A BRIEF INTRODUCTION TO CONSTRUCTIVE TYPE THEORY

ANSTEN KLEV

Contents

1. Judgements and categories 2
1.1. Forms of categorical judgement 2
1.2. Categories 4
1.3. General rules of judgemental equality 6
1.4. Propositions 6
1.5. Forms of hypothetical judgement 7
1.6. Assumptions and other speech acts 9
1.7. Hypothetical judgements with more than one assumption 10
2. Rules 12
2.1. Cartesian product of a family of sets 12
2.2. The logical interpretation of the cartesian product 14
2.3. Disjoint union of a family of sets 15
2.4. The logical interpretation of the disjoint union of a family of sets 17
2.5. Disjoint union of two sets 18
2.6. Finite sets 18
2.7. The natural numbers 20
2.8. Propositional identity 21
Exercises 26
Solutions 27
References 32

Martin-Löf's constructive type theory is a formal language developed in order to reason constructively about mathematics. It is thus a formal language conceived primarily as a tool to reason *with* rather than a formal language conceived primarily as a mathematical system to reason *about*. Constructive type theory is therefore much closer in spirit to Frege's ideography and the language of Russell and Whitehead's *Principia Mathematica* than to the majority of logical systems ("logics") studied by contemporary logicians. Since constructive type theory is designed as a language to reason with, much attention is paid to the explanation of basic concepts. This is perhaps the main reason why the style of presentation of constructive type theory differs somewhat from the style of presentation typically found in, for instance, ordinary logic textbooks. For those new to the system it might be useful

to approach an introduction such as the one given below more as a language course than as a course in mathematics.

1. Judgements and categories

Statements made in constructive type theory are called *judgements*. *Judgement* is thus a technical term, chosen because of its long pedigree in the history logic (cf. e.g. Martin-Löf 1996, 2011 and Sundholm 2009). Judgement thus understood is a logical notion and not, as it is commonly understood in contemporary philosophy, a psychological notion. As in traditional logic, a judgement may be categorical or hypothetical. Categorical judgements are conceptually prior to hypothetical judgements, hence we must begin by explaining them.

1.1. **Forms of categorical judgement.** There are two basic forms of categorical judgement in constructive type theory:

$$a : \mathscr{C}$$
$$a = b : \mathscr{C}$$

The first is read "a is an object of the category $\mathscr{C}$" and the second is read "a and b are identical objects of the category $\mathscr{C}$". Ordinary grammatical analysis of $a : \mathscr{C}$ yields a as subject, $\mathscr{C}$ as predicate, and the colon as a copula. We thus call the predicate $\mathscr{C}$ in $a : \mathscr{C}$ a *category*. This use of the term 'category' is in accordance with one of the original meanings of the Greek *katēgoria*, namely as predicate. It is also in accordance with a common use of the term 'category' in current philosophy.[1] We require, namely, that any category $\mathscr{C}$ occurring in a judgement of constructive type theory be associated with

- a criterion of application, which tells us what a $\mathscr{C}$ is; that a meets this criterion is precisely what is expressed in $a : \mathscr{C}$;
- a criterion of identity, which tells us what it is for a and b to be identical $\mathscr{C}$'s; that a and b together meet this criterion is precisely what is expressed in $a = b : \mathscr{C}$.

What the categories of constructive type theory are will be explained below.

In constructive type theory any object belongs to a category. The theory recognizes something as an object only if it can appear in a judgement of the form $a : \mathscr{C}$ or $a = b : \mathscr{C}$. Since with any category there is associated a criterion of identity, we can recover Quine's precept of "no entity without identity" (Quine, 1969, p. 23) as

> no object without category +
> no category without a criterion of identity.

Thus we derive Quine's precept from two of the fundamental principles of constructive type theory. We shall have more to say later about the treatment of identity in constructive type theory.

[1] See, in particular, the definition of category given by Dummett (1973, pp. 75–76), a definition that has been taken over by, for instance, Hale and Wright (2001).

Neither semantically nor syntactically does $a : \mathscr{C}$ agree with the basic form of statement in predicate logic:

$$F(a)$$

In $F(a)$ a function F is applied to an argument a (in general there may be more than one argument). The judgement $a : \mathscr{C}$, by contrast, does not have function–argument form. In fact, the '$a : \mathscr{C}$'-form of judgement is closer to the 'S is P'-form of traditional, syllogistic logic than to the function–argument form of modern, Fregean logic. Since we have required that the predicate $\mathscr{C}$ be associated with criteria of application and identity, the judgement $a : \mathscr{C}$ can, however, be compared only with a special case of the 'S is P'-form, for no such requirement is in general laid on the predicate P in a judgement of Aristotle's syllogistics—it can be any general term. To understand the restriction that P be associated with criteria of application and identity in terms of traditional logic, we may invoke Aristotle's doctrine of predicables from the *Topics*. A predicable may be thought of as a certain relation between the S and the P in an 'S is P'-judgement. Aristotle distinguishes four predicables: genus, definition, *idion* or *proprium*, and accident. That P is a genus of S means that P reveals a *what*, or a *what-it-is*, of the subject S; a genus of S may thus be proposed in answer to a question of what S is. The class of judgements of Aristotelian syllogistics to which judgements of the form $a : \mathscr{C}$ may be compared is the class of judgements whose predicate is a genus of the subject. Provided the judgement $a : \mathscr{C}$ is correct, the category $\mathscr{C}$ is namely an answer to the question of what a is; we may thus think of $\mathscr{C}$ as the genus of a. Aristotle's other predicables will not concern us here.

Being a natural number is in a clear sense a *what* of 7. The number 7 is also a prime number; but being prime is not a *what* of 7 in the sense that being a natural number is, even though 7 is necessarily, and perhaps even essentially, a prime number. Following Almog (1991) we may say that being prime is one of the *how*s of 7. This difference between the *what* and the *how* of a thing captures quite well the difference in semantics between a judgement $a : \mathscr{C}$ of constructive type theory and a sentence $F(a)$ of predicate logic. In the predicate-logical language of arithmetic we do not express the fact that 7 is a number by means of a sentence of the form $F(a)$. That the individual terms of the language of arithmetic denote numbers is rather a feature of the interpretation of the language that we may express in the metalanguage.[2] We do, however, say in the language of arithmetic that 7 is prime by means of a sentence of the form $F(a)$, for instance as $\mathbf{Pr}(7)$. It is therefore natural to suggest that by means of the form of statement $F(a)$ we express a *how*, but not the *what*, of the object a. The opposite holds for the form of statement $a : \mathscr{C}$—by means of this we express the *what*, but not the *how*, of the object a. Thus, in constructive type theory we do say that 7 is a number by means of a judgement, namely as $7 : \mathbf{N}$, where $\mathbf{N}$ is the category of natural numbers; but we do not say that 7 is prime by means of a similar judgement such as $7 : \mathbf{Pr}$. Precisely how we express in constructive type theory that 7 is prime will

[2]Compare Carnap's treatment of what he calls *Allwörter* ('universal words' in the English translation) in §§ 76, 77 of *Logische Syntax der Sprache* (Carnap, 1934).

become clear only later; it will then be seen that we express the primeness of 7 by a judgement of the form

$$p : \mathbf{Pr}(7)$$

where $\mathbf{Pr}(7)$ is a *proposition* and p is a *proof* of this proposition. The proposition $\mathbf{Pr}(7)$ has function–argument form, just as the atomic sentences of ordinary predicate logic.

1.2. **Categories.** The forms of judgement $a : \mathscr{C}$ and $a = b : \mathscr{C}$ are only schematic forms. The specific forms of categorical judgement employed in constructive type theory are obtained from these schematic forms by specifying the categories of the theory. There is then a choice to be made, namely between what may be called a higher-order and a lower-order presentation of the theory. The higher-order presentation results in a conceptually somewhat cleaner theory, but for pedagogical purposes the lower-order presentation is preferable, both because it requires less machinery and because it is the style of presentation found in the standard references of Martin-Löf (1975b, 1982, 1984) and Nordström et al. (1990, chs. 4–16). We shall therefore follow this style of presentation. The categories are then the following. There is a category **set** of sets in the sense of Martin-Löf; and for any **set** A, A itself is a category. We therefore have the following four forms of categorical judgement:

$$A : \mathbf{set}$$
$$A = B : \mathbf{set}$$

and for any **set** A,

$$a : A$$
$$a = b : A$$

In the higher-order presentation the categories are **type** and α, for any **type** α. The higher-order presentation in a sense subsumes the lower-order presentation, since we have there, firstly, as an axiom **set** : **type**, hence **set** itself is a category; and secondly, there is a rule to the effect that if A : **set**, then A : **type**, hence also any **set** A will be a category. The higher-order presentation can be found in (Nordström et al., 1990, chs. 19–20) and (Nordström et al., 2000).

We have so far only given names to our categories. To justify calling **set** as well as any **set** A a category we must specify the criteria of application and identity of **set** and of A, for any **set** A. Thus we have to explain four things: what a **set** is, what identical **set**s are, what an element of a **set** A is, and what identical elements of a **set** A are. By giving these explanations we also explain the four forms of categorical judgement $A : \mathbf{set}$, $A = B : \mathbf{set}$, $a : A$, and $a = b : A$. Our explanations follow those given by Martin-Löf (1984, pp. 7–10)

We explain the form of judgement A : **set** as follows. A **set** A is defined by saying what a *canonical* element of A is and what equal canonical elements of A are. (Instead of 'canonical element' one can also say 'element of canonical form'.) What the canonical elements are, as well as what equal canonical elements are, of a **set** A is determined by the so-called introduction rules associated with A. For instance, the introduction rules associated with the **set** of natural numbers **N** are

as follows.

$$0 : \mathbf{N} \qquad 0 = 0 : \mathbf{N} \qquad \frac{n : \mathbf{N}}{\mathbf{s}(n) : \mathbf{N}} \qquad \frac{n = m : \mathbf{N}}{\mathbf{s}(n) = \mathbf{s}(m) : \mathbf{N}}$$

By virtue of these rules 0 is a canonical element of $\mathbf{N}$, as is $\mathbf{s}(n)$ provided n is a $\mathbf{N}$, which does not have to be canonical. Moreover, 0 is the same canonical element of $\mathbf{N}$ as 0, and $\mathbf{s}(n)$ is the same canonical element of $\mathbf{N}$ as $\mathbf{s}(m)$ provided $n = m : \mathbf{N}$. It is required that the specification of what equal canonical elements of a **set** A are renders this relation reflexive, symmetric, and transitive.

The form of judgement $A = B$: **set** means that from a's being a canonical element of A we may infer that a is also a canonical element of B, and *vice versa*; and that from a and b's being identical canonical elements of A we may infer that they are also identical canonical elements of B, and *vice versa*.

Thus we have given the criteria of application and identity for the category **set**.

Suppose that A is a **set**. Then we know how the canonical elements of A are formed as well as how equal canonical elements of A are formed. The judgement $a : A$ means that a is a programme which, when executed, evaluates to a canonical element of A. For instance, once one has introduced the addition function, $+$, and the definitions $1 = \mathbf{s}(0) : \mathbf{N}$ and $2 = \mathbf{s}(1) : \mathbf{N}$, one can see that $2 + 2$ is an element of $\mathbf{N}$, since it evaluates to $\mathbf{s}(2+1)$, which is of canonical form. A canonical element of a **set** A evaluates to itself; hence, any canonical element of A is an element of A.

The judgement $a = b : A$ presupposes the judgements $a : A$ and $b : A$. Hence, if we can make the judgement $a = b : A$, then we know that both a and b evaluate to canonical objects of A. The judgement $a = b : A$ means that a and b evaluate to equal canonical elements of A. The value of a canonical element a of a **set** A is taken be a itself. Hence, if b evaluates to a, then we have $a = b : A$.

Thus we have given the criteria of application and identity for the category A, for any **set** A.

A note on terminology is here in order. 'Set' is the term used by Martin-Löf from (Martin-Löf, 1984) onwards for what in earlier writings of his were called types.[3] A set in the sense of Martin-Löf is a very different thing from a set in the sense of ordinary axiomatic set theory. In the latter sense a set is typically conceived of as an object belonging to the cumulative hierarchy V. It is, however, this hierarchy V itself rather than any individual object belonging to V that should be regarded as a set in the sense of Martin-Löf. A set in the sense of Martin-Löf is in effect a domain of individuals, and V is precisely a domain of individuals. That was certainly the idea of Zermelo in his paper on models of set theory (Zermelo, 1930): he there speaks of such models as *Mengenbereiche*, domains of sets. And Aczel (1978) has defined a set in the sense of Martin-Löf that is "a type theoretic reformulation of the classical conception of the cumulative hierarchy of types" (ibid. p. 61). It is

[3]This older terminology is retained for instance in Homotopy Type Theory (The Univalent Foundations Program, 2013); what is there called a set (ibid., Definition 3.1.1) is only a special case of a set in Martin-Löf's sense, namely a set over which every identity proposition has at most one proof.

in order to mark this difference in conception that we denote a set in the sense of Martin-Löf with boldface type, thus writing '**set**'.[4]

1.3. **General rules of judgemental equality.** Recall that it is required when defining a **set** A that the relation of being equal canonical elements then specified be reflexive, symmetric, and transitive. From the explanation of the form of judgement $a = b : A$ it is then easy to see that the relation of so-called judgemental identity, namely the relation expressed to hold between a and b by means of the judgement $a = b : A$, is also reflexive, symmetric, and transitive. Thus the following three rules are justified.

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

The explanation of the form of judgement $A = B : \mathbf{set}$ justifies the same rules at the level of **set**s.

$$\frac{A : \mathbf{set}}{A = A : \mathbf{set}} \qquad \frac{A = B : \mathbf{set}}{B = A : \mathbf{set}} \qquad \frac{A = B : \mathbf{set} \qquad B = C : \mathbf{set}}{A = C : \mathbf{set}}$$

They also justify the following two important rules.

$$\frac{a : A \qquad A = B : \mathbf{set}}{a : B} \qquad \frac{a = b : A \qquad A = B : \mathbf{set}}{a = b : B}$$

1.4. **Propositions.** The notion of proposition has already been alluded to above; and it is reasonable to expect that a system of logic should give some account of this notion. In constructive type theory there is a category **prop** of propositions. The reason this category was not explicitly introduced above is that it is identified in constructive type theory with the category **set**. Thus we have

$$\mathbf{prop} = \mathbf{set}$$

The identification of these two categories[5] is the manner in which the so-called Curry–Howard isomorphism (cf. Howard, 1980) is implemented in constructive type theory. This "isomorphism" is one of the fundamental principles on which the theory rests.

When regarding A as a proposition, the elements of A are thought of as the proofs of A. Thus *proof* is employed as a technical term for elements of propositions. A proposition is, accordingly, identified with the **set** of its proofs. That a proposition is true means that it is inhabited.

By the identification of **set** and **prop** the meaning explanation of the four basic forms of categorical judgement carries over to the explanation of the similar forms

$$\begin{aligned} A &: \mathbf{prop} \\ A = B &: \mathbf{prop} \\ a &: A \\ a = b &: A \end{aligned}$$

[4]For more discussion of the difference between Martin-Löf's and other notions of set, see Granström (2011, pp. 53–63) and Klev (2014, pp. 138–140).

[5]In the higher-order presentation this identification can be made in the language itself, namely as the judgement $\mathbf{prop} = \mathbf{set} : \mathbf{type}$.

To define a **prop** one must lay down what are the canonical proofs of A and what are identical canonical proofs of A. That the propositions A and B are identical means that from a's being a canonical proof of A we may infer that it is also a canonical proof of B, and *vice versa*; and that from a and b's being identical canonical proofs of A we may infer that they are also identical canonical proofs of B, and *vice versa*. Thus, by the identification of **set** and **prop** we get for free a criterion of identity for propositions. That a is a proof of A means that a is a method which, when executed, evaluates to a canonical proof of A. That a and b are identical proofs of A means that a and b evaluate to identical canonical proofs of A. Thus we have provided a criterion of identity for proofs.

Let us illustrate the concept of a canonical proof in the case of conjunction. A canonical proof of $A \wedge B$ is a proof that ends in an application of $\wedge$-introduction

$$\frac{\begin{array}{cc}\mathcal{D}_1 & \mathcal{D}_2 \\ A & B\end{array}}{A \wedge B}$$

where $\mathcal{D}_1$ is a proof of A and $\mathcal{D}_2$ a proof of B. An example of a non-canonical proof is therefore

$$\frac{\begin{array}{cc}\mathcal{D}_1 & \mathcal{D}_2 \\ C \supset (A \wedge B) & C\end{array}}{A \wedge B}$$

where $\mathcal{D}_1$ is a proof of $C \supset (A \wedge B)$ and $\mathcal{D}_2$ a proof of C.

The proofs occurring in the above illustration are in tree form. Proofs in the technical sense of constructive type theory are not given in tree form, but rather as the subjects a of judgements of the form $a : A$, where A is a **prop**. Proofs in this sense are in effect terms in a certain rich typed lambda-calculus and they are often called proof objects (this term was introduced by Diller and Troelstra 1984).

We may introduce a new form of judgement 'A true' governed by the following rule of inference

$$\frac{a : A}{A \text{ true}}$$

Thus, provided we have found a proof a of A, we may infer A true. The conclusion A true can be seen as suppressing the proof a of A displayed in $a : A$.

1.5. **Forms of hypothetical judgement.** One of the characteristic features of constructive type theory is that it recognizes hypothetical judgement as a form of statement distinct from the assertion of the truth of an implicational proposition $A \supset B$. In fact, hypothetical judgements are fundamental to the theory. It is, for instance, hypothetical judgements that give rise to the various dependency structures in constructive type theory, by virtue of which it is a dependent type theory.

Assume A : **set**. Then we have the following four forms of hypothetical judgement with one assumption:

$$\begin{aligned} &x : A \vdash B : \mathbf{set} \\ &x : A \vdash B = C : \mathbf{set} \\ &x : A \vdash b : B \\ &x : A \vdash b = c : B \end{aligned}$$

We have used the turnstile symbol, $\vdash$, to separate the antecedent, or assumption, of the judgement from the consequent. In (Martin-Löf, 1984) the notation used is

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

for what we here write $x : A \vdash B : \mathbf{set}$. We read this judgement as "B is a **set** under the assumption $x : A$". Similar remarks apply to the other three forms of hypothetical judgement. Let us consider the more precise meaning explanations of these forms of judgement.

A judgement of the form $x : A \vdash B : \mathbf{set}$ means that

> $B[a/x] : \mathbf{set}$ whenever $a : A$, and
> $B[a/x] = B[a'/x] : \mathbf{set}$ whenever $a = a' : A$.

Here '$B[a/x]$' signifies the result of substituting 'a' for 'x' in 'B'. Thus we may think of B as a function from A into **set**; or using a different terminology, B may be thought of as a family of **set**s over A. We are assuming that x is the only free variable in B and that A contains no free variables, hence that the judgement $A : \mathbf{set}$ holds categorically, that is, under no assumptions. It follows that $B[a/x]$ is a closed term, hence that $B[a/x] : \mathbf{set}$ holds categorically; by the explanation given of the form of categorical judgement $A : \mathbf{set}$ we therefore know the meaning of $B[a/x] : \mathbf{set}$. Thus we see that the meaning of a hypothetical judgement is explained in terms of the meaning of categorical judgements. It holds in general that the meaning explanation of hypothetical judgements is thus reduced to the meaning explanation of categorical judgements.

The explanation of the form of judgement $x : A \vdash B : \mathbf{set}$ justifies the following two rules.

$$\frac{a : A \qquad x : A \vdash B : \mathbf{set}}{B[a/x] : \mathbf{set}} \qquad \frac{a = a' : A \qquad x : A \vdash B : \mathbf{set}}{B[a/x] = B[a'/x] : \mathbf{set}}$$

Note that by the second rule here, substitution into **set**s is extensional with respect to judgemental identity.

We note that the notion of substitution is here understood only informally and that the notation $B[a/x]$ belongs to the metalanguage. The notion of substitution can be made precise, and a notation for substitution introduced into the language of constructive type theory itself; but it would take us too far afield to get into the details of that (cf. Martin-Löf 1992 and Tasistro 1993).

A judgement of the form $x : A \vdash B = C : \mathbf{set}$ means that

> $B[a/x] = C[a/x] : \mathbf{set}$ whenever $a : A$.

Hence, in this case we may think of B and C as identical families of **set**s over A. The explanation justifies the following rule.

$$\frac{a : A \qquad x : A \vdash B = C : \mathbf{set}}{B[a/x] = C[a/x] : \mathbf{set}}$$

A judgement of the form $x : A \vdash b : B$ means that

> $b[a/x] : B[a/x]$ whenever $a : A$, and
> $b[a/x] = b[a'/x] : B[a/x]$ whenever $a = a' : A$.

Here we are presupposing $x : A \vdash B : \mathbf{set}$, hence we know that $B[a/x] : \mathbf{set}$ whenever $a : A$, and therefore we also know the meaning of $b[a/x] : B[a/x]$ and $b[a/x] = b[a'/x] : B[a/x]$ whenever $a : A$ and $a = a' : A$. The judgement $x : A \vdash b : B$ can be understood as saying that b is a function from A into the family B; that is to say, b is a function that for any $a : A$ yields an element $b[a/x]$ of the **set** $B[a/x]$. The explanation justifies the following two rules.

$$\frac{a : A \qquad x : A \vdash b : B}{b[a/x] : B[a/x]} \qquad \frac{a = a' : A \qquad x : A \vdash b : B}{b[a/x] = b[a'/x] : B[a/x]}$$

Note that by the second rule here, substitution into elements of **set**s is extensional with respect to judgemental identity.

A judgement of the form $x : A \vdash b = c : B$ means that

$$b[a/x] = c[a/x] : B[a/x] \text{ whenever } a : A.$$

Thus, in this case, b and c are identical functions into the family B. The explanation justifies the following rule.

$$\frac{a : A \qquad x : A \vdash b = c : B}{b[a/x] = c[a/x] : B[a/x]}$$

1.6. **Assumptions and other speech acts.** All the three notions of proposition, categorical judgement and hypothetical judgement can be seen to be presupposed by what is arguably the most natural interpretation of natural deduction derivations (cf. Sundholm, 2006). Consider the following natural deduction proof sketch:

$$\dfrac{\dfrac{\begin{array}{c}A\\ \mathcal{D}_1\\ B\end{array}}{A \supset B} \qquad \begin{array}{c}\mathcal{D}_2\\ A\end{array}}{B}$$

Here $\mathcal{D}_1$ is a proof of B from A, and $\mathcal{D}_2$ is a closed proof of A. Let us regard this natural deduction proof sketch as a representation of an actual mathematical demonstration and let us consider which speech acts the individual formulae here then represent. The topmost A represents an assumption, namely the assumption that the proposition A is true. The formula A that is the conclusion of $\mathcal{D}_2$ is the conclusion of a closed proof; this formula therefore represents the categorical judgement, or assertion, that A is true; the same considerations apply to $A \supset B$ and to the final conclusion B. The B that is the conclusion of $\mathcal{D}_1$ represents neither an assumption nor a categorical assertion; it rather represents a hypothetical judgement, namely the judgement that B is true on the hypothesis that A is true. Let us also note the formula A occurring as a subformula in $A \supset B$. In the given proof sketch this formula A represents neither an assumption nor a categorical assumption nor a hypothetical judgement. It rather represents a proposition that is a part of a more complex proposition $A \supset B$, which in the given proof is asserted categorically to be true.

Thus we see that in order to make the semantics of natural deduction derivations explicit we should employ a notation that is able to distinguish not only propositions from judgements, but also categorical judgements from hypothetical judgements, and perhaps also assumptions from all of these. Assumptions can, however, be

subsumed under hypothetical judgements, since we may regard the assumption of some categorical judgement J as the assertion of J on the hypothesis that J. In particular, the assumption of $a : A$ and the assumption that the proposition A is true may be analyzed as

$$a : A \vdash a : A \quad \text{and} \quad A \text{ true} \vdash A \text{ true}$$

respectively. In constructive type theory one can therefore make the semantics of the above natural deduction proof sketch explicit as follows

$$\dfrac{\dfrac{\begin{array}{c}A \text{ true} \vdash A \text{ true} \\ \mathcal{D}_1 \\ A \text{ true} \vdash B \text{ true}\end{array}}{A \supset B \text{ true}} \qquad \begin{array}{c}\mathcal{D}_2 \\ A \text{ true}\end{array}}{B \text{ true}}$$

From the meaning explanation of hypothetical judgements it is clear that the following rule is justified.

$$\frac{A : \mathbf{set}}{x : A \vdash x : A}$$

Nordström et al. (1990, p. 37) call this the rule of assumption, since it in effect allows us to introduce assumptions.

1.7. **Hypothetical judgements with more than one assumption.** The forms of hypothetical judgement where the number of hypotheses is $n > 1$ are explained by induction on n. We consider the case of $n = 2$ for illustration. We assume that $A_1 : \mathbf{set}$ and $x : A_1 \vdash A_2 : \mathbf{set}$. Thus A_1 is a **set** categorically, while A_2 is a family of **set**s over A_1. The four forms of judgement to be considered are the following.

$$\begin{aligned} x : A_1, x_2 : A_2 &\vdash B : \mathbf{set} \\ x : A_1, x_2 : A_2 &\vdash B = C : \mathbf{set} \\ x : A_1, x_2 : A_2 &\vdash b : B \\ x : A_1, x_2 : A_2 &\vdash b = c : B \end{aligned}$$

The first of these judgements means that $B[a_1/x, a_2/x_2] : \mathbf{set}$ whenever $a_1 : A_1$ and $a_2 : A_2[a_1/x_1]$ and that $B[a_1/x_1, a_2/x_2] = B[a_1'/x, a_2'/x_2] : \mathbf{set}$ whenever $a_1 = a_1' : A_1$ and $a_2 = a_2' : A_2$. Note that A_2 here in general may be a family of **set**s over A_1. Which member of the family the second argument a_2 is taken from depends on the first argument a_1. Thus B is a family of **set**s over A_1 and A_2, where A_2 itself may be a family of **set**s over A_1.

The meaning of the third judgement is that $b[a_1/x, a_2/x_2] : B[a_1/x, a_2/x_2]$ whenever $a_1 : A_1$ and $a_2 : A_2[a_1/x_1]$, and that $b[a_1/x, a_2/x_2] = b[a_1'/x, a_2'/x_2] : B[a_1/x, a_2/x_2]$ whenever $a_1 = a_1' : A_1$ and $a_2 = a_2' : A_2$. Thus b is a binary function whose first argument is an element of A_1; if this element is a_1, then the second argument is an element of $A_2[a_1/x_1]$; if the second argument is a_2, then the value $b[a_1/x, a_2/x_2]$ is an element of $B[a_1/x, a_2/x_2]$. Here one sees the complex dependency structures that can be expressed in constructive type theory.

It should be clear how the explanation of the second and fourth forms of judgement above, as well as the explanation for arbitrary n, should go.

Let J be any categorical judgement, that is, a judgement of one of the forms $B : \mathbf{set}$, $B = C : \mathbf{set}$, $b : B$, $b = b' : B$. In a hypothetical judgement

$$x_1 : A_1, \ldots, x_n : A_n \vdash J$$

we call the sequence of hypotheses $x_1 : A_1, \ldots, x_n : A_n$ a *context*. A judgement of the form

$$x_1 : A_1, \ldots, x_n : A_n \vdash B : \mathbf{set}$$

may thus be expressed by saying that B is a **set** in the context $x_1 : A_1, \ldots, x_n : A_n$. Let Γ be a context. From the meaning explanation of hypothetical judgements one sees that rules of the following kind are justified.

$$\frac{\Gamma \vdash B : \mathbf{set} \qquad \Gamma \vdash J}{\Gamma, y : B \vdash J}$$

These rules may be called rules of weakening, in accordance with the terminology used in sequent calculus.

With the general hypothetical form of judgement explained we may introduce a notion of category in a wider sense, in effect what is called a category in (Martin-Löf, 1984, pp. 21–23). Let us write the four general forms of judgement in the style of Martin-Löf, namely as follows.

$$\begin{aligned} B &: \mathbf{set} \;\; (x_1 : A_1, \ldots, x_n : A_n) \\ B = C &: \mathbf{set} \;\; (x_1 : A_1, \ldots, x_n : A_n) \\ b &: B \;\; (x_1 : A_1, \ldots, x_n : A_n) \\ b = c &: B \;\; (x_1 : A_1, \ldots, x_n : A_n) \end{aligned}$$

In a grammatical analysis of the first of these it is natural to view not only **set** but everything that is to the right of the colon, namely

$$\mathbf{set} \;\; (x_1 : A_1, \ldots, x_n : A_n)$$

as the predicate. The relation between the notions of predicate and category thus suggests that we may regard this as a category. Indeed, this may be regarded as the category of families of **set**s in n variables ranging over the **set**s or families of **set**s $A_1, \ldots, A_n$, among which there may be dependency relations as explained for the case of $n = 2$ above. Likewise we may regard

$$B \;\; (x_1 : A_1, \ldots, x_n : A_n)$$

as a category. It is the category of n-ary functions from $A_1, \ldots, A_n$ into the family B (again keeping dependency relations in mind).

Thus we may extend the notion of category to include not only **set** and A for any **set** A, but also n-ary families of **set**s and n-ary functions into a **set** A. Note that these are indeed categories in the present sense since they are associated with criteria of application and identity, namely through the explanation of the general forms of hypothetical judgement.

2. Rules

So far we have only the frame of a language, namely an explanation of its basic forms of statement as well as explanations of the basic notions of set, proposition, element of a set, and proof of a proposition. The frame is filled by the introduction of symbols signifying sets, operations for forming sets, and operations for forming elements of sets. These symbols are not explained one by one, but rather in groups. The meaning of the symbols in a given group is determined by rules of four kinds:

- Formation rules
- Introduction rules
- Elimination rules
- Equality, or computation, rules

The inclusion of formation rules in the language itself is a distinctive feature of constructive type theory. The introduction- and elimination rules are like those of Gentzen (1933), though generalized to the syntax of constructive type theory so as also to cover the construction of proof objects. The equality rules correspond to the reduction rules of Prawitz (1965). The best way of getting a grip on these notions is by looking at concrete examples, which we now proceed to do.

In the following we shall in most cases write $A[b, c]$ and $a[b, c]$, etc., instead of $A[b/x, c/y]$ and $a[b/x, c/y]$, etc. That is, for ease of readability we shall usually omit to mention the variables for which b, c, etc. are substituted in A, a, etc. Which variables are replaced will usually be clear from the context. Although variables are not mentioned, square brackets will still stand for substitution and not for function application.

2.1. **Cartesian product of a family of sets.** Given a **set** A and a family B of **set**s over A we can form the product of B over A. That is the content of the Π-formation rule:

$$\text{(Π-form)} \qquad \frac{A : \mathbf{set} \qquad x : A \vdash B : \mathbf{set}}{(\Pi x : A)B : \mathbf{set}}$$

This rule lays down the conditions for when we can judge that $(\Pi x : A)B$ is a **set**. There is a second Π-formation rule that lays down the conditions for when we can judge that two sets of the form $(\Pi x : A)B$ are identical:

$$\frac{A = A' : \mathbf{set} \qquad x : A \vdash B = B' : \mathbf{set}}{(\Pi x : A)B = (\Pi x : A')B' : \mathbf{set}}$$

All formation-, introduction-, and elimination rules are paired with identity rules of this kind, but we shall state these rules explicitly only in the present case of Π.

The conclusion of Π-formation says that $(\Pi x : A)B$ is **set**. Since we have the right to judge that C is a **set** only if we can say what the canonical elements of C are as well as what equal canonical elements of C are, we see that the rule of Π-formation requires justification.

The required justification is provided by the Π-introduction rules:

$$\text{(Π-intro)} \qquad \frac{x : A \vdash b : B}{\lambda x.b : (\Pi x : A)B} \qquad \frac{x : A \vdash b = b' : B}{\lambda x.b = \lambda x.b' : (\Pi x : A)B}$$

According to this rule a canonical element of $(\Pi x : A)B$ has the form $\lambda x.b$, where $b[a] : B[a]$ whenever $a : A$. Note that such a b is of a category different from the category of $\lambda x.b$. Namely, b is of category B $(x : A)$ whereas $\lambda x.b$ is of category $(\Pi x : A)B$. It was noted above that we may regard such a b as a function from A into the family B. We may think of $\lambda x.b$ as an individual that codes this function. The λ-operator is thus similar to Frege's course-of-values operator (cf. e.g. Frege, 1893, § 9) which, given a function $f(x)$, yields an individual $\grave{\alpha} f(\alpha)$. Note, however, that $\lambda x.b$ belongs to a separate **set** $(\Pi x : A)B$ and not to the domain A of the function b; whence we cannot make sense of applying the function b to $\lambda x.b$, hence a contradiction along the lines of Russell's Paradox cannot be derived.

The role of the elements of $(\Pi x : A)B$ as codes of functions is made clear by the Π-elimination rule:

$$\text{(Π-elim)} \qquad \frac{c : (\Pi x : A)B \qquad a : A}{\mathbf{ap}(c, a) : B[a]} \qquad \frac{c = c' : (\Pi x : A)B \qquad a = a' : A}{\mathbf{ap}(c, a) = \mathbf{ap}(c', a') : B[a]}$$

The conclusion of this rule asserts that $\mathbf{ap}(c, a)$ is an element of the **set** $B[a]$. Since we have the right to judge that c is an element of a **set** C only if we can specify how to compute c to a canonical element of C, we see that the rule of Π-elimination requires justification.

The required justification is provided by the rule of Π-equality, which specifies how $\mathbf{ap}(c, a)$ is computed in the case where c is of canonical form, namely $\lambda x.b$.

$$\text{(Π-eq)} \qquad \frac{x : A \vdash b : B \qquad a : A}{\mathbf{ap}(\lambda x.b, a) = b[a] : B[a]}$$

We can now justify Π-elimination as follows. By the assumption $c : (\Pi x : A)B$ we know how to evaluate c to canonical form $\lambda x.b$, where $x : A \vdash b : B$; thus we have $c = \lambda x.b : (\Pi x : A)B$. But then also $\mathbf{ap}(c, a) = \mathbf{ap}(\lambda x.b, a) : B[a]$, so $\mathbf{ap}(c, a) = b[a] : B[a]$, whence the value of $\mathbf{ap}(c, a)$ is equal to the value of $b[a]$; by the assumption $x : A \vdash b : B$ we know how to find this value.

From the Π-equality rule we see that **ap** is an application operator; as such it is similar to the function $x^\frown y$, satisfying the equation $d^\frown \grave{\alpha} f(\alpha) = f(d)$, defined by Frege (1893, § 34).

We have now seen that the Π-introduction rules make possible the justification of the Π-formation rule and that the Π-equality rule makes possible the justification of the Π-elimination rule. These relations of justification hold in general and not only in the case of Π.

The advantage of the higher-order presentation of constructive type theory is most readily seen when we ask about the categories of Π, λ, and **ap**. Intuitively we may think of Π as a certain higher-order function that takes a **set** A and a family of **set**s B over A and yields a **set** $(\Pi x : A)B$. But we have no means of naming the category of such a function in the language frame introduced here. In the higher-order presentation such a name is easily constructed; indeed we then express the category assignment of Π by means of the judgement $\Pi : (X : \mathbf{set})((X)\mathbf{set})\mathbf{set}$. Similar remarks apply to λ and **ap**, and in fact to all of the various symbols that

we are now in the process of introducing into the language (apart from the constant **set**s $\mathbf{N}_n$ and $\mathbf{N}$ to be introduced below—these are of category **set**).

2.2. **The logical interpretation of the cartesian product.** Recall that **prop** = **set**. Hence we may regard a family B of **set**s over a **set** A as a family of propositions over A. A family of propositions over A is a function from A into the category of propositions; it is thus a propositional function.

Let us consider B as a propositional function over A and $(\Pi x : A)B$ as a proposition, and let us ask what a canonical proof of this proposition looks like. Such a canonical proof has the form $\lambda x.b$, where $x : A \vdash b : B$, and is in effect a code of the function b. This function b takes an element a of A and yields a proof $b[a]$ of the proposition $B[a]$. Keeping in mind the Brouwer–Heyting–Kolmogorov interpretation of the logical constants (cf. e.g. Troelstra and van Dalen, 1988, pp. 9–10), we see thus that $(\Pi x : A)B$, when regarded as a proposition, is the proposition $(\forall x : A)B$, which intuitively says that all elements of A have the property B. Note that this proposition is not written $\forall x B$ as in ordinary predicate logic; rather, the domain of quantification, A, is explicitly mentioned.

On the understanding of Π as $\forall$, we can recover the rule of $\forall$-introduction from the rule of Π-introduction by employing the form of judgement 'C true' as follows.

$$\frac{x : A \vdash B \text{ true}}{(\forall x : A)B \text{ true}}$$

That is to say, if B is true for any $x : A$, then $(\forall x : A)B$ is true. Let us also consider the version of $\forall$-introduction where the proof objects have not been suppressed:

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

Here we should think of b as an open proof of B, a proof depending on a parameter $x : A$. For instance, A may be the natural numbers, $\mathbf{N}$, and B may be the propositional function that for any element n of $\mathbf{N}$ yields the proposition that n is either even or odd; b is then a proof of the proposition that x is either even or odd, where x is a generic or arbitrary natural number. By binding x we get a proof $\lambda x.b$ of $(\forall x : A)B$ where x is no longer free; if x is the only free variable in b, then $\lambda x.b$ is a closed proof of $(\forall x : A)B$.

Since the domain of quantification is explicitly mentioned in $(\forall x : A)B$, it also has to be mentioned in the $\forall$-elimination rule:

$$\frac{(\forall x : A)B \text{ true} \qquad a : A}{B[a] \text{ true}}$$

Making the proof-objects explicit yields the following $\forall$-elimination rule.

$$\frac{c : (\forall x : A)B \qquad a : A}{\mathbf{ap}(c, a) : B[a]}$$

The rule says that if c is a proof of $(\forall x : A)B$ and $a : A$, then $\mathbf{ap}(c, a)$ is a proof of $B[a]$.

The Π-equality rule can now be seen to correspond to the $\forall$-reduction of Prawitz (1965, p. 37) at the level of proof objects. We shall illustrate this in the case of $\supset$, to which we now turn.

Suppose $B : \mathbf{set}$. Then, by weakening, $x : A \vdash B : \mathbf{set}$ holds. In this case an element of $(\Pi x : A)B$ codes a function from the **set** A to the **set** B. Since x is not free in B in this case, we may write $A \to B$ instead of $(\Pi x : A)B$, thereby also indicating that this is the function space from A to B. Regarding both A and B as propositions, and again keeping in mind the Brouwer–Heyting–Kolmogorov interpretation of the logical connectives, it is clear that $A \to B$ can be interpreted as the implication $A \supset B$.

The Π-introduction and elimination rules become $\supset$-introduction and elimination in this case. A canonical proof object of $A \supset B$ has the form $\lambda x.b$, where b is an open proof from A to B. Given a proof of $c : A \supset B$ and a proof $a : A$, then $\mathbf{ap}(c, a)$ is a proof of B.

The Π-equality rule yields the following rule of $\supset$-equality.

$$\frac{x : A \vdash b : B \qquad a : A}{\mathbf{ap}(\lambda x.b, a) = b[a] : B}$$

Here a is a proof of A; b is an open proof of B from A; $\lambda x.b$ is a proof of $A \supset B$ obtained by extending b with one application of $\supset$-introduction; $\mathbf{ap}(\lambda x.b, a)$ is the proof of B got by applying $\supset$-elimination to $\lambda x.b$ and a; and $b[a]$ is a proof of B got from b by supplying it in the suitable sense with the proof a of A. The $\supset$-equality rule says that $\mathbf{ap}(\lambda x.b, a)$ and $b[a]$ are equal proofs of B. Using the standard notation of natural deduction this equality can be expressed as follows (where we write $\mathcal{D}_1$ instead of b and $\mathcal{D}_2$ instead of a).

$$\dfrac{\begin{array}{c}A\\ \mathcal{D}_1\\ B\end{array}}{\dfrac{A \supset B \qquad A}{B}} \quad \mathcal{D}_2 \quad = \quad \begin{array}{c}\mathcal{D}_2\\ A\\ \mathcal{D}_1\\ B\end{array}$$

By replacing '=' here with a sign for Prawitz's reduction relation, one sees that what is displayed here is just the rule of $\supset$-reduction. Thus the rule of $\supset$-equality can be read as saying that a proof containing a "detour" like that in the proof on the left hand side above is identical to the proof got by deleting this detour by means of a $\supset$-reduction.

2.3. **Disjoint union of a family of sets.** Given a **set** A and a family B of **set**s over A we can form the disjoint union of the family B. That is the content of Σ-formation:

$$(\Sigma\text{-form}) \qquad \frac{A : \mathbf{set} \qquad x : A \vdash B : \mathbf{set}}{(\Sigma x : A)B : \mathbf{set}}$$

According to the rule of Σ-introduction, the canonical elements of $(\Sigma x : A)B$ are pairs:

$$(\Sigma\text{-intro}) \qquad \frac{a : A \qquad b : B[a]}{\langle a, b\rangle : (\Sigma x : A)B}$$

Assume $A : \mathbf{set}$, $x : A \vdash B : \mathbf{set}$. Then we may form $(\Sigma x : A)B : \mathbf{set}$. Assume further that C is a family of **set**s over $(\Sigma x : A)B$, that is, assume $z : (\Sigma x : A)B \vdash$

$C : \mathbf{set}$. The rule of Σ-elimination is as follows:

$$\text{(Σ-elim)} \qquad \frac{c : (\Sigma x : A)B \qquad x : A, y : B \vdash d : C[\langle x, y\rangle]}{\mathbf{E}(c, xy.d) : C[c]}$$

We may think of the binary function d as a unary function on the canonical elements of $(\Sigma x : A)B$—it takes $\langle a, b\rangle$, where $a : A$ and $b : B[a]$, and yields an element $d[a, b]$ of $C[\langle a, b\rangle]$. The Σ-elimination rule provides us with a function $c \mapsto \mathbf{E}(c, xy.d)$ defined for all elements c (not only canonical ones) of $(\Sigma x : A)B$.

Two clarificatory remarks pertaining to Σ-elimination are in order here. The first remark concerns the premiss $x : A, y : B \vdash d : C[\langle x, y\rangle]$. By the preliminary assumption $z : (\Sigma x : A)B \vdash C : \mathbf{set}$, the variable z, ranging over $(\Sigma x : A)B$, occurs (or, is allowed to occur) in C. Since $x : A, y : B \vdash \langle x, y\rangle : (\Sigma x : A)B$ holds by Σ-introduction, the substitution of $\langle x, y\rangle$ for z in C in the context $x : A, y : B$ makes sense. The second remark concerns the conclusion $\mathbf{E}(c, xy.d) : C[c]$. The operation $\mathbf{E}$ is variable-binding: it binds the free variables x and y in d. This is symbolized by prefixing d with x and y inside $\mathbf{E}(-, -)$.[6]

The Σ-equality rule tells us how to compute $\mathbf{E}(c, xy.d)$ when c is in canonical form.

$$\text{(Σ-eq)} \qquad \frac{a : A \qquad b : B(a) \qquad x : A, y : B \vdash d : C[\langle x, y\rangle]}{\mathbf{E}(\langle a, b\rangle, xy.d) = d[a, b] : C[\langle a, b\rangle]}$$

The rule of Σ-elimination introduces a non-canonical element $\mathbf{E}(c, xy.d)$ in $C[c]$. To justify this rule we have to explain how to evaluate this non-canonical element to canonical form. This is done by reference to the rule of Σ-equality. First, evaluate $c : (\Sigma x : A)B$ to get a pair $\langle a, b\rangle$, where $a : A$ and $b : B[a]$. We have

$$\mathbf{E}(c, xy.d) = \mathbf{E}(\langle a, b\rangle, xy.d) = d[a, b] : C[\langle a, b\rangle]$$

by Σ-equality. By the premiss $x : A, y : B \vdash d : C[\langle x, y\rangle]$ we know how to compute $d[a, b]$ to obtain a canonical element of $C[\langle a, b\rangle]$; since $C[c] = C[\langle a, b\rangle] : \mathbf{set}$, this will also be a canonical element of $C[c]$.

By means of $\mathbf{E}$ we can define projection operations, which justifies our speaking of the canonical elements of $(\Sigma x : A)B$ as pairs. For the first projection we put $C = A$ and $d = x$ in the rule of Σ-elimination, thereby obtaining:

$$\frac{c : (\Sigma x : A)B \qquad x : A, y : B \vdash x : A}{\mathbf{E}(c, xy.x) : A}$$

By Σ-equality we have in this case:

$$\mathbf{E}(\langle a, b\rangle, xy.x) = x[a/x, b/y] = a : A$$

We may therefore define the first projection **fst** as follows.

$$c : (\Sigma x : A)B \vdash \mathbf{fst}(c) = \mathbf{E}(c, xy.x)$$

[6]In the higher-order presentation there is only one variable-binding operation, namely abstraction, by means of which higher-order functions are formed. The $\mathbf{E}$ above is then a higher-order function whose second argument is itself a higher-order function $d : (x : A)(y : B)C(\langle x, y\rangle)$. To make the notation for λ in the Π-introduction rule accord with the notation here used for $\mathbf{E}$ we should write $\lambda(x.b)$ instead of $\lambda x.b$. The latter is preferred here as it is more familiar.

For the second projection we put $C = B[\mathbf{fst}(z)]$ and $d = y$ in the rule of Σ-elimination:

$$\frac{c : (\Sigma x : A)B \qquad x : A, y : B \vdash y : B[\mathbf{fst}(\langle x, y\rangle)]}{\mathbf{E}(c, xy.y) : B[\mathbf{fst}(c)]}$$

The second premiss here is valid since $x : A, y : B \vdash B[\mathbf{fst}(\langle x, y\rangle)] = B[x] = B : \mathbf{set}$ holds. By Σ-equality we have

$$\mathbf{E}(\langle a, b\rangle, xy.y) = y[a/x, b/y] = b : B[\mathbf{fst}(\langle a, b\rangle)]$$

But $\mathbf{fst}(\langle a, b\rangle) = a : A$, hence

$$B[\mathbf{fst}(\langle a, b\rangle)] = B[a] : \mathbf{set}$$

We therefore define the second projection by

$$c : (\Sigma x : A)B \vdash \mathbf{snd}(c) = \mathbf{E}(c, xy.y)$$

The following four rules are then justified

$$\frac{c : (\Sigma x : A)B}{\mathbf{fst}(c) : A} \qquad \frac{a : A \qquad b : B[a]}{\mathbf{fst}(\langle a, b\rangle) = a : A}$$

$$\frac{c : (\Sigma x : A)B}{\mathbf{snd}(c) : B[\mathbf{fst}(c)]} \qquad \frac{a : A \qquad b : B[a]}{\mathbf{snd}(\langle a, b\rangle) = b : B[a]}$$

2.4. **The logical interpretation of the disjoint union of a family of sets.** If we regard B as a propositional function over A, then $(\Sigma x : A)B$ can be regarded as the existentially quantified proposition $(\exists x : A)B$. A canonical proof of $(\exists x : A)B$ is a pair $\langle a, b\rangle$ where $a : A$ and $b : B[a]$; that is to say, a is a witness and b is a proof that a indeed has the property B. When suppressing proof objects and employing the form of judgement D true, the rule of Σ-elimination becomes $\exists$-elimination:

$$\frac{(\exists x : A)B \text{ true} \qquad x : A, B \text{ true} \vdash C \text{ true}}{C \text{ true}}$$

In ordinary natural deduction the assumption $x : A$ in the second premiss is usually not made explicit.

If $B : \mathbf{set}$ holds categorically, then the rules for Σ yield rules for ordinary cartesian product. On the logical interpretation, the cartesian product becomes conjunction. Indeed the Σ-formation and introduction rules then become:

$$\frac{A : \mathbf{prop} \qquad B : \mathbf{prop}}{A \wedge B : \mathbf{prop}} \qquad \frac{a : A \qquad b : B}{\langle a, b\rangle : A \wedge B}$$

The Σ-elimination rule, with and without proof objects, becomes:

$$\frac{A \wedge B \text{ true} \qquad A \text{ true}, B \text{ true} \vdash C \text{ true}}{C \text{ true}}$$

$$\frac{c : A \wedge B \qquad x : A, y : B \vdash d : C[\langle x, y\rangle]}{\mathbf{E}(c, xy.d) : C[c]}$$

This is a generalization of the ordinary rules of $\wedge$-elimination also found in Schroeder-Heister (1984, p. 1294). The ordinary rules are obtained as a special case by letting C be A or B. We remark that in the higher-order presentation a generalized elimination rule in this sense can also be given for Π (cf. Nordström et al., 1990, pp.

51–52); using this generalized elimination rule instead of the rule of Π-elimination presented above in fact yields a strictly stronger theory, as shown by Garner (2009).

2.5. **Disjoint union of two sets.** Given two **set**s we may form their disjoint union. That is content of the rule of +-formation.

(+-form) $$\frac{A : \mathbf{set} \qquad B : \mathbf{set}}{A + B : \mathbf{set}}$$

A canonical element of $A + B$ is an element of A or an element of B together with the information that it comes from A or B respectively. Thus there are two rules of +-introduction:

(+-intro) $$\frac{a : A}{\mathbf{i}(a) : A + B} \qquad \frac{b : B}{\mathbf{j}(b) : A + B}$$

Assume $A : \mathbf{set}$, $B : \mathbf{set}$, and $z : A + B \vdash C : \mathbf{set}$. The rule of +-elimination is:

(+-elim) $$\frac{c : A + B \qquad x : A \vdash d : C[\mathbf{i}(x)] \qquad y : B \vdash e : C[\mathbf{j}(y)]}{\mathbf{D}(c, x.d, y.e) : C[c]}$$

The rule can be glossed as follows. Assume that C is a family of **set**s over $A + B$ and that we are given a function d which takes an $a : A$ to an element $d[a]$ of $C[\mathbf{i}(a)]$ and a function e which takes a $b : B$ to an element $e[b]$ of $C[\mathbf{j}(b)]$. Then $C[c]$ is inhabited for any $c : C$, namely by $\mathbf{D}(c, x.d, y.e)$. How to compute $\mathbf{D}(c, x.d, y.e)$ is determined by the +-equality rules. Since there are two +-introduction rules, there are also two +-equality rules.

(+-eq) $$\frac{a : A \qquad x : A \vdash d : C[\mathbf{i}(x)] \qquad y : B \vdash e : C[\mathbf{j}(y)]}{\mathbf{D}(\mathbf{i}(a), x.d, y.e) = d[a] : C[\mathbf{i}(a)]}$$

$$\frac{b : B \qquad x : A \vdash d : C[\mathbf{i}(x)] \qquad y : B \vdash e : C[\mathbf{j}(y)]}{\mathbf{D}(\mathbf{j}(b), x.d, y.e) = e[b] : C[\mathbf{j}(b)]}$$

In the logical interpretation + becomes disjunction $\vee$.

2.6. **Finite sets.** We have introduced operations for constructing **set**s from other **set**s or families of **set**s; but so far we have no basic **set**s to start from. We shall now provide a scheme of rules which, when specified for any particular natural number n, gives us a **set** with n canonical elements.

In the following n is a generic natural number. There is a **set** $\mathbf{N}_n$:

($\mathbf{N}_n$-form) $$\mathbf{N}_n : \mathbf{set}$$

The **set** $\mathbf{N}_n$ has n canonical elements, each introduced by its own introduction rule; thus $\mathbf{N}_n$ has n introduction rules:

($\mathbf{N}_n$-intro) $$m_1 : \mathbf{N}_n, \ldots, m_n : \mathbf{N}_n$$

The $\mathbf{N}_n$-elimination rule can be seen as a principle of proof by n cases. We assume that C is a family of **set**s over $\mathbf{N}_n$; that is, we assume $z : \mathbf{N}_n \vdash C : \mathbf{set}$.

($\mathbf{N}_n$-elim) $$\frac{m : \mathbf{N}_n \qquad c_1 : C[m_1] \qquad \ldots \qquad c_n : C[m_n]}{\mathbf{case}_n(m, c_1, \ldots, c_n) : C[m]}$$

Thus, assume that for each canonical $m_k : \mathbf{N}_n$ we have an element $c_k : C[m_k]$. The $\mathbf{N}_m$-elimination rule allows us to infer that for any $m : \mathbf{N}_n$ there is an element,

namely $\mathbf{case}_n(m, c_1, \ldots, c_n)$, in $C[m]$. How to compute $\mathbf{case}_n(m, c_1, \ldots, c_n)$ to canonical form is determined by the $\mathbf{N}_n$-equality rules. Since there are n $\mathbf{N}_n$-introduction rules, there are also n $\mathbf{N}_n$-equality rules, one for each introduction rule. We state the rule for a generic $k \leq n$.

$$(\mathbf{N}_n\text{-eq}) \qquad \frac{c_1 : C[m_1] \qquad \ldots \qquad c_n : C[m_n]}{\mathbf{case}_n(m_k, c_1, \ldots, c_n) = c_k : C[m_k]}$$

On the basis of this rule we may explain how to compute $\mathbf{case}_n(m, c_1, \ldots, c_n) : C(c)$. Evaluate $m : \mathbf{N}_n$, thereby obtaining a canonical element $m_k : \mathbf{N}_n$. Since $m = m_k : \mathbf{N}_n$, we have both $\mathbf{case}_n(m, c_1, \ldots, c_n) = \mathbf{case}_n(m_k, c_1, \ldots, c_n) : C[m_k]$ and $C[m_k] = C[m] : \mathbf{set}$. Therefore, $\mathbf{case}_n(m, c_1, \ldots, c_n) = c_k : C[m]$, hence the value of $\mathbf{case}_n(m, c_1, \ldots, c_n)$ equals the value of c_k, which we know how to compute by the premiss $c_k : C[m_k]$.

We may give $\mathbf{N}_2$ the name **bool**; the canonical elements of $\mathbf{N}_2$ the names t and f; and the expression $\mathbf{case}_2(m, c_1, c_2)$ may be written $\mathbf{if}\, m\, \mathbf{then}\, c_1\, \mathbf{else}\, c_2$. Let C be a family of **set**s over **bool**; that is, assume $z : \mathbf{bool} \vdash C : \mathbf{set}$ We have the following two rules of **bool**-elimination, which we here state as one rule with two conclusions:

$$\frac{c : C[\mathsf{t}] \qquad d : C[\mathsf{f}]}{\begin{array}{c}\mathbf{if}\, \mathsf{t}\, \mathbf{then}\, c\, \mathbf{else}\, d = c : C[\mathsf{t}] \\ \mathbf{if}\, \mathsf{f}\, \mathbf{then}\, c\, \mathbf{else}\, d = d : C[\mathsf{f}]\end{array}}$$

Familiar boolean functions can be defined from t, f, and $\mathbf{if}\, m\, \mathbf{then}\, c_1\, \mathbf{else}\, c_2$ as follows.

$$\begin{aligned} a : \mathbf{bool}, b : \mathbf{bool} &\vdash a\, \mathbf{and}\, b = \mathbf{if}\, a\, \mathbf{then}\, b\, \mathbf{else}\, \mathsf{f} : \mathbf{bool} \\ a : \mathbf{bool}, b : \mathbf{bool} &\vdash a\, \mathbf{or}\, b = \mathbf{if}\, a\, \mathbf{then}\, \mathsf{t}\, \mathbf{else}\, b : \mathbf{bool} \\ a : \mathbf{bool} &\vdash \mathbf{not}\, a = \mathbf{if}\, a\, \mathbf{then}\, \mathsf{f}\, \mathbf{else}\, \mathsf{t} : \mathbf{bool} \end{aligned}$$

The **set** $\mathbf{N}_0$ has 0, that is no, introduction rules; but it does have an elimination rule:

$$\frac{m : \mathbf{N}_0}{\mathbf{case}_0(m) : C[m]}$$

Thus, in particular, if we are given $C : \mathbf{set}$ and $m : \mathbf{N}_0$, then we may infer that C is inhabited, namely by $\mathbf{case}_0(m)$. Since there is no $\mathbf{N}_0$-introduction rule, neither is there a $\mathbf{N}_0$-equality rule. The justification of $\mathbf{N}_0$-elimination is therefore different in character from the justification of the other elimination rules. A rule of inference is justified if we can make the conclusion of the rule evident on the assumption that its premisses are known (cf. e.g. Sundholm, 2012). Since $\mathbf{N}_0$ has no introduction rule and therefore no canonical elements, the premiss $m : \mathbf{N}_0$ of $\mathbf{N}_0$-elimination cannot be known. The rule of $\mathbf{N}_0$-elimination is therefore vacuously justified, for the assumption that its premiss is known cannot be fulfilled.

In the logical interpretation $\mathbf{N}_0$ becomes absurdity $\bot$. In constructive type theory absurdity is thus the proposition that by definition has no introduction rule. The rule of $\bot$-elimination is the principle of *ex falso quodlibet*:

$$\frac{\bot\ \mathrm{true}}{C\ \mathrm{true}}$$

We define the negation of a proposition A to be the proposition $A \supset \bot$.[7]

$$A : \mathbf{prop} \vdash \neg A = A \supset \bot$$

The following rules are then derivable.

$$\frac{A : \mathbf{prop}}{\neg A : \mathbf{prop}} \qquad \frac{x : A \vdash b : \bot}{\lambda x.b : \neg A} \qquad \frac{b : \neg A \qquad a : A}{\mathbf{ap}(b, a) : C}$$

2.7. **The natural numbers.** We shall not need the natural numbers in later chapters, but it may nevertheless be useful to see that the primitive notions of arithmetic can be introduced by Gentzen–Prawitz style rules.

(**N**-form) $$\mathbf{N} : \mathbf{set}$$

The canonical elements of $\mathbf{N}$ are 0 and $\mathbf{s}(n)$, where n is any $\mathbf{N}$, not necessarily of canonical form:

(**N**-intro) $$0 : \mathbf{N} \qquad \frac{n : \mathbf{N}}{\mathbf{s}(n) : \mathbf{N}}$$

The rule of **N**-elimination is simultaneously a principle of definition by recursion and proof by mathematical induction. Assume that C is a family of **set**s over $\mathbf{N}$; that is, assume $z : \mathbf{N} \vdash C : \mathbf{set}$.

(**N**-elim) $$\frac{n : \mathbf{N} \qquad d : C[0] \qquad x : \mathbf{N}, y : C[x] \vdash e : C[\mathbf{s}(x)]}{\mathbf{R}(n, d, xy.e) : C[n]}$$

We are given an element $d : C[0]$ together with a function e that takes $k : \mathbf{N}$ and $c : C[k]$ and yields $e[k, c] : C[\mathbf{s}(k)]$. The rule tells us that for an arbitrary $n : \mathbf{N}$ the **set** $C[n]$ is then inhabited, namely by $\mathbf{R}(n, d, xy.e)$.

To see that **N**-elimination encapsulates the ordinary principle of proof by induction over the natural numbers, assume that C is a propositional function over $\mathbf{N}$. Then d is a proof of the base case $C[0]$ and e is a proof of the induction step; that is, e is an open proof from $C[k]$ to $C[\mathbf{s}(k)]$. The conclusion of **N**-elimination says that $C[n]$ is inhabited, that is, true, for an arbitrary $n : \mathbf{N}$.

The rule of **N**-equality tells us how to compute $\mathbf{R}(n, d, xy.e)$ when n is of canonical form. Since there are two **N**-introduction rules, there are also two **N**-equality rules, which we here state more simply without the premisses:

(**N**-eq)
$$\begin{aligned} \mathbf{R}(0, d, xy.e) &= d : C[0] \\ \mathbf{R}(\mathbf{s}(n), d, xy.e) &= e[n, \mathbf{R}(n, d, xy.e)] : C[\mathbf{s}(n)] \end{aligned}$$

This gives in particular:

$$\mathbf{R}(\mathbf{s}(0), d, xy.e) = e[0, \mathbf{R}(0, d, xy.e)] = e[0, d] : C[\mathbf{s}(0)]$$
$$\mathbf{R}(\mathbf{s}(\mathbf{s}(0)), d, xy.e) = e[\mathbf{s}(0), \mathbf{R}(\mathbf{s}(0), d, xy.e)] = e[\mathbf{s}(0), e[0, d]] : C[\mathbf{s}(\mathbf{s}(0))]$$
$$\mathbf{R}(\mathbf{s}(\mathbf{s}((0))), d, xy.e) = e[\mathbf{s}(\mathbf{s}((0)), e[\mathbf{s}(0), e[0, d]]] : C[\mathbf{s}(\mathbf{s}(\mathbf{s}(0)))]$$

[7] In the higher-order presentation $\neg$ may be defined in the empty context, namely as follows.

$$\neg = [A]A \supset \bot : (\mathbf{prop})\mathbf{prop}$$

Thus, $\neg$ is a function which takes a **prop** A as argument and yields a **prop** $\neg A$ as value. Similar considerations apply to the definitions of **fst** and **snd** above as well as to the definitions given below pertaining to **N**.

It should be clear from these few computations that $\mathbf{R}$ provides a means for defining functions by recursion.

In general, to compute $\mathbf{R}(m, d, xy.e)$ first evaluate m to get either 0 or $\mathbf{s}(n)$. In the first case output d, in the second case continue by computing $e[n, \mathbf{R}(n, d, xy.e)]$.

By letting C be $\mathbf{N}$ and e be $\mathbf{s}(y)$ in $\mathbf{N}$-elimination, one can infer $a : \mathbf{N}, b : \mathbf{N} \vdash \mathbf{R}(b, a, xy.\mathbf{s}(y)) : \mathbf{N}$. The reader may check that we here have the *definiens* of the addition function, in other words, that addition can be defined as follows.

$$a : \mathbf{N}, b : \mathbf{N} \vdash a + b = \mathbf{R}(b, a, xy.\mathbf{s}(y)) : \mathbf{N}$$

To define multiplication we let C be $\mathbf{N}$, d be 0, and e be $y + a$.

$$a : \mathbf{N}, b : \mathbf{N} \vdash a \times b = \mathbf{R}(b, 0, xy.(y + a)) : \mathbf{N}$$

2.8. **Propositional identity.** In the language developed so far we can express identities by means of judgements $a = b : \mathscr{C}$. Judgements can, however, not be operated on by the propositional operators, that is, for instance by conjunction or universal quantification. Since we want to be able to operate on identity statements with the propositional operations, it is clear that we need identity propositions. Thus, for each **set** A we wish to introduce a binary propositional function $x =_A y$ of identity over A. Instead of $x =_A y$, we shall write $\mathbf{Id}(A, x, y)$. The rule of $\mathbf{Id}$-formation says that given a **set** A and two elements a, b of A, there is a proposition $\mathbf{Id}(A, a, b)$.

(**Id**-form)
$$\frac{A : \mathbf{set} \qquad a : A \qquad b : A}{\mathbf{Id}(A, a, b) : \mathbf{prop}}$$

Note that by this rule there is no identity proposition between a and b unless a and b belong to the same **set**. Hence, assuming that Julius Caesar and the number 7 do not belong to the same **set**, there is no proposition to the effect that Julius Caesar and 7 are identical.

It is only with the introduction of $\mathbf{Id}$ that we are able to define propositional functions in our language. Namely, given a **set** A we now have a propositional function $\mathbf{Id}(A, x, y)$ over A, by means of which other propositional functions can be defined. For instance, we may now define $x \leq y$ over $\mathbf{N}$ by

$$x : \mathbf{N}, y : \mathbf{N} \vdash x \leq y = (\exists z : \mathbf{N})\mathbf{Id}(\mathbf{N}, x + z, y) : \mathbf{prop}$$

Note the use of judgemental identity $A = B : \mathbf{prop}$ here: the definition in effect identifies two propositional functions in the context $x : \mathbf{N}, y : \mathbf{N}$. It may be useful to see how one may define the propositional function $\mathbf{Pr}(x)$, saying that x is a prime number, in constructive type theory. In ordinary predicate logic with restricted quantifiers we may use a definition such as the following (for the purposes of this presentation let us assume that 0 and 1 are prime numbers).

$$Pr(n) \equiv \forall x, y \leq n(x \times y = n \supset (x = 1 \vee x = n))$$

In constructive type theory restricted quantification may be defined as quantification over $(\exists z : \mathbf{N})z \leq n$. Such quantification makes sense, since $(\exists z : \mathbf{N})z \leq n$ is a **set** for any $n : \mathbf{N}$. An element of this **set** is a pair $\langle k, p \rangle$, where k is a $\mathbf{N}$ and p is a

proof of $k \leq n$. We may define $\mathbf{Pr}$ as follows.

$$n : \mathbf{N} \vdash \mathbf{Pr}(n) = (\forall x, y : (\exists z : \mathbf{N}) z \leq n) \\ ((\mathbf{fst}(x) \times \mathbf{fst}(y) =_{\mathbf{N}} n) \supset (\mathbf{fst}(x) =_{\mathbf{N}} 1 \vee \mathbf{fst}(x) =_{\mathbf{N}} n)) : \mathbf{prop}$$

Here we have used the notation $x =_{\mathbf{N}} y$ instead of the official $\mathbf{Id}(\mathbf{N}, x, y)$, and we have contracted the two quantifiers by writing $(\forall x, y : (\exists z : \mathbf{N}) z \leq n)$.

To justify the rule of **Id**-formation we have to specify what is a canonical proof of $\mathbf{Id}(A, a, b)$. What, for instance, should be a canonical proof of the proposition $\mathbf{Id}(\mathbf{N}, 0, 0)$? The Brouwer–Heyting–Kolmogorov explanation will not help us in answering this question, since it is silent about identity propositions. Since we want $\mathbf{Id}(\mathbf{N}, x, y)$ to be the relation of identity over $\mathbf{N}$ and since a proposition is here taken to be true if it is inhabited as a set, it is clear that we simply have to introduce a proof of $\mathbf{Id}(\mathbf{N}, 0, 0)$ by stipulation; we call this proof $\mathbf{refl}(\mathbf{N}, 0)$. The rule of **Id**-introduction is as follows.

(**Id**-intro) $$\frac{a : A}{\mathbf{refl}(A, a) : \mathbf{Id}(A, a, a)}$$

Thus, provided $a : A$, we stipulate that there is a proof $\mathbf{refl}(A, a) : \mathbf{Id}(A, a, a)$. We here emphasize the aspect of stipulation, but in fact, all introduction rules are purely stipulatory in nature. An introduction rule stipulates that the canonical elements of the **set** under consideration look such and such. The **Id**-introduction rule is therefore no different from other introduction rules in this regard.

The **Id**-introduction rule *is* different from other introduction rules in that it does not immediately yield an answer to the question of what is a canonical element of $\mathbf{Id}(A, a, b)$, that is of a **set** of the form introduced by **Id**-formation. It yields an answer only to the question of what is a canonical element of $\mathbf{Id}(A, a, a)$. Since no introduction rule has $\mathbf{Id}(A, a, b)$ as the predicate $\mathscr{C}$ of its conclusion $c : \mathscr{C}$, it is clear that the only way in which we can come to judge that c is a canonical element of $\mathbf{Id}(A, a, b)$ is on the basis of an identity judgement of the form $C = \mathbf{Id}(A, a, b) : \mathbf{prop}$, where $c : C$ is the conclusion of an application of an introduction rule. It is, moreover, clear that any such C must have the form $\mathbf{Id}(A', a', a')$, where $A = A' : \mathbf{set}$, $a = a' : A$ and $b = a' : A$. A canonical element of $\mathbf{Id}(A, a, b)$ is therefore of the form $\mathbf{refl}(A', a')$ where $A = A' : \mathbf{set}$, $a = a' : A$ and $b = a' : A$.

Martin-Löf (1971), in a paper concerned with natural deduction rather than type theory, provided a general scheme of introduction- and elimination rules as well as reduction procedures for so-called inductively defined predicates. The rules provided above for $\mathbf{N}_n$ and $\mathbf{N}$ follow this scheme, although generalizing the rules to the syntax of constructive type theory. Also the identity predicate of ordinary predicate logic is covered by this rule scheme (Martin-Löf, 1971, p. 190). The ordinary binary identity predicate, which Martin-Löf designates by E, is the predicate that has the introduction rule

$$Exx$$

with no premisses. It should be clear how the rule of **Id**-introduction above generalizes this rule to the syntax of constructive type theory. Martin-Löf's scheme

yields the following elimination rule for E:

$$\frac{Etu \qquad C[z/x, z/y]}{C[t/x, u/y]}$$

Here C is any formula of the language, and $C[z/x, z/y]$ is the result of substituting the variable z for both x and y in C. If we assume that x and y are all and only the free variables in C, then we may think of C as defining a binary relation over the underlying domain. That we can prove $C[z/x, z/y]$ means that the relation defined by C is reflexive. The E-elimination rule allows us to infer that the relation defined by C is true of t and u provided we have a derivation of Etu. Thus the rule says in effect that E is the smallest reflexive relation over the underlying domain.

The rule of **Id**-elimination generalizes the E-elimination rule to the syntax of constructive type theory; it generalizes the E-elimination rule also in allowing the relation C occurring in the minor premiss to include as argument a proof object of the identity proposition being eliminated. Assume $A : \mathbf{set}$, $a : A$, $b : A$, and $x : A, y : A, u : \mathbf{Id}(A, x, y) \vdash C : \mathbf{set}$.

(**Id**-elim) $$\frac{p : \mathbf{Id}(A, a, b) \qquad z : A \vdash d : C[z, z, \mathbf{refl}(A, z)]}{\mathbf{J}(p, z.d) : C[a, b, p]}$$

Thus, if we have a proof p of the proposition $\mathbf{Id}(A, a, b)$ and a function d taking any a' of A to a proof that the ternary relation C holds of the triple $a', a', \mathbf{refl}(A, a')$; then **Id**-elimination allows us to infer that there is a proof $\mathbf{J}(p, z.d)$ of $C[a, b, p]$.

The rule of **Id**-equality tells us how to compute $\mathbf{J}(p, z.d)$ when p is of canonical form, $\mathbf{refl}(A, a)$.

(**Id**-eq) $$\mathbf{J}(\mathbf{refl}(A, a), z.d) = d[a] : C[a, a, \mathbf{refl}(A, a)]$$

The rule of **Id**-elimination can now be justified as follows. To evaluate $\mathbf{J}(p, z.d)$ first evaluate p to get a canonical element of $\mathbf{Id}(A, a, b)$. As noted above, such a canonical element has the form $\mathbf{refl}(A', a')$, where $A = A' : \mathbf{set}$, $a = a' : A$ and $b = a' : A$. Hence,

$$\mathbf{J}(p, z.d) = \mathbf{J}(\mathbf{refl}(A', a'), z.d) = d[a'] : C[a', a', \mathbf{refl}(A', a')].$$

By the assumption $z : A \vdash d : C[z, z, \mathbf{refl}(z)]$ we know how to evaluate $d[a']$ to canonical form. It remains then only to see that $C[a', a', \mathbf{refl}(A', a')] = C[a, b, p] : \mathbf{set}$, but this follows from the judgemental identities $A = A' : \mathbf{set}$, $a = a' : A$, $b = a' : A$ and $p = \mathbf{refl}(A', a') : \mathbf{Id}(A, a, b)$ together with the extensionality of substitution into **set**s with respect to judgemental identity.

In many applications of **Id**-elimination the family C in its minor premiss does not depend on the **set** $\mathbf{Id}(A, a, b)$ of its major premiss. Thus, C will then be a **set** already in the context $x : A, y : A$; that is to say, C will be a binary relation over A. What is required then for an application of **Id**-elimination is that we have a function d witnessing that C is a reflexive relation over A; more precisely, that $d[a]$ is a proof of $C[a, a]$ for any $a : A$. We shall now demonstrate how **Id**-elimination is used in practice by showing that the relation $\mathbf{Id}(A, x, y)$ is symmetric and transitive, and by showing that if $F[a]$ and $\mathbf{Id}(A, a, b)$ are true, then so is $F[b]$, that is, by showing

the indiscernibility of elements related by $\mathbf{Id}(A, x, y)$. The main task in each case is to find a suitable C and a suitable function d taking $a : A$ and yielding a proof of $C[a, a]$.

For symmetry let C be $\mathbf{Id}(A, y, x)$. It is clear that $z : A \vdash \mathbf{refl}(A, z) : \mathbf{Id}(A, z, z)$, so we let d be $\mathbf{refl}(A, z)$. If we insert these data into **Id**-elimination we get:

$$\frac{p : \mathbf{Id}(A, a, b) \qquad z : A \vdash \mathbf{refl}(A, z) : \mathbf{Id}(A, z, z)}{\mathbf{J}(p, z.\mathbf{refl}(A, z)) : \mathbf{Id}(A, b, a)}$$

Hence, from a proof $p : \mathbf{Id}(A, a, b)$ we get a proof $\mathbf{J}(p, z.\mathbf{refl}(A, z)) : \mathbf{Id}(A, b, a)$. Assuming that p here is a closed term, one can argue, on the basis of the explanation of what a canonical proof of an identity proposition is, that p is identical with the proof $\mathbf{J}(p, z.\mathbf{refl}(A, z))$. Namely, the judgement $p : \mathbf{Id}(A, a, b)$ means that p, since it is a closed term, evaluates to a proof of the form $\mathbf{refl}(A', a')$, where $A = A' : \mathbf{set}$, $a = a' : A$, and $b = a' : A$; hence, $p = \mathbf{refl}(A', a') : \mathbf{Id}(A, a, b)$. Therefore,

$$\mathbf{J}(p, z.\mathbf{refl}(A, z)) = \mathbf{J}(\mathbf{refl}(A', a'), z.\mathbf{refl}(A, z)) = \mathbf{refl}(A, a') : \mathbf{Id}(A, b, a)$$

Since $A = A' : \mathbf{set}$, we get:

$$p = \mathbf{refl}(A', a') = \mathbf{refl}(A, a') = \mathbf{J}(p, z.\mathbf{refl}(A, z)) : \mathbf{Id}(A, b, a)$$

It should be emphasized that this reasoning i) presupposes that p is closed and ii) is not a computation in the theory itself; the computation in any particular case depends on what the given proof object $p : \mathbf{Id}(A, a, b)$ looks like.

For transitivity we let C be $\mathbf{Id}(A, y, c) \supset \mathbf{Id}(A, x, c)$ for an arbitrary $c : A$. We have $\lambda u.u : \mathbf{Id}(A, z, c) \supset \mathbf{Id}(A, z, c)$, so we let d be $\lambda u.u$. Inserting these data into **Id**-elimination yields:

$$\frac{p : \mathbf{Id}(A, a, b) \qquad z : A \vdash \lambda u.u : \mathbf{Id}(A, z, c) \supset \mathbf{Id}(A, z, c)}{\mathbf{J}(p, \lambda u.u) : \mathbf{Id}(A, b, c) \supset \mathbf{Id}(A, a, c)}$$

Hence, from a proof $p : \mathbf{Id}(A, a, b)$ and a proof $q : \mathbf{Id}(A, b, c)$, we get a proof $\mathbf{ap}(\mathbf{J}(p, \lambda u.u), q) : \mathbf{Id}(A, a, c)$. Note that $\lambda u.u$ does not depend on $z : A$, therefore no variable gets bound by this application of **Id**-elimination. Assuming that p is closed, we can argue as above that $p = \mathbf{refl}(A', a') : \mathbf{Id}(A, a, b)$; whence by **Id**-equality and Π-equality we get:

$$\mathbf{ap}(\mathbf{J}(p, \lambda u.u), q) = \mathbf{ap}(\mathbf{J}(\mathbf{refl}(A', a'), \lambda u.u), q) = \mathbf{ap}(\lambda u.u, q) = q : \mathbf{Id}(A, a, c)$$

Again it must be emphasized that this argument is not the same as an actual computation in the theory.

For the inidiscernibility of elements a, b for which $\mathbf{Id}(A, a, b)$ is true, let F be a propositional function over A, that is $x : A \vdash F : \mathbf{prop}$. Let C be $F[x] \supset F[y]$. Again we have $\lambda u.u : F[z] \supset F[z]$, hence **Id**-elimination yields:

$$\frac{p : \mathbf{Id}(A, a, b) \qquad z : A \vdash \lambda u.u : F[z] \supset F[z]}{\mathbf{J}(p, \lambda u.u) : F[a] \supset F[b]}$$

Hence, from a proof $p : \mathbf{Id}(A, a, b)$ and a proof $q : F[a]$, we get a proof

$$\mathbf{ap}(\mathbf{J}(p, \lambda u.u), q) : F[b]$$

As in the case of transitivity above we can argue that $\mathbf{ap}(\mathbf{J}(p, \lambda u.u), q) = q : F[b]$ on the assumption that p is closed.

The second **Id**-formation rule, the rule that governs when two sets of the form $\mathbf{Id}(A, a, b)$ are identical, is as follows.

$$\frac{A = A' : \mathbf{set} \qquad a = a' : A \qquad b = b' : A}{\mathbf{Id}(A, a, b) = \mathbf{Id}(A', a', b') : \mathbf{set}}$$

Employing this rule together with the general rules governing judgemental identity, one sees that the following rule is derivable:

$$\frac{a = b : A}{\mathbf{refl}(A, a) : \mathbf{Id}(A, a, b)}$$

The rule of identity elimination employed in (Martin-Löf, 1984) lays down that one can also go the other way:

$$\frac{p : \mathbf{Id}(A, a, b)}{a = b : A}$$

In the literature this rule is sometimes called extensional identity elimination, whereas the rule of **Id**-elimination is called intensional. Subsequent metamathematical work has shown that this extensional rule has several undesirable consequences, perhaps the strongest of which is that in the presence of this rule judgements of the form $a = b : A$ become undecidable in general (Hofmann, 1995, Theorem 3.2.1). From the normalization theorem of Martin-Löf (1975b) for the system employing the intensional **Id**-elimination rule it follows that judgements of the form $a = b : A$ are decidable in this system. For this and other reasons,[8] most presentations of constructive type theory therefore prefer the intensional system.

It follows from Martin-Löf's normalization theorem that if $p : \mathbf{Id}(A, a, b)$ is demonstrable in the empty context, then so is $a = b : A$; hence if one can construct a closed proof p of $\mathbf{Id}(A, a, b)$, then the judgemental identity $a = b : A$ is demonstrable (cf. Martin-Löf, 1975b, Theorem 3.14). In a non-empty context, however, one can in general not infer $a = b : A$ from $p : \mathbf{Id}(A, a, b)$. For instance, using Σ-elimination one can prove

$$z : A \times B \vdash \mathbf{E}(z, xy.\mathbf{refl}(A \times B, \langle x, y\rangle)) : \mathbf{Id}(A \times B, z, \langle \mathbf{fst}(z), \mathbf{snd}(z)\rangle)$$

But there is no way of demonstrating the corresponding judgemental identity

$$z : A \times B \vdash z = \langle \mathbf{fst}(z), \mathbf{snd}(z)\rangle : A \times B$$

since z and $\langle \mathbf{fst}(z), \mathbf{snd}(z)\rangle$ are different normal forms in the context $z : A \times B$ (this example is taken from Martin-Löf 1975a, pp. 103–104). We see therefore that judgemental identity $a = b : A$ and propositional identity $\mathbf{Id}(A, a, b)$ do not only have different logical form, but also that they differ in logical strength. These are therefore two essentially different notions of identity.

[8] In Homotopy Type Theory the main reason to prefer the intensional **Id**-elimination rule is that it does not entail

$$(\forall p, q : \mathbf{Id}(A, a, b))\mathbf{Id}(\mathbf{Id}(A, a, b), p, q) \text{ true}$$

for arbitrary A and $a, b : A$, a result that was established by Hofmann and Streicher (1998).

Exercises

The word 'proof' has been reserved for proof objects. Derivations in constructive type theory are often called demonstrations. Most exercises below ask the reader to demonstrate A true for a given proposition A. What is then intended is that a demonstration be given whose conclusion has the form $a : A$.

1. Let $A : \mathbf{prop}$, $B : \mathbf{prop}$. Demonstrate the following judgements.
   a) $A \supset (B \supset A)$ true
   b) $(A \wedge B) \supset (A \vee B)$ true
   c) $\neg(A \vee B) \supset \neg A$ true

2. Let $D : \mathbf{set}$, $x : D \vdash A : \mathbf{prop}$, and $x : D \vdash B : \mathbf{prop}$. Demonstrate the following judgements.
   a) $(\exists x : D)(A \wedge B) \supset ((\exists x : D)A \wedge (\exists x : D)B)$ true
   b) $(\forall x : D)(A \wedge B) \supset ((\forall x : D)A \wedge (\forall x : D)B)$ true
   c) $(\exists x : D)(A \vee B) \supset ((\exists x : D)A \vee (\exists x : D)B)$ true

3. Let $D : \mathbf{set}$ and $x : D, y : D \vdash R : \mathbf{prop}$. That is to say, R is a binary relation over D. Demonstrate

$$(\exists x : D)(\forall y : D)R[x, y] \supset (\forall y : D)(\exists x : D)R[x, y] \text{ true}$$

(Note that $R[x, y] \equiv R[x/x, y/y] \equiv R$.)

4. In its class-theoretic form, the syllogism of Barbara may be formulated as follows.

$$\begin{array}{ll} & \text{All } A\text{'s are } B. \\ & \text{All } B\text{'s are } C. \\ \hline \therefore & \text{All } A\text{'s are } C. \end{array}$$

Formalize Barbara in constructive type theory. Assume that you have been given proof objects of the premisses; construct a proof object of the conclusion.

5. Formalize in constructive type theory the following class-theoretical reasoning. "Everything is an A; whatever is an A is a B; hence, everything is a B."

   Given a proof object of the two premisses, construct a proof object of the conclusion.

6. Let $A : \mathbf{set}$, $B : \mathbf{set}$, and assume $c : A \to B$. Demonstrate

$$\mathbf{Id}(A, a, a') \supset \mathbf{Id}(B, \mathbf{ap}(c, a), \mathbf{ap}(c, a')) \text{ true}$$

7. Let $A : \mathbf{set}$, $B : \mathbf{set}$. Assume $p : \mathbf{Id}(A, a, a')$ and $q : \mathbf{Id}(B, b, b')$. Demonstrate

$$\mathbf{Id}(A \times B, \langle a, b\rangle, \langle a', b'\rangle) \text{ true}$$

8. Demonstrate
   a) $(\forall x : \mathbf{bool})(\mathbf{Id}(\mathbf{bool}, x, \mathsf{t}) \vee \mathbf{Id}(\mathbf{bool}, x, \mathsf{f}))$ true
   b) $(\forall x : \mathbf{N})(\mathbf{Id}(\mathbf{N}, x, 0) \vee (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, x, \mathbf{s}(y)))$ true

Solutions

1. In the solutions to these exercises we will include all the details. Each demonstration begins from the judgements $A : \mathbf{prop}$ and $B : \mathbf{prop}$. These judgements could also be included in the demonstration as hypotheses, namely by placing them to the left of $\vdash$. Here we rather choose to regard these judgements as given. We may think of an interlocutor who has taken responsibility for these judgements, $A : \mathbf{prop}$ and $B : \mathbf{prop}$, and view it is our task to make evident each of $A \supset (B \supset A)$ true, $(A \wedge B) \supset (A \vee B)$ true, and $\neg(A \vee B) \supset \neg A$ true.[9]

a)

$$\dfrac{\dfrac{\dfrac{A : \mathbf{prop}}{x : A \vdash x : A} \quad \dfrac{B : \mathbf{prop}}{x : A \vdash B : \mathbf{prop}}}{x : A, y : B \vdash x : A}}{\dfrac{x : A \vdash \lambda y.x : B \supset A}{\lambda x.\lambda y.x : A \supset (B \supset A)}}$$

b)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A : \mathbf{prop} \qquad B : \mathbf{prop}}{A \wedge B : \mathbf{prop}}}{x : A \wedge B \vdash x : A \wedge B}}{x : A \wedge B \vdash \mathbf{fst}(x) : A}}{x : A \wedge B \vdash \mathbf{i}(\mathbf{fst}(x)) : A \vee B}}{\lambda x.\mathbf{i}(\mathbf{fst}(x)) : (A \wedge B) \supset (A \vee B)}$$

There is a similar demonstration having as conclusion

$$\lambda x.\mathbf{j}(\mathbf{snd}(x)) : (A \wedge B) \supset (A \vee B).$$

c)

$$\dfrac{\dfrac{\dfrac{\dfrac{A : \mathbf{prop} \qquad B : \mathbf{prop}}{A \vee B : \mathbf{prop}}}{\neg(A \vee B) : \mathbf{prop}}}{y : \neg(A \vee B) \vdash y : \neg(A \vee B)} \quad \dfrac{\dfrac{A : \mathbf{prop}}{x : A \vdash x : A}}{x : A \vdash \mathbf{i}(x) : A \vee B}}{\dfrac{\dfrac{y : \neg(A \vee B), x : A \vdash \mathbf{ap}(y, \mathbf{i}(x)) : \bot}{y : \neg(A \vee B) \vdash \lambda x.\mathbf{ap}(y, \mathbf{i}(x)) : \neg A}}{\lambda y.\lambda x.\mathbf{ap}(y, \mathbf{i}(x)) : \neg(A \vee B) \supset \neg A}}$$

2. Here we shall use the natural deduction style of presenting demonstrations. That means that we shall not exhibit the hypotheses, but take these to be implicitly understood. The reader may want to supply the missing hypotheses.

a) To save space we shall write $\mathbf{p}$ instead of **fst** and $\mathbf{q}$ instead of **snd**.

$$\dfrac{\dfrac{\dfrac{y : (\exists x : D)(A \wedge B)}{\mathbf{p}(y) : D} \quad \dfrac{y : (\exists x : D)(A \wedge B)}{\mathbf{q}(y) : (A \wedge B)[\mathbf{p}(y)]}}{\mathbf{p}(\mathbf{q}(y)) : A[\mathbf{p}(y)]}}{\langle \mathbf{p}(y), \mathbf{p}(\mathbf{q}(y)) \rangle : (\exists x : D)A} \quad \dfrac{\dfrac{\dfrac{y : (\exists x : D)(A \wedge B)}{\mathbf{p}(y) : D} \quad \dfrac{y : (\exists x : D)(A \wedge B)}{\mathbf{q}(y) : (A \wedge B)[\mathbf{p}(y)]}}{\mathbf{q}(\mathbf{q}(y)) : B[\mathbf{p}(y)]}}{\langle \mathbf{p}(y), \mathbf{q}(\mathbf{q}(y)) \rangle : (\exists x : D)B}$$

$$\dfrac{\langle \mathbf{p}(y), \mathbf{p}(\mathbf{q}(y)) \rangle : (\exists x : D)A \qquad \langle \mathbf{p}(y), \mathbf{q}(\mathbf{q}(y)) \rangle : (\exists x : D)B}{\langle \langle \mathbf{p}(y), \mathbf{p}(\mathbf{q}(y)) \rangle, \langle \mathbf{p}(y), \mathbf{q}(\mathbf{q}(y)) \rangle \rangle : (\exists x : D)A \wedge (\exists x : D)B}$$

[9] This dialogical view on demonstration has been developed in some recent lectures of Per Martin-Löf.

b)

$$\dfrac{\dfrac{\dfrac{\dfrac{z : (\forall x : D)(A \wedge B) \qquad x : D}{\mathbf{ap}(z, x) : A \wedge B}}{\mathbf{fst}(\mathbf{ap}(z, x)) : A}}{\lambda x.\mathbf{fst}(\mathbf{ap}(z, x)) : (\forall x : D)A} \qquad \dfrac{\dfrac{\dfrac{z : (\forall x : D)(A \wedge B) \qquad x : D}{\mathbf{ap}(z, x) : A \wedge B}}{\mathbf{snd}(\mathbf{ap}(z, x)) : B}}{\lambda x.\mathbf{snd}(\mathbf{ap}(z, x)) : (\forall x : D)B}}{\dfrac{\langle \lambda x.\mathbf{fst}(\mathbf{ap}(z, x)), \lambda x.\mathbf{snd}(\mathbf{ap}(z, x)) \rangle : (\forall x : D)A \wedge (\forall x : D)B}{\lambda z.\langle \lambda x.\mathbf{fst}(\mathbf{ap}(z, x)), \lambda x.\mathbf{snd}(\mathbf{ap}(z, x)) \rangle : (\forall x : D)(A \wedge B) \supset ((\forall x : D)A \wedge (\forall x : D)B)}}$$

c) In the solution to this exercise we use both Σ-elimination and $+$-elimination. To save space we define $\mathbf{Dis} = (\exists x : D)A \vee (\exists x : D)B : \mathbf{prop}$.

$$\dfrac{\dfrac{z : (\exists x : D)(A \vee B) \qquad \dfrac{y : A \vee B \qquad \dfrac{\dfrac{x : D \qquad v : A}{\langle x, v \rangle : (\exists x : D)A}}{\mathbf{i}(\langle x, v \rangle) : \mathbf{Dis}} \qquad \dfrac{\dfrac{x : D \qquad w : B}{\langle x, w \rangle : (\exists x : D)B}}{\mathbf{j}(\langle x, w \rangle) : \mathbf{Dis}}}{\mathbf{D}(y, v.\mathbf{i}(\langle x, v \rangle), w.\mathbf{j}(\langle x, w \rangle)) : \mathbf{Dis}}}{\mathbf{E}(z, xy.\mathbf{D}(y, v.\mathbf{i}(\langle x, v \rangle), w.\mathbf{j}(\langle x, w \rangle))) : \mathbf{Dis}}}{\lambda z.\mathbf{E}(z, xy.\mathbf{D}(y, v.\mathbf{i}(\langle x, v \rangle), w.\mathbf{j}(\langle x, w \rangle))) : (\exists x : D)(A \vee B) \supset \mathbf{Dis}}$$

Note that, since A and B are propositional functions over D, the judgement $v : A$ is made in the context $x : D, v : A$; the judgement $w : B$ is made in the context $x : D, w : B$; and the judgment $y : A \vee B$ is made in the context $x : D, y : A \vee B$. The variables w and v get bound by the application of $\mathbf{D}$, whereas x and y get bound by the application of $\mathbf{E}$.

3. Again we use the natural deduction style of presenting demonstrations.

$$\dfrac{\dfrac{\dfrac{\dfrac{z : (\exists x : D)(\forall y : D)R[x, y]}{\mathbf{fst}(z) : D} \qquad \dfrac{\dfrac{z : (\exists x : D)(\forall y : D)R[x, y]}{\mathbf{snd}(z) : (\forall y : D)R[\mathbf{fst}(z), y]} \qquad y : D}{\mathbf{ap}(\mathbf{snd}(z), y) : R[\mathbf{fst}(z), y)]}}{\langle \mathbf{fst}(z), \mathbf{ap}(\mathbf{snd}(z), y) \rangle : (\exists x : D)R[x, y]}}{\lambda y.\langle \mathbf{fst}(z), \mathbf{ap}(\mathbf{snd}(z), y) \rangle : (\forall y : D)(\exists x : D)R[x, y]}}{\lambda z.\lambda y.\langle \mathbf{fst}(z), \mathbf{ap}(\mathbf{snd}(z), y) \rangle : (\exists x : D)(\forall y : D)R[x, y] \supset (\forall y : D)(\exists x : D)R[x, y]}$$

4. The important observation is that the classes A, B, and C are defined over a universe of discourse. The universe of discourse is made explicit in constructive type theory. Thus we let $D : \mathbf{set}$ and we let A, B, and C be propositional functions over D, that is we assume $x : D \vdash A : \mathbf{prop}$, $x : D \vdash B : \mathbf{prop}$, and $x : D \vdash C : \mathbf{prop}$. That all A's are B is *not* formalized as $(\forall x : D)(A \supset B)$. This proposition says that all D's have the property of being "B–if–A". The formalization is rather $(\forall z : (\exists x : D)A)B[\mathbf{fst}(z)]$. A **set** of the form $(\exists x : D)A$ may be understood as formalizing the idea of "the D's such that A" or "the D's that are A" (cf. Ranta, 1995, pp. 61–64). The proposition $(\forall z : (\exists x : D)A)B[\mathbf{fst}(z)]$ can therefore be understood as expressing that all the D's that are A are B. Since B is a propositional function over D, $B[\mathbf{fst}(z)]$ is a family over $(\exists x : D)A$ in the variable z. The formalization, then, is as follows.

$$\frac{\begin{array}{r}(\forall z : (\exists x : D)A)B[\mathbf{fst}(z)]\ \text{true}\\ (\forall z : (\exists x : D)B)C[\mathbf{fst}(z)]\ \text{true}\end{array}}{\therefore\quad (\forall z : (\exists x : D)A)C[\mathbf{fst}(z)]\ \text{true}}$$

Assume now that we are given

$$\begin{array}{l} p : (\forall z : (\exists x : D)A)B[\mathbf{fst}(z)]\\ q : (\forall z : (\exists x : D)B)C[\mathbf{fst}(z)]\end{array}$$

To save space we define

$$\begin{array}{l}\mathbf{P} = (\forall z : (\exists x : D)A)B[\mathbf{fst}(z)] : \mathbf{prop}\\ \mathbf{Q} = (\forall z : (\exists x : D)B)C[\mathbf{fst}(z)] : \mathbf{prop}\end{array}$$

We construct a proof of $(\forall z : (\exists x : D)A)C[\mathbf{fst}(z)]$ as follows.

$$\dfrac{\dfrac{q : \mathbf{Q} \qquad \dfrac{\dfrac{z : (\exists x : D)A}{\mathbf{fst}(z) : D} \qquad \dfrac{p : \mathbf{P} \qquad z : (\exists x : D)A}{\mathbf{ap}(p, z) : B[\mathbf{fst}(z)]}}{\langle \mathbf{fst}(z), \mathbf{ap}(p, z)\rangle : (\exists x : D)B}}{\mathbf{ap}(q, \langle \mathbf{fst}(z), \mathbf{ap}(p, z)\rangle) : C[\mathbf{fst}(z)]}}{\lambda z.\mathbf{ap}(q, \langle \mathbf{fst}(z), \mathbf{ap}(p, z)\rangle) : (\forall z : (\exists x : D)A)C[\mathbf{fst}(z)]}$$

5. Again we must be careful to remember the universe of discourse. Hence, when we say that everything is an A, we mean that everything in the universe of discourse is an A. We have the following formalization. We assume $D : \mathbf{set}$, $x : D \vdash A : \mathbf{set}$, and $x : D \vdash B : \mathbf{set}$.

$$\frac{\begin{array}{r}(\forall x : D)A\ \text{true}\\ (\forall z : (\exists x : D)A)B[\mathbf{fst}(z)]\ \text{true}\end{array}}{\therefore\quad (\forall x : D)B\ \text{true}}$$

Now assume

$$\begin{array}{l} q : (\forall x : D)A\\ p : (\forall z : (\exists x : D)A)B[\mathbf{fst}(z)]\end{array}$$

We construct a proof of $(\forall x : D)B$ as follows. We use $\mathbf{P}$ as in the previous exercise.

$$\dfrac{\dfrac{\dfrac{p : \mathbf{P} \qquad \dfrac{x : D \qquad \dfrac{q : (\forall x : D)A \qquad x : D}{\mathbf{ap}(q, x) : A}}{\langle x, \mathbf{ap}(q, x)\rangle : (\exists x : D)A}}{\mathbf{ap}(p, \langle x, \mathbf{ap}(q, x)\rangle) : B[\mathbf{fst}(\langle x, \mathbf{ap}(q, x)\rangle)]} \qquad \dfrac{\mathbf{fst}(\langle x, \mathbf{ap}(q, x)\rangle) = x : D}{B[\mathbf{fst}(\langle x, \mathbf{ap}(q, x)\rangle)] = B : \mathbf{set}}}{\mathbf{ap}(p, \langle x, \mathbf{ap}(q, x)\rangle) : B}}{\lambda x.\mathbf{ap}(p, \langle x, \mathbf{ap}(q, x)\rangle) : (\forall x : D)B}$$

Note here the use of the extensionality of substitution into **set**s as well as the use of the principle that we can infer $a : B$ from $a : A$ and $A = B : \mathbf{set}$. We rely on the syntactic identity $B[x] \equiv B$.

6. We apply **Id**-elimination.

$$\dfrac{\dfrac{p : \mathbf{Id}(A, a, a') \qquad \dfrac{x : A \vdash \mathbf{ap}(c, x) = \mathbf{ap}(c, x) : B}{x : A \vdash \mathbf{refl}(B, \mathbf{ap}(c, x)) : \mathbf{Id}(B, \mathbf{ap}(c, x), \mathbf{ap}(c, x))}}{\mathbf{J}(p, x.\mathbf{refl}(B, \mathbf{ap}(c, x))) : \mathbf{Id}(B, \mathbf{ap}(c, a), \mathbf{ap}(c, a'))}}{\lambda p.\mathbf{J}(p, x) : \mathbf{Id}(A, a, a') \supset \mathbf{Id}(B, \mathbf{ap}(c, a), \mathbf{ap}(c, a'))}$$

An alternative demonstration is the following.

$$\dfrac{q : \mathbf{Id}(A,a,a') \qquad \dfrac{\dfrac{\dfrac{\dfrac{x : A \vdash \mathbf{ap}(c,x) = \mathbf{ap}(c,x) : B}{x : A \vdash \mathbf{refl}(B, \mathbf{ap}(c,x)) : \mathbf{Id}(B, \mathbf{ap}(c,x), \mathbf{ap}(c,x))}}{x : A, p : \mathbf{Id}(A,x,x) \vdash \mathbf{refl}(B, \mathbf{ap}(c,x)) : \mathbf{Id}(B, \mathbf{ap}(c,x), \mathbf{ap}(c,x))}}{x : A \vdash \lambda p.\mathbf{refl}(B, \mathbf{ap}(c,x)) : \mathbf{Id}(A,x,x) \supset \mathbf{Id}(B, \mathbf{ap}(c,x), \mathbf{ap}(c,x))}}{\mathbf{J}(q, x.\lambda p.\mathbf{refl}(B, \mathbf{ap}(c,x))) : \mathbf{Id}(A,a,a') \supset \mathbf{Id}(B, \mathbf{ap}(c,a), \mathbf{ap}(c,a'))}$$

7. We apply **Id**-elimination twice.

$$\dfrac{q : \mathbf{Id}(B,b,b') \qquad \dfrac{p : \mathbf{Id}(A,a,a') \qquad x : A, y : B \vdash \mathbf{refl}(\langle x,y\rangle) : \mathbf{Id}(A \times B, \langle x,y\rangle, \langle x,y\rangle)}{y : B \vdash \mathbf{J}(p, x.\mathbf{refl}(\langle x,y\rangle)) : \mathbf{Id}(A \times B, \langle a,y\rangle, \langle a',y\rangle)}}{\mathbf{J}(q, y.\mathbf{J}(p, x.\mathbf{refl}(\langle x,y\rangle))) : \mathbf{Id}(A \times B, \langle a,b\rangle, \langle a',b'\rangle)}$$

8. *a*) We apply **bool**-elimination with $\mathbf{Id}(\mathbf{bool}, x, \mathsf{t}) \vee \mathbf{Id}(\mathbf{bool}, x, \mathsf{f})$ as our C. Hence we need a proof object of $\mathbf{Id}(\mathbf{bool}, \mathsf{t}, \mathsf{t}) \vee \mathbf{Id}(\mathbf{bool}, \mathsf{t}, \mathsf{f})$ and a proof object of $\mathbf{Id}(\mathbf{bool}, \mathsf{f}, \mathsf{t}) \vee \mathbf{Id}(\mathbf{bool}, \mathsf{f}, \mathsf{f})$. These are constructed as follows.

$$\dfrac{\dfrac{\mathsf{t} : \mathbf{bool}}{\mathbf{refl}(\mathsf{t}) : \mathbf{Id}(\mathbf{bool}, \mathsf{t}, \mathsf{t})}}{\mathbf{i}(\mathbf{refl}(\mathsf{t})) : \mathbf{Id}(\mathbf{bool}, \mathsf{t}, \mathsf{t}) \vee \mathbf{Id}(\mathbf{bool}, \mathsf{t}, \mathsf{f})} \qquad \dfrac{\dfrac{\mathsf{f} : \mathbf{bool}}{\mathbf{refl}(\mathsf{f}) : \mathbf{Id}(\mathbf{bool}, \mathsf{f}, \mathsf{f})}}{\mathbf{j}(\mathbf{refl}(\mathsf{f})) : \mathbf{Id}(\mathbf{bool}, \mathsf{f}, \mathsf{t}) \vee \mathbf{Id}(\mathbf{bool}, \mathsf{f}, \mathsf{f})}$$

Continuing the demonstration, we write the three premisses of **bool**-elimination below each other.

$$\dfrac{\dfrac{\begin{array}{c} x : \mathbf{bool} \\ \mathbf{i}(\mathbf{refl}(\mathsf{t})) : \mathbf{Id}(\mathbf{bool}, \mathsf{t}, \mathsf{t}) \vee \mathbf{Id}(\mathbf{bool}, \mathsf{t}, \mathsf{f}) \\ \mathbf{j}(\mathbf{refl}(\mathsf{f})) : \mathbf{Id}(\mathbf{bool}, \mathsf{f}, \mathsf{t}) \vee \mathbf{Id}(\mathbf{bool}, \mathsf{f}, \mathsf{f}) \end{array}}{\mathbf{if}\, x\, \mathbf{then}\, \mathbf{i}(\mathbf{refl}(\mathsf{t}))\, \mathbf{else}\, \mathbf{j}(\mathbf{refl}(\mathsf{f})) : \mathbf{Id}(\mathbf{bool}, x, \mathsf{t}) \vee \mathbf{Id}(\mathbf{bool}, x, \mathsf{f})}}{\lambda x.\mathbf{if}\, x\, \mathbf{then}\, \mathbf{i}(\mathbf{refl}(\mathsf{t}))\, \mathbf{else}\, \mathbf{j}(\mathbf{refl}(\mathsf{f})) : (\forall x : \mathbf{bool})(\mathbf{Id}(\mathbf{bool}, x, \mathsf{t}) \vee \mathbf{Id}(\mathbf{bool}, x, \mathsf{f}))}$$

b) In this solution we sometimes omit parentheses and write, for instance, $\mathbf{s}x$ instead of $\mathbf{s}(x)$. We aim to apply **N**-elimination with $\mathbf{Id}(\mathbf{N}, x, 0) \vee (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, x, \mathbf{s}y)$ as our C. Hence we need a proof object d of $\mathbf{Id}(\mathbf{N}, 0, 0) \vee (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, 0, \mathbf{s}y)$ and a function e which, given a proof object

$$w : \mathbf{Id}(\mathbf{N}, x, 0) \vee (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, x, y)$$

yields a proof object

$$e[w] : \mathbf{Id}(\mathbf{N}, \mathbf{s}x, 0) \vee (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, \mathbf{s}x, y)$$

The proof object d is easily constructed.

$$\dfrac{\dfrac{0 : \mathbf{N}}{\mathbf{refl}(0) : \mathbf{Id}(\mathbf{N}, 0, 0)}}{\mathbf{i}(\mathbf{refl}(0)) : \mathbf{Id}(\mathbf{N}, 0, 0) \vee (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, 0, \mathbf{s}y)}$$

To find the function e requires more work. Since we are given

$$w : \mathbf{Id}(\mathbf{N}, x, 0) \vee (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, x, y)$$

it is natural to try to construct $e[w]$ by means of $\vee$-elimination. Assume, therefore, first that we are given $q : \mathbf{Id}(\mathbf{N}, x, 0)$.

$$\dfrac{\dfrac{\dfrac{q : \mathbf{Id}(\mathbf{N}, x, 0) \qquad \mathbf{refl}(\mathbf{s}z) : \mathbf{Id}(\mathbf{N}, \mathbf{s}z, \mathbf{s}z)}{\mathbf{J}(q, z.\mathbf{refl}z) : \mathbf{Id}(\mathbf{N}, \mathbf{s}x, \mathbf{s}0)}}{\langle \mathbf{s}0, \mathbf{J}(q, z.\mathbf{refl}z)\rangle : (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, \mathbf{s}x, y)}}{\mathbf{j}\langle \mathbf{s}0, \mathbf{J}(q, z.\mathbf{refl}z)\rangle : \mathbf{Id}(\mathbf{N}, \mathbf{s}x, 0) \vee (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, \mathbf{s}x, y)}$$

Next assume that we are given $p : (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, x, \mathbf{s}y)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{p : (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, x, \mathbf{s}y)}{\mathbf{snd}p : \mathbf{Id}(\mathbf{N}, x, \mathbf{s}(\mathbf{fst}p))} \qquad \mathbf{refl}(\mathbf{s}z) : \mathbf{Id}(\mathbf{N}, \mathbf{s}z, \mathbf{s}z)}{\mathbf{J}(\mathbf{snd}p, z.\mathbf{refl}(\mathbf{s}z)) : \mathbf{Id}(\mathbf{N}, \mathbf{s}x, \mathbf{ss}(\mathbf{fst}p))}}{\langle \mathbf{ss}(\mathbf{fst}p), \mathbf{J}(\mathbf{snd}p, z.\mathbf{refl}(\mathbf{s}z))\rangle : (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, \mathbf{s}x, y)}}{\mathbf{j}\langle \mathbf{ss}(\mathbf{fst}p), \mathbf{J}(\mathbf{snd}p, z.\mathbf{refl}(\mathbf{s}z))\rangle : \mathbf{Id}(\mathbf{N}, \mathbf{s}x, 0) \vee (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, \mathbf{s}x, y)}$$

With $w : \mathbf{Id}(\mathbf{N}, x, 0) \vee (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, x, y)$ as major premiss, $\vee$-elimination then yields

$$\begin{aligned}&\mathbf{D}(w, q.\mathbf{j}\langle \mathbf{s}0, \mathbf{J}(q, z.\mathbf{refl}z)\rangle, p.\mathbf{j}\langle \mathbf{ss}(\mathbf{fst}p), \mathbf{J}(\mathbf{snd}p, z.\mathbf{refl}(\mathbf{s}z))\rangle) :\\ &\mathbf{Id}(\mathbf{N}, \mathbf{s}x, 0) \vee (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, \mathbf{s}x, y)\end{aligned}$$

With the assumption $x : \mathbf{N}$ as major premiss, $\mathbf{N}$-elimination yields

$$\begin{aligned}&\mathbf{R}(x, \mathbf{i}(\mathbf{refl}0), w.\mathbf{D}(w, q.\mathbf{j}\langle \mathbf{s}0, \mathbf{J}(q, z.\mathbf{refl}z)\rangle, p.\mathbf{j}\langle \mathbf{ss}(\mathbf{fst}p), \mathbf{J}(\mathbf{snd}p, z.\mathbf{refl}(\mathbf{s}z))\rangle)) :\\ &\mathbf{Id}(\mathbf{N}, x, 0) \vee (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, x, \mathbf{s}y)\end{aligned}$$

Note that w has become bound here, so only x is free. A proof object of

$$(\forall x : \mathbf{N})(\mathbf{Id}(\mathbf{N}, x, 0) \vee (\exists y : \mathbf{N})\mathbf{Id}(\mathbf{N}, x, \mathbf{s}y))$$

is then constructed by means of Π-introduction.

References

Aczel, P. (1978). The type theoretic interpretation of constructive set theory. In Macintyre, A., Pacholski, L., and Paris, J., editors, *Logic Colloquium 77*, pages 55–66. North-Holland, Amsterdam.

Almog, J. (1991). The What and the How. *Journal of Philosophy*, 88:225–244.

Carnap, R. (1934). *Logische Syntax der Sprache*. Julius Springer, Vienna.

Diller, J. and Troelstra, A. (1984). Realizability and intuitionistic logic. *Synthese*, 60:253–282.

Dummett, M. (1973). *Frege. Philosophy of Language*. Duckworth, London. Cited from the second edition (1981).

Frege, G. (1893). *Grundgesetze der Arithmetik I*. Hermann Pohle, Jena.

Garner, R. (2009). On the strength of dependent products in the type theory of Martin-Löf. *Annals of Pure and Applied Logic*, 160:1–12.

Gentzen, G. (1933). Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431.

Granström, J. G. (2011). *Treatise on Intuitionistic Type Theory*. Springer, Dordrecht.

Hale, B. and Wright, C. (2001). To bury Caesar.... In *The Reason's Proper Study*, pages 335–396. Oxford University Press, Oxford.

Hofmann, M. (1995). *Extensional Concepts in Intensional Type Theory*. PhD thesis, University of Edinburgh.

Hofmann, M. and Streicher, T. (1998). The groupoid interpretation of type theory. In Sambin, G. and Smith, J. M., editors, *Twenty-five Years of Constructive Type Theory*, pages 83–111. Oxford University Press, Oxford.

Howard, W. A. (1980). The formulae-as-types notion of construction. In Seldin, J. P. and Hindley, J. R., editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, London.

Klev, A. (2014). *Categories and Logical Syntax*. PhD thesis, Leiden University.

Martin-Löf, P. (1971). Hauptsatz for the intuitionistic theory of iterated inductive definitions. In Fenstad, J. E., editor, *Proceedings of the Second Scandinavian Logic Symposium*, pages 179–216. North-Holland, Amsterdam.

Martin-Löf, P. (1975a). About models for intuitionistic type theories and the notion of definitional equality. In Kanger, S., editor, *Proceedings of the Third Scandinavian Logic Symposium*, pages 81–109. North-Holland, Amsterdam.

Martin-Löf, P. (1975b). An intuitionistic theory of types: Predicative part. In Rose, H. E. and Shepherdson, J. C., editors, *Logic Colloquium '73*, pages 73–118. North-Holland, Amsterdam.

Martin-Löf, P. (1982). Constructive mathematics and computer programming. In Cohen, J. L., Łoś, J., et al., editors, *Logic, Methodology and Philosophy of Science VI, 1979*, pages 153–175. North-Holland, Amsterdam.

Martin-Löf, P. (1984). *Intuitionistic Type Theory*. Bibliopolis, Naples.

Martin-Löf, P. (1992). Substitution calculus. Lecture notes. Available at `http://archive-pml.github.io/martin-lof/pdfs/Substitution-calculus-1992`.

pdf.

Martin-Löf, P. (1996). On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1:11–60.

Martin-Löf, P. (2011). When did 'judgement' come to be a term of logic? Lecture held at *École Normale Supérieure* on 14 October 2011. Video recording available at http://savoirs.ens.fr//expose.php?id=481.

Nordström, B., Petersson, K., and Smith, J. (1990). *Programming in Martin-Löf's Type Theory.* Oxford University Press, Oxford.

Nordström, B., Petersson, K., and Smith, J. M. (2000). Martin-Löf's type theory. In Abramsky, S., Gabbay, D., and Maibaum, T. S. E., editors, *Handbook of Logic in Computer Science. Volume 5: Logic and Algebraic Methods*, pages 1–37. Oxford University Press, Oxford.

Prawitz, D. (1965). *Natural Deduction.* Almqvist & Wiksell, Stockholm.

Quine, W. V. (1969). *Ontological Relativity and Other Essays.* Columbia University Press, New York.

Ranta, A. (1995). *Type-Theoretical Grammar.* Oxford University Press, Oxford.

Schroeder-Heister, P. (1984). A natural extension of natural deduction. *Journal of Symbolic Logic*, 49:1284–1300.

Sundholm, B. G. (2006). Semantic values for natural deduction derivations. *Synthese*, 148:623–638.

Sundholm, B. G. (2009). A century of judgement and inference, 1837-1936: Some strands in the development of logic. In Haaparanta, L., editor, *The Development of Modern Logic*, pages 263–317. Oxford University Press, Oxford.

Sundholm, B. G. (2012). "Inference versus consequence" revisited: inference, consequence, conditional, implication. *Synthese*, 187:943–956.

Tasistro, A. (1993). Formulation of Martin-Löf's theory of types with explicit substitutions. Master's thesis, Chalmers University of Technology, Gothenburg.

The Univalent Foundations Program (2013). *Homotopy Type Theory: Univalent Foundations of Mathematics.* http://homotopytypetheory.org/book, Institute for Advanced Study, Princeton.

Troelstra, A. and van Dalen, D. (1988). *Constructivism in Mathematics.* North-Holland, Amsterdam.

Zermelo, E. (1930). Über Grenzzahlen und Mengenbereiche. *Fundamenta Mathematicae*, 16:29–47.