(a)$. We can write this induction rule as the existence of a term

$$\mathrm{Ind}_{\sum} : \prod_{X:\,\mathcal{U}} \prod_{\Pi:\,X\to\mathcal{U}} \prod_{R:\,\left(\sum_{x:\,X} \Pi(x)\right)\to\mathcal{U}} \prod_{\Phi:\,\prod_{x:\,X}\prod_{y:\,\Pi(x)} R((x,y))} \prod_{z:\,\sum_{x:\,X}\Pi(x)} R(z)$$

satisfying the computation rule

$$\mathrm{Ind}_{\sum}(A, P, Q, G, (a, b)) \equiv \mathrm{Ind}_{\sum_{x:\,A} P(x)}(Q, G, (a, b)) \equiv G(a)(b)$$

i.e.,

$$F \equiv \mathrm{Ind}_{\sum}(A, P, Q, G) \equiv \mathrm{Ind}_{\sum_{x:\,A} P(x)}(Q, G).$$

Next we explain why $\sum_{x:\,A} P(x)$ is called the type of dependent pairs.

Remark 2.7.2. *If $P\colon A \to \mathcal{U}$ is the constant type family $B\colon \mathcal{U}$, then*

$$\sum_{x\colon A} P(x) \equiv \sum_{x\colon A} B \equiv A \times B.$$

Proof. Exercise. □

Proposition 2.7.3. *The induction rule $\mathrm{Ind}_{\sum_{x\colon A} P(x)}$ implies the recursion rule $\mathrm{Rec}_{\sum_{x\colon A} P(x)}$ i.e., if there is a term*

$$\mathrm{Ind}_{\sum}\colon \prod_{X\colon \mathcal{U}} \prod_{\Pi\colon X\to\mathcal{U}} \prod_{R\colon \left(\sum_{x\colon X}\Pi(x)\right)\to\mathcal{U}} \prod_{\Phi\colon \prod_{x\colon X}\prod_{y\colon \Pi(x)} R((x,y))} \prod_{z\colon \sum_{x\colon X}\Pi(x)} R(z),$$

satisfying the computation rule $\mathrm{Ind}_{\sum_{x\colon A} P(x)}(Q, G, (a, b)) \equiv G(a)(b)$, then there is a term

$$\mathrm{Rec}_{\sum}\colon \prod_{X,Z\colon \mathcal{U}} \prod_{\Pi\colon X\to\mathcal{U}} \prod_{\Phi\colon \prod_{x\colon X}\left(\Pi(x)\to Z\right)} \left[\left(\sum_{x\colon X}\Pi(x)\right)\to Z\right],$$

satisfying the computation rule $\mathrm{Rec}_{\sum_{x\colon A} P(x)}(C, G, (a, b)) \equiv G(a)(b)$.

Proof. Exercise. □

Proposition 2.7.4. *Let $A\colon \mathcal{U}$ and $P\colon A \to \mathcal{U}$.*
(i) *The first projection function*

$$\mathtt{pr}_{1,\Sigma} : \left(\sum_{x\colon A} P(x)\right) \to A, \quad \mathtt{pr}_{1,\Sigma}((a, b)) \equiv a, \quad a\colon A, b\colon P(a),$$

is defined through the recursion principle for $\sum_{x\colon A} P(x)$.
(ii) *The second projection (dependent) function*

$$\mathtt{pr}_{2,\Sigma}\colon \prod_{z:\sum_{x\colon A} P(x)} P(\mathtt{pr}_{1,\Sigma}(z)), \quad \mathtt{pr}_{2,\Sigma}((a, b)) \equiv b, \quad a\colon A, b\colon P(a),$$

is defined through the induction principle for $\sum_{x\colon A} P(x)$.

Proof. (i) With the use of $\mathrm{Rec}_{\sum}$, if $G : \prod_{x:A}(P(x) \to A)$ is defined by

$$G \equiv \lambda_{x\colon A}.\left(\lambda_{y\colon P(x)}.x\right),$$

then there is a function

$$\mathtt{pr}_{1,\sum} : \left(\sum_{x\colon A} P(x)\right) \to A$$

that satisfies the computation rule

$$\mathtt{pr}_{1,\sum}((a, b)) \equiv G(a, b) \equiv G(a)(b) \equiv \left(\lambda_{y\colon P(a)}.a\right)(b) \equiv a,$$

for every $a\colon A$ and $b\colon P(x)$.

(ii) Use the induction principle $\mathrm{Ind}_{\sum}$ for the type family

$$Q\colon \left(\sum_{x\colon A} P(x)\right) \to \mathcal{U},$$

defined by

$$Q \equiv \lambda_{z\colon \sum_{x\colon A} P(x)}.P\big(\mathtt{pr}_{1,\sum}(z)\big),$$

and an appropriately defined term

$$G\colon \prod_{x\colon A}\prod_{y\colon P(x)} Q((x,y)).$$ □

Next we write the dependent version of Proposition 2.5.4.

Proposition 2.7.5. *Let $A\colon \mathcal{U}$, $P\colon A \to \mathcal{U}$ and*

$$G\colon \prod_{x\colon A} \big(P(x) \to C\big).$$

If the projection functions $\mathtt{pr}_{1,\sum}$ and $\mathtt{pr}_{2,\sum}$, together with their computation rules are given, then the recursion term $\mathrm{Rec}_{\sum}(A, C, P, G)$, together with its computation rule, are derivable.

Proof. Exercise. □

2.8 The type-theoretic axiom of choice

In MLTT the type-theoretic formulation of the following form of the axiom of choice

$$\forall_{x\in A}\exists_{y\in B}R(x,y) \Rightarrow \exists_{f\in \mathbb{F}(A,B)}\forall_{x\in A}R(x,f(x)),$$

where $R \subseteq A \times B$, is provable! Although the type-theoretic axiom of choice is the direct translation of the axiom of choice in MLTT, it does not involve a "choice", as the classical one. The reason behind its provability is the proof-relevant character of the types involved.

Theorem 2.8.1. *If $A, B\colon \mathcal{U}$ and $R\colon A \to B \to \mathcal{U}$, there is a term*

$$\mathrm{AC}_{A,B,R}\colon \left(\prod_{x\colon A}\sum_{y\colon B} R(x,y)\right) \to \left(\sum_{\phi\colon A\to B}\prod_{x\colon A} R(x,\phi(x))\right).$$

Proof. Let a term

$$G\colon \prod_{x\colon A}\sum_{y\colon B} R(x,y).$$

By η-conversion we have that

$$G \equiv \lambda_{x\colon A}.\left[G(x)\colon \sum_{y\colon B} R(x,y)\right].$$

We show that this term G carries enough information to define its value

$$\mathrm{AC}_{A,B,R}(G)\colon \sum_{\phi\colon A\to B}\prod_{x\colon A} R(x,\phi(x)).$$

Clearly

$$\mathtt{pr}_{1,\sum}\big(G(x)\big)\colon B \quad \& \quad \mathtt{pr}_{2,\sum}\big(G(x)\big)\colon R\big(x, \mathtt{pr}_{1,\sum}\big(G(x)\big)\big).$$

Hence

$$f_G \equiv \Big[\lambda_{x\colon A}.\mathtt{pr}_{1,\sum}\big(G(x)\big)\colon B\Big]\colon A \to B,$$

$$F_G \equiv \Big[\lambda_{x\colon A}.\mathtt{pr}_{2,\sum}\big(G(x)\big)\colon R\big(x, \mathtt{pr}_{1,\sum}\big(G(x)\big)\big)\Big]\colon \prod_{x\colon A} R(x, f_G(x)).$$

Consequently, we define

$$\mathrm{AC}_{A,B,R} \equiv \lambda_{\Phi\colon \prod_{x\colon A}\sum_{y\colon B} R(x,y)}.\big(f_\Phi, F_\Phi\big),$$

where

$$f_\Phi \equiv \Big[\lambda_{x\colon A}.\mathtt{pr}_{1,\sum}\big(\Phi(x)\big)\colon B\Big]\colon A \to B,$$

$$F_\Phi \equiv \Big[\lambda_{x\colon A}.\mathtt{pr}_{2,\sum}\big(\Phi(x)\big)\colon R\big(x, \mathtt{pr}_{1,\sum}\big(\Phi(x)\big)\big)\Big]\colon \prod_{x\colon A} R(x, f_\Phi(x)). \qquad \square$$

The converse to Theorem 2.8.1 also holds.

Proposition 2.8.2. *If $A, B\colon \mathcal{U}$ and $R\colon A \to B \to \mathcal{U}$, there is a term*

$$\mathrm{CA}_{A,B,R}\colon \Big(\sum_{\phi\colon A\to B}\prod_{x\colon A} R(x,\phi(x))\Big) \to \Big(\prod_{x\colon A}\sum_{y\colon B} R(x,y)\Big).$$

Proof. Exercise. $\square$

2.9 The empty type

Next we define the type corresponding to the empty set.

Definition 2.9.1. Form$_{\mathbf{0}}$:

$$\frac{}{\mathbf{0}\colon \mathcal{U}}\ .$$

I.e., the empty type $\mathbf{0}$ is in $\mathcal{U}$.

Intro$_{\mathbf{0}}$: *There is no such rule i.e., there is no canonical element of $\mathbf{0}$.*

Rec$_{\mathbf{0}}$:

$$\frac{A\colon \mathcal{U}}{\mathrm{Efq}_A\colon \mathbf{0} \to A}\ .$$

I.e., if $A\colon \mathcal{U}$, there is always a term $\mathrm{Efq}_A\colon \mathbf{0} \to A$ without giving some computation rule for it.

Ind$_{\mathbf{0}}$:

$$\frac{P\colon \mathbf{0} \to \mathcal{U}}{\mathrm{EFQ}_P\colon \prod_{x\colon \mathbf{0}} P(x)}\ .$$

I.e., if $P\colon \mathbf{0} \to \mathcal{U}$, there is always a term $\mathrm{EFQ}_P\colon \prod_{x\colon \mathbf{0}} P(x)$ without giving some computation rule for it.

Formula / Type	Proof-term
$\bot$ / $\mathbf{0}$	–

Table 2.7: There is no term or "proof" of the type-theoretic absurdity

The rule $\mathrm{Rec}_{\mathbf{0}}$ corresponds to the intuitionistic principle *ex falso quodlibet*

$$\bot \to A,$$

for every formula A. We can write the above rules as terms

$$\mathrm{Rec}_{\mathbf{0}} \equiv \mathrm{Efq}\colon \prod_{X\colon \mathcal{U}} (\mathbf{0} \to X),$$

$$\mathrm{Ind}_{\mathbf{0}} \equiv \mathrm{EFQ}\colon \prod_{\Pi\colon \mathbf{0}\to\mathcal{U}} \prod_{x\colon \mathbf{0}} \Pi(x).$$

Trivially, the existence of $\mathrm{Ind}_{\mathbf{0}}$ implies the existence of $\mathrm{Rec}_{\mathbf{0}}$.

Definition 2.9.2. *If $A\colon \mathcal{U}$, the negation $\neg A$ of A is the type in $\mathcal{U}$ defined by*

$$\neg A \equiv A \to \mathbf{0}.$$

We say that A is inhabited, if there is a closed term $a\colon A$, and we say that A is not inhabited, or empty, if $\neg A$ is inhabited.

Remark 2.9.3. (i) *The type $\mathbf{0}$ is empty.*
(ii) *If $A\colon \mathcal{U}$ is inhabited, then a term of type $\mathbf{0} \to A$ is definable.*

Proof. (i) By Definition 2.9.2 it suffices to show that $\neg\mathbf{0} \equiv \mathbf{0} \to \mathbf{0}$ is inhabited. Clearly, $\mathrm{Efq}_{\mathbf{0}}\colon \mathbf{0} \to \mathbf{0}$.
(ii) If $a\colon A$, let $f\colon \mathbf{0} \to A$ be defined by $f \equiv \lambda_{x\colon \mathbf{0}}.a$. □

2.10 The unit type

Next we define the type-theoretic version of a singleton.

Definition 2.10.1. $\mathrm{Form}_{\mathbf{1}}$:

$$\frac{}{\mathbf{1}\colon \mathcal{U}}\ .$$

I.e., the unit type $\mathbf{1}$ is in $\mathcal{U}$.
$\mathrm{Intro}_{\mathbf{1}}$:

$$\frac{}{0_{\mathbf{1}}\colon \mathbf{1}}\ .$$

I.e., the unit type has a single canonical term $0_{\mathbf{1}}$.

$\mathrm{Rec}_{\mathbf{1}}$:

$$\frac{A\colon \mathcal{U},\ a\colon A}{f\colon \mathbf{1}\to A,\ f(0_{\mathbf{1}})\equiv a}\ .$$

$\mathrm{Ind}_{\mathbf{1}}$:

$$\frac{P\colon \mathbf{1}\to\mathcal{U},\ a\colon P(0_{\mathbf{1}})}{F\colon \prod_{x\colon \mathbf{1}} P(x),\ F(0_{\mathbf{1}})\equiv a}\ .$$

By $\mathrm{Rec}_{\mathbf{1}}$, if $A\colon \mathcal{U}$, then to define a function $f\colon \mathbf{1}\to A$, it suffices to know a term $a\colon A$. Similarly, if $P\colon \mathbf{1}\to\mathcal{U}$, then by $\mathrm{Ind}_{\mathbf{1}}$ to define a dependent function $F\colon \prod_{x\colon \mathbf{1}} P(x)$, it suffices to know a term $a\colon P(0_{\mathbf{1}})$.

Formula / Type	Proof-term
$\top$ / $\mathbf{1}$	$0_{\mathbf{1}}\colon \mathbf{1}$

Table 2.8: There is term or "proof" of the type-theoretic verum (truth)

Remark 2.10.2. *If $A\colon \mathcal{U}$, there is a term $1_A\colon A\to\mathbf{1}$.*

Proof. Exercise. □

2.11 The type of booleans

Next we define the type-theoretic version of the set of truth-values.

Definition 2.11.1. $\mathrm{Form}_{\mathbf{2}}$:

$$\frac{}{\mathbf{2}\colon \mathcal{U}}\ .$$

I.e., the type of booleans $\mathbf{2}$ is in $\mathcal{U}$.

$\mathrm{Intro}_{\mathbf{2}}$:

$$\frac{}{0_{\mathbf{2}}\colon \mathbf{2}}\ ,\quad \frac{}{1_{\mathbf{2}}\colon \mathbf{2}}\ .$$

I.e., the type of booleans has the canonical terms $0_{\mathbf{2}}, 1_{\mathbf{2}}$.

$\mathrm{Rec}_{\mathbf{2}}$:

$$\frac{A\colon \mathcal{U},\ a,b\colon A}{f\colon \mathbf{2}\to A,\ f(0_{\mathbf{2}})\equiv a,\ f(1_{\mathbf{2}})\equiv b}\ .$$

$\mathrm{Ind}_{\mathbf{2}}$:

$$\frac{P\colon \mathbf{2}\to\mathcal{U},\ a\colon P(0_{\mathbf{2}}),\ b\colon P(1_{\mathbf{2}})}{F\colon \prod_{x\colon \mathbf{2}} P(x),\ F(0_{\mathbf{2}})\equiv a\ F(1_{\mathbf{2}})\equiv b}\ .$$

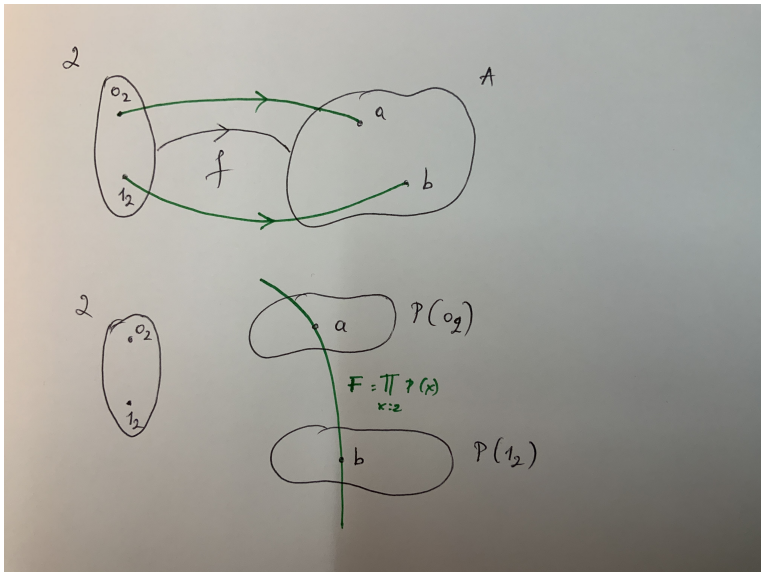


Figure 2.10: $\mathrm{Rec}_{\mathbf{2}}$ and $\mathrm{Ind}_{\mathbf{2}}$

By $\mathrm{Rec}_{\mathbf{2}}$, if $A : \mathcal{U}$, then to define a function $f : \mathbf{2} \to A$, it suffices to know two terms $a, b : A$. Similarly, if $P : \mathbf{2} \to \mathcal{U}$, then by $\mathrm{Ind}_{\mathbf{2}}$ to define a dependent function $F : \prod_{x:\mathbf{2}} P(x)$, it suffices to know terms $a : P(0_{\mathbf{2}})$ and $b : P(1_{\mathbf{2}})$.

Remark 2.11.2. (i) *There is a function* $\mathtt{sw} : \mathbf{2} \to \mathbf{2}$ *such that* $\mathtt{sw}(0_{\mathbf{2}}) \equiv 1_{\mathbf{2}}$ *and* $\mathtt{sw}(1_{\mathbf{2}}) \equiv 0_{\mathbf{2}}$.
(ii) *There is a function* $f : \mathbf{2} \to \mathbf{1} + \mathbf{1}$ *such that* $f(0_{\mathbf{2}}) \equiv \mathrm{inl}(0_{\mathbf{1}})$ *and* $f(1_{\mathbf{2}}) \equiv \mathrm{inr}(0_{\mathbf{1}})$.
(iii) *There is a function* $g : \mathbf{1} + \mathbf{1} \to \mathbf{2}$ *such that* $g\big(\mathrm{inl}(0_{\mathbf{1}})\big) \equiv 0_{\mathbf{2}}$ *and* $g\big(\mathrm{inr}(0_{\mathbf{1}})\big) \equiv 1_{\mathbf{2}}$.

Proof. Exercise. □

Notice that we cannot show yet that $0_{\mathbf{2}}$ and $1_{\mathbf{2}}$ are "different" terms. Later we will see that with the help of the above functions f, g the type $\mathbf{2}$ and the coproduct $\mathbf{1} + \mathbf{1}$ not only are logically equivalent, but also they can be identified. All recursion and induction principles mentioned so far are relative to a given universe and we have a recursion and induction such principle for each universe in the hierarchy of universes. This fact can be used as follows.

Proposition 2.11.3. *If* $A, B : \mathcal{U}$, *there is a type family* $P : \mathbf{2}' \to \mathcal{U}$, *where* $\mathbf{2}' : \mathcal{U}'$ *is the type of booleans in the immediate successor universe* $\mathcal{U}'$ *to* $\mathcal{U}$, *such that*

$$P(0_{\mathbf{2}}) \equiv A \quad \& \quad P(1_{\mathbf{2}}) \equiv B.$$

Moreover, there are non-trivial logical equivalences

$$\prod_{x:\mathbf{2}} P(x) \leftrightarrow A \times B \quad \& \quad \sum_{x:\mathbf{2}} P(x) \leftrightarrow A + B.$$

Proof. As $A, B, \mathcal{U} : \mathcal{U}'$, by $\mathrm{Rec}_{\mathbf{2}'}$ in $\mathcal{U}'$, there is such a type family P in $\mathcal{U}'$. The required logical equivalences are straightforward to define. □

Later we will see that the above types can be identified, hence the product and coproduct of types are redundant in the presence of the type of booleans, the dependent products , the dependent pairs, and the hierarchy of universes.

2.12 The type of naturals

Next we define the type-theoretic version of the set of natural numbers.

Definition 2.12.1. $\mathrm{Form}_{\mathbf{N}}$:

$$\frac{}{\mathbf{N}\colon \mathcal{U}}\ .$$

I.e., the type of natural numbers $\mathbf{N}$ *is in* $\mathcal{U}$.
$\mathrm{Intro}_{\mathbf{N}}$:

$$\frac{}{0_{\mathbf{N}}\colon \mathbf{N}}\ ,\quad \frac{n\colon \mathbf{N}}{\mathtt{succ}(n)\colon \mathbf{N}}\ ,$$

I.e., the type of naturals has the canonical terms $0_{\mathbf{N}}, \mathtt{succ}(0_{\mathbf{N}}), \mathtt{succ}(\mathtt{succ}(0_{\mathbf{N}}))$, *etc..*
$\mathrm{Rec}_{\mathbf{N}}$:

$$\frac{A\colon \mathcal{U},\ a\colon A,\ s\colon \mathbf{N}\to A\to A}{f\colon \mathbf{N}\to A,\ f(0_{\mathbf{N}})\equiv a,\ f(\mathtt{succ}(n))\equiv s\big(n, f(n)\big)}\ .$$

$\mathrm{Ind}_{\mathbf{N}}$:

$$\frac{P\colon \mathbf{N}\to\mathcal{U},\ a\colon P(0_{\mathbf{N}}),\ S\colon \prod_{x\colon \mathbf{N}}\big(P(x)\to P(\mathtt{succ}(x)\big)}{F\colon \prod_{x\colon \mathbf{N}}P(x),\ F(0_{\mathbf{N}})\equiv a,\ F(\mathtt{succ}(n))\equiv S(n,F(n))}\ .$$

We define the following canonical terms of $\mathbf{N}$:

$$1\equiv \mathtt{succ}(0),$$

$$2\equiv \mathtt{succ}(1)\equiv \mathtt{succ}(\mathtt{succ}(0)),$$

$$3\equiv \mathtt{succ}(2)\equiv \mathtt{succ}(\mathtt{succ}(\mathtt{succ}(0))),$$

$$4\equiv \mathtt{succ}(3)\equiv \mathtt{succ}(\mathtt{succ}(\mathtt{succ}(\mathtt{succ}(0)))),$$

and so on. Recall the set-theoretic induction principle $\mathrm{Ind}_{\mathbb{N}}$ for $\mathbb{N}$:

$$\big[P(0)\ \&\ \forall_{n\in\mathbb{N}}(P(n)\Rightarrow P(\mathrm{Succ}(n)))\big]\Rightarrow \forall_{n\in\mathbb{N}}(P(n)),$$

where $P(n)$ is an arbitrary formula on the set of naturals $\mathbb{N}$. The induction principle of the type $\mathbf{N}$ is a proof-relevant version of $\mathrm{Ind}_{\mathbb{N}}$, as $a\colon P(0_{\mathbf{N}})$ witnesses that $P(0)$ is true, $S\colon \prod_{x\colon \mathbf{N}}\big(P(x)\to P(\mathtt{succ}(x)\big)$ witnesses that for an arbitrary $n\colon \mathbf{N}$, if $a\colon P(n)$, then $[S(n)](a)\colon P(\mathtt{succ}(n)$, and hence $F\colon \prod_{x\colon \mathbf{N}}P(x)$ witnesses the universal validity of P on $\mathbf{N}$. The computation-part of $\mathrm{Ind}_{\mathbf{N}}$ expresses the fact that the way F proves $\prod_{x\colon \mathbf{N}}P(x)$ is determined from the proof-terms a and S.

By $\mathrm{Rec}_{\mathbf{N}}$, if $A\colon \mathcal{U}$, to define a function $f\colon \mathbf{N}\to A$, it suffices to know a term $a\colon A$ and a function $s\colon \mathbf{N}\to A\to A$. In this case $f\colon \mathbf{N}\to A$ satisfies the computation rules

$$f(0_{\mathbf{N}})\equiv a,$$

$$f(\mathtt{succ}(n))\equiv s(n,f(n)),$$

and we say that it is defined by *primitive recursion.*

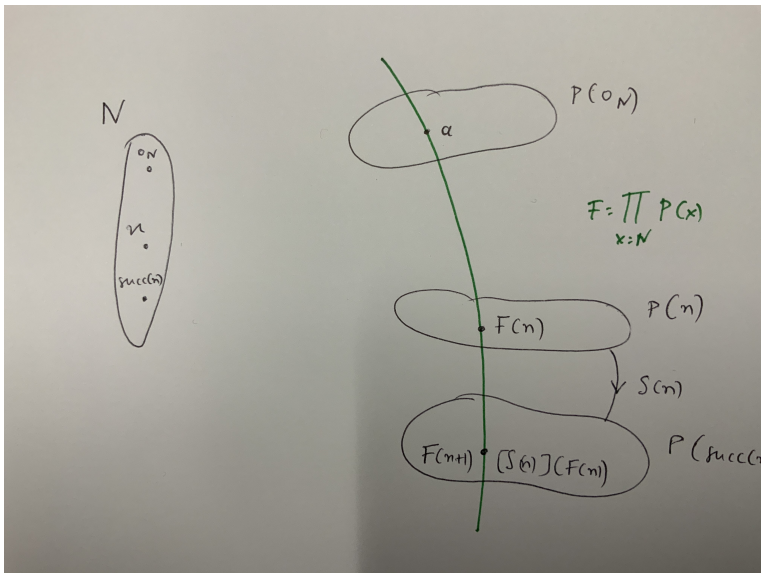


Figure 2.11: $\mathrm{Ind}_{\mathbf{N}}$

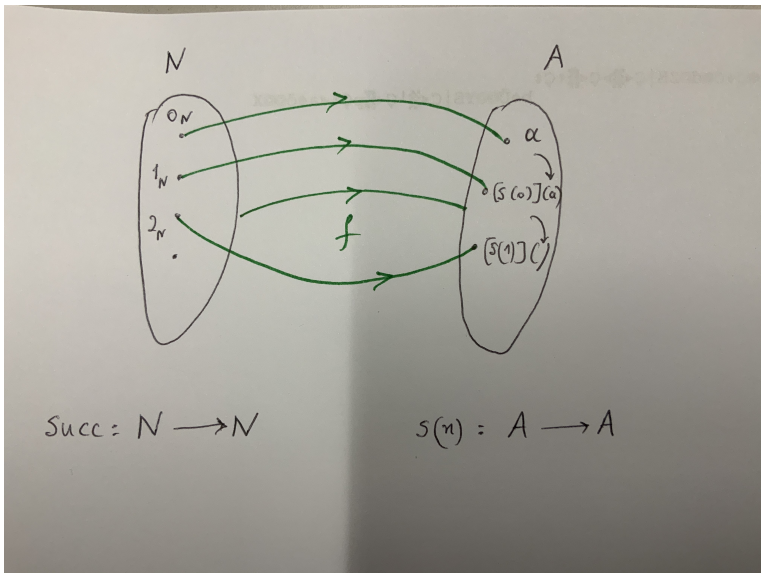


Figure 2.12: $\mathrm{Rec}_{\mathbf{N}}$

Example 2.12.2. *There is a function* $\mathrm{pred}\colon \mathbf{N} \to \mathbf{N}$ *satisfying the computation rules*

$$\mathrm{pred}(0_{\mathbf{N}}) \equiv 0_{\mathbf{N}},$$

$$\mathrm{pred}(\mathtt{succ}(n)) \equiv n.$$

Use $\mathrm{Rec}_{\mathbf{N}}$ *on* $s\colon \mathbf{N} \to \mathbf{N} \to \mathbf{N}$*, where* $s(m,n) \equiv m$*. Clearly,* $\mathrm{pred}(2) \equiv \mathrm{pred}(\mathtt{succ}(1)) \equiv 1$.

Example 2.12.3. There is a function $\mathrm{double}\colon \mathbf{N} \to \mathbf{N}$ satisfying the computation rules

$$\mathrm{double}(0_{\mathbf{N}}) \equiv 0_{\mathbf{N}},$$

$$\mathrm{double}(\mathtt{succ}(n)) \equiv \mathtt{succ}(\mathtt{succ}(\mathrm{double}(n))).$$

It is straightforward to show that $\mathrm{double}(2) \equiv 4$.

Example 2.12.4. There is a function $\mathrm{add}\colon \mathbf{N} \to \mathbf{N} \to \mathbf{N}$ satisfying the computation rules

$$\mathrm{add}(0_{\mathbf{N}}, n) \equiv n,$$

$$\mathrm{add}(\mathtt{succ}(m), n) \equiv \mathtt{succ}(\mathrm{add}(m,n)).$$

It is straightforward to show that $\mathrm{add}(2,2) \equiv 4$. Usually we write $m + n \equiv \mathrm{add}(m,n)$, where $m, n\colon \mathbf{N}$.

The types $\mathbf{N}$ and 1 are logically equivalent, but as expected, they will not be equivalent in an interesting sense. In other words, logical equivalence is not an interesting notion of equivalence for types. Actually, *any two inhabited types are logically equivalent*!

Remark 2.12.5. *There is a term* $(f,g)\colon \mathbf{N} \leftrightarrow \mathbf{1}$.

Proof. Exercise. □

If $P\colon \mathbf{N} \to \mathcal{U}$, then by $\mathrm{Ind}_{\mathbf{N}}$ to define a dependent function $F\colon \prod_{x\colon \mathbf{N}} P(x)$, it suffices to know terms $a\colon P(0_{\mathbf{N}})$ and a dependent function $S\colon \prod_{x\colon \mathbf{N}} \big(P(x) \to P(\mathtt{succ}(x))\big)$. Clearly, $\mathrm{Ind}_{\mathbf{N}}$ is the type-theoretic version of the set-theoretic induction principle for natural numbers that we mentioned in section 1.2.

Example 2.12.6. Let $\mathbf{N}'$ be the type of natural numbers in $\mathcal{U}'$. There is a type family $\mathrm{Fin}\colon \mathbf{N}' \to \mathcal{U}$ satisfying the computation rules

$$\mathrm{Fin}(0_{\mathbf{N}'}) \equiv \mathbf{1},$$

$$\mathrm{Fin}(\mathtt{succ}(n')) \equiv \mathbf{1} + \mathrm{Fin}(n').$$

If one defines a term

$$S\colon \prod_{x\colon \mathbf{N}'} \big(\mathrm{Fin}(x') \to \mathrm{Fin}(\mathtt{succ}(x'))\big),$$

one gets a dependent function

$$F\colon \prod_{x\colon \mathbf{N}'} \mathrm{Fin}(x')$$

with the help of $\mathrm{Ind}_{\mathbf{N}'}$.

Definition 2.12.7. *The successor function* $\mathtt{succ}\colon \mathbf{N} \to \mathbf{N}$ *is defined by* $\mathtt{succ} \equiv \lambda_{x\colon \mathbf{N}}.\mathtt{succ}(x)$.

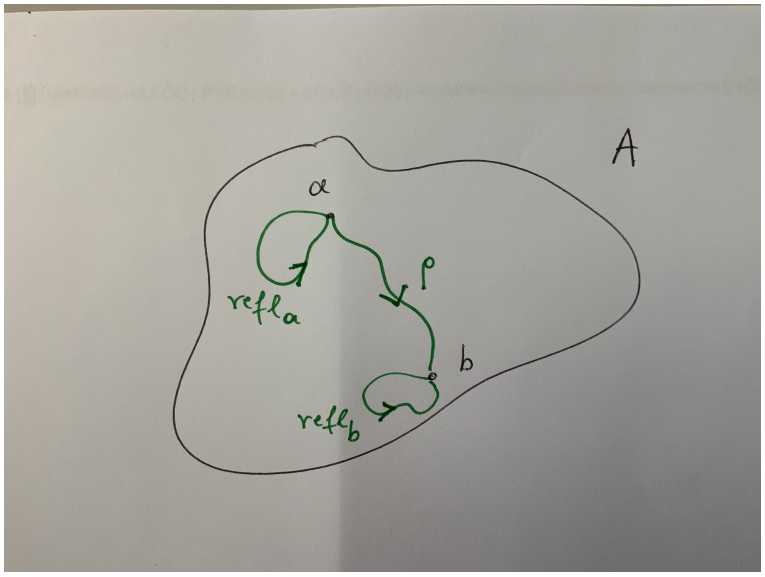


Figure 2.13: Terms of the equality type as paths

2.13 The equality type family

Next follows the type-theoretic version of the set-theoretic equality. The equality between terms a, b of a type A in a universe $\mathcal{U}$ is a new type

$$a =_A b$$

in $\mathcal{U}$, and its induction principle is the most important proof-tool of MLTT. We represent a term $p\colon a =_A b$ as a *path* from $a\colon A$ to $b\colon A$, while the introduced, canonical path $\mathtt{refl}_a\colon a =_A a$ as a *loop* at $a\colon A$. This representation is motivated by the homotopic interpretation of MLTT given by Voevodsky and discussed later in the course.

Definition 2.13.1. *If $A\colon \mathcal{U}$, the type family $=_A\colon A \to A \to \mathcal{U}$, where*

$$a =_A b \equiv =_A (a, b)\colon \mathcal{U},$$

is defined by the following rules. Notice that we use Greek letter $\pi, \varpi, \rho, \sigma$ for variables of type $x =_A y$ and Latin letters p, q, r, s for closed terms of the equality type.

Intro$_{=_A}$*:*

$$\mathtt{refl}_A\colon \prod_{x\colon A} x =_A x.$$

If $a\colon A$, we denote for simplicity the object $\mathtt{refl}_A(a)\colon a =_A a$ by $\mathtt{refl}_a$.

Ind$_{=_A}$*:*

$$\frac{Q\colon \prod_{x,y\colon A} \prod_{\pi\colon x =_A y} \mathcal{U}, \quad R\colon \prod_{x\colon A} Q(x, x, \mathtt{refl}_x)}{\widehat{R}\colon \prod_{x,y\colon A} \prod_{\pi\colon x =_A y} Q(x, y, \pi),\ \widehat{R}(a, a, \mathtt{refl}_a) \equiv R(a)}\ .$$

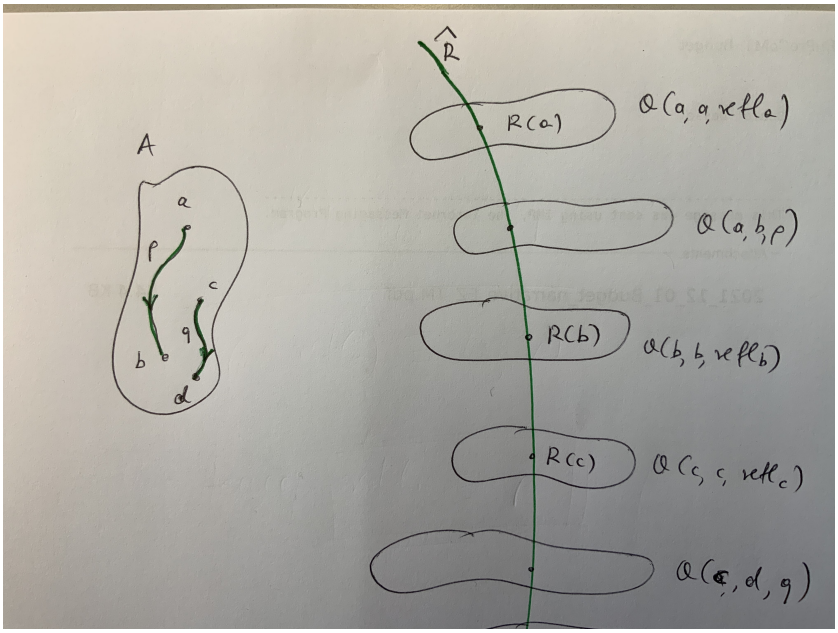


Figure 2.14: The J-rule

The type $a =_A b$ is also called the *identity type* from a to b, and the induction principle $\mathrm{Ind}_{=_A}$ is also called Martin-Löf's *J-rule*. The J-rule expresses an *extension-property*, since, according to it, if

$$Q\colon \prod_{x,y\colon A} \prod_{\pi\colon x =_A y} \mathcal{U} \equiv \prod_{x,y\colon A} (x =_A y \to \mathcal{U}),$$

together with a term

$$R\colon \prod_{x\colon A} Q(x, x, \mathtt{refl}_x)$$

are given, then there is a dependent function

$$\widehat{R}\colon \prod_{x,y\colon A} \prod_{\pi\colon x =_A y} Q(x, y, \pi)$$

that *extends* R in the sense that for every $a\colon A$

$$\widehat{R}(a, a, \mathtt{refl}_a) \equiv R(a).$$

The notation $\widehat{R}$ does not indicate a unique term of that type that satisfies the above computation rule, although, if the axiom of function extensionality is added to MLTT, then uniqueness of extension, up to the corresponding equality, is provable (see Proposition 3.1.4). We can package the induction principle $\mathrm{Ind}_{=_A}$ into a single function

$$J_A \equiv \mathrm{Ind}_{=_A} : \prod_{\Phi\colon \prod_{x,y\colon A} \prod_{\pi\colon x =_A y} \mathcal{U}} \prod_{\Theta\colon \prod_{x\colon A} \Phi(x,x,\mathtt{refl}_x)} \left(\prod_{x,y\colon A} \prod_{\pi\colon x =_A y} \Phi(x, y, \pi) \right)$$

with the defining equation

$$J_A(Q, R, a, a, \mathtt{refl}_a) \equiv R(a).$$

The recursion rule corresponding to the equality types of $A\colon \mathcal{U}$ takes the following form.

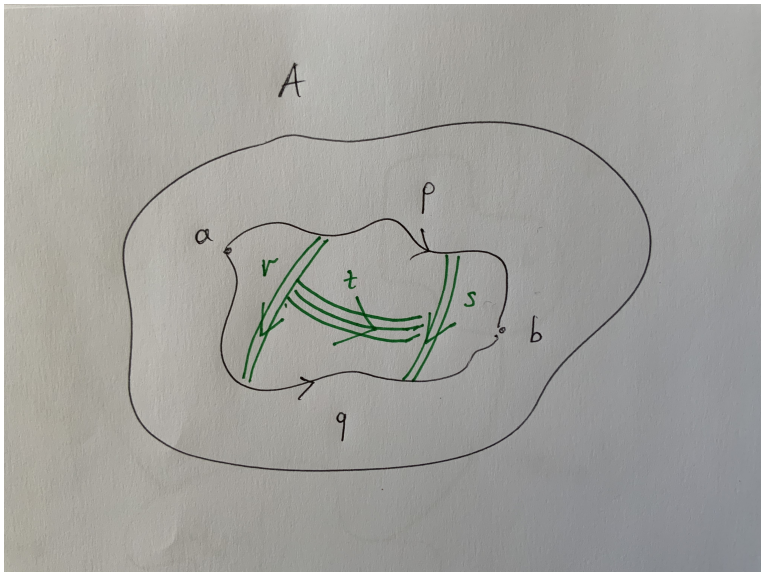


Figure 2.15: The infinity structure of the equality types

Corollary 2.13.2 ($\mathrm{Rec}_{=_A}$)**.** *Let $A, B : \mathcal{U}$ and $f : A \to B$. There is a term*

$$\widehat{f} : \prod_{x,y:\, A} \prod_{\pi:\, x =_A y} B \equiv \prod_{x,y:\, A} (x =_A y \to B)$$

such that $\widehat{f}(a, a, \mathtt{refl}_a) \equiv f(a)$, for every $a : A$.

Proof. Let

$$Q : \prod_{x,y:\, A} \prod_{\pi:\, x =_A y} \mathcal{U} \equiv \prod_{x,y:\, A} \big(x =_A y \to \mathcal{U}\big),$$

defined by $Q(a, b, p) \equiv B$ i.e.,

$$Q \equiv \lambda_{x:\, A} \lambda_{y:\, A} \lambda_{\pi:\, x =_A y}.B.$$

Moreover, let

$$R : \prod_{x:\, A} Q(x, x, \mathtt{refl}_x) \equiv \prod_{x:\, A} B \equiv A \to B,$$

defined by $R \equiv f$. By the J-rule there is a term $\widehat{f}$ with $\widehat{f}(a, a, \mathtt{refl}_a) \equiv R(a) \equiv f(a)$. □

Notice that the above rule follows trivially without the use of the J-rule, hence it is not interesting. As $a =_A b$ is a type in $\mathcal{U}$, if $p, q : a =_A b$, then we have the new type

$$p =_{a =_A b} q.$$

This procedure of forming new equality types can be repeated infinitely many times. If $r, s : p =_{a =_A b} q$, we get the new type

$$r =_{p =_{a =_A b} q} s.$$

Although at the moment we cannor say, if the terms $p, q : a =_A b$ are always equal or not, we can associate a family of equality types as a bove to every type $A : \mathcal{U}$. The next chapter is devoted in the understanding of the equality type family and to the study of its implications.

Chapter 3

General properties of the equality type-family

In this chapter we present applications of the J-rule to the study of the general equality type family $=_A \colon A \to A \to \mathcal{U}$. The general "technique" of applying the J-rule in a proof is the following:

Step 1: Create a type $Q(a, b, p)$ out of the given data $A \colon U$, $a, b \colon A$, $p \colon a =_A b$, and the property to be proven i.e., **translate what is to be proven into a type** $Q(a, b, p)$.

Step 2: Define through Step 1 the term

$$Q \colon \prod_{x,y \colon A} \prod_{\pi \colon x =_A y} \mathcal{U} \equiv \prod_{x,y \colon A} (x =_A y \to \mathcal{U}).$$

Step 3: Define the term

$$R \colon \prod_{x \colon A} Q(x, x, \mathtt{refl}_x)$$

by unfolding $Q(x, x, \mathtt{refl}_x)$. I most cases the definition of $R(x)$ is straightforward.

Step 4: Apply the J-rule on Q and R to get the required term

$$\widehat{R} \colon \prod_{x,y \colon A} \prod_{\pi \colon x =_A y} Q(x, y, \pi)$$

with the computation rule $\widehat{R}(a, a, \mathtt{refl}_a) \equiv R(a)$.

3.1 Transport and least reflexive relation

Next we show that two equal terms of a type have the same properties. This result is also known as *indiscernability of identicals*, and it expresses the fact that if $a =_A b$, and $a \colon A$ satisfies property P i.e., the type $P(a)$ is inhabited, then $b \colon A$ satisfies P too i.e., the type $P(b)$ is also inhabited.

Proposition 3.1.1 (Transport)**.** *If $A \colon U$, $a, b \colon A$, $p \colon a =_A b$, and $P \colon A \to \mathcal{U}$, there is a function $p_* \colon P(a) \to P(b)$ such that, if $p \equiv \mathtt{refl}_a$, then $(\mathtt{refl}_a)_* \equiv \mathrm{id}_{P(a)}$.*

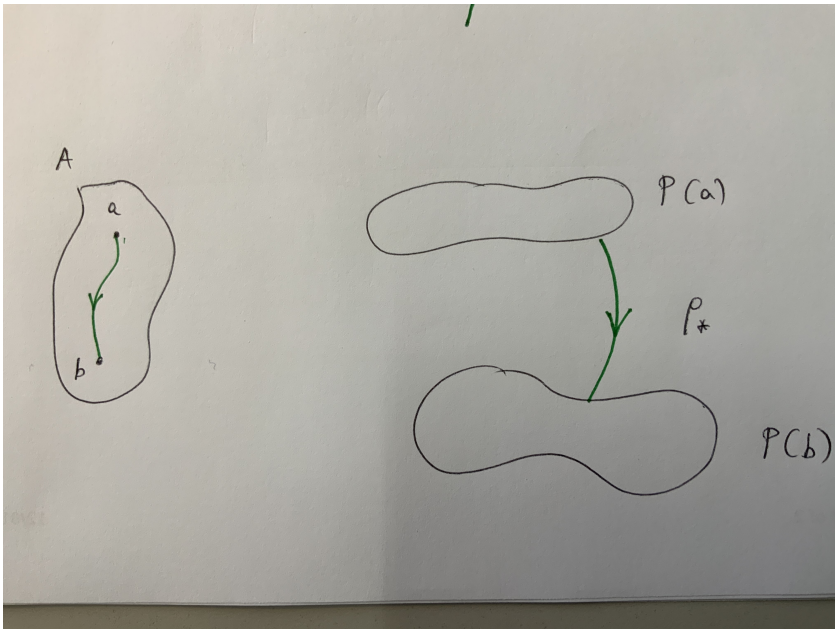


Figure 3.1: The transport $p_*\colon P(a) \to P(b)$ of an equality term $p\colon a =_A b$

Proof. Let

$$Q\colon \prod_{x,y\colon A} \prod_{\pi\colon x =_A y} \mathcal{U} \equiv \prod_{x,y\colon A} \big(x =_A y \to \mathcal{U}\big),$$

defined by

$$Q(a,b,p) \equiv P(a) \to P(b)$$

i.e.,

$$Q \equiv \lambda_{x\colon A}\lambda_{y\colon B}\lambda_{\pi\colon x =_A y}.\big(P(x) \to P(y)\big).$$

Let

$$R\colon \prod_{x\colon A} Q(x,x,\mathtt{refl}_x),$$

defined by

$$R(a) \equiv \mathrm{id}_{P(a)}\colon C(a,a,\mathtt{refl}_a) \equiv P(a) \to P(a)$$

i.e.,

$$R \equiv \lambda_{x\colon A}.\mathrm{id}_{P(x)}.$$

By the J-rule there is

$$\widehat{R}\colon \prod_{x,y\colon A} \prod_{\pi\colon x =_A y} (P(x) \to P(y))$$

such that

$$\widehat{R}(a,a,\mathtt{refl}_a) \equiv \mathrm{id}_{P(a)}.$$

We define $p_* \equiv \widehat{R}(a,b,p)$. Hence, if $p \equiv \mathtt{refl}_a$, we get $(\mathtt{refl}_a)_* \equiv \widehat{R}(a,a,\mathtt{refl}_a) \equiv \mathrm{id}_{P(a)}$. □

Hence, if $a =_A b$ and $P(a)$ holds i.e., there is a term $u\colon P(a)$, then $P(b)$ also holds, since $p_*(u)\colon P(b)$. If necessary, we write p^P_*, instead of p_*, to denote the transport of the equality term $p\colon a =_A b$ with respect to the type family P over $A\colon \mathcal{U}$. We can package the above result in a single term as follows:

$$\mathtt{Transport}_A : \prod_{\Pi\colon A\to\mathcal{U}} \prod_{x,y\colon A} \prod_{\pi\colon x=_A y} \big(\Pi(x) \to \Pi(y)\big)$$

$$\mathtt{Transport}_A(P, a, a, \mathtt{refl}_a) \equiv \mathrm{id}_{P(a)}, \quad a\colon A.$$

Next we show that the J-rule implies that the type family

$$=_A\colon A \to A \to \mathcal{U}$$

is the least reflexive relation on A i.e., a relation of the form

$$R\colon A \to A \to \mathcal{U}.$$

Proposition 3.1.2 (Least reflexive relation)**.** *Let $A\colon \mathcal{U}$ and $R\colon A \to A \to \mathcal{U}$ a reflexive relation on A i.e., there is a term*

$$\mathtt{refl}_R\colon \prod_{x\colon A} R(x, x).$$

Then there is a term

$$\widehat{\mathtt{refl}_R}\colon \prod_{x,y\colon A} \prod_{\pi\colon x=_A y} R(x, y) \equiv \prod_{x,y\colon A} \big(x =_A y \to R(x, y)\big).$$

such that $\widehat{\mathtt{refl}_R}(a, a, \mathtt{refl}_a) \equiv \mathtt{refl}_R(a)$, for every $a\colon A$.

Proof. Let the term

$$Q\colon \prod_{x,y\colon A} \prod_{\pi\colon x=_A y} \mathcal{U},$$

defined by

$$Q(a, b, p) \equiv R(a, b); \quad a\colon A, b\colon A, p\colon a =_A b.$$

Let

$$\mathtt{refl}_R\colon \prod_{x\colon A} Q(x, x, \mathtt{refl}_x) \equiv \prod_{x\colon A} R(x, x).$$

By the J-rule there is

$$\widehat{\mathtt{refl}_R}\colon \prod_{x,y\colon A} \prod_{\pi\colon x=_A y} Q(x, y, \pi) \equiv \prod_{x,y\colon A} R(x, y),$$

such that $\widehat{\mathtt{refl}_R}(a, a, \mathtt{refl}_a) \equiv \mathtt{refl}_R(a)$. □

We can package the above result in a single term as follows:

$$\mathtt{LeastRefl}_A\colon \prod_{\Sigma\colon A\to A\to\mathcal{U}} \prod_{\sigma\colon \prod_{x\colon A}\Sigma(x,x)} \prod_{x,y\colon A} \prod_{\pi\colon x=_A y} \Sigma(x, y),$$

$$\mathtt{LeastRefl}_A(R, \mathtt{refl}_R, a, a, \mathtt{refl}_a) \equiv \mathtt{refl}_R(a), \quad a\colon A.$$

Proposition 3.1.3. *The following are equivalent:*
(i) *There is a term*

$$\mathtt{Transport}_A : \prod_{\Pi\colon A\to\mathcal{U}} \prod_{x,y\colon A} \prod_{\pi\colon x=_A y} \big(\Pi(x) \to \Pi(y)\big)$$

such that $\mathtt{Transport}_A(P, a, a, \mathtt{refl}_a) \equiv \mathrm{id}_{P(a)}$, *for every* $a\colon A$.
(ii) *There is a term*

$$\mathtt{LeastRefl}_A\colon \prod_{\Sigma\colon A\to A\to\mathcal{U}} \prod_{\sigma\colon \prod_{x\colon A}\Sigma(x,x)} \prod_{x,y\colon A} \prod_{\pi\colon x=_A y} \Sigma(x, y),$$

such that $\mathtt{LeastRefl}_A(R, \mathtt{refl}_R, a, a, \mathtt{refl}_a) \equiv \mathtt{refl}_R(a)$, *for every* $a\colon A$.

Proof. Exercise. □

Proposition 3.1.4 (Pointwise equality of extensions)**.** *Let terms*

$$Q\colon \prod_{x,y\colon A} \prod_{\pi\colon x=_A y} \mathcal{U} \equiv \prod_{x,y\colon A} (x =_A y \to \mathcal{U}),$$

$$R\colon \prod_{x\colon A} Q(x, x, \mathtt{refl}_x).$$

If there are dependent functions

$$F, G\colon \prod_{x,y\colon A} \prod_{\pi\colon x=_A y} Q(x, y, \pi),$$

such that

$$F(x, x, \mathtt{refl}_x) \equiv R(x) \equiv G(x, x, \mathtt{refl}_x),$$

for every $x\colon A$, *then*

$$\prod_{x,y\colon A} \prod_{\pi\colon x=_A y} F(x, y, \pi) =_{Q(x,y,\pi)} G(x, y, \pi).$$

Proof. Exercise. □

One needs the axiom of function extensionality to conclude that

$$F =_{\prod_{x,y\colon A}\prod_{\pi\colon x=_A y} Q(x,y,\pi)} G.$$

3.2 The uniqueness-rule associated to the equality type-family

Let the term

$$Q\colon \prod_{x\colon A} \prod_{\pi\colon x=_A x} \mathcal{U} \equiv \prod_{x\colon A} (x =_A x \to \mathcal{U}),$$

defined by

$$Q(x, \pi) \equiv \pi =_{x=_A x} \mathtt{refl}_x.$$

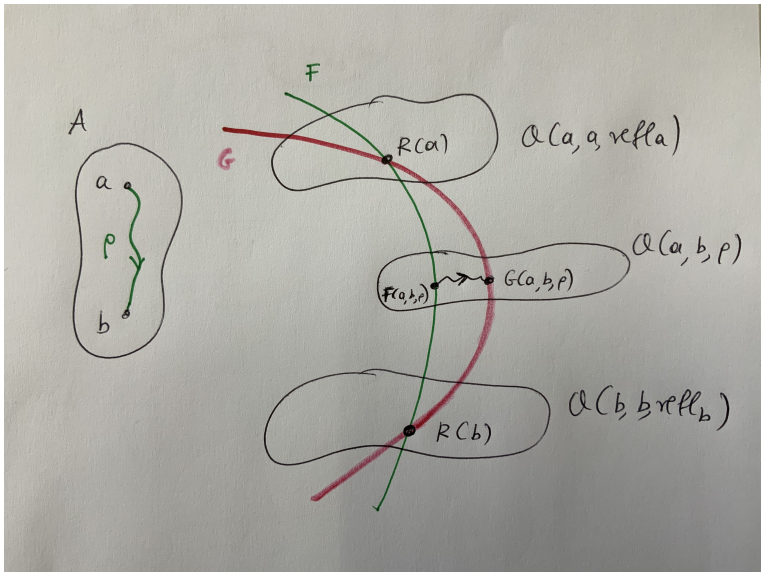


Figure 3.2: The pointwise uniqueness of two extensios of R

Let also the term

$$R\colon \prod_{x\colon A} Q(x, \mathtt{refl}_x) \equiv \prod_{x\colon A} \big(\mathtt{refl}_x =_{x=_A x} \mathtt{refl}_x\big),$$

defined by

$$R(x) \equiv \mathtt{refl}_{\mathtt{refl}_x}.$$

The J-rule does not imply (why?) from the existence of the term R the existence of a term

$$F\colon \prod_{x\colon A} \prod_{\pi\colon x=_A x} \big(\pi =_{x=_A x} \mathtt{refl}_x\big)$$

i.e., we cannot prove (in this way) that all terms of type $a =_A a$ are equal to $\mathtt{refl}_a$. Next follows the uniqueness-rule that corresponds to the equality type.

Proposition 3.2.1 (Uniqueness-rule of the equality type family)**.** *If $A\colon \mathcal{U}$ and $x\colon A$, let the singleton type*[1]

$$S_x \equiv \sum_{y\colon A} (x =_A y).$$

There is a term

$$\mathtt{Unique}_A\colon \prod_{x,y\colon A} \prod_{\pi\colon x=_A y} \big((x, \mathtt{refl}_x) =_{S_x} (y, \pi)\big),$$

such that for every $a\colon A$ we have that

$$\mathtt{Unique}_A\big(a, a, \mathtt{refl}_a)\big) \equiv \mathtt{refl}_{(a, \mathtt{refl}_a)}.$$

[1] if A is a set, and $a \in A$, the singleton $\{a\}$ can be defined as the set $\{a\} = \{x \in A \mid a = x\}$. In MLTT if $A\colon \mathcal{U}$ and $a\colon A$, the singleton type is the corresponding $\sum$-type

$$S_a \equiv \sum_{x\colon A} (a =_A x).$$

Proof. Let the term

$$Q\colon \prod_{x,y\colon A}\prod_{\pi\colon x=_A y}\mathcal{U} \equiv \prod_{x\colon A}\big(x =_A y \to \mathcal{U}\big),$$

defined by

$$Q(x,y,\pi) \equiv \big((x,\mathtt{refl}_x) =_{S_x} (y,\pi)\big).$$

Let also the term

$$R\colon \prod_{x\colon A} Q(x,x,\mathtt{refl}_x) \equiv \prod_{x\colon A}\big((x,\mathtt{refl}_x) =_{S_x} (x,\mathtt{refl}_x)\big),$$

defined by

$$R(x) \equiv \mathtt{refl}_{(x,\mathtt{refl}_x)}.$$

Then by the J-rule we get $\mathtt{Unique}_A \equiv \widehat{R}$. □

Corollary 3.2.2 (Contractibility of the singleton type)**.** *If $A\colon \mathcal{U}$ and $a\colon A$, let the singleton type*

$$S_a \equiv \sum_{y\colon A}(a =_A y).$$

There is a term

$$\mathtt{Unique}_a\colon \prod_{z\colon S_a}\big((a,\mathtt{refl}_a) =_{S_a} z\big),$$

such that

$$\mathtt{Unique}_a\big(a,\mathtt{refl}_a)\big) \equiv \mathtt{refl}_{(a,\mathtt{refl}_a)}.$$

Proof. To find such a dependent function over S_a we use the induction rule $\mathrm{Ind}_{\sum}$. Let $P_a\colon A \to \mathcal{U}$, defined by $P_a(y) \equiv (a =_A y)$, Clearly,

$$S_a \equiv \sum_{y\colon A} P_a(y).$$

Let $Q_a\colon S_a \to \mathcal{U}$, defined by

$$Q_a(z) \equiv \big((a,\mathtt{refl}_a) =_{S_a} z\big).$$

We define the term

$$G_a\colon \prod_{y\colon A}\prod_{\pi\colon P_a(y)} Q_a\big((y,\pi)\big) \equiv \prod_{y\colon A}\prod_{\pi\colon a=_A y}\big((a,\mathtt{refl}_a) =_{S_a} (y,\pi)\big)$$

by $G_a \equiv \mathtt{Unique}_A(a)$, where the term $\mathtt{Unique}_A$ is given in Proposition 3.2.1. By $\mathrm{Ind}_{\sum}$ there is a term

$$\mathtt{Unique}_a\colon \prod_{z\colon S_a} Q_a(z) \equiv \prod_{z\colon S_a}\big((a,\mathtt{refl}_a) =_{S_a} z\big),$$

such that

$$\mathtt{Unique}_a\big((b,p)\big) \equiv G_a(b,p) \equiv \mathtt{Unique}_A\big(a,(b,p)\big).$$

Hence

$$\mathtt{Unique}_a\big(a,\mathtt{refl}_a)\big) \equiv \mathtt{Unique}_A\big(a,(a,\mathtt{refl}_a)\big) \equiv \mathtt{refl}_{(a,\mathtt{refl}_a)}.$$

□

Although we cannot show that $\mathtt{refl}_a$ is equal in $a =_A a$ to an arbitrary term $p\colon a =_A a$, the *pair* $(a, \mathtt{refl}_a)$ id equal in S_a to the arbitrary pair (b, p), where $p\colon a =_A b$. This is the "uniqueness"-rule associated to the equality type.

Definition 3.2.3. *If $A\colon \mathcal{U}$, we call A contractible, if there is a term $a\colon A$, the center of contraction, such that*

$$\prod_{x\colon A} a =_A x.$$

According to Corollary 3.2.2, S_a is contractible with $(a, \mathtt{refl}_a)$ center of contraction.

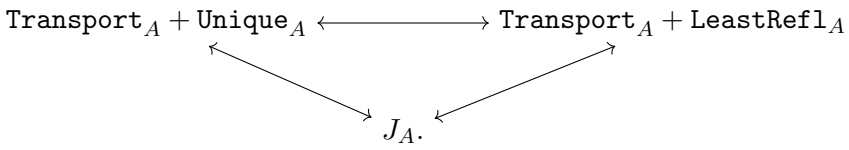
Corollary 3.2.4 (Equality reflection rule)**.** *Suppose that we add to* MLTT *the following equality reflection rule:*

$$\frac{A\colon \mathcal{U},\ a, b\colon A,\quad p\colon a =_A b}{a \equiv b}\ .$$

Then if $p\colon a =_A a$, we get $p \equiv \mathtt{refl}_a$.

Proof. Exercise. □

So far we have seen that the J-rule for $A\colon \mathcal{U}$ implies both $\mathtt{Transport}_A$ and $\mathtt{Unique}_A$. Following [10], we show the converse i.e., $\mathtt{Transport}_A$ and $\mathtt{Unique}_A$ (or $\mathtt{LeastRefl}_A$ and $\mathtt{Unique}_A$) imply the J-rule[2]. Combining this result with Proposition 3.1.3 we get the following diagram of equivalences:



Theorem 3.2.5 (Coquand)**.** *If $A\colon \mathcal{U}$, then terms $\mathtt{Transport}_A$ and $\mathtt{Unique}_A$, or equivalently $\mathtt{LeastRefl}_A$ and $\mathtt{Unique}_A$, together with their computation rules imply the J-rule for A together with its computation-rule.*

Proof. Let terms

$$Q\colon \prod_{x,y\colon A} \prod_{\pi\colon x =_A y} \mathcal{U} \equiv \prod_{x,y\colon A} (x =_A y \to \mathcal{U}),$$

$$R\colon \prod_{x\colon A} Q(x, x, \mathtt{refl}_x).$$

Let

$$G_x\colon \prod_{y\colon A} \big((x =_A y) \to \mathcal{U}\big),$$

defined by

$$G_x \equiv \lambda_{y\colon A} \lambda_{\pi\colon x =_A y}. Q(x, y, \pi)\colon \mathcal{U}.$$

If $S_x \equiv \sum_{y\colon A}(x =_A y)$, then by $\mathrm{Rec}_{\sum}$ there is a function $T_x\colon S_x \to \mathcal{U}$, such that

$$T_x\big((b, p)\big) \equiv G_x(b, p) \equiv Q(x, b, p).$$

[2]The proof in [10] uses the based version of the J-rule, which we avoid here.

By hypothesis we have that

$$\mathtt{Unique}_A(x,y,\pi)\colon (x,\mathtt{refl}_x) =_{S_x} (y,\pi),$$

hence by $\mathtt{Transport}_A$ we have that

$$\big[\mathtt{Unique}_A(x,y,\pi)\big]_*^{T_x}\colon T_x\big((x,\mathtt{refl}_x)\big) \to T_x\big((y,\pi)\big)$$

i.e.,

$$\big[\mathtt{Unique}_A(x,y,\pi)\big]_*^{T_x}\colon Q(x,x,\mathtt{refl}_x) \to Q(x,y,\pi).$$

Hence we define

$$\widehat{R}\colon \prod_{x,y\colon A}\prod_{\pi\colon x=_A y} Q(x,y,\pi)$$

by

$$\widehat{R} \equiv \lambda_{x\colon A}\lambda_{y\colon A}\lambda_{\pi\colon x=_A y}.\big[\mathtt{Unique}_A(x,y,\pi)\big]_*^{T_x}\big(R(x)\big).$$

By the computation rules associated to $\mathtt{Unique}_A$ and $\mathtt{Transport}_A$ we get

$$\begin{aligned}
\widehat{R}(a,a,\mathtt{refl}_a) &\equiv \big[\mathtt{Unique}_A(a,a,\mathtt{refl}_a)\big]_*^{T_a}\big(R(a)\big)\\
&\equiv \big[\mathtt{refl}_{(a,\mathtt{refl}_a)}\big]_*^{T_a}\big(R(a)\big)\\
&\equiv \mathrm{id}_{T_a\left((a,\mathtt{refl}_a)\right)}\big(R(a)\big)\\
&\equiv \mathrm{id}_{Q(a,a,\mathtt{refl}_a)}\big(R(a)\big)\\
&\equiv R(a).
\end{aligned}$$

□

3.3 The based version of the J-rule

In [28] Paulin-Mohring introduced the following based-version of the J-rule, which is the induction principle of the inductive definition of the type family $=_a\colon A \to \mathcal{U}$, where $a\colon A$ and $=_a(b) \equiv (a =_A b)\colon \mathcal{U}$, for every $a\colon A$, with introduction rule $\mathtt{refl}_a\colon a =_A a$.

Definition 3.3.1. *If $A\colon \mathcal{U}$ and $a\colon A$, the type family $=_a\colon A \to \mathcal{U}$, where*

$$a =_A b \equiv =_a(b)\colon \mathcal{U},$$

is defined by the following rules.

Intro$_{=_a}$:

$$\mathtt{refl}_a\colon a =_A a.$$

Ind$_{=_a}$:

$$\frac{Q_a\colon \prod_{x\colon A}\prod_{\pi\colon a=_A x}\mathcal{U}, \quad r\colon Q_a(a,\mathtt{refl}_a)}{\widehat{r}\colon \prod_{x\colon A}\prod_{\pi\colon a=_A x} Q_a(x,\pi),\ \widehat{r}(a,\mathtt{refl}_a) \equiv r}.$$

Ind$_{=_a} \equiv j_a$ *is the based version of Martin-Löf's J-rule. The j-rule is the term*

$$j\colon \prod_{y\colon A} Q_y\colon \prod_{x\colon A}\prod_{\pi\colon y=_A x}\mathcal{U}\ \sigma\colon Q_y(x,\mathtt{refl}_x)\ x\colon A\ \pi\colon y=_A x Q_y(x,\pi),$$

satisfying the computation-rule $j(a,Q_a,r,a,\mathtt{refl}_a) \equiv r$.

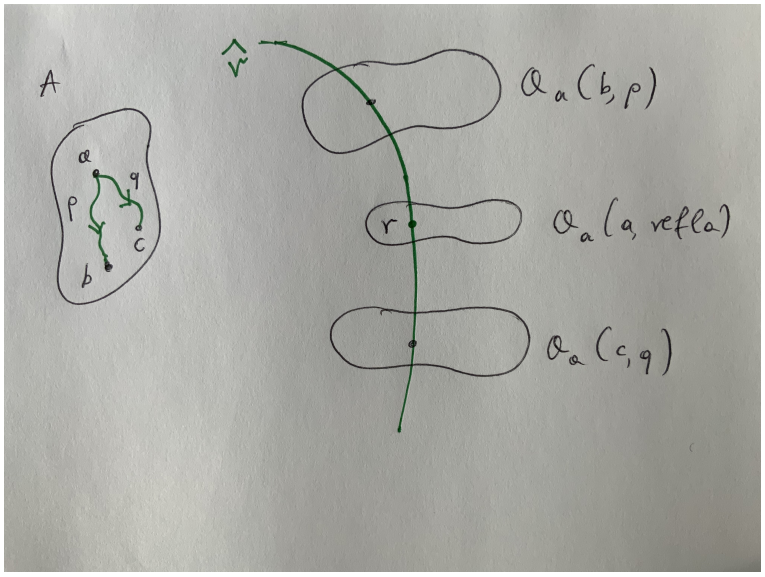


Figure 3.3: The j-rule

The j_a-rule expresses an *extension-property*, since, according to it, if

$$Q_a\colon \prod_{x\colon A}\prod_{\pi\colon a=_A x} \mathcal{U}$$

and $r\colon Q_a(a,\mathtt{refl}_a)$, then we can "extend" the term r to a dependent function

$$\widehat{r}\colon \prod_{x\colon A}\prod_{\pi\colon a=_A x} Q_a(x,\pi).$$

As in the case of the J-rule, the j-rule does not imply (why?) the existence of a term

$$F\colon \prod_{x\colon A}\prod_{\pi\colon x=_A x} (\pi =_{x=_A x} \mathtt{refl}_x).$$

Next follow the based-versions of $\mathtt{Transport}_A$, of $\mathtt{LeastRefl}_A$, and of $\mathtt{Unique}_A$.

Proposition 3.3.2. *Let $A\colon \mathcal{U}$, $a\colon A$, and let the singleton type $S_a \equiv \sum_{x\colon A} a =_A x$. The following are shown with the use of the based version of the J-rule:*

(i) *There is a term*

$$\mathtt{Transport}_a\colon \prod_{\Pi_a\colon A\to\mathcal{U}}\prod_{x\colon A}\prod_{\pi\colon a=_A x} (\Pi_a(a)\to\Pi_a(x)),$$

such that $\mathtt{Transport}_a(P_a,a,\mathtt{refl}_a)\equiv \mathrm{id}_{P_a(a)}$.

(ii) *There is a term*

$$\mathtt{LeastRefl}_a\colon \prod_{\Sigma_a\colon A\to\mathcal{U}}\prod_{x\colon \Sigma_a(a)}\prod_{x\colon A}\prod_{\pi\colon a=_A x} \Sigma_a(x),$$

such that $\mathtt{LeastRefl}_a(R_a,r,a,\mathtt{refl}_a)\equiv r$.

(iii) *There is a term*

$$\mathtt{Unique}_a \colon \prod_{z \colon S_a} (a, \mathtt{refl}_a) =_{S_a} z,$$

such that $\mathtt{Unique}_a\big((a, \mathtt{refl}_a)\big) \equiv \mathtt{refl}_{(a, \mathtt{refl}_a)}$.

Proof. Exercise. □

Proposition 3.3.3. *The j-rule implies the J_A-rule.*

Proof. Exercise. □

Theorem 3.3.4. *The J_A-rule implies the j-rule.*

Proof. As we have seen, the J_A-rule implies $\mathtt{Transport}_A$ and $\mathtt{Unique}_A$. It is straightforward to show that $\mathtt{Transport}_A$ implies $\mathtt{Transport}_a$ and $\mathtt{Unique}_A$ implies $\mathtt{Unique}_a$, for every $a \colon A$. Similarly to Theorem 3.2.5, the rules $\mathtt{Transport}_a$ and $\mathtt{Unique}_a$ imply the j_a-rule. Hence, the J_A-rule implies the j-rule. □

3.4 The equality type family is an equivalence relation

An equality relation must satisfy the properties of an equivalence relation. In this section we show that the equality type family $=_A \colon A \to A \to \mathcal{U}$ is, in type-theoretic terms, an equivalence relation.

Definition 3.4.1. *If $A \colon \mathcal{U}$, then $R \colon A \to A \to \mathcal{U}$ is called reflexive, if there is a term*

$$\mathtt{refl}_R \colon \prod_{x \colon A} R(x, x).$$

R is called symmetric, if there is a term

$$\mathtt{symm}_R \colon \prod_{x,y \colon A} \big(R(x, y) \to R(y, x)\big).$$

R is called transitive, if there is a term

$$\mathtt{trans}_R \colon \prod_{x,y,z \colon A} \big(R(x, y) \to R(y, z) \to R(x, z)\big).$$

Moreover, R is called an equivalence relation, if it is reflexive, symmetric and transitive.

Clearly, $=_A$ is reflexive, with $\mathtt{refl}_{=_A} \equiv \mathtt{refl}_A$.

Proposition 3.4.2. *Let $A \colon \mathcal{U}$.*

(i) *There is a term*

$$\mathtt{symm}_A \colon \prod_{x,y \colon A} \big(x =_A y \to y =_A x\big).$$

such that $\mathtt{symm}_A(a, a, \mathtt{refl}_a) \equiv \mathtt{refl}_a$.

(ii) *There is a term*

$$\mathtt{trans}_A \colon \prod_{x,y,z \colon A} \big(x =_A y \to y =_A z \to x =_A z\big).$$

such that $\mathtt{trans}_A(a, a, a, \mathtt{refl}_a, \mathtt{refl}_a) \equiv \mathtt{refl}_a$.

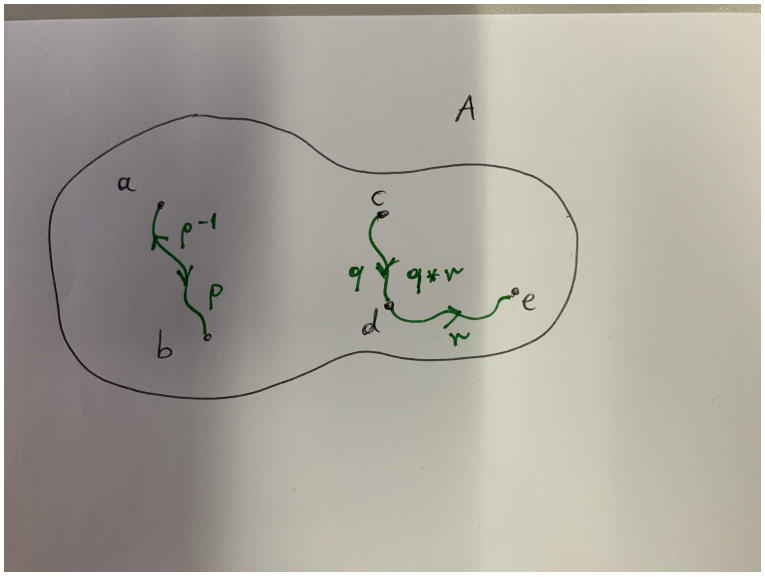


Figure 3.4: The inverse and the concatenation of equality terms

Proof. (i) Let the term $Q\colon \prod_{x,y\colon A}(x =_A y) \to \mathcal{U}$, defined by

$$Q(x, y, \pi) \equiv y =_A x.$$

As $\mathtt{refl}_x\colon Q(x, x, \mathtt{refl}_x) \equiv x =_A x$, by the J_A-rule we get the required term $\mathtt{symm}_A$. Clearly, $\mathtt{symm}_A(a, a, \mathtt{refl}_a) \equiv \mathtt{refl}_a$.
(ii) Let the term $Q'\colon \prod_{x,y\colon A}(x =_A y) \to \mathcal{U}$, defined by

$$Q'(x, y, \pi) \equiv \prod_{z\colon A}(y =_A z \to x =_A z).$$

Let

$$R\colon \prod_{x\colon A} Q'(x, x, \mathtt{refl}_x) \equiv \prod_{x\colon A}\prod_{z\colon A}\big(x =_A z \to x =_A z\big),$$

defined by $R(x) \equiv \lambda_{z\colon A}\lambda_{\pi\colon x=_A z}.\pi$ By the J_A-rule we get the required term

$$\mathtt{trans}_A \equiv \widehat{R}\colon \prod_{x,y\colon A} Q'(x, y, \pi) \equiv \prod_{x,y,z\colon A}\big(x =_A y \to y =_A z \to x =_A z\big),$$

satisfying, $\mathtt{trans}_A(a, a, a, \mathtt{refl}_a, \mathtt{refl}_a) \equiv [R(a)](\mathtt{refl}_a) \equiv \mathtt{refl}_a$. □

Definition 3.4.3. *If $A\colon \mathcal{U}$, $a, b, c, d, e\colon A$, $p\colon a =_A b$, $q\colon c =_A d$ and $r\colon d =_A e$, we use the following notations:*

$$p^{-1} \equiv \mathtt{symm}_A(a, b, p),$$
$$q * r \equiv \mathtt{trans}_A(c, d, e, q, r),$$

and we call $p^{-1}\colon b =_A a$ the inverse proof-term of p and $q * r\colon c =_A e$ the composite term, or the concatenation of q and r.

Hence, the computation-rules of $\mathtt{symm}_A$ and $\mathtt{trans}_A$ are rewritten as follows:

$$(\mathtt{refl}_a)^{-1} \equiv \mathtt{refl}_a \quad \& \quad \mathtt{refl}_a * \mathtt{refl}_a \equiv \mathtt{refl}_a.$$

3.5 The higher groupoid structure of a type

Proposition 3.5.1. *Let $A\colon \mathcal{U}$, $a,b,c,d\colon A$, $p\colon a =_A b$, $q\colon b =_A c$ and $r\colon c =_A d$.*
(i) $p * \mathtt{refl}_b =_{a=_A b} p =_{a=_A b} \mathtt{refl}_a * p$.
(ii) $p^{-1} * p =_{b=_A b} \mathtt{refl}_b$.
(iii) $p * p^{-1} =_{a=_A a} \mathtt{refl}_a$.
(iv) $(p^{-1})^{-1} =_{a=_A b} p$.
(v) $p * (q * r) =_{a=_A d} (p * q) * r$.
(vi) $(p * q)^{-1} =_{c=_A a} q^{-1} * p^{-1}$.

Proof. (i) Let $Q(x,y,\pi) \equiv \pi * \mathtt{refl}_y =_{x=_A y} \pi$. By the computation-rule of the term $\mathtt{trans}_A$ we have that

$$R(x) \equiv \mathtt{refl}_{\mathtt{refl}_x}\colon Q(x,x,\mathtt{refl}_x) \equiv \mathtt{refl}_x * \mathtt{refl}_x = \mathtt{refl}_x \equiv \mathtt{refl}_x = \mathtt{refl}_x.$$

We use the J_A-rule to get a term

$$\widehat{R}\colon \prod_{x,y\colon A} \prod_{\pi\colon x=_A y} \pi * \mathtt{refl}_y =_{x=_A y} \pi,$$

such that $\widehat{R}(a,a,\mathtt{refl}_a) \equiv \mathtt{refl}_{\mathtt{refl}_a}$.
(ii)-(vi) Exercise. □

Because of Proposition 3.5.1(i) we draw the equality term $\mathtt{refl}_a\colon a =_A a$ as the constant path a, and the loop-representation of $\mathtt{refl}_a$ in Figure 2.13 is not accurate. Because of Proposition 3.5.1(iii)-(iv) we draw the inverse equality term p^{-1} as the inverse of the path p (see Figure 3.4). Next we see how the transport of a proof term $p\colon a =_A b$ depends on the type family $P\colon A \to \mathcal{U}$, when P is defined through an equality type. Notice that the figures drawn suggest the behavior of these transports!

Proposition 3.5.2. *Let $A\colon \mathcal{U}$, $a,b,c\colon A$, $p\colon a =_A b$, and let the type families*

$$P\colon A \to \mathcal{U}, \quad P(x) \equiv c =_A x,$$

$$Q\colon A \to \mathcal{U}, \quad Q(x) \equiv x =_A c,$$

$$R\colon A \to \mathcal{U}, \quad R(x) \equiv x =_A x.$$

(i) $p^P_*\colon P(a) \to P(b) \equiv (c =_A a) \to (c =_A b)$ *is given by* $p^P_*(q) = q * p$.
(ii) $p^Q_*\colon Q(a) \to Q(b) \equiv (a =_A c) \to (b =_A c)$ *is given by* $p^Q_*(q) = p^{-1} * q$.
(iii) $p^R_*\colon R(a) \to R(b) \equiv (a =_A a) \to (b =_A b)$ *is given by* $p^R_*(q) = p^{-1} * q * p$.

Proof. (i) Let $T(x,y,\pi) \equiv \pi^P_*(\varpi) = \varpi * \pi$, where $\varpi\colon c =_A x$. Then, the type

$$\begin{aligned} T(x,x,\mathtt{refl}_x) &\equiv (\mathtt{refl}_x)^P_*(\varpi) = \varpi * \mathtt{refl}_x \\ &\equiv \mathrm{id}_{P(x)}(\varpi) = \varpi * \mathtt{refl}_x \\ &\equiv \varpi = \varpi * \mathtt{refl}_x \end{aligned}$$

is inhabited by Proposition 3.5.1(i) and the term $\mathtt{symm}_A$. Then we use the J-rule.
(ii) and (iii) Exercise. □

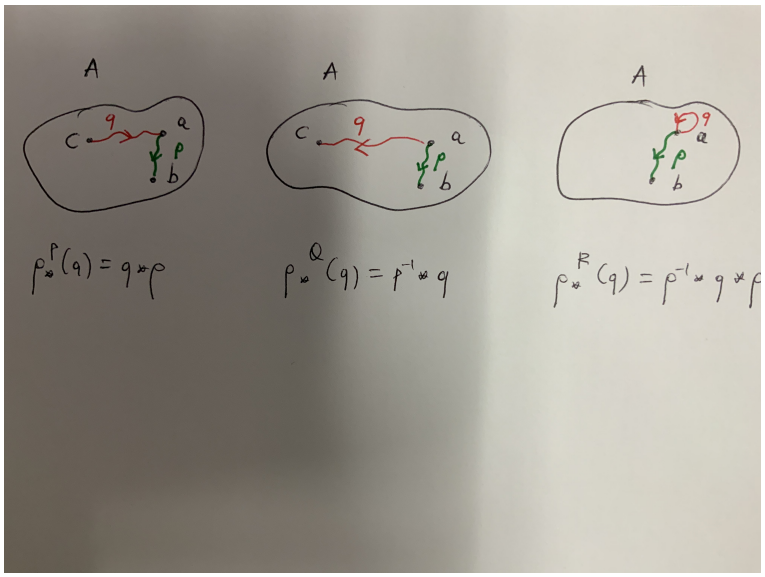


Figure 3.5: The calculation of basic transports w.r.t. equality

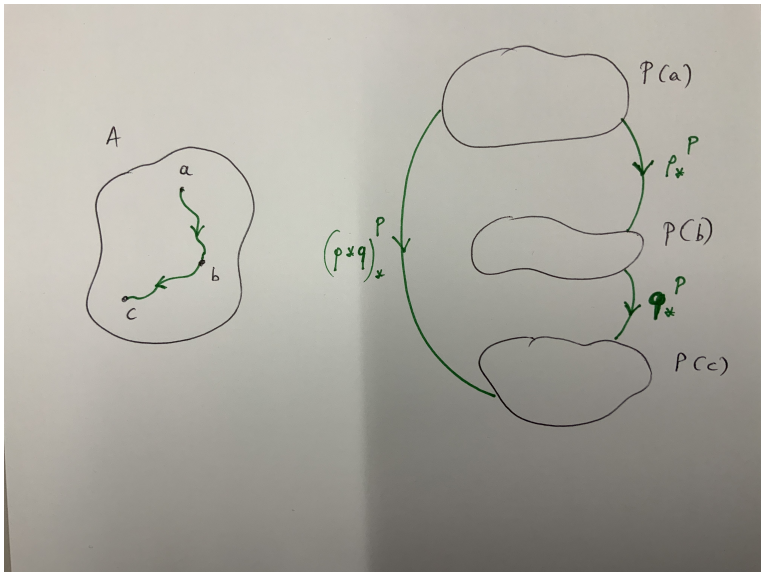


Figure 3.6: The transport of the concatenation "is" the composition of transports

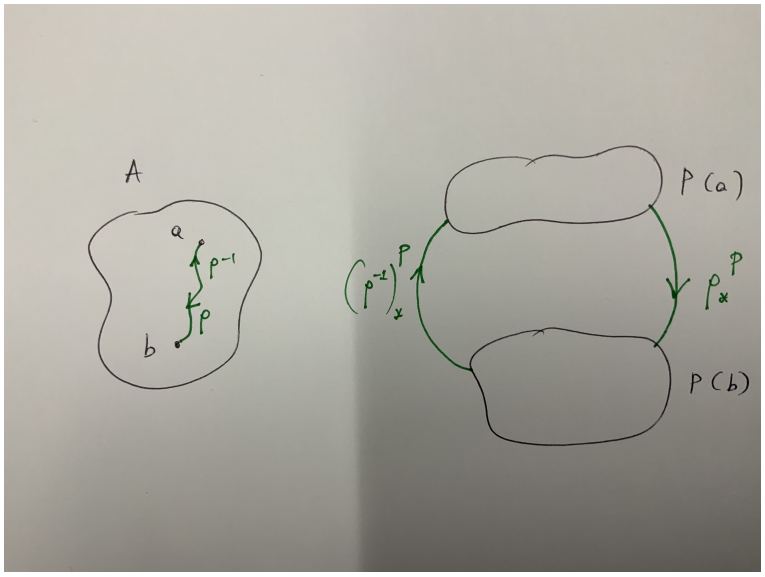


Figure 3.7: The transport of the inverse equality term "is" the inverse of the transport

Next we show how transport relates to concatenation of equality terms.

Proposition 3.5.3. *Let* $A\colon \mathcal{U}, a, b, c\colon A$, $p\colon a =_A b$ *and* $q\colon b =_A c$. *If* $P\colon A \to \mathcal{U}$, $u\colon P(a)$ *and* $v\colon P(b)$, *the following hold:*
(i) $(p * q)^P_*(u) =_{P(c)} \left[q^P_*\right]\left(p^P_*(u)\right)$.
(ii) $u =_{P(a)} \left[(p)^{-1})^P_*\right]\left(p^P_*(u)\right)$.
(iii) $v =_{P(b)} \left[\left(p^P_*\right]\left((p^{-1})^P_*(u)\right)\right)$.

Proof. Exercise. □

Proposition 3.5.4. *Let* $A\colon \mathcal{U}, a, b\colon A$, $p, q\colon a =_A b$ *and* $\boldsymbol{\alpha}\colon p =_A q$. *If* $P\colon A \to \mathcal{U}$, *then*

$$\prod_{u\colon P(a)} \left(p^P_*(u) =_{P(b)} q^P_*(u)\right).$$

Proof. Exercise. □

3.6 Loop spaces and the Eckmann-Hilton theorem

We give a flavor of the study of higher equality terms (paths). We also show that the concatenation of equal equality terms preserves equality. We use bold letters for higher equality terms.

Proposition 3.6.1 (Right horizontal composition of a 2-dimensional equality term with an 1-dimensional one)**.** *Let* $A\colon \mathcal{U}$, $a, b, c\colon A$, $p, q\colon a =_A b$, *and* $r\colon b =_A c$. *If* $\boldsymbol{\alpha}\colon p =_{a=_A b} q$, *there is a term*

$$\boldsymbol{\alpha} *_{\mathtt{right}} r\colon p * r =_{a=_A c} q * r.$$

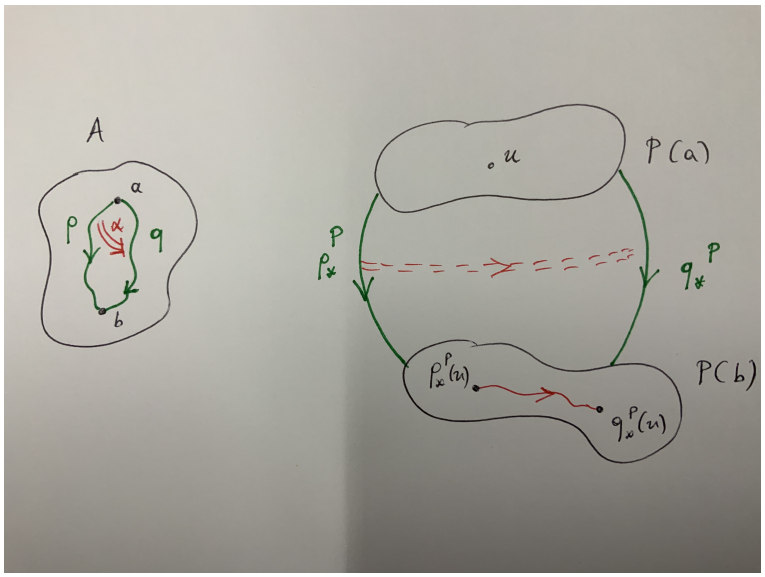


Figure 3.8: Equal equality terms induce "equal" transports

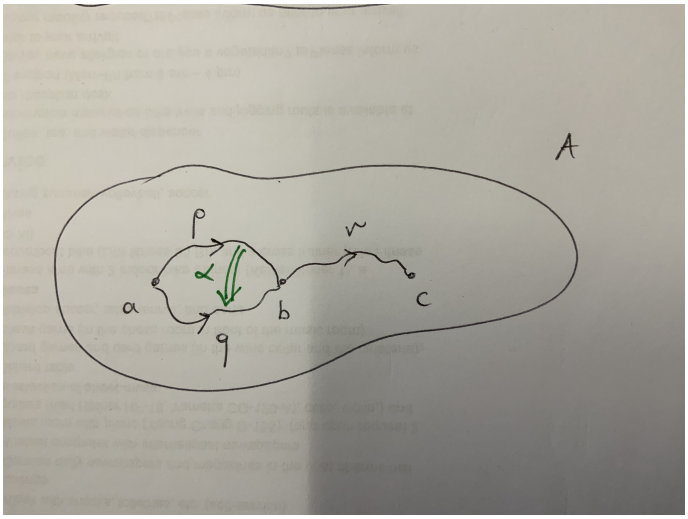


Figure 3.9: The right horizontal composition $\boldsymbol{\alpha} *_{\mathtt{right}} r \colon p * r =_{a =_A c} q * r$

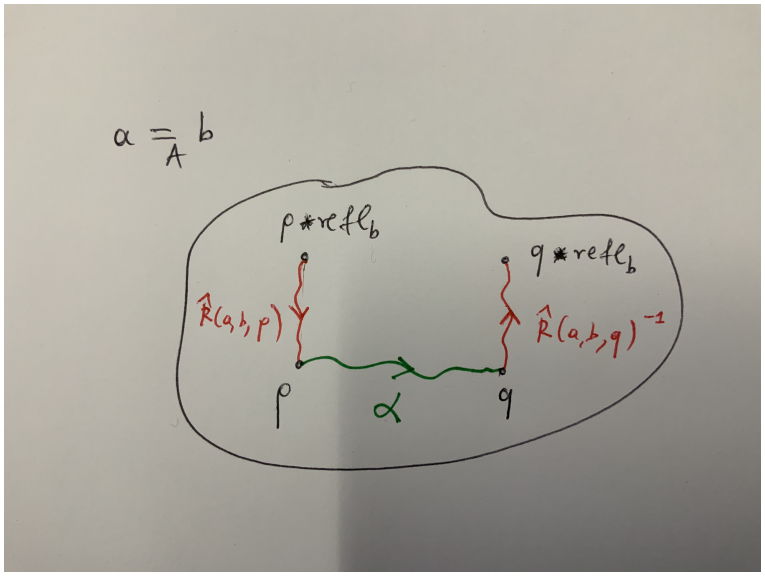


Figure 3.10: A proof term $t\colon p * \mathtt{refl}_b = q * \mathtt{refl}_b$

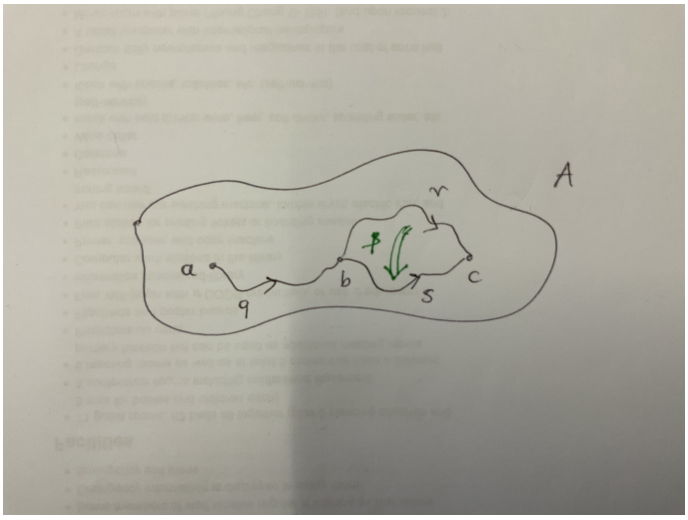


Figure 3.11: The left horizontal composition $q *_{\mathtt{left}} \boldsymbol{\beta}\colon q * r =_{a=_A c} q * s$

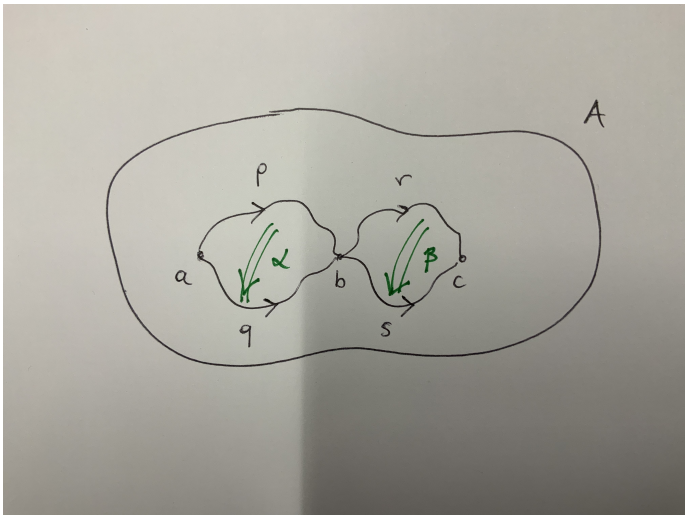


Figure 3.12: The horizontal composition $\boldsymbol{\alpha} \circledast \boldsymbol{\beta} \colon p * r = q * s$

Proof. We apply the based J-rule on $r \colon b =_A c$. Let $Q_b \colon \prod_{z \colon A} \prod_{\pi \colon b =_A z} \mathcal{U}$, defined by $Q_b(z, \pi) \equiv p * \pi = q * \pi$. We need to find a term of type

$$t \colon Q_b(b, \mathtt{refl}_b) \equiv p * \mathtt{refl}_b = q * \mathtt{refl}_b.$$

By Proposition 3.5.1(i) there are terms $\widehat{R}(a, b, p) \colon p * \mathtt{refl}_b = p$, and $\widehat{R}(a, b, q) \colon q * \mathtt{refl}_b = q$, hence we define $t \equiv \widehat{R}(a, b, p) * \boldsymbol{\alpha} * [\widehat{R}(a, b, q)]^{-1}$ (see Figure 3.10). □

Proposition 3.6.2 (Left horizontal composition of a 2-dimensional equality term with an 1-dimensional one)**.** *Let $A \colon \mathcal{U}$, $a, b, c \colon A$, $q \colon a =_A b$, and $r, s \colon b =_A c$. If $\boldsymbol{\beta} \colon r =_{a =_A b} s$, there is a term*

$$q *_{\mathtt{left}} \boldsymbol{\beta} \colon q * r =_{a =_A c} q * s.$$

Proof. We work similarly to the proof of Proposition 3.6.1, using the terms $\widehat{S}(b, c, r) \colon \mathtt{refl}_b * r = r$ and $\widehat{S}(b, c, s) \colon \mathtt{refl}_b * s = s$. □

Definition 3.6.3 (Horizontal compositions of 2-dimensional equality terms)**.** *Let $A \colon \mathcal{U}$, $a, b, c \colon A$, $q \colon a =_A b$, and $r, s \colon b =_A c$. If $\boldsymbol{\alpha} \colon p =_{a =_A b} q$ and $\boldsymbol{\beta} \colon r =_{a =_A b} s$, we define the horizontal compositions*

$$\boldsymbol{\alpha} \circledast \boldsymbol{\beta} \colon p * r = q * s \quad \textit{where} \quad p * r = q * r \ \&\ q * r = q * s,$$

$$\boldsymbol{\alpha} \circledast' \boldsymbol{\beta} \colon p * r = q * s \quad \textit{where} \quad p * r = p * s \ \&\ p * s = q * s,$$

by

$$\boldsymbol{\alpha} \circledast \boldsymbol{\beta} \equiv (\boldsymbol{\alpha} *_{\mathtt{right}} r) * (q *_{\mathtt{left}} \boldsymbol{\beta}),$$

$$\boldsymbol{\alpha} \circledast' \boldsymbol{\beta} \equiv (p *_{\mathtt{left}} \boldsymbol{\beta}) * (\boldsymbol{\alpha} *_{\mathtt{right}} s).$$

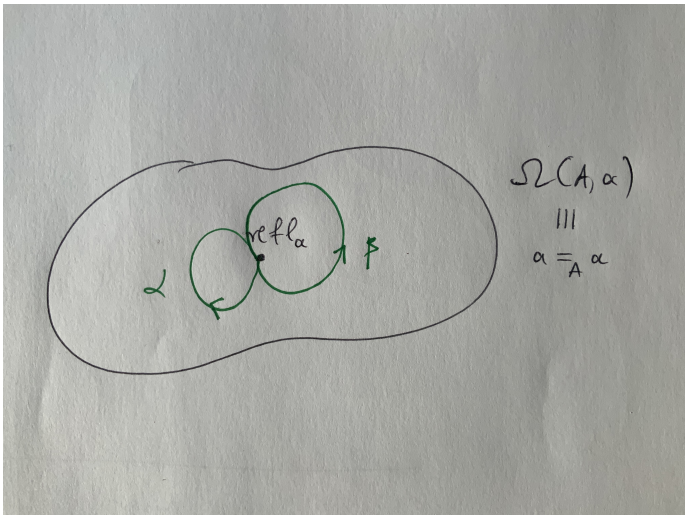


Figure 3.13: The composition $\boldsymbol{\alpha} * \boldsymbol{\beta} \colon \mathtt{refl}_a = \mathtt{refl}_a$ of $\boldsymbol{\alpha}, \boldsymbol{\beta} \colon \mathtt{refl}_a =_A \mathtt{refl}_a$

Lemma 3.6.4. *Let terms* $\boldsymbol{\alpha} \circledast \boldsymbol{\beta}, \boldsymbol{\alpha} \circledast' \boldsymbol{\beta} \colon p * r = q * s$ *as above, where* $a \equiv b \equiv c$ *and* $p \equiv q \equiv r \equiv s \equiv \mathtt{refl}_a$.

(i) $\boldsymbol{\alpha} \circledast \boldsymbol{\beta} = \boldsymbol{\alpha} * \boldsymbol{\beta}$.

(ii) $\boldsymbol{\alpha} \circledast' \boldsymbol{\beta} = \boldsymbol{\beta} \circledast \boldsymbol{\alpha}$.

(iii) $\boldsymbol{\alpha} \circledast \boldsymbol{\beta} = \boldsymbol{\alpha} \circledast' \boldsymbol{\beta}$.

Proof. (i) By the computation-rules of the terms $\widehat{R}$ and $\widehat{S}$ in the proof of Proposition 3.5.1(i) we get

$$\begin{aligned}
\boldsymbol{\alpha} \circledast \boldsymbol{\beta} &\equiv (\boldsymbol{\alpha} *_{\mathtt{right}} \mathtt{refl}_a) * (\mathtt{refl}_a *_{\mathtt{left}} \boldsymbol{\beta}) \\
&\equiv \big(\widehat{R}(a, a, \mathtt{refl}_a) * \boldsymbol{\alpha} * [\widehat{R}(a, a, \mathtt{refl}_a)]^{-1}\big) \\
&\quad * \big([\widehat{S}(a, a, \mathtt{refl}_a)]^{-1} * \boldsymbol{\beta} * \widehat{S}(a, a, \mathtt{refl}_a)\big) \\
&\equiv \big(\mathtt{refl}_{\mathtt{refl}_a} * \boldsymbol{\alpha} * [\mathtt{refl}_{\mathtt{refl}_a}]^{-1}\big) * \big(\mathtt{refl}_{\mathtt{refl}_a} * \boldsymbol{\beta} * [\mathtt{refl}_{\mathtt{refl}_a}]^{-1}\big) \\
&\equiv \big(\mathtt{refl}_{\mathtt{refl}_a} * \boldsymbol{\alpha} * \mathtt{refl}_{\mathtt{refl}_a}\big) * \big(\mathtt{refl}_{\mathtt{refl}_a} * \boldsymbol{\beta} * \mathtt{refl}_{\mathtt{refl}_a}\big) \\
&= \boldsymbol{\alpha} * \boldsymbol{\beta}.
\end{aligned}$$

(ii) We work similarly to (i).

(iii) By induction on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. □

Definition 3.6.5. *Let* $A \colon \mathcal{U}$, *and* $a \colon A$. *The pair* (A, a) *is called a pointed type, and* a *is its base-point. The loop space of* (A, a) *is the pointed type*

$$\Omega(A, a) \equiv \big(a =_A a, \mathtt{refl}_a\big)$$

of the loops at $a \colon A$. *If* $n \colon \mathbf{N}$, *we define recursively the* n*-loop space* $\Omega^n(A, a)$ *as follows:*

$$\Omega^0(A, a) \equiv (A, a),$$

$$\Omega^{n+1}(A, a) \equiv \Omega^n\big(\Omega(A, a)\big).$$

Hence, we have that

$$\Omega^1(A,a) \equiv \Omega^0\big(\Omega(A,a)\big) \equiv \Omega(A,a),$$

and

$$\begin{aligned}\Omega^2(A,a) &\equiv \Omega^1\big(\Omega(A,a)\big)\\ &\equiv \Omega\big(\Omega(A,a)\big)\\ &\equiv \Omega\big(a =_A a, \mathtt{refl}_a\big)\\ &\equiv \big(\mathtt{refl}_a =_{a=_A a} \mathtt{refl}_a, \mathtt{refl}_{\mathtt{refl}_a}\big).\end{aligned}$$

Clearly, the concatenation of loops is a loop i.e., using the same symbol $*$ for simplicity, there is a function

$$*\colon \Omega(A,a) \times \Omega(A,a) \to \Omega(A,a),$$

$$(\boldsymbol{\alpha}, \boldsymbol{\beta}) \mapsto \boldsymbol{\alpha} * \boldsymbol{\beta}.$$

Theorem 3.6.6 (Eckmann-Hilton theorem)**.** *Let $A\colon \mathcal{U}$, and $a\colon A$. If $\boldsymbol{\alpha}, \boldsymbol{\beta}\colon \Omega^2(A,a)$, then*

$$\boldsymbol{\alpha} * \boldsymbol{\beta} = \boldsymbol{\beta} * \boldsymbol{\alpha}.$$

Proof. It follows immediately from Lemma 3.6.4. □

Chapter 4

Equality and basic types

In this chapter we examine how the equality type family $=_A : A \to A \to \mathcal{U}$ interacts with the basic types of MLTT, other than the equality type-family itself and the universes.

4.1 Equality and the function type

Functions preserve equality in the following proof-relevant sense.

Proposition 4.1.1. *Let $A, B : \mathcal{U}$ and $f : A \to B$. There is a term*

$$\mathtt{ap}_f : \prod_{x,y:\, A} x =_A y \to f(x) =_B f(y),$$

such that $\mathtt{ap}_f(a, a, \mathtt{refl}_a) \equiv \mathtt{refl}_{f(a)}$, for every $a : A$. If $a, b : A$ are fixed we may also write

$$\mathtt{ap}_f : a =_A b \to f(a) =_B f(b),$$

$$p \mapsto f(p) \equiv \mathtt{ap}_f(p),$$

with $f(\mathtt{refl}_a) \equiv \mathtt{ap}_f(\mathtt{refl}_a) \equiv \mathtt{refl}_{f(a)}$, for every $a : A$.

Proof. Let the term $Q : \prod_{x,y:\, A} (x =_A y) \to \mathcal{U}$, defined by $Q(x, y, \pi) \equiv f(x) =_B f(y)$. Clearly,

$$\mathtt{refl}_{f(x)} : Q(x, x, \mathtt{refl}_x) \equiv f(x) =_B f(x).$$

The required term and its computation rule follow immediately from the J-rule. □

Hence, $\mathtt{ap}_f$ is "applied" to equality terms of type $a =_A b$ and outputs equality terms of type $f(a) =_B f(b)$. The application of a function can be iterated as follows.

Definition 4.1.2. *If $A, B : \mathcal{U}, a, b : A$ and $f : A \to B$. As $\mathtt{ap}_f : a =_A b \to f(a) =_B f(b)$, if $p, q : a =_A b$, we get the application of $\mathtt{ap}_f$, or the second-application function of f i.e.,*

$$\mathtt{ap}_f^2 \equiv \mathtt{ap}_{\mathtt{ap}_f} : p =_{a=_A b} q \to f(p) =_{f(a)=_B f(b)} f(q),$$

$$r \mapsto f^2(r) \equiv \mathtt{ap}_f^2(r),$$

such that $f^2(\mathtt{refl}_p) \equiv \mathtt{ap}_f^2(\mathtt{refl}_p) \equiv \mathtt{refl}_{f(p)}$, for every $p : a =_A b$. Actually, we have that

$$\mathtt{ap}_f^2 : \prod_{x,y:\, A} \prod_{p,q:\, x=_A y} p =_{a=_A b} q \to f(p) =_{f(a)=_B f(b)} f(q),$$

such that $\mathtt{ap}_f^2(a, b, p, p, \mathtt{refl}_p) \equiv \mathtt{refl}_{f(p)}$.

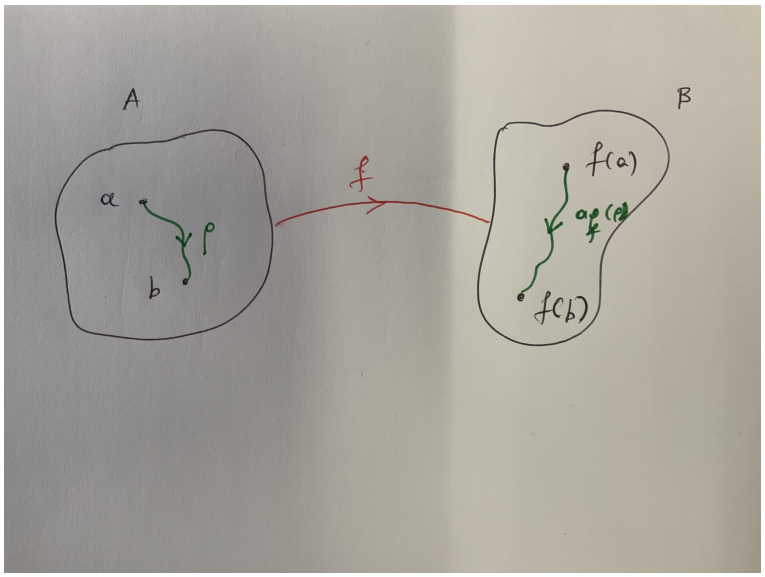


Figure 4.1: If $p\colon a =_A b$, then $\mathtt{ap}_f(p)\colon f(a) =_B f(b)$

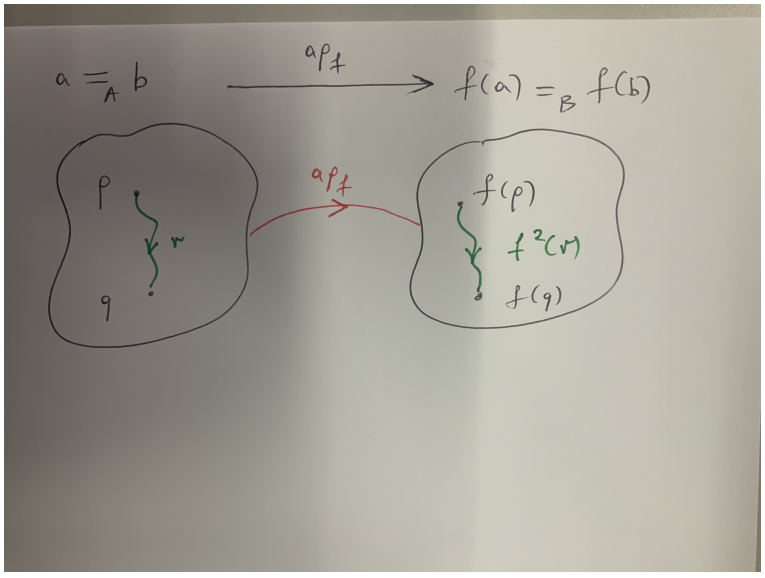


Figure 4.2: The second-application function of $f\colon A \to B$

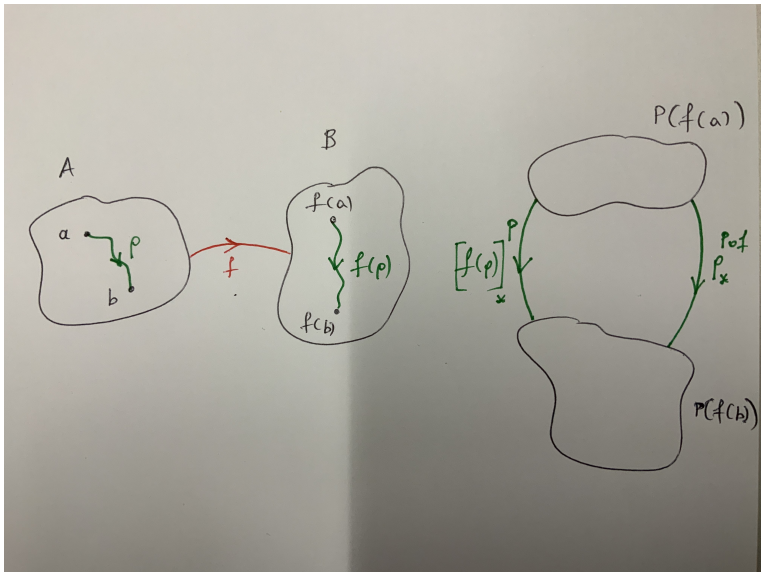


Figure 4.3: The transport $p_*^{P \circ f}$

Proposition 4.1.3. *Let $A, B, C : \mathcal{U}, a, b, c : A$, $f : A \to B$ and $g : B \to C$. If $p : a =_A b$ and $q : b =_A c$, the following hold:*
(i) $f(p * q) =_{f(a)=_B f(c)} f(p) * f(q)$.
(ii) $f\left(p^{-1}\right) =_{f(b)=_B f(a)} f(p)^{-1}$.
(iii) $g(f(p)) =_{g(f(a))=_C g(f(b))} (g \circ f)(p)$.
(iv) $\mathrm{id}_A(p) =_{f(a)=_B f(a)} p$.
(v) *If $b_0 : B$ and $h \equiv \lambda_{x:A}.b_0$, then $\mathtt{ap}_h$ is a constant function.*

Proof. (i) Let the term $Q : \prod_{x,y:A} (x =_A y) \to \mathcal{U}$, defined by

$$Q(x, y, \pi) \equiv \prod_{z:A} \prod_{\varpi : y =_A z} f(\pi * \varpi) = f(\pi) * f(\varpi).$$

Hence,

$$\begin{aligned} Q(x, x, \mathtt{refl}_x) &\equiv \prod_{z:A} \prod_{\varpi : x =_A z} f(\mathtt{refl}_x * \varpi) = f(\mathtt{refl}_x) * f(\varpi) \\ &\equiv \prod_{z:A} \prod_{\varpi : x =_A z} f(\mathtt{refl}_x * \varpi) = \mathtt{refl}_{f(x)} * f(\varpi). \end{aligned}$$

By Proposition 3.5.1(i) $\mathtt{refl}_x * \varpi = \varpi$, hence using the second-application function we get a term of type $f(\mathtt{refl}_x * \varpi) = f(\varpi)$. By Proposition 3.5.1(i) again we get $\mathtt{refl}_{f(x)} * f(\varpi) = f(\varpi)$. As $\mathtt{refl}_{f(\varpi)} : f(\varpi) = f(\varpi)$, we find a term of type $Q(x, x, \mathtt{refl}_x)$.
(ii)-(v) Exercise. □

Proposition 4.1.4. *Let $A, B : \mathcal{U}, f : A \to B$. If $P : B \to \mathcal{U}$, let*

$$P \circ f : A \to \mathcal{U}, \quad (P \circ f)(x) \equiv P(f(x)).$$

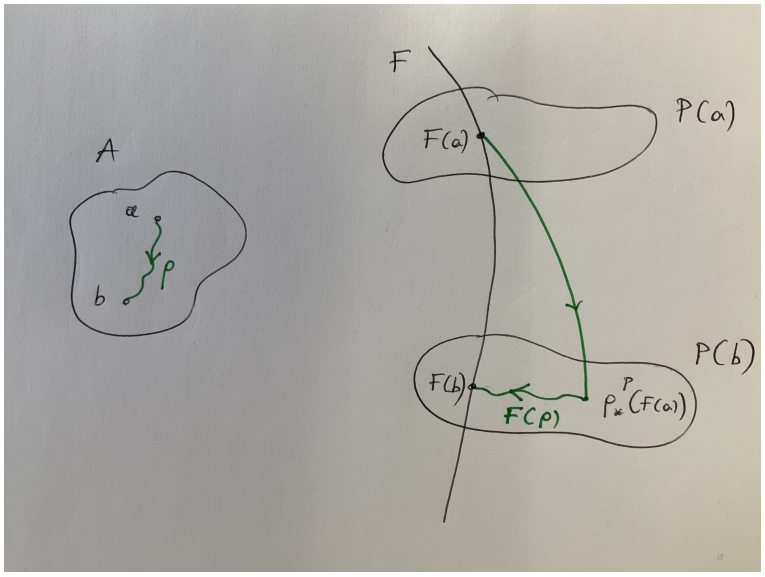


Figure 4.4: The application $F(p)$ of a dependent function F an an equality term $p\colon a =_A b$

If $p\colon a =_A b$*, then* $p_*^{P\circ f}\colon P(f(a)) \to P(f(b))$*. If* $u\colon P(f(a))$*, one can determine the term* $p_*^{P\circ f}(u)\colon P(f(b))$*.*

Proof. Exercise. □

The equality of functions implies their pointwise equality.

Proposition 4.1.5. *Let* $A, B\colon \mathcal{U}$*. There is a term*

$$\mathtt{eqtopointeq}\colon \prod_{\phi,\theta\colon A\to B} \prod_{\pi\colon \phi =_{A\to B}\theta} \prod_{x\colon A} \phi(x) =_B \theta(x),$$

such that $\mathtt{eqtopointeq}(f, f, \mathtt{refl}_f, a) \equiv \mathtt{refl}_{f(a)}$*, for every* $a\colon A$*.*

Proof. Exercise. □

4.2 Equality and the dependent function type

Next follows the dependent version of the application of a function.

Proposition 4.2.1. *Let* $A\colon \mathcal{U}$*,* $P\colon A \to \mathcal{U}$*, and* $F\colon \prod_{x\colon A} P(x)$*. There is a term*

$$\mathtt{ap}_F\colon \prod_{x,y\colon A} \prod_{\pi\colon x=_A y} \pi_*^P(F(x)) =_{P(y)} F(y),$$

$$(a, b, p) \mapsto F(p) \equiv \mathtt{ap}_F(a, b, p),$$

such that $\mathtt{ap}_F(a, a, \mathtt{refl}_a) \equiv \mathtt{refl}_{F(a)}$*, for every* $a\colon A$*.*

Proof. Exercise. □

One studies the dependent application as the non-dependent one.

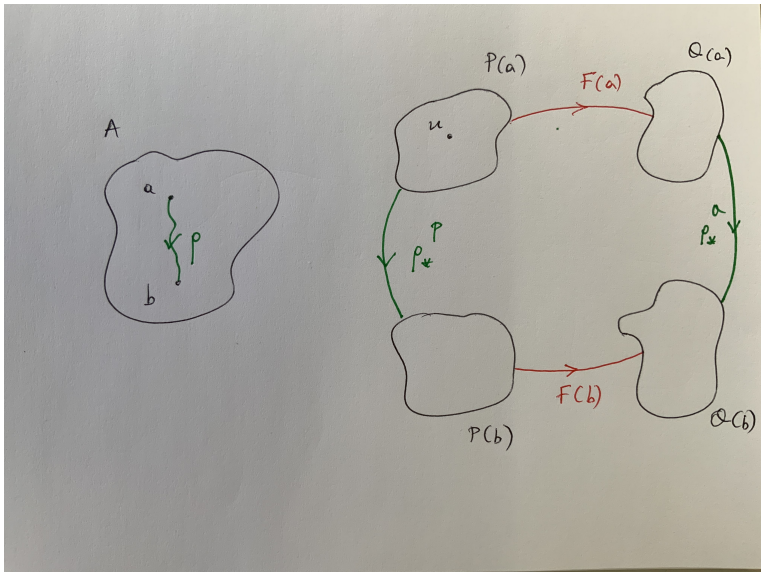


Figure 4.5: The relation between the transports and a dependent function

Proposition 4.2.2. *Let* $A\colon \mathcal{U}$*,* $P, Q\colon A \to \mathcal{U}$*, and*

$$F\colon \prod_{x\colon A} \big(P(x) \to Q(x)\big).$$

If $p\colon a =_A b$*, then* $p^P_*\colon P(a) \to P(b)$ *and* $p^Q_*\colon Q(a) \to Q(b)$*. If* $u\colon P(a)$*, then*

$$F\big(b, p^P_*(u)\big) =_{Q(b)} p^Q_*\big(F(a,u)\big).$$

Proof. Exercise. □

The equality of dependent functions implies their pointwise equality.

Proposition 4.2.3. *Let* $A\colon \mathcal{U}$ *and* $P\colon A \to \mathcal{U}$*. There is a term*

$$\mathtt{EqtoPointEq}\colon \prod_{\Phi,\Theta\colon \prod_{x\colon A} P(x)} \prod_{\pi\colon \Phi =_{\prod_{x\colon A} P(x)} \Theta} \prod_{x\colon A} \Phi(x) =_{P(x)} \Theta(x),$$

such that $\mathtt{EqtoPointEq}(F, F, \mathtt{refl}_F, a) \equiv \mathtt{refl}_{F(a)}$*, for every* $a\colon A$*.*

Proof. Exercise. □

4.3 Equality and the product type

Proposition 4.3.1. *If* $A, B\colon \mathcal{U}$*, there is a term*

$$M_{A\times B}\colon \prod_{z\colon A\times B} z =_{A\times B} \big(\mathtt{pr}_1(z), \mathtt{pr}_2(z)\big),$$

such that $M_{A\times B}\big((a,b)\big) \equiv \mathtt{refl}_{(a,b)}$*, for every* $a\colon A$ *and* $b\colon A$*.*

Proof. Let $P\colon A \times B \to \mathcal{U}$, defined by $P(z) \equiv z =_{A\times B} \big(\mathtt{pr}_1(z), \mathtt{pr}_2(z)\big)$. In order to use the induction rule for $A \times B$, we need to find

$$\begin{aligned} G\colon & \prod_{x\colon A}\prod_{y\colon B}\Big((x,y) =_{A\times B} \big(\mathtt{pr}_1((x,y)), \mathtt{pr}_2((x,y))\big)\Big) \\ & \equiv \prod_{x\colon A}\prod_{y\colon B}(x,y) =_{A\times B} (x,y). \end{aligned}$$

For that we define $G(x,y) \equiv \mathtt{refl}_{(x,y)}$. □

Corollary 4.3.2. *If $A, B\colon \mathcal{U}$ and $c\colon A \times B$, there are $a\colon A, b\colon A$ such that $c =_{A\times B} (a,b)$.*

Proof. It follows immediately from Proposition 4.3.1. □

Proposition 4.3.3. *If $A, B\colon \mathcal{U}$, there is a term*

$$\mathtt{eqtocompeq}_\times\colon \prod_{z,w\colon A\times B}(z =_{A\times B} w) \to \big(\mathtt{pr}_1(z) =_A \mathtt{pr}_1(w) \times \mathtt{pr}_2(z) =_B \mathtt{pr}_2(w)\big),$$

such that $\mathtt{eqtocompeq}_\times(c,c,\mathtt{refl}_c) \equiv \big(\mathtt{refl}_{\mathtt{pr}_1(c)}, \mathtt{refl}_{\mathtt{pr}_2(c)}\big)$*, for every $c\colon A \times B$.*

Proof. If $\pi\colon z =_{A\times B} w$, let $\mathtt{eqtocompeq}_\times(z,w,\pi) \equiv \big(\mathtt{pr}_1(\pi), \mathtt{pr}_2(\pi)\big)$, and he required computation-rule follows from the computation-rule of $\mathtt{ap}_f$ in Proposition 4.1.1. □

The above function has an "inverse".

Proposition 4.3.4. *If $A, B\colon \mathcal{U}$, there is a term*

$$\mathtt{compeqtoeq}_\times\colon \prod_{z,w\colon A\times B}\big(\mathtt{pr}_1(z) =_A \mathtt{pr}_1(w) \times \mathtt{pr}_2(z) =_B \mathtt{pr}_2(w)\big) \to (z =_{A\times B} w),$$

such that $\mathtt{compeqtoeq}_\times(c,c,\mathtt{refl}_{\mathtt{pr}_1(c)},\mathtt{refl}_{\mathtt{pr}_1(c)}) \equiv \mathtt{refl}_c$*, for every $c\colon A \times B$.*

Proof. If we find a term

$$N\colon \prod_{x,x'\colon A}\prod_{y,y'\colon B}(x =_A x' \times y =_B y') \to (x,y) =_{A\times B} (x',y'),$$

then by Proposition 4.3.1 we get

$$z = \big(\mathtt{pr}_1(z), \mathtt{pr}_2(z)\big) = \big(\mathtt{pr}_1(w), \mathtt{pr}_2(w)\big) = w.$$

In order to use the J-rule, it suffices to find a term

$$N'\colon \prod_{x,x'\colon A}\prod_{\pi\colon x=_A x'}\Big(\prod_{y,y'\colon B}\prod_{\varpi\colon y=_B y'}(x,y) =_{A\times B} (x',y')\Big).$$

Let

$$Q_A(x,x',\pi) \equiv \prod_{y,y'\colon B}\prod_{\varpi\colon y=_B y'}(x,y) =_{A\times B} (x',y').$$

We need to find a term

$$R_A \colon Q_A(x, x, \mathtt{refl}_x) \equiv \prod_{y,y' \colon B} \prod_{\varpi \colon y =_B y'} (x, y) =_{A \times B} (x, y').$$

For that let

$$Q_B \colon \prod_{y,y' \colon B} \prod_{\varpi \colon y =_B y'} \mathcal{U},$$

defined by

$$Q_B(y, y', \varpi) \equiv (x, y) =_{A \times B} (x, y').$$

Clearly,

$$R_B(y) \equiv \mathtt{refl}_{(x,y)} \colon Q_B(y, y, \mathtt{refl}_y) \equiv (x, y) =_{A \times B} (x, y).$$

Hence by the J_B-rule, let

$$R_A \equiv \widehat{R_B} \colon Q_A(x, x, \mathtt{refl}_x) \equiv \prod_{y,y' \colon B} \prod_{\varpi \colon y =_B y'} (x, y) =_{A \times B} (x, y').$$

Hence by the J_A-rule, let

$$N' \equiv \widehat{R_A} \equiv \widehat{\widehat{R_B}} \colon \prod_{x,x' \colon A} \prod_{\pi \colon x =_A x'} \left(\prod_{y,y' \colon B} \prod_{\varpi \colon y =_B y'} (x, y) =_{A \times B} (x', y') \right). \qquad \square$$

The two maps $\mathtt{eqtocompeq}_\times$ and $\mathtt{compeqtoeq}_\times$ can be shown to be "inverse" to each other.

4.4 Equality and the dependent pair type

The results of section 4.3 are generalised to the case of dependent pairs.

Proposition 4.4.1. *If $A \colon \mathcal{U}$ and $P \colon A \to \mathcal{U}$, there is a term*

$$M_{\sum_{x \colon A} P(x)} \colon \prod_{z \colon \sum_{x \colon A} P(x)} z =_{\sum_{x \colon A} P(x)} \left(\mathtt{pr}_1^\Sigma(z), \mathtt{pr}_2^\Sigma(z)\right),$$

such that $M_{\sum_{x \colon A} P(x)}\big((a, u)\big) \equiv \mathtt{refl}_{(a,u)}$, for every $a \colon A$ and $u \colon P(a)$.

Proof. Exercise. $\square$

Corollary 4.4.2. *If $A \colon \mathcal{U}$ and $P \colon A \to \mathcal{U}$, there are terms $a \colon A$ and $u \colon P(a)$ such that*

$$c =_{\sum_{x \colon A} P(x)} (a, u).$$

Proof. It follows immediately from Proposition 4.4.1. $\square$

Proposition 4.4.3. *If $A \colon \mathcal{U}$ and $P \colon A \to \mathcal{U}$, there is a term $\mathtt{eqtocompeq}_\Sigma$ of type*

$$\prod_{z,w \colon \sum_{x \colon A} P(x)} \left(\left(z =_{\sum_{x \colon A} P(x)} w\right) \to \sum_{\pi \colon \mathtt{pr}_1^\Sigma(z) =_A \mathtt{pr}_1^\Sigma(w)} \pi_*^P\left(\mathtt{pr}_2^\Sigma(z) =_{P(\mathtt{pr}_1^\Sigma(z))} \mathtt{pr}_2^\Sigma(w)\right) \right),$$

such that $\mathtt{eqtocompeq}_\Sigma(c, c, \mathtt{refl}_c) \equiv \left(\mathtt{refl}_{\mathtt{pr}_1^\Sigma(c)}, \mathtt{refl}_{\mathtt{pr}_2^\Sigma(c)}\right)$, for every $c \colon \sum_{x \colon A} P(x)$.

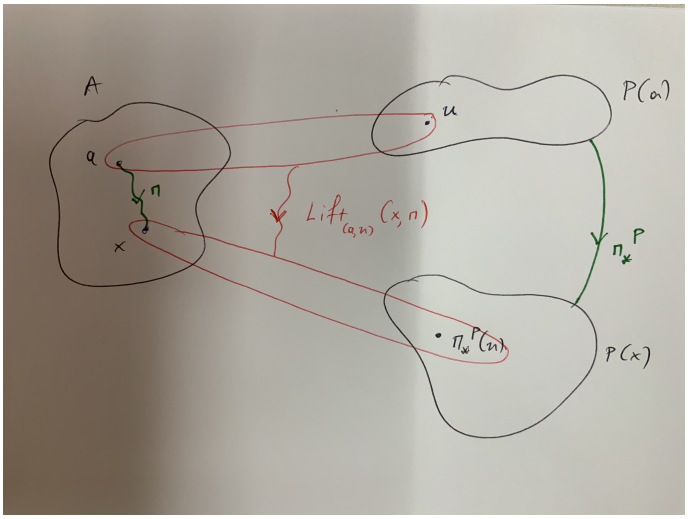


Figure 4.6: The path π is lifted to the path $\mathtt{Lift}_{(a,u)}(x,\pi)$ in $\sum_{x:\,A} P(x)$

Proof. Let the dependent function

$$Q\colon \prod_{z,w:\,\sum_{x:\,A} P(x)} \prod_{\varpi:\,z=w} \mathcal{U}$$

defined by

$$Q(z,w,\varpi) \equiv \sum_{\pi:\,\mathtt{pr}_1^\Sigma(z)=_A \mathtt{pr}_1^\Sigma(w)} \pi_*^P\big(\mathtt{pr}_2^\Sigma(z)\big) =_{P(\mathtt{pr}_1^\Sigma(z))} \mathtt{pr}_2^\Sigma(w).$$

hence

$$Q(z,z,\mathtt{refl}_z) \equiv \sum_{\pi:\,\mathtt{pr}_1^\Sigma(z)=_A \mathtt{pr}_1^\Sigma(z)} \pi_*^P\big(\mathtt{pr}_2^\Sigma(z)\big) =_{P(\mathtt{pr}_1^\Sigma(z))} \mathtt{pr}_2^\Sigma(z).$$

Since $\mathtt{pr}_1^\Sigma\colon \sum_{x:\,A} P(x) \to A$, we get

$$\mathtt{ap}_{\mathtt{pr}_1^\Sigma}(\mathtt{refl}_z) \equiv \mathtt{pr}_1^\Sigma(\mathtt{refl}_z) \equiv \mathtt{refl}_{\mathtt{pr}_1^\Sigma(z)} : \mathtt{pr}_1^\Sigma(z) =_A \mathtt{pr}_1^\Sigma(z),$$

hence,

$$\big(\mathtt{refl}_{\mathtt{pr}_1^\Sigma(c)}, \mathtt{refl}_{\mathtt{pr}_2^\Sigma(c)}\big) : Q(z,z,\mathtt{refl}_z),$$

and the required term follows from the J-rule. □

Proposition 4.4.4. *If $A\colon \mathcal{U}$ and $P\colon A \to \mathcal{U}$, there is a term $\mathtt{compeqtoeq}_\Sigma$ of type*

$$\prod_{z,w:\,\sum_{x:\,A} P(x)} \left[\sum_{\pi:\,\mathtt{pr}_1^\Sigma(z)=_A \mathtt{pr}_1^\Sigma(w)} \pi_*^P\big(\mathtt{pr}_2^\Sigma(z)\big) =_{P(\mathtt{pr}_1^\Sigma(w))} \mathtt{pr}_2^\Sigma(w) \right] \to \big(z =_{\sum_{x:\,A} P(x)} w\big),$$

such that $\mathtt{compeqtoeq}_\Sigma(c,c,\mathtt{refl}_{\mathtt{pr}_1^\Sigma(c)}, \mathtt{refl}_{\mathtt{pr}_2^\Sigma(c)}) \equiv \mathtt{refl}_c$, for every $c\colon \sum_{x:\,A} P(x)$.

Proof. Exercise. □

Proposition 4.4.5 (The path-lifting property)**.** *If* $A\colon \mathcal{U}$*,* $a\colon A$*,* $P\colon A \to \mathcal{U}$*, and* $u\colon P(a)$*, there is a term*

$$\mathtt{Lift}_{(a,u)}\colon \prod_{x\colon A}\prod_{\pi\colon a =_A x}(a,u) =_{\sum_{x\colon A} P(x)} \big(x, \pi^P_*(u)\big),$$

such that $\mathtt{Lift}_{(a,u)}(a, \mathtt{refl}_a) \equiv \mathtt{refl}_{(a,u)}$*.*

Proof. Exercise. □

4.5 Equality and the coproduct type

The coproduct $A + B$ is the union of $\mathrm{inl}(A)$ and $\mathrm{inr}(B)$.

Proposition 4.5.1. *If* $A, B\colon \mathcal{U}$*, there is a term of type*

$$\prod_{z\colon A+B}\left(\sum_{w\colon A} z =_{A+B} \mathrm{inl}(w) + \sum_{u\colon B} z =_{A+B} \mathrm{inr}(u)\right).$$

Proof. Exercise. □

Next we show that the left and right injections are indeed injections i.e.,

$$\mathrm{inl}(a_0) =_{A+B} \mathrm{inl}(a) \to a_0 =_A a, \qquad a_0, a\colon A$$

$$\mathrm{inr}(b_0) =_{A+B} \mathrm{inr}(b) \to b_0 =_B b, \qquad b_0, b\colon B.$$

The converse "implications" follow immediately by the application of the functions $\mathrm{inl}\colon A \to A + B$ and $\mathrm{inr}\colon B \to A + B$ i.e., if $p\colon a_0 =_A a$, then $\mathrm{inl}(p)\colon \mathrm{inl}(a_0) =_{A+B} \mathrm{inl}(a)$, and if $q\colon b_0 =_B b$, then $\mathrm{inr}(q)\colon \mathrm{inr}(b_0) =_{A+B} \mathrm{inr}(b)$. Moreover, we show that $A + B$ is the disjoint union of A and B (or of $\mathrm{inl}(A)$ and $\mathrm{inr}(B)$) i.e., we prove that the following type is inhabited:

$$\mathrm{inl}(a_0) =_{A+B} \mathrm{inr}(b_0) \to \mathbf{0},$$

for every $a_0\colon A$ and $b_0\colon B$. For that the proof-technique of "code" will be used. If $a_0\colon A$ and $b_0\colon B$, then by Rec_+ (in the next universe $\mathcal{U}'$) there are type-families $\mathtt{code}_l\colon A + B \to \mathcal{U}$ and $\mathtt{code}_r\colon A + B \to \mathcal{U}$ with

$$\mathtt{code}_l(\mathrm{inl}(a)) \equiv a_0 =_A a \quad \& \quad \mathtt{code}_l(\mathrm{inr}(b)) \equiv \mathbf{0},$$

$$\mathtt{code}_r(\mathrm{inl}(b)) \equiv b_0 =_B b \quad \& \quad \mathtt{code}_r(\mathrm{inl}(a)) \equiv \mathbf{0},$$

for every $a\colon A$ and $b\colon B$.

Proposition 4.5.2. *If* $A, B\colon \mathcal{U}$*,* $a_0\colon A$ *and* $b_0\colon B$*, there are terms*

$$\mathtt{encode}_l\colon \prod_{z\colon A+B}\prod_{\pi\colon \mathrm{inl}(a_0) =_{A+B} z} \mathtt{code}_l(z),$$

$$\mathtt{encode}_r\colon \prod_{z\colon A+B}\prod_{\pi\colon \mathrm{inr}(b_0) =_{A+B} z} \mathtt{code}_r(z).$$

Proof. If $z\colon A + B$ and $\pi\colon \mathrm{inl}(a_0) =_{A+B} z$, then

$$\pi_*^{\mathtt{code}_l}\colon \mathtt{code}_l(\mathrm{inl}(a_0)) \to \mathtt{code}_l(z) \equiv (a_0 =_A a_0) \to \mathtt{code}_l(z).$$

Hence we define

$$\mathtt{encode}_l(z, \pi) \equiv \pi_*^{\mathtt{code}_l}(\mathtt{refl}_{a_0}).$$

For the definition of $\mathtt{encode}_r$ we work similarly. □

To prove the injectivity of inl we use the term

$$i\colon \mathrm{inl}(a_0) =_{A+B} \mathrm{inl}(a) \to a_0 =_A a,$$

$$i(p) \equiv \mathtt{encode}_l(\mathrm{inl}(a), p)\colon \mathtt{code}_l(\mathrm{inl}(a) \equiv a_0 =_A a.$$

To prove the injectivity of inr we use the term

$$j\colon \mathrm{inr}(b_0) =_{A+B} \mathrm{inr}(b) \to b_0 =_B b,$$

$$j(q) \equiv \mathtt{encode}_r(\mathrm{inr}(b), q)\colon \mathtt{code}_r(\mathrm{inr}(b) \equiv b_0 =_B b.$$

To prove that $A + B$ is the disjoint union of A and B we use the term

$$k\colon \mathrm{inl}(a_0) =_{A+B} \mathrm{inr}(b_0) \to \mathbf{0},$$

$$k(r) \equiv \mathtt{encode}_l(\mathrm{inr}(b_0), r)\colon \mathtt{code}_l(\mathrm{inr}(b_0) \equiv \mathbf{0}.$$

Corollary 4.5.3. *There is a term of type* $0_{\mathbf{2}} =_{\mathbf{2}} 1_{\mathbf{2}} \to \mathbf{0}$.

Proof. Exercise. Actually, a more accurate formulation of this is about $\mathbf{2}'\colon \mathcal{U}'$. □

Proposition 4.5.4. *If* $A, B\colon \mathcal{U}$, $a\colon A$ *and* $b\colon B$, *there are terms of type*

$$\prod_{z\colon A+B} \prod_{\pi\colon \mathrm{inl}(a)=z} \sum_{x\colon A} z =_{A+B} \mathrm{inl}(x),$$

$$\prod_{z\colon A+B} \prod_{\pi\colon \mathrm{inr}(b)=z} \sum_{y\colon B} z =_{A+B} \mathrm{inr}(y)$$

Proof. Exercise. □

4.6 Equality and the unit type

Proposition 4.6.1. *There is a term*

$$M_{\mathbf{1}}\colon \prod_{x\colon \mathbf{1}} (x =_{\mathbf{1}} 0_{\mathbf{1}}),$$

such that $M_{\mathbf{1}}(0_{\mathbf{1}}) \equiv \mathtt{refl}_{0_{\mathbf{1}}}$.

Proof. By $\mathrm{Ind}_{\mathbf{1}}$ it sufficed to find a term of type $0_{\mathbf{1}} =_{\mathbf{1}} 0_{\mathbf{1}}$. This term is $\mathtt{refl}_{0_{\mathbf{1}}}$. □

Proposition 4.6.2. *If* $x, y\colon 0_{\mathbf{1}}$, *then there are functions* $f\colon \mathbf{1} \to x =_{0_{\mathbf{1}}} y$ *and* $g\colon x =_{0_{\mathbf{1}}} y \to \mathbf{1}$, *which are inverse to each other.*

Proof. Let $g \equiv \lambda_{\pi:\, x=_{0_\mathbf{1}} y}.0_\mathbf{1}$, and let $f\colon \mathbf{1} \to x =_{0_\mathbf{1}} y$ be defined by $\mathrm{Rec}_\mathbf{1}$ through the term

$$f(0_\mathbf{1}) \equiv M_\mathbf{1}(x) * [M_\mathbf{1}(y)]^{-1}\colon x =_{0_\mathbf{1}} y,$$

where $M_\mathbf{1}$ is determined in Proposition 4.6.1. Clearly,

$$\prod_{x:\,\mathbf{1}} g(f(x)) =_\mathbf{1} x$$

is inhabited, as by $\mathrm{Ind}_\mathbf{1}$ we have that $g(f(0_\mathbf{1})) =_\mathbf{1} 0_\mathbf{1}$. Next we use the J-rule to inhabit the type

$$\prod_{x,y:\,\mathbf{1}} \prod_{\pi:\, x=_{0_\mathbf{1}} y} f(g(\pi)) = \pi.$$

If $Q(x, y, \pi) \equiv f(g(\pi) = \pi$, then it suffices to inhabit the type

$$Q(x, x, \mathtt{refl}_x) \equiv f(g(\mathtt{refl}_x)) = \mathtt{refl}_x.$$

This follows from the following equalities

$$\begin{aligned} f(g(\mathtt{refl}_x)) &\equiv f(0_\mathbf{1}) \\ &\equiv M_\mathbf{1}(x) * [M_\mathbf{1}(x)]^{-1} \\ &= \mathtt{refl}_x. \end{aligned}$$

□

Hence all proofs of equality between $x, y\colon \mathbf{1}$ are equal (why?).

4.7 Equality and the type of naturals

Next we use the code-method to show some of the Peano axioms (see 1.3.3) for the type of naturals $\mathbf{N}'\colon \mathcal{U}'$, where $\mathbf{N}'$ is the type of natural numbers in the next universe $\mathcal{U}'$ to $\mathcal{U}$. For that we use only primitive recursion $\mathrm{Rec}_{\mathbf{N}'}$.

Theorem 4.7.1. *There is a function* $\mathtt{code}_{\mathbf{N}'}\colon \mathbf{N}' \to \mathbf{N}' \to \mathcal{U}$ *satisfying the following conditions:*

$$\mathtt{code}_{\mathbf{N}'}(0_{\mathbf{N}}{}', 0_{\mathbf{N}}{}') \equiv \mathbf{1},$$

$$\mathtt{code}_{\mathbf{N}'}(\mathtt{succ}'(m'), 0_{\mathbf{N}}{}') \equiv \mathbf{0},$$

$$\mathtt{code}_{\mathbf{N}'}(0_\mathbf{N}, \mathtt{succ}'(n')) \equiv \mathbf{0},$$

$$\mathtt{code}_{\mathbf{N}'}(\mathtt{succ}'(m'), \mathtt{succ}'(n')) \equiv \mathtt{code}_{\mathbf{N}'}(m', n').$$

Proof. There is $h\colon \mathbf{N}' \to \mathcal{U}$ satisfying

$$h(0_{\mathbf{N}}{}') \equiv \mathbf{1},$$

$$h(\mathtt{succ}'(n')) \equiv \mathbf{0}.$$

To define h we use $\mathrm{Rec}_{\mathbf{N}'}$, where $\mathbf{1}\colon \mathcal{U}$ and $s'\colon \mathbf{N}' \to \mathcal{U} \to \mathcal{U}$ is the term

$$s' \equiv \lambda_{x':\,\mathbf{N}'}.\lambda_{X:\,\mathcal{U}}.\mathbf{0}.$$

Clearly, $\mathtt{code}_{\mathbf{N}'}(0_{\mathbf{N}}{}') \equiv h$. Then we define

$$s\colon \mathbf{N}' \to (\mathbf{N}' \to \mathcal{U}) \to (\mathbf{N}' \to \mathcal{U}),$$

such that

$$\mathtt{code}_{\mathbf{N}'}(\mathtt{succ}'(m'), 0_{\mathbf{N}}{}') \equiv \mathbf{0} \equiv [s(m', \mathtt{code}_{\mathbf{N}'}(m')](0_{\mathbf{N}}{}'),$$

and

$$\mathtt{code}_{\mathbf{N}'}(\mathtt{succ}'(m'), \mathtt{succ}'(n')) \equiv \mathtt{code}_{\mathbf{N}'}(m', n')$$
$$\equiv [\mathtt{code}_{\mathbf{N}'}(m')](n') \equiv [s(m', \mathtt{code}_{\mathbf{N}'}(m')](\mathtt{succ}'(n')).$$

For that we define

$$s \equiv \lambda x'\colon \mathbf{N}'.\lambda_{f'\colon \mathbf{N}' \to \mathcal{U}}.s(m', f'),$$

where $s(m', f')\colon \mathbf{N}' \to \mathcal{U}$ satisfies the conditions:

$$[s(m', f')](0_{\mathbf{N}}{}') \equiv \mathbf{0},$$

$$[s(m', f')](\mathtt{succ}'(n')) \equiv f(n).$$

The existence of such a function $s(m', f')$ follows again from $\mathrm{Rec}_{\mathbf{N}'}$ (exercise). □

To get a function $\mathtt{code}_{\mathbf{N}}\colon \mathbf{N} \to \mathbf{N} \to \mathcal{U}$ that satisfies the same definitional equalities as $\mathtt{code}_{\mathbf{N}'}$, we need to have a function $i\colon \mathbf{N} \to \mathbf{N}'$ in $\mathcal{U}'$ such that $i(0_{\mathbf{N}}) \equiv 0_{\mathbf{N}}{}'$ and $i(\mathtt{succ}(n)) \equiv \mathtt{succ}'(i(n))$, and then define $\mathtt{code}_{\mathbf{N}}(m, n) \equiv \mathtt{code}_{\mathbf{N}'}(i(m), i(n))$.

Lemma 4.7.2. *There is a term*

$$R\colon \prod_{x'\colon \mathbf{N}'} \mathtt{code}_{\mathbf{N}'}(x', x'),$$

such that

$$R(0_{\mathbf{N}}{}') \equiv 0_{\mathbf{1}},$$
$$R(\mathtt{succ}'(n')) \equiv R(n').$$

Proof. Let $P'\colon \mathbf{N}' \to \mathcal{U}$ with $P'(x') \equiv \mathtt{code}_{\mathbf{N}'}(x', x')$. Clearly, $0_{\mathbf{1}}\colon P(0_{\mathbf{N}}{}') \equiv \mathtt{code}_{\mathbf{N}'}(0_{\mathbf{N}}{}', 0_{\mathbf{N}}{}') \equiv \mathbf{1}$. We define next

$$S \prod_{x'\colon \mathbf{N}'} \mathtt{code}_{\mathbf{N}'}(x', x') \to \mathtt{code}_{\mathbf{N}'}(\mathtt{succ}'(x'), \mathtt{succ}'(x'))$$

by $S(x', z) \equiv z$. By $\mathrm{Ind}_{\mathbf{N}'}$ $R(0_{\mathbf{N}}{}') \equiv 0_{\mathbf{1}}$ and

$$R(\mathtt{succ}'(n')) \equiv S(n', R(n')) \equiv R(n').$$ □

Theorem 4.7.3. *There are terms*

$$\mathtt{encode}_{\mathbf{N}'}\colon \prod_{x', y'\colon \mathbf{N}'} \big(x' =_{\mathbf{N}'} y' \to \mathtt{code}_{\mathbf{N}'}(x', y')\big),$$

$$\mathtt{decode}_{\mathbf{N}'}\colon \prod_{x', y'\colon \mathbf{N}'} \big(\mathtt{code}_{\mathbf{N}'}(x', y') \to x' =_{\mathbf{N}'} y'\big).$$

Proof. If $\pi\colon x' =_{\mathbf{N}'} y'$, then $\mathtt{code}_{\mathbf{N}'}(x', -)\colon \mathbf{N}' \to \mathcal{U}$, hence $\mathtt{code}_{\mathbf{N}'}(x', -)\colon \mathbf{N}' \to \mathcal{U}'$. Consequently,

$$\pi_*^{\mathtt{code}_{\mathbf{N}'}(x', -)}\colon \mathtt{code}_{\mathbf{N}'}(x', x') \to \mathtt{code}_{\mathbf{N}'}(x', y').$$

Hence we define

$$\mathtt{encode}_{\mathbf{N}'}(x', y', \pi) \equiv \pi_*^{\mathtt{code}_{\mathbf{N}'}(x', -)}\big(R(x')\big).$$

To define $\mathtt{decode}_{\mathbf{N}'}$, let $Q\colon \mathbf{N}' \to \mathcal{U}'$ with

$$Q(x') \equiv \prod_{y'\colon \mathbf{N}'} \big(\mathtt{code}_{\mathbf{N}'}(x', y') \to x' =_{\mathbf{N}'} y'\big).$$

It is easy to inhabit $Q(0_{\mathbf{N}}')$. Then one needs to define a term

$$S \prod_{x'\colon \mathbf{N}'} \big(Q(x') \to Q(\mathtt{succ}'(x'))\big)$$

i.e., if

$$\phi\colon \prod_{y'\colon \mathbf{N}'} \big(\mathtt{code}_{\mathbf{N}'}(x', y') \to x' =_{\mathbf{N}'} y'\big),$$

$$[S(x')](\phi)\colon \prod_{y'\colon \mathbf{N}'} \big(\mathtt{code}_{\mathbf{N}'}(\mathtt{succ}'(x'), y') \to \mathtt{succ}'(x') =_{\mathbf{N}'} y'\big).$$

This follows by $\mathrm{Ind}_{\mathbf{N}'}$ again. □

One can show that $\mathtt{encode}_{\mathbf{N}'}(x', y')$ and $\mathtt{decode}_{\mathbf{N}'}(x', y')$ are inverse to each other.

Corollary 4.7.4. *There is a term of type* $\mathtt{succ}'(x') =_{\mathbf{N}'} \mathtt{succ}'(y') \to x' =_{\mathbf{N}'} y'$*, for every* $x', y'\colon \mathbf{N}'$.

Proof. Exercise. □

Corollary 4.7.5. *There is a term of type* $\mathtt{succ}'(x') =_{\mathbf{N}'} 0_{\mathbf{N}}' \to \mathbf{0}$*, for every* $x'\colon \mathbf{N}'$.

Proof. Exercise. □

If we define addition on $\mathbf{N}'$, the remaining Peano axioms are also provable.

Chapter 5

Homotopy type theory

In this chapter we present the basics of Homotopy Type Theory (HoTT), which is an extension of intensional Martin-Löf Type Theory with Voevodsky's axiom of univalence and higher inductive types. The axiom of univalence embodies the interplay between the universe and the equality type.

5.1 Homotopies

Definition 5.1.1. *Let $A\colon \mathcal{U}$ $P\colon A \to \mathcal{U}$ and $F, G\colon \prod_{x\colon A} P(x)$. A homotopy from F to G is a term*

$$H\colon F \sim G \equiv \prod_{x\colon A} F(x) =_{P(x)} G(x).$$

Consequently, if $A, B\colon \mathcal{U}$ and $f, g\colon A \to B$, a homotopy from f to g is term

$$H\colon f \sim f \equiv \prod_{x\colon A} f(x) =_B g(x).$$

As types are interpreted as spaces, equality proofs as paths, and function-terms as "continuous" functions, a homotopy H from$f$ to g corresponds to a continuous deformation of f to g i.e., to a homotopy H from topology:

$$H\colon A \times [0,1] \overset{cnt}{\to} B,$$

$$H(x,0) = f(x), \quad x \in A,$$

$$H(x,1) = g(x), \quad x \in A.$$

In this case $H(x,-)\colon [0,1] \to Y$ is a continuous path from $f(x)$ to $g(x)$ in B, for every $x \in A$.

Proposition 5.1.2. *If $A, B\colon \mathcal{U}$, the following types are inhabited:*

$$\prod_{f\colon A\to B} f \sim f,$$

$$\prod_{f,g\colon A\to B} f \sim g \to g \sim f,$$

$$\prod_{f,g,h\colon A\to B} f \sim g \to g \sim h \to f \sim h.$$

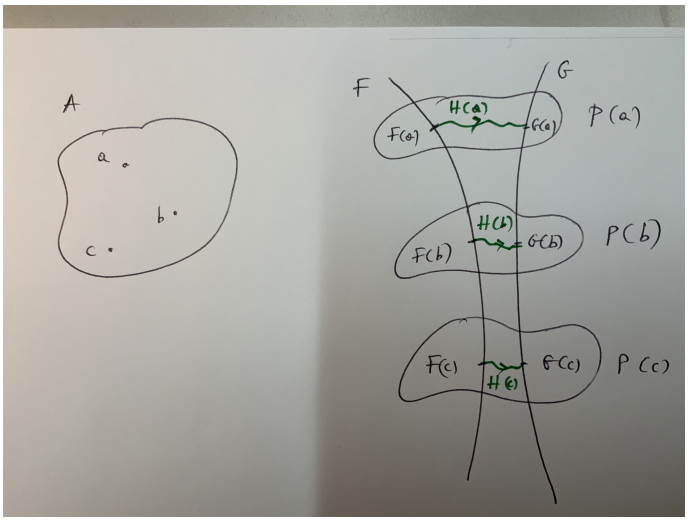


Figure 5.1: A homotopy H between dependent functions F, G

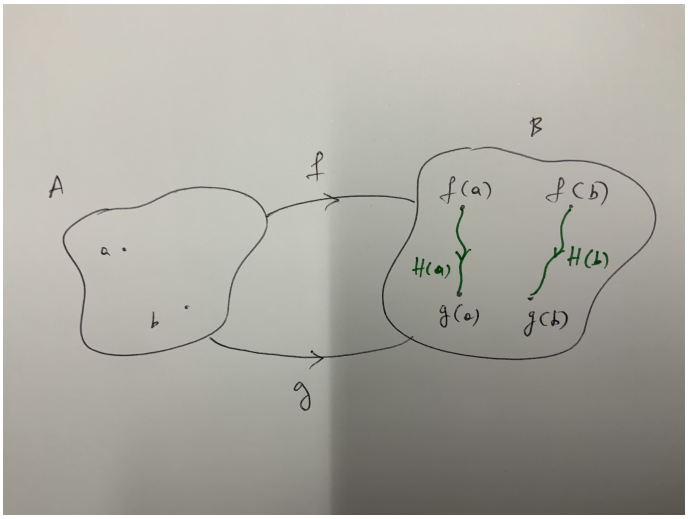


Figure 5.2: A homotopy H between non-dependent functions f, g

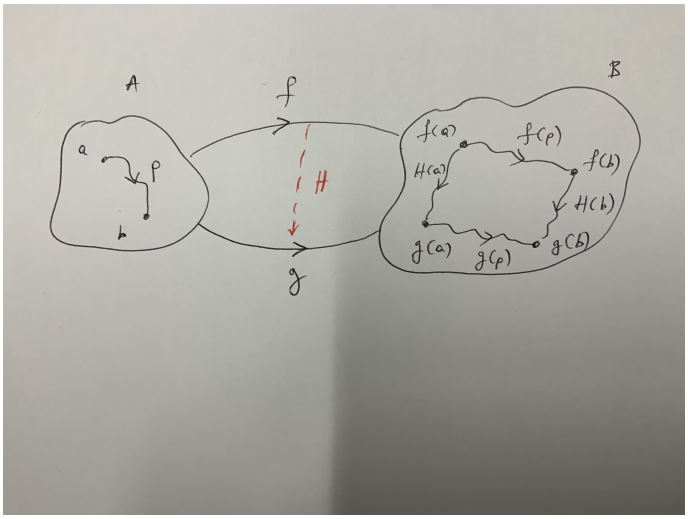


Figure 5.3: A homotopy H behaves like a natural transformation

Proof. Exercise. □

Definition 5.1.3. *If* $H\colon f \sim g$*, let* $H^{-1}\colon g \sim f$*, defined by* $H^{-1}(x) \equiv H(x)^{-1}$*. If* $G\colon g \sim h$*, let* $H * G\colon f \sim h$*, defined by* $(H * G)(x) \equiv H(x) * G(x)$*, for every* $x\colon A$*.*

Proposition 5.1.4. *If* $A, B\colon \mathcal{U}$*,* $f, g\colon A \to B$ *and* $p\colon a =_A b$*, then if* $H\colon f \sim g$*, we have that*

$$H(a) * g(p) = f(p) * H(b).$$

Proof. Exercise. □

Proposition 5.1.5. *If* $A\colon \mathcal{U}$*,* $f\colon A \to A$*, and* $a\colon A$*, then if* $H\colon f \sim \mathrm{id}_A$*, we have that*

$$H(f(a)) = f(H(a))).$$

Proof. Exercise. □

5.2 Equivalences

Next concept is the direct type-theoretic translation of the homotopic equivalence between two topological spaces. Although this is going to be a very workable notion of equivalence of types, it is the notion of Voevodsky's equivalence that will fully capture the equivalence of types in HoTT.

Definition 5.2.1. *If* $A, B\colon \mathcal{U}$ *and* $f\colon A \to B$*, the type "*f *is a quasi-inverse" is defined by*

$$\mathtt{qinv}(f) \equiv \sum_{g\colon B \to A} [(f \circ g \sim \mathrm{id}_B) \times (g \circ f \sim \mathrm{id}_A)].$$

Th quasi-equivalence between A, B *is the type*

$$A \simeq_q B \equiv \sum_{f\colon A \to B} \mathtt{qinv}(f).$$

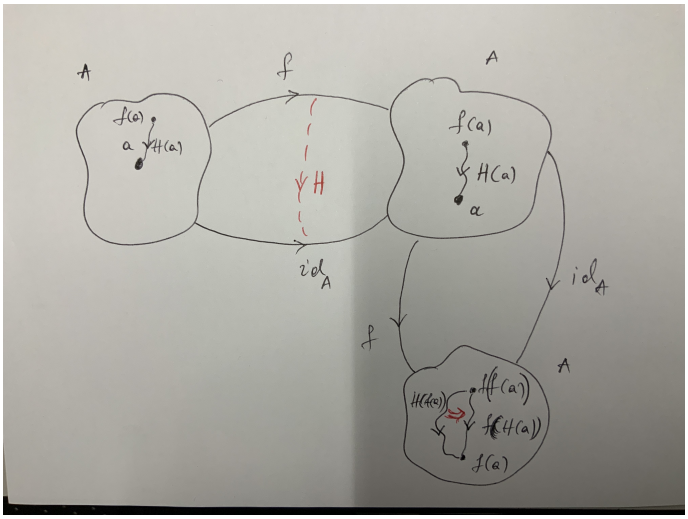


Figure 5.4: The geometry of Proposition 5.1.5

For example, if $A : \mathcal{U}$, then $(\mathrm{id}_A, \mathtt{refl}_A, \mathtt{refl}_A) : \mathtt{qinv}(\mathrm{id}_A)$, where

$$\mathtt{refl}_A : \mathrm{id}_A \circ \mathrm{id}_A \sim \mathrm{id}_A \equiv \prod_{x:\, A} x =_A x.$$

If $p : a =_A b$ and $P : A \to \mathcal{U}$, then by Proposition 3.5.4(ii)-(iii) we get $\mathtt{qinv}(p_*^P)$.

Definition 5.2.2. *If $A, B : \mathcal{U}$ and $f : A \to B$, the type "f is an equivalence" is defined by*

$$\mathtt{isequiv}(f) \equiv \left(\sum_{g:\, B \to A} (f \circ g) \sim \mathrm{id}_B \right) \times \left(\sum_{h:\, B \to A} (h \circ f) \sim \mathrm{id}_A \right).$$

The equivalence, or Voevodsky-equivalence, between A, B is the type

$$A \simeq_{\mathcal{U}} B \equiv \sum_{f:\, A \to B} \mathtt{isequiv}(f).$$

When the universe is clear from the context we also write $A \simeq B$.

The two notions of equivalence for a function $f : A \to B$ are logically equivalent.

Proposition 5.2.3. *If $A, B : \mathcal{U}$ and $f : A \to B$, the following hold:*
(i) $\mathtt{qinv}(f) \to \mathtt{isequiv}(f)$.
(ii) $\mathtt{isequiv}(f) \to \mathtt{qinv}(f)$.

Proof. (i) Let $\mathrm{qi} : \mathtt{qinv}(f) \to \mathtt{isequiv}(f)$, defined by

$$\mathrm{qi}(g, H, G) \equiv \big((g, H), (g, G)\big).$$

(ii) Let $\mathrm{iq} : \mathtt{isequiv}(f) \to \mathtt{qinv}(f)$, defined by

$$\mathrm{iq}\big((g, G), (h, H)\big) \equiv (e, G', H'),$$

where $e \equiv h \circ f \circ g$. □

The equivalence of type is an equivalence relation on $\mathcal{U}$.

Proposition 5.2.4. *If $A, B, C : \mathcal{U}$, the following types are inhabited.*
(i) $A \simeq A$.
(ii) $A \simeq B \to B \simeq A$.
(iii) $A \simeq B \to B \simeq B \simeq C \to A \simeq C$.

Proof. Exercise. □

Definition 5.2.5. *Let $A, B, C : \mathcal{U}$. If*

$$(e, p) : A \simeq B \equiv \sum_{e : A \to B} \mathtt{isequiv}(e),$$

we only write $e : A \simeq B$, as the type $\mathtt{isequiv}(e)$ will be shown to be a mere proposition. Moreover, we write $e^{-1} : B \simeq A$. If $j : B \simeq C$, let $j \circ e : A \simeq C$.

5.3 Function extensionality axiom

By Proposition 4.1.5 if $A, B : \mathcal{U}$, there is a term

$$\mathtt{eqtopointeq} : \prod_{\phi,\theta : A \to B} \prod_{\pi : \phi = \theta} \prod_{x : A} \phi(x) =_B \theta(x),$$

such that $\mathtt{eqtopointeq}(f, f, \mathtt{refl}_f, a) \equiv \mathtt{refl}_{f(a)}$, for every $a : A$. Hence, if $f, g : A \to B$, there is a function

$$\mathtt{eqtopointeq}_{f,g} : f =_{A \to B} g \to f \sim g \equiv \prod_{x : A} f(x) =_B g(x),$$

such that

$$\mathtt{eqtopointeq}_{f,f}(\mathtt{refl}_f) \equiv \lambda_{x : A}.\mathtt{refl}_{f(x)}.$$

Function extensionality axiom (`FunExt`): The following function is an equivalence

$$\mathtt{eqtopointeq}_{f,g} : f =_{A \to B} g \to f \sim g.$$

Let $\mathtt{funext}_{f,g} : f \sim g \to f =_{A \to B} g$ be the quasi-inverse to $\mathtt{eqtopointeq}_{f,g}$. By this we mean the following. If $\big((g, G), (h, H)\big) : \mathtt{isequiv}(\mathtt{eqtopointeq}_{f,g})$, then

$$\mathrm{iq}\big((g, G), (h, H)\big) \equiv (\mathtt{funext}_{f,g}, G', H') : \mathtt{qinv}(\mathtt{eqtopointeq}_{f,g}).$$

By the definition of the type $\mathtt{qinv}(\mathtt{eqtopointeq}_{f,g})$ we get

$$\mathtt{funext}_{f,g} \circ \mathtt{eqtopointeq}_{f,g} \sim \mathrm{id}_{f=g}$$

i.e.,

$$G' : \prod_{\pi : f=g} \mathtt{funext}_{f,g}\big(\mathtt{eqtopointeq}_{f,g}(\pi)\big) = \pi.$$

Moreover,

$$\mathtt{eqtopointeq}_{f,g} \circ \mathtt{funext}_{f,g} \sim \mathrm{id}_{f \sim g}$$

i.e.,

$$H' \colon \prod_{K \colon f \sim g} \mathtt{eqtopointeq}_{f,g}\big(\mathtt{funext}_{f,g}(K)\big) = K.$$

Hence, by G' we get

$$\begin{aligned} \mathtt{refl}_f &= \mathtt{funext}_{f,f}\big(\mathtt{eqtopointeq}_{f,f}(\mathtt{refl}_f)\big) \\ &\equiv \mathtt{funext}_{f,f}\big(\lambda_{x \colon A}.\mathtt{refl}_{f(x)}\big)\big). \end{aligned}$$

Proposition 5.3.1. *Let $A, B \colon \mathcal{U}$, $f, g, h \colon A \to B$, $p \colon f = g$ and $q \colon g = h$. As $\mathtt{eqtopointeq}_{f,g} \colon f = g \to f \sim g$ and $\mathtt{eqtopointeq}_{g,h} \colon g = h \to g \sim h$, we have that*

$$\mathtt{eqtopointeq}_{f,g}(p) \colon f \sim g \quad \& \quad \mathtt{eqtopointeq}_{g,h}(q) \colon g \sim h,$$

hence

$$\mathtt{funext}_{f,g}\big(\mathtt{eqtopointeq}_{f,g}(p)\big) \colon f = g \quad \& \quad \mathtt{funext}_{g,h}\big(\mathtt{eqtopointeq}_{g,h}(q)\big) \colon g = h.$$

Considering the inverse and the concatenation of homotopies from Definition 5.1.3, we get

$$\big[\mathtt{eqtopointeq}_{f,g}(p)\big]^{-1} \colon g \sim f \quad \& \quad \mathtt{eqtopointeq}_{f,g}(p) * \mathtt{eqtopointeq}_{g,h}(q) \colon f \sim h.$$

(i) $\mathtt{funext}_{g,f}\left(\big[\mathtt{eqtopointeq}_{f,g}(p)\big]^{-1}\right) = p^{-1}$.

(ii) $\mathtt{funext}_{f,h}\Big(\mathtt{eqtopointeq}_{f,g}(p) * \mathtt{eqtopointeq}_{g,h}(q)\Big) = p * q$.

Proof. Exercise. □

By Proposition 4.2.3 if $A \colon \mathcal{U}$ and $P \colon A \to \mathcal{U}$, there is a term

$$\mathtt{EqtoPointEq} \colon \prod_{\Phi,\Theta \colon \prod_{x \colon A} P(x)} \pi \colon \Phi = \prod_{x \colon A} P(x) \Theta \prod_{x \colon A} \Phi(x) =_{P(x)} \Theta(x),$$

such that $\mathtt{EqtoPointEq}(F, F, \mathtt{refl}_F, a) \equiv \mathtt{refl}_{F(a)}$, for every $a \colon A$. If $F, G \colon \prod_{x \colon A} P(x)$, then

$$\mathtt{EqtoPointEq}_{F.G} \colon F =_{\prod_{x \colon A} P(x)} G \to F \sim G \equiv \prod_{x \colon A} F(x) =_{P(x)} G(x),$$

$$\mathtt{EqtoPointEq}_{F,F}(\mathtt{refl}_F) \equiv \lambda_{x \colon A}.\mathtt{refl}_{F(x)}.$$

Function extensionality axiom-(dependent): The following function is an equivalence

$$\mathtt{EqtoPointEq}_{F,G} \colon F =_{\prod_{x \colon A} P(x)} G \to F \sim G.$$

If $\mathtt{Funext}_{F,F} \colon F \sim G \to F =_{\prod_{x \colon A} P(x)} G$ is its quasi-inverse, we work as above.

5.4 Propositions and sets

A type A is a mere proposition, or simply a proposition, if it is true or not i.e., if it is inhabited or not. In the first case, all inhabitands of A must must be equal, as its truth (inhabitedness) is only that matters.

Definition 5.4.1. *If $A\colon \mathcal{U}$, the type "A is a proposition" is defined by*

$$\mathtt{isProp}(A) \equiv \prod_{x,y\colon A} x =_A y.$$

Clearly, $\mathbf{0}, \mathbf{1}$ are propositions. Voevodsky's equivalence generalises logical equivalence between propositions.

Proposition 5.4.2. *Let $A, B\colon \mathcal{U}$, such that* $\mathtt{isProp}(A)$ *and* $\mathtt{isProp}(B)$.

(i) *If A is inhabited, then $A \simeq \mathbf{1}$.*

(ii) *If $A \leftrightarrow B$, then $A \simeq B$.*

(iii) $\mathtt{isProp}(A)$ *is a mere proposition.*

(iv) $\mathtt{isProp}(A \times B)$.

(v) *If $C\colon \mathcal{U}$, then* $\mathtt{isProp}(C \to A)$.

(vi) $\mathtt{isProp}(\neg A)$.

Proof. Exercise. □

Clearly, $\mathbf{2}$ is not a proposition, hence propositions are not closed under coproducts (why?). Similarly, they are not closed under dependent pairs. If $P\colon A \to \mathcal{U}$ such that $\mathtt{isProp}(P(x))$, for every $x\colon A$, then by Proposition 4.3.4 we get

$$\mathtt{pr}_1(z) = \mathtt{pr}_1(w) \to z =_{\sum_{x\colon A} P(x)} w.$$

Using the axiom of function extensionality, the type $\mathtt{qinv}(\mathrm{id}_A)$ is written as follows:

$$\begin{aligned}
\mathtt{qinv}(\mathrm{id}_A) &\equiv \sum_{g\colon A\to A} [(\mathrm{id}_A \circ g \sim \mathrm{id}_A) \times (g \circ \mathrm{id}_A \sim \mathrm{id}_A)] \\
&\equiv \sum_{g\colon A\to A} [g \sim \mathrm{id}_A \times g \sim \mathrm{id}_A) \\
&\equiv \sum_{g\colon A\to A} \left[\left(\prod_{x\colon A} g(x) =_A x\right) \times \left(\prod_{x\colon A} g(x) =_A x\right)\right] \\
&\simeq \sum_{g\colon A\to A} \left[\left(g =_{A\to A} \mathrm{id}_A\right) \times \left(\prod_{x\colon A} g(x) =_A x\right)\right] \\
&\simeq \prod_{x\colon A} x =_A x,
\end{aligned}$$

since by Corollary 3.2.2 the type $\sum_{g\colon A\to A} \left(g =_{A\to A} \mathrm{id}_A\right)$ is contractible. It can be shown that there is a type A such that

$$\prod_{x\colon A} x =_A x$$

is not a mere proposition. Hence, the type $\mathtt{qinv}(f)$ is not a mere proposition too. However, one can show that

$$\mathtt{isProp}(\mathtt{isequiv}(f)).$$

hence, to show that two proofs of $A \simeq B$ are equal, it suffices to show that the corresponding equivalences are equal functions of type $A \to B$. Tis is an important advantage of Voevodsky's notion of equivalence.

5.5 Sets

A type A is a set, if it is characterised by its inhabitands i.e., its inhabitands matter and not the proofs of their equalities.

Definition 5.5.1. *If $A : \mathcal{U}$, the type "A is a set" is defined by*

$$\mathtt{isSet}(A) \equiv \prod_{x,y:\, A} \prod_{\pi,\varpi:\, x =_A y} \pi = \varpi.$$

Proposition 5.5.2. *If $A : \mathcal{U}$, then* $\mathtt{isProp}(A) \to \mathtt{isSet}(A)$.

Proof. If $F : \mathtt{isProp}(A)$ and $x : A$, let $P_x : A \to \mathcal{U}$, defined by $P_x(y) \equiv x =_A y$. Let $G_x : \prod_{y:\, A} P_x(y)$, defined by $G_x(y) \equiv F(x, y)$. If $y, z : A$, by Proposition 3.5.2(i) we have that

$$\mathtt{ap}_G(y, z) : \prod_{\pi:\, y =_A z} \pi_*^P(G(y)) = G(z) \equiv \prod_{\pi:\, y =_A z} G(y) * \pi = G(z).$$

Hence, $\pi = G(y)^{-1} * G(z)$. □

Similarly, one shows that if $\mathtt{isSet}(A)$, then

$$\prod_{x,y:\, A} \prod_{\pi,\varpi:\, x =_A y} \prod_{r,s:\, \pi = \varpi} r = s.$$

The types $\mathbf{2}$ and $\mathbb{N}$ are sets (why?), which are not propositions. We need the univalence axiom, in order to show that there are types which are not sets.

5.6 Voevodsky's axiom of univalence

The axiom of univalence asserts that $A =_{\mathcal{U}} B \equiv_{\mathcal{U}'} A \equiv_{\mathcal{U}} B$.

Proposition 5.6.1. *If $A, B : \mathcal{U}$, there is a function in $\mathcal{U}'$*

$$\mathtt{eqtoequiv}_{A,B} : A =_{\mathcal{U}} B \to A \simeq B,$$

such that $\mathtt{eqtoequiv}_{A,A}(\mathtt{refl}_A) \equiv \big((\mathrm{id}_A, \mathtt{refl}_A), (\mathrm{id}_A, \mathtt{refl}_A)\big)$.

Proof. Let the dependent function

$$Q : \prod_{X,Y:\, \mathcal{U}} \prod_{\pi:\, X =_{\mathcal{U}} Y} \mathcal{U}',$$

defined by $Q(X, Y, \pi) \equiv X \simeq Y$. As

$$\big((\mathrm{id}_X, \mathtt{refl}_X), (\mathrm{id}_X, \mathtt{refl}_X)\big) : Q(X, X, \mathtt{refl}_X) \equiv X \simeq X,$$

the existence of the required function follows from the J-rule (in the next universe $\mathcal{U}'$). □

Another, more informative proof of Proposition 5.6.1 goes as follows. Let the type family $\mathrm{id}_{\mathcal{U}} : \mathcal{U} \to \mathcal{U}'$, where $\mathrm{id}_{\mathcal{U}} \equiv \lambda_{X:\mathcal{U}}.X : \mathcal{U}'$. If $p : A =_{\mathcal{U}} B$, then

$$p_*^{\mathrm{id}_{\mathcal{U}}} : \mathrm{id}_{\mathcal{U}}(A) \to \mathrm{id}_{\mathcal{U}}(B) \equiv A \to B.$$

By Proposition 3.5.4(ii)-(iii) $p_*^{\mathrm{id}_{\mathcal{U}}}$ is an equivalence, such that $(\mathtt{refl}_A)_*^{\mathrm{id}_{\mathcal{U}}} = \mathrm{id}_{\mathrm{id}_{\mathcal{U}}(A)} = \mathrm{id}_A$.

Proposition 5.6.2. *Let $A : \mathcal{U}, P : A \to \mathcal{U}$ and $q : a =_A b$. Then*

$$q_*^P = \mathtt{eqtoequiv}_{P(a),P(b)}\big(\mathtt{ap}_P(q)\big).$$

Proof. Exercise. □

Voevodsky's univalence axiom (UA): The following function is an equivalence in $\mathcal{U}'$

$$\mathtt{eqtoequiv}_{A,B} : A =_{\mathcal{U}} B \to A \simeq B.$$

Let

$$\mathtt{ua}_{A,B} : A \simeq B \to A =_{\mathcal{U}} B$$

be the quasi-inverse to $\mathtt{eqtoequiv}_{A,B}$. By this we mean the following. If

$$\big((g, G), (h, H)\big) : \mathtt{isequiv}(\mathtt{eqtoequiv}_{A,B}),$$

then

$$\mathrm{iq}\big((g, G), (h, H)\big) \equiv (\mathtt{ua}_{A,B}, G', H') : \mathtt{qinv}(\mathtt{ua}_{A,B}).$$

By the definition of the type $\mathtt{qinv}(\mathtt{ua}_{A,B})$ we get

$$\mathtt{ua}_{A,B} \circ \mathtt{eqtoequiv}_{A,B} \sim \mathrm{id}_{A=_{\mathcal{U}}B}$$

i.e.,

$$G' : \prod_{\pi :\, A=_{\mathcal{U}}B} \mathtt{ua}_{A,B}\big(\mathtt{eqtoequiv}_{A,B}(\pi)\big) = \pi.$$

Moreover,

$$\mathtt{eqtoequiv}_{A,B} \circ \mathtt{ua}_{A,B} \sim \mathrm{id}_{A\sim B}$$

i.e.,

$$H' : \prod_{K :\, A\sim B} \mathtt{eqtoequiv}_{A,B}\big(\mathtt{ua}_{A,B}(K)\big) = K.$$

Hence, by G' we get

$$\begin{aligned} \mathtt{refl}_A &= \mathtt{ua}_{A,A}\big(\mathtt{eqtoequiv}_{A,A}(\mathtt{refl}_A)\big) \\ &\equiv \mathtt{ua}_{A,A}\big(\big((\mathrm{id}_A, \mathtt{refl}_A), (\mathrm{id}_A, \mathtt{refl}_A)\big)\big). \end{aligned}$$

Proposition 5.6.3. *Let $A, B : \mathcal{U}$, $p : A =_{\mathcal{U}} B$ and $q : B =_{\mathcal{U}} C$. As $\mathtt{eqtoequiv}_{A,B} : A =_{\mathcal{U}} B \to A \simeq B$ and $\mathtt{eqtoequiv}_{B,C} : B =_{\mathcal{U}} C \to B \simeq C$, we have that*

$$\mathtt{eqtoequiv}_{A,B}(p) : A \simeq B \quad \& \quad \mathtt{eqtoequiv}_{B,C}(q) : B \simeq C,$$

hence

$$\mathtt{ua}_{A,B}\big(\mathtt{eqtoequiv}_{A,B}(p)\big) : A =_{\mathcal{U}} B \quad \& \quad \mathtt{ua}_{B,C}\big(\mathtt{eqtoequiv}_{B,C}(q)\big) : B =_{\mathcal{U}} C.$$

Considering the inverse and the composition of equivalences from Definition 5.2.5, we get

$$\left[\mathtt{eqtoequiv}_{A,B}(p)\right]^{-1}\colon B \simeq A \quad \& \quad \mathtt{eqtoequiv}_{B,C}(q) \circ \mathtt{eqtoequiv}_{A,B}(p)\colon A \simeq C.$$

(i) $\mathtt{ua}_{B,A}\left(\left[\mathtt{eqtoequiv}_{A,B}(p)\right]^{-1}\right) = p^{-1}$.

(ii) $\mathtt{ua}_{A,C}\left(\mathtt{eqtoequiv}_{B,C}(q) \circ \mathtt{eqtoequiv}_{A,B}(p)\right) = p * q$.

Proof. We work similarly to the proof of Proposition 5.3.1. □

Corollary 5.6.4. *Let $A, B\colon \mathcal{U}$, $e\colon A \simeq B$ and $j\colon B \simeq C$. Considering the inverse and the composition of equivalences from Definition 5.2.5, we have that:*
(i) $\left[\mathtt{ua}_{A,B}(e)\right]^{-1} = \mathtt{ua}_{B,A}\left(e^{-1}\right)$.
(ii) $\mathtt{ua}_{A,B}(e) * \mathtt{ua}_{B,C}(j) = \mathtt{ua}_{A,C}(j \circ e)$.

Proof. Exercise. □

Proposition 5.6.5. *Let $A\colon \mathcal{U}, P\colon A \to \mathcal{U}$ and $l\colon a =_A a$. If $f\colon P(a) \to P(a)$ such that $\mathtt{isequiv}(f)$ and $\mathtt{ap}_P(l) = \mathtt{ua}_{P(a),P(a)}(f)$, then*

$$l_*^P = f.$$

Proof. Exercise. □

The universe $\mathcal{U}$ is not a set.

Proposition 5.6.6. $\mathtt{isSet}(\mathcal{U}) \to \mathbf{0}$.

Proof. By Definition 5.5.1

$$\mathtt{isSet}(\mathcal{U}) \equiv \prod_{X,Y\colon \mathcal{U}} \prod_{\pi,\varpi\colon X =_{\mathcal{U}} Y} \pi = \varpi.$$

Clearly, $\mathtt{isequiv}(\mathtt{sw}_{\mathbf{2}})$. Hence, $\mathtt{ua}_{\mathbf{2},\mathbf{2}}(\mathtt{sw}_{\mathbf{2}})\colon \mathbf{2} =_{\mathcal{U}} \mathbf{2}$. Of course, $\mathtt{refl}_{\mathbf{2}}\colon \mathbf{2} =_{\mathcal{U}} \mathbf{2}$ too. Let

$$\mathtt{ua}_{\mathbf{2},\mathbf{2}}(\mathtt{sw}_{\mathbf{2}}) = \mathtt{refl}_{\mathbf{2}}.$$

Hence

$$\mathtt{eqtoequiv}_{\mathbf{2},\mathbf{2}}\left(\mathtt{ua}_{\mathbf{2},\mathbf{2}}(\mathtt{sw}_{\mathbf{2}})\right) = \mathtt{eqtoequiv}_{\mathbf{2},\mathbf{2}}\left(\mathtt{refl}_{\mathbf{2}}\right).$$

Consequently,

$$\mathtt{sw}_{\mathbf{2}} = \mathrm{id}_{\mathbf{2}},$$

which implies $0_{\mathbf{2}} = 1_{\mathbf{2}}$. From that we get a term of type $\mathbf{0}$. □

5.7 Propositional truncation as a higher inductive type

If $A\colon \mathcal{U}$, its (-1)-truncation $||A||\colon \mathcal{U}$ is a mere proposition generated by A by truncating, or squashing, all information regarding the elements of A. In HoTT the truncation $||A|$ of A is defined as a *higher inductive type* i.e., introduction rules for canonical elements and paths between them are included in its definition.

Definition 5.7.1. $\mathrm{Form}_{||A||}$:

$$\frac{A\colon \mathcal{U}}{||A||\colon \mathcal{U}}\ .$$

$\mathrm{Intro}_{||A||}$:

$$\frac{a\colon A}{|a|\colon ||A||}\ , \qquad \frac{c,d\colon ||A||}{p_{c,d}\colon c =_{||A||} d}\ .$$

$\mathrm{Rec}_{||A||}$:

$$\frac{B\colon \mathcal{U},\ b,b'\colon B,\ q_{b,b'}\colon b =_B b',\ f\colon A\to B}{|f|\colon ||A||\to B,\ |f|(|a|)\equiv f(a), a\colon A,\ |f|(p_{c,d}) = q_{|f|(c),|f|(d)}}\ .$$

$\mathrm{Ind}_{||A||}$:

$$\frac{P\colon ||A||\to\mathcal{U},\ G\colon \prod_{x\colon A} P(|x|),\ c,d\colon ||A||,\ u\colon P(c), w\colon P(d),\ q_{c,d}\colon (p_{c,d})^P_*(u) =_{P(d)} w; c,d\colon ||A||}{F\colon \prod_{x\colon ||A||} P(x),\ F(|a|)\equiv G(a); a\colon A,\ F(p_{c,d}) = q_{c,d}}\ .$$

The first introduction rule introduces a function $|.|\colon A\to ||A||$, where

$$|.| \equiv \lambda_{x\colon A}.|x|\colon ||A||.$$

Clearly, if A is inhabited, then $||A||$ is also inhabited. The second introduction rule introduces a term of type

$$\big(\lambda_{x,y\colon ||A||}.p_{x,y}\big)\colon \mathtt{isProp}\big(||A||\big).$$

The rule $\mathrm{Rec}_{||A||}$ says that if B is a competitor type to A i.e., a mere proposition, and if $f\colon A\to B$ competes with $|.|$, then the following diagram commutes

$$\begin{array}{ccc} A & & \\ {\scriptstyle |.|}\downarrow & \searrow^{f} & \\ ||A|| & \xrightarrow[|f|]{} & B. \end{array}$$

In other words, if B follows from A, then it follows from $||A||$ too. The equality $|f|(p_{c,d}) = q_{|f|(c),|f|(d)}$ is a natural compatibility condition that needs to be added, because of the second introduction rule. The rule $\mathrm{Ind}_{||A||}$ is the dependent analogue to the rule $\mathrm{Rec}_{||A||}$. Notice that if P is the constant type family B over A, then by the J-rule we get $(p_{c,d})^P_*(u) =_B u$, and the equality $(p_{c,d})^P_*(u) =_B w$ induces the equality $u =_B w$. As expected, the truncation of a proposition A is shown to be equivalent to A, hence by univalence equal to A.

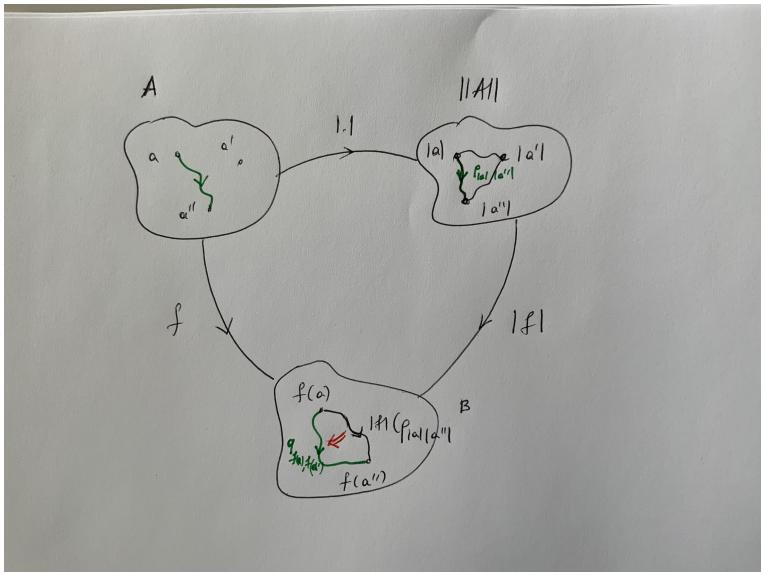


Figure 5.5: $\mathrm{Rec}_{||A||}$

Proposition 5.7.2. *Let* $A : \mathcal{U}$ *such that* $\mathtt{isProp}(A)$. *Then* $A \simeq ||A||$.

Proof. As A and $||A||$ are mere propositions, it suffices to show that $A \leftrightarrow ||A||$. Clearly,

$$\begin{array}{ccc} A & & \\ {\scriptstyle |.|}\downarrow & \searrow^{\mathrm{id}_A} & \\ ||A|| & \xrightarrow[|\mathrm{id}_A|]{} & A. \end{array}$$

$|.| : A \to ||A||$, while $|\mathrm{id}_A| : ||A|| \to A$ □

The 0-truncation of A, which defines a set from A, is defined similarly.

5.8 Propositionally truncated logic

We can use the axiom of univalence, and the propositional truncation $||A||$ of A as a HIT, in order to translate a bounded formula of first-order logic into HoTT. In Table 5.1 A, B are mere propositions in $\mathcal{U}$, S is a set in $\mathcal{U}$, and $P : S \to \mathcal{U}$, such that $P(x)$ is a proposition, for every $x : S$. In this way, all types in HoTT given on Table 5.1 are mere propositions. Notice that $A \Leftrightarrow B$ corresponds to $A \leftrightarrow B$, which is equivalent to $A \simeq B$, as A, B are mere propositions. By `UA` this type is equivalent to $A =_{\mathcal{U}} B$. To show that in this case $A =_{\mathcal{U}} B$ is a proposition, we use `UA` again. By it

$$A =_{\mathcal{U}} B \simeq A \simeq B \equiv \sum_{\phi : A \to B} \mathtt{isequiv}(\phi),$$

and we use the fact that both $A \to B$ and $\mathtt{isequiv}(\phi)$ are mere propositions. Within univalent logic, if $\mathtt{isSet}(A), \mathtt{isSet}(B)$, and if $R : A \to B \to \mathcal{U}$, such that $\mathtt{isProp}(R(a, b)$, for every $a : A$

and $b\colon B$, the set-theoretic axiom of choice is translated as the following type in univalent logic:

$$\Big(\prod_{x\colon A}\Big\|\sum_{y\colon B}R(x,y)\Big\|\Big)\to\Big\|\sum_{\phi\colon A\to B}\prod_{x\colon A}R(x,\phi(x))\Big\|.$$

This type cannot be shown to be inhabited, as the information on the sigma-type is suppressed through truncation.

Corollary 5.8.1 (Unique-choice)**.** *Let $A\colon\mathcal{U}$ and $P\colon A\to\mathcal{U}$, such that* $\mathtt{isProp}(P(x))$ *and $||P(x)||$ is inhabited, for every $x\colon A$. Then the type*

$$\prod_{x\colon A}P(x)$$

is inhabited.

Proof. By Proposition 5.7.2 $P(x)\simeq||P(x)$, for every $x\colon A$. Then a term of type $\prod_{x\colon A}P(x)$ is defined by the inhabitedness of $P(x)$, for every $x\colon A$. □

5.9 The higher inductive type interval

Next we define an abstract notion of a unit interval as a HIT.

Definition 5.9.1. $\mathrm{Form}_{\boldsymbol{I}}$:

$$\frac{}{\boldsymbol{I}\colon\mathcal{U}}\ .$$

$\mathrm{Intro}_{\boldsymbol{I}}$:

$$\frac{}{0_{\boldsymbol{I}}\colon\boldsymbol{I}}\ ,\quad\frac{}{1_{\boldsymbol{I}}\colon\boldsymbol{I}}\ ,\quad\frac{}{\mathtt{seg}\colon 0_{\boldsymbol{I}}=_{\boldsymbol{I}}1_{\boldsymbol{I}}}\ .$$

I.e., the interval $\boldsymbol{I}$ has the canonical terms $0_{\boldsymbol{I}},1_{\boldsymbol{I}}$ and an equality path $\mathtt{seg}$ *between them.*

$\mathrm{Rec}_{\boldsymbol{I}}$:

$$\frac{B\colon\mathcal{U},\ b_0,b_1\colon B,\ p\colon b_0=_B b_1}{f\colon\boldsymbol{I}\to B,\ f(0_{\boldsymbol{I}})\equiv b_0,\ f(1_{\boldsymbol{I}})\equiv b_1,\ f(\mathtt{seg})=p}\ .$$

$\mathrm{Ind}_{\boldsymbol{I}}$:

$$\frac{P\colon\boldsymbol{I}\to\mathcal{U},\ b_0\colon P(0_{\boldsymbol{I}}),\ b_1\colon P(1_{\boldsymbol{I}}),\ p\colon\mathtt{seg}^P_*(b_0)=_{P(1_{\boldsymbol{I}})}b_1}{F\colon\prod_{x\colon\boldsymbol{I}}P(x),\ F(0_{\boldsymbol{I}})\equiv b_0\ F(1_{\boldsymbol{I}})\equiv b_1,\ F(\mathtt{seg})=p}\ .$$

The following obvious equivalence supports the homotopic interpretation of the equality type as a path-type:

$$a=_A b\simeq\sum_{\phi\colon I\to A}\big(f(0_{\boldsymbol{I}})=_A a\big)\times\big(f(1_{\boldsymbol{I}})=_A b\big).$$

As $b_1\equiv\mathtt{seg}^P_*(b_0)\colon P(1_{\boldsymbol{I}})$ and

$$\mathtt{refl}_{b_1}\colon\mathtt{seg}^P_*(b_0)=_{P(1_{\boldsymbol{I}})}b_1,$$

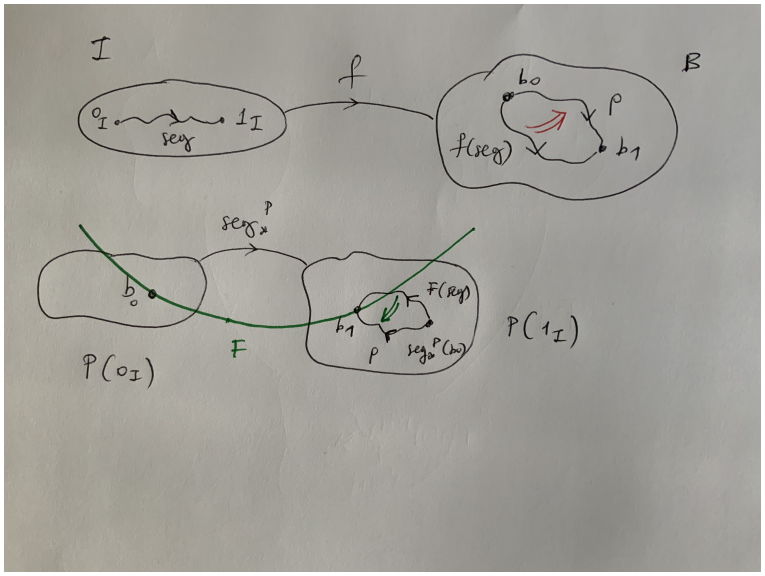


Figure 5.6: $\mathrm{Rec}_{\boldsymbol{I}}$ and $\mathrm{Ind}_{\boldsymbol{I}}$

in order to find

$$F\colon \prod_{x\colon \boldsymbol{I}} P(x),$$

it suffices to find $b_0\colon P(0_{\boldsymbol{I}})$. This remark implies immediately the contractibility of $\boldsymbol{I}$ i.e.,

$$\prod_{x\colon \boldsymbol{I}} 0_{\boldsymbol{I}} =_{\boldsymbol{I}} x.$$

The interval type implies the axiom of function-extensionality.

Theorem 5.9.2. MLTT $+\, \boldsymbol{I} \vdash \mathtt{FunExt}$.

Proof. Let $f, g\colon A \to B$ and $H\colon f \sim g$. By the equivalence

$$f =_{A\to B} g \simeq \sum_{\phi\colon I\to(A\to B)} \phi(0_{\boldsymbol{I}}) = f \times \phi(1_{\boldsymbol{I}}) = g,$$

it suffices to find $h\colon I \to (A \to B)$ such that $h(0_{\boldsymbol{I}}) = f$ and $h(1_{\boldsymbol{I}}) = g$. If $x\colon A$, then $H(x)\colon f(x) =_B g(x)$. By $\mathrm{Rec}_{\boldsymbol{I}}$ there is $h_x\colon \boldsymbol{I} \to B$, such that $h_x(0_{\boldsymbol{I}}) \equiv f(x), h_x(1_{\boldsymbol{I}}) \equiv g(x)$ and $h_x(\mathtt{seg}) = H(x)$. Let

$$h \equiv \lambda_{i\colon I}\lambda_{x\colon A}.h_x(i).$$

Then we get

$$h(0_{\boldsymbol{I}}) \equiv \lambda_{x\colon A}.h_x(0_{\boldsymbol{I}}) \equiv \lambda_{x\colon A}.f(x) \equiv f,$$
$$h(1_{\boldsymbol{I}}) \equiv \lambda_{x\colon A}.h_x(1_{\boldsymbol{I}}) \equiv \lambda_{x\colon A}.g(x) \equiv g.$$

□

Bounded formula	Type in HoTT
$\top$	$\mathbf{1}$
$\bot$	$\mathbf{0}$
$A \wedge B$	$A \times B$
$A \Rightarrow B$	$A \to B$
$A \Leftrightarrow B$	$A =_{\mathcal{U}} B$
$\neg A$	$A \to \mathbf{0}$
$A \vee B$	$\|\|A + B\|\|$
$\forall_{x \in S} P(x)$	$\prod_{x:\, S} P(x)$
$\exists_{x \in S} P(x)$	$\left\|\left\| \sum_{x:\, S} P(x) \right\|\right\|$

Table 5.1: Propositionally truncated logic, or univalent logic

Bibliography

[1] P. Aczel, M. Rathjen: *Constructive Set Theory*, book draft, 2010.

[2] S. Awodey: *Category Theory*, Oxford University Press, 2010.

[3] S. Awodey, M. A. Warren. Homotopy theoretic models of identity types, Mathematical Proceedings of the Cambridge Philosophical Society, 2009, 146:45-55.

[4] E. Bishop: *Foundations of Constructive Analysis*, McGraw-Hill, 1967.

[5] E. Bishop and D. S. Bridges: *Constructive Analysis*, Grundlehren der math. Wissenschaften 279, Springer-Verlag, Heidelberg-Berlin-New York, 1985.

[6] A. Blass: Existence of bases implies the axiom of choice, Contemporary Mathematics, Volume 31, 1984, 31-33.

[7] D. S. Bridges and F. Richman: *Varieties of Constructive Mathematics*, Cambridge University Press, 1987.

[8] G. Cantor: Über unendliche, lineare Punktmannichfaltigkeiten, Nummer 3. Mathematische Annalen, 20, 1882, 113–121.

[9] C. C. Chang, A. H. Keisler: *Model Theory*, Dover, 2012.

[10] T. Coquand: A remark on singleton types, manuscript, 2014.

[11] J. Dugundji: *Topology*, Allyn and Bacon,1966.

[12] H. B. Enderton: *Elements of Set Theory*, Academic Press, 1977.

[13] M. Escardó: Using Yoneda rather than J to present the identity type, preprint, in http://www.cs.bham.ac.uk/∼mhe/yoneda/yoneda.html.

[14] G. Gentzen: Über das Verhältnis zwischen intuitionistischer und klassischer Arithmetik, Galley proof, Mathematische Annalen (received 15th March 1933). First published in English translation in *The collected papers of Gerhard Gentzen*, M.E. Szabo (editor), 53-67, Amsterdam (North-Holland).

[15] G. Gentzen: Untersuchungen über das logische Schließen I, II, Mathematische Zeitschrift, 39, 1935, 176-210, 405-431.

[16] K. Gödel: Zur intuitionistisohen Arithmetik und Zahlentheorie, in : Ergebnisse eines mathematischen Kolloquiums, Heft 4 (for 1931-1932, appeared in 1933), 34-38. Translated into English in *The undecidable*, M. Davis (editor), 75-81, under the title "On intuitionistic arithmetic and number theory". For corrections of the translation see review in J.S.L. 31 (1966), 484-494.

[17] R. Goldblatt: *Topoi, The Categorial Analysis of Logic*, Dover, 2006.

[18] M. Hofmann, T. Streicher: The groupoid interpretation of type theory, G. Sambin, J. M. Smith (Eds.) *Twenty-five years of constructive type theory* (Venice, 1995), volume 36 of Oxford Logic Guides, Oxford University Press, 1998, 83-111.

[19] S. Huber: *Cubical Interpretations of Type Theory*, PhD Thesis, University of Gothenburg, 2016.

[20] C. Kapulkin, P. LeFanu Lumsdaine, V. Voevodsky: The simplicial model of univalent foundations, arXiv:1211.2851, 2012.

[21] K. Kunen: *Set Theory*, Individual author and College Publications, 2013.

[22] T. Jech: *Set theory*, Springer, 2002.

[23] A. N. Kolmogorov. On the principle of the excluded middle (in Russian). Mat. Sb., 32, 1925, 646-667.

[24] J. Lambek, P. J. Scott: *Introduction to higher order categorical logic*, Cambridge University Press, 1986.

[25] P. Martin-Löf: An intuitionistic theory of types: predicative part, in H. E. Rose and J. C. Shepherdson (Eds.) *Logic Colloquium'73*, pp.73-118, North-Holland, 1975.

[26] P. Martin-Löf: *Intuitionistic type theory: Notes by Giovanni Sambin on a series of lectures given in Padua, June 1980*, Napoli: Bibliopolis, 1984.

[27] P. Martin-Löf: An intuitionistic theory of types, in G. Sambin, J. M. Smith (Eds.) *Twenty-five years of constructive type theory* (Venice, 1995), volume 36 of Oxford Logic Guides, Oxford University Press, 1998, 127-172.

[28] C. Paulin-Mohring: Inductive Definitions in the System Coq - Rules and Properties, in M. Bezem, J. F. Groote (Eds.) Proceedings of TLCA, LNM 664, Springer, 1993.

[29] I. Petrakis: *Families of Sets in Bishop Set Theory*, Habilitationsschrift, LMU, 2020.

[30] E. Rijke: *Homotopy Type Theory*, Master Thesis, Utrecht University 2012.

[31] H. Rogers: *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, 1967.

[32] H. Rubin, J. E. Rubin: *Equivalents of the axiom of choice II*, North-Holland, 1985.

[33] H. Schwichtenberg, A. Troelstra: *Basic Proof Theory*, Cambridge University Press 1996.

[34] H. Schwichtenberg, S. Wainer: *Proofs and Computations*, Cambridge University Press 2012.

[35] T. Streicher: *Realizability*, Lecture Notes, TU Darmstadt, 2018.

[36] The Univalent Foundations Program: *Homotopy Type Theory: Univalent Foundations of Mathematics*, Institute for Advanced Study, Princeton, 2013.

[37] V. Voevodsky. A very short note on the homotopy λ-calculus, in http://www.math.ias.edu/∼vladimir/Site3/Univalent_Foundations_files /Hlambda_short_current.pdf, 2006