PER MARTIN-LÖF

INTUITIONISTIC TYPE THEORY

Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980

BIBLIOPOLIS

STUDIES IN PROOF THEORY

LECTURE NOTES

1

ISBN 88-7088-105-9

Printed in Italy by « Grafitalia »
Via Censi dell'Arco, 25 - Cercola (Napoli)

CONTENTS

Introductory remarks 1
Propositions and judgements 3
Explanations of the forms of judgement 7
Propositions 11
Rules of equality 14
Hypothetical judgements and substitution rules 16
Judgements with more than one assumption and contexts 19
Sets and categories 21
General remarks on the rules 24
Cartesian product of a family of sets 26
Definitional equality 31
Applications of the cartesian product 32
Disjoint union of a family of sets 39
Applications of the disjoint union 42
The axiom of choice 50
The notion of such that 53
Disjoint union of two sets 55
Propositional equality 59
Finite sets 65
Consistency 69
Natural numbers 71
Lists 77
Wellorderings 79
Universes 87

Preface

These lectures were given in Padova at the Laboratorio per Ricerche di Dinamica dei Sistemi e di Elettronica Biomedica of the Consiglio Nazionale delle Ricerche during the month of June 1980. I am indebted to Dr. Enrico Pagello of that laboratory for the opportunity of so doing. The audience was made up by philosophers, mathematicians and computer scientists. Accordingly, I tried to say something which might be of interest to each of these three categories. Essentially the same lectures, albeit in a somewhat improved and more advanced form, were given later in the same year as part of the meeting on Konstruktive Mengenlehre und Typentheorie which was organized in Munich by Prof. Dr. Helmut Schwichtenberg, to whom I am indebted for the invitation, during the week 29 September - 3 October 1980.

The main improvement of the Munich lectures, as compared with those given in Padova, was the adoption of a systematic higher level (Ger. Stufe) notation which allows me to write simply

$$\Pi(A,B),\ \Sigma(A,B),\ W(A,B),\ \lambda(b),$$
$$E(c,d),\ D(c,d,e),\ R(c,d,e),\ T(c,d)$$

instead of

$$(\Pi x \in A)B(x),\ (\Sigma x \in A)B(x),\ (Wx \in A)B(x),\ (\lambda x)b(x),$$
$$E(c,(x,y)d(x,y)),\ D(c,(x)d(x),(y)e(y)),\ R(c,d,(x,y)e(x,y)),$$
$$T(c,(x,y,z)d(x,y,z)),$$

respectively. Moreover, the use of higher level variables and constants makes it possible to formulate the elimination and equality rules for the cartesian product in such a way that they follow the

same pattern as the elimination and equality rules for all the other type forming operations. In their new formulation, these rules read

Π-elimination

$$\frac{c \in \Pi(A,B) \qquad \begin{matrix}(y(x) \in B(x)\ (x \in A)) \\ d(y) \in C(\lambda(y))\end{matrix}}{F(c,d) \in C(c)}$$

and

Π-equality

$$\frac{\begin{matrix}(x \in A) \\ b(x) \in B(x)\end{matrix} \qquad \begin{matrix}(y(x) \in B(x)\ (x \in A)) \\ d(y) \in C(\lambda(y))\end{matrix}}{F(\lambda(b),d) = d(b) \in C(\lambda(b))}$$

respectively. Here y is a bound function variable, F is a new non-canonical (eliminatory) operator by means of which the binary application operation can be defined, putting

$$Ap(c,a) \equiv F(c,(y)y(a)),$$

and $y(x) \in B(x)$ $(x \in A)$ is an assumption, itself hypothetical, which has been put within parentheses to indicate that it is being discharged. A program of the new form F(c,d) has value e provided c has value λ (b) and d(b) has value e. This rule for evaluating F(c,d) reduces to the lazy evaluation rule for Ap(c,a) when the above definition is being made. Choosing C(z) to be B(a), thus independent of z, and d(y) to be y(a), the new elimination rule reduces to the old one and the new equality rule to the first of the two old equality rules. Moreover, the second of these, that is, the rule

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

can be derived by means of the I-rules in the same way as the rule

$$\frac{c \in \Sigma(A,B)}{c = (p(c),q(c))) \in \Sigma(A,B)}$$

is derived by way of example on p. 62 of the main text. Conversely, the new elimination and equality rules can be derived from the old ones by making the definition

$$F(c,d) \equiv d((x)Ap(c,x)).$$

So, actually, they are equivalent.

It only remains for me to thank Giovanni Sambin for having undertaken, at his own suggestion, the considerable work of writing and typing these notes, thereby making the lectures accessible to a wider audience.

Stockholm, January 1984,

Per Martin-Löf

Introductory remarks

Mathematical logic and the relation between logic and mathematics have been interpreted in at least three different ways:

(1) mathematical logic as symbolic logic, or logic using mathematical symbolism;

(2) mathematical logic as foundations (or philosophy) of mathematics;

(3) mathematical logic as logic studied by mathematical methods, as a branch of mathematics.

We shall here mainly be interested in mathematical logic in the second sense. What we shall do is also mathematical logic in the first sense, but certainly not in the third.

The principal problem that remained after Principia Mathematica was completed was, according to its authors, that of justifying the axiom of reducibility (or, as we would now say, the impredicative comprehension axiom). The ramified theory of types was predicative, but it was not sufficient for deriving even elementary parts of analysis. So the axiom of reducibility was added on the pragmatic ground that it was needed, although no satisfactory justification (explanation) of it could be provided. The whole point of the ramification was then lost, so that it might just as well be abolished. What then remained was the simple theory of types. Its official justification (Wittgenstein, Ramsey) rests on the interpretation of propositions as truth values and propositional functions (of one or several variables) as truth functions. The laws of the classical propositional logic are then clearly valid, and so are the quantifier laws, as long as quantification is restricted to finite domains. However, it does not seem possible to make sense of quantification over infinite domains, like the

domain of natural numbers, on this interpretation of the notions of proposition and propositional function. For this reason, among others, what we develop here is an intuitionistic theory of types, which is also predicative (or ramified). It is free from the deficiency of Russell's ramified theory of types, as regards the possibility of developing elementary parts of mathematics, like the theory of real numbers, because of the presence of the operation which allows us to form the cartesian product of any given family of sets, in particular, the set of all functions from one set to another.

In two areas, at least, our language seems to have advantages over traditional foundational languages. First, Zermelo-Fraenkel set theory cannot adequately deal with the foundational problems of category theory, where the category of all sets, the category of all groups, the category of functors from one such category to another etc. are considered. These problems are coped with by means of the distinction between sets and categories (in the logical or philosophical sense, not in the sense of category theory) which is made in intuitionistic type theory. Second, present logical symbolisms are inadequate as programming languages, which explains why computer scientists have developed their own languages (FORTRAN, ALGOL, LISP, PASCAL, ...) and systems of proof rules (Hoare[1], Dijkstra[2], ...). We have shown elsewhere[3] how the additional richness of type theory, as compared with first order predicate logic, makes it usable as a programming language.

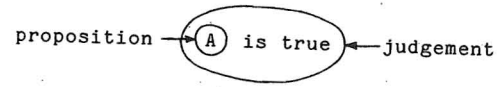
[1] C. A. Hoare, An axiomatic basis of computer programming, Communications of the ACM, Vol. 12, 1969, pp. 576-580 and 583.

[2] E. W. Dijkstra, A discipline of Programming, Prentice Hall, Englewood Cliffs, N.J., 1976.

[3] P. Martin-Löf, Constructive mathematics and computer programming, Logic, Methodology and Philosophy of Science VI, Edited by L. J. Cohen, J. Los, H. Pfeiffer and K.-P. Podewski, North-Holland, Amsterdam, 1982, pp. 153-175.

Propositions and judgements

Here the distinction between proposition (Ger. Satz) and assertion or judgement (Ger. Urteil) is essential. What we combine by means of the logical operations ($\bot, \supset, \&, \vee, \forall, \exists$) and hold to be true are propositions. When we hold a proposition to be true, we make a judgement:



In particular, the premisses and conclusion of a logical inference are judgements.

The distinction between propositions and judgements was clear from Frege to Principia. These notions have later been replaced by the formalistic notions of formula and theorem (in a formal system), respectively. Contrary to formulas, propositions are not defined inductively. So to speak, they form an open concept. In standard textbook presentations of first order logic, we can distinguish three quite separate steps:

(1) inductive definition of terms and formulas,

(2) specification of axioms and rules of inference,

(3) semantical interpretation.

Formulas and deductions are given meaning only through semantics, which is usually done following Tarski and assuming set theory.

What we do here is meant to be closer to ordinary mathematical practice. We will avoid keeping form and meaning (content) apart. Instead we will at the same time display certain forms of judgement and inference that are used in mathematical proofs and explain them semantically. Thus we make explicit what is usually implicitly taken for

granted. When one treats logic as any other branch of mathematics, as in the metamathematical tradition originated by Hilbert, such judgements and inferences are only partially and formally represented in the so-called object language, while they are implicitly used, as in any other branch of mathematics, in the so-called metalanguage.

Our main aim is to build up a system of formal rules representing in the best possible way informal (mathematical) reasoning. In the usual natural deduction style, the rules given are not quite formal. For instance, the rule

$$\frac{A}{A \vee B}$$

takes for granted that A and B are formulas, and only then does it say that we can infer $A \vee B$ to be true when A is true. If we are to give a formal rule, we have to make this explicit, writing

$$\frac{A \text{ prop.} \qquad B \text{ prop.} \qquad A \text{ true}}{A \vee B \text{ true}}$$

or

$$\frac{A, B \text{ prop.} \qquad \vdash A}{\vdash A \vee B}$$

where we use, like Frege, the symbol $\vdash$ to the left of A to signify that A is true. In our system of rules, this will always be explicit.

A rule of inference is justified by explaining the conclusion on the assumption that the premisses are known. Hence, before a rule of inference can be justified, it must be explained what it is that we must know in order to have the right to make a judgement of any one of the various forms that the premisses and conclusion can have.

We use four forms of judgement:

(1) A is a set (abbr. A set),

(2) A and B are equal sets (A = B),

(3) a is an element of the set A (a ∈ A),

(4) a and b are equal elements of the set A (a = b ∈ A).

(If we read ∈ literally as ἐστί , then we might write A ∈ Set, A = B ∈ Set, a ∈ El(A), a = b ∈ El(A), respectively.) Of course, any syntactic variables could be used; the use of small letters for elements and capital letters for sets is only for convenience. Note tha in ordinary set theory, a ∈ b and a = b are propositions, while they are judgements here. A judgement of the form A = B has no meaning unless we already know A and B to be sets. Likewise, a judgement of the form a ∈ A presupposes that A is a set, and a judgement of the form a = b ∈ A presupposes, first, that A is a set, and, second, that a an b are elements of A.

Each form of judgement admits of several different readings, as in the table:

A set	a ∈ A	
A is a set	a is an element of the set A	A is nonempty
A is a proposition	a is a proof (construction) of the proposition A	A is true
A is an intention (expectation)	a is a method of fulfilling (realizing) the intention (expectation) A	A is fulfillabl (realizable)
A is a problem (task)	a is a method of solving the problem (doing the task) A	A is solvable

The second, logical interpretation is discussed toghether with rules below. The third was suggested by Heyting[4] and the fourth by Kolmogorov[5]. The last is very close to programming. "a is a method ..." can be read as "a is a program ...". Since programming languages have a formal notation for the program a, but not for A, we complete the sentence with "... which meets the specification A". In Kolmogorov's interpretation, the word problem refers to something to be done and the word program to how to do it. The analogy between the first and the second interpretation is implicit in the Brouwer-Heyting interpretation of the logical constants. It was made more explicit by Curry and Feys[6], but only for the implicational fragment, and it was extended to intuitionistic first order arithmetic by Howard[7]. It is the only known way of interpreting intuitionistic logic so that the axiom of choice becomes valid.

To distinguish between proofs of judgements (usually in tree-like form) and proofs of propositions (here identified with elements, thus to the left of $\in$) we reserve the word construction for the latter and use it when confusion might occur.

[4] A. Heyting, Die intuitionistische Grundlegung der Mathematik, Erkenntnis, Vol. 2, 1931, pp. 106-115.

[5] A. N. Kolmogorov, Zur Deutung der intuitionistischen Logik, Mathematische Zeitschrift, Vol. 35, 1932, pp. 58-65.

[6] H. B. Curry and R. Feys, Combinatory Logic, Vol. 1, North-Holland, Amsterdam, 1958, pp. 312-315.

[7] W. A. Howard, The formulae-as-types notion of construction, To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Academic Press, London, 1980, pp. 479-490.

Explanations of the forms of judgement

For each one of the four forms of judgement, we now explain what a judgement of that form means. We can explain what a judgement, say of the first form, means by answering one of the following three questions:

What is a set?

What is it that we must know in order to have the right to judge something to be a set?

What does a judgement of the form "A is a set" mean?

The first is the ontological (ancient Greek), the second the epistemological (Descartes, Kant, ...) and the third the semantical (modern) way of posing essentially the same question. At first sight, we could assume that a set is defined by prescribing how its elements are formed. This we do when we say that the set of natural numbers N is defined by giving the rules:

$$0 \in N \qquad \frac{a \in N}{a' \in N}$$

by which its elements are constructed. However, the weakness of this definition is clear: 10^{10}, for instance, though not obtainable with the given rules, is clearly an element of N, since we know that we can bring it to the form a' for some $a \in N$. We thus have to distinguish the elements which have a form by which we can directly see that they are the result of one of the rules, and call them canonical, from all other elements, which we will call noncanonical.

But then, to be able to define when two noncanonical elements are equal, we must also prescribe how two equal canonical elements are formed. So:

(1) a set A is defined by prescribing how a canonical element of A is formed as well as how two equal canonical elements of A are formed.

This is the explanation of the meaning of a judgement of the form A is a set. For example, to the rules for N above, we must add

$$0 = 0 \in N \qquad \text{and} \qquad \frac{a = b \in N}{a' = b' \in N}$$

To take another example, $A \times B$ is defined by the rule

$$\frac{a \in A \qquad b \in B}{(a,b) \in A \times B}$$

which prescribes how canonical elements are formed, and the rule

$$\frac{a = c \in A \qquad b = d \in B}{(a,b) = (c,d) \in A \times B}$$

by means of which equal canonical elements are formed. There is no limitation on the prescription defining a set, except that equality between canonical elements must always be defined in such a way as to be reflexive, symmetric and transitive.

Now suppose we know A and B to be sets, that is, we know how canonical elements and equal canonical elements of A and B are formed. Then we stipulate:

(2) two sets A and B are equal if

$$\frac{a \in A}{a \in B} \quad \text{(that is, } \frac{a \in A}{a \in B} \text{ and } \frac{a \in B}{a \in A}\text{)}$$

and

$$\frac{a = b \in A}{a = b \in B}$$

for arbitrary canonical elements a, b.

This is the meaning of a judgement of the form A = B.

When we explain what an element of a set A is, we must assume we know that A is a set, that is, in particular, how its canonical elements are formed. Then:

(3) an element a of a set A is a method (or program) which, whe executed, yields a canonical element of A as result.

This is the meaning of a judgement of the form $a \in A$. Note that here we assume the notion of method as primitive. The rules of computatic (execution) of the present language will be such that the computatic of an element a of a set A terminates with a value b as soon as the outermost form of b tells that it is a canonical element of A (norma order or lazy evaluation). For instance, the computation of $2 + 2 \in$ gives the value $(2 + 1)'$, which is a canonical element of N since $2 + 1 \in N$.

Finally:

(4) two arbitrary elements a, b of a set A are equal if, when executed, a and b yield equal canonical elements of A as result

This is the meaning of a judgement of the form $a = b \in A$. This definition makes good sense since it is part of the definition of a set what it means for two canonical elements of the set to be equal.

Example. If $e, f \in A \times B$, then e and f are methods which yield canonical elements $(a,b), (c,d) \in A \times B$, respectively, as results, and $e = f \in A \times B$ if $(a,b) = (c,d) \in A \times B$, which in turn holds if $a = c \in A$ and $b = d \in B$.

Propositions

Classically, a proposition is nothing but a truth value, that is, an element of the set of truth values, whose two elements are the true and the false. Because of the difficulties of justifying the rules for forming propositions by means of quantification over infinite domains, when a proposition is understood as a truth value, this explanation is rejected by the intuitionists and replaced by saying that

> a proposition is defined by laying down what counts as a proof of the proposition,

and that

> a proposition is true if it has a proof, that is, if a proof of it can be given[8].

Thus, intuitionistically, truth is identified with provability, though of course not (because of Gödel's incompleteness theorem) with derivability within any particular formal system.

The explanations of the meanings of the logical operations, which fit together with the intuitionistic conception of what a proposition is, are given by the standard table:

[8] D. Prawitz, Intuitionistic logic: a philosophical challenge, Logic and Philosophy, Edited by G. H. von Wright, Martinus Nijhoff, The Hague, pp. 1-10.

a proof of the proposition	consists of
$\bot$	-
$A \,\&\, B$	a proof of A and a proof of B
$A \vee B$	a proof of A or a proof of B
$A \supset B$	a method which takes any proof of A into a proof of B
$(\forall x)B(x)$	a method which takes an arbitrary individual a into a proof of B(a)
$(\exists x)B(x)$	an individual a and a proof of B(a)

the first line of which should be interpreted as saying that there is nothing that counts as a proof $\bot$.

The above table can be made more explicit by saying:

a proof of the proposition	has the form
$\bot$	-
$A \,\&\, B$	(a,b), where a is a proof of A and b is a proof of B
$A \vee B$	i(a), where a is a proof of A, or j(b), where b is a proof of B
$A \supset B$	$(\lambda x)b(x)$, where b(a) is a proof of B provided a is a proof of A
$(\forall x)B(x)$	$(\lambda x)b(x)$, where b(a) is a proof of B(a) provided a is an individual
$(\exists x)B(x)$	(a,b), where a is an individual and b is a proof of B(a)

As it stands, this table is not strictly correct, since it shows proofs of canonical form only. An arbitary proof, in analogy with an arbitrary element of a set, is a method of producing a proof of canonical form.

If we take seriously the idea that a proposition is defined by laying down how its canonical proofs are formed (as in the second table above) and accept that a set is defined by prescribing how its canonical elements are formed, then it is clear that it would only lead to unnecessary duplication to keep the notions of proposition and set (and the associated notions of proof of a proposition and element of a set) apart. Instead, we simply identify them, that is, treat them as one and the same notion. This is the formulae-as-types (propositions-as-sets) interpretation on which intuitionistic type theory is based.

Rules of equality

We now begin to build up a system of rules. First, we give the following rules of equality, which are easily explained using the fact that they hold, by definition, for canonical elements:

Reflexivity

$$\frac{a \in A}{a = a \in A} \qquad \frac{A \text{ set}}{A = A}$$

Symmetry

$$\frac{a = b \in A}{b = a \in A} \qquad \frac{A = B}{B = A}$$

Transitivity

$$\frac{a = b \in A \qquad b = c \in A}{a = c \in A} \qquad \frac{A = B \qquad B = C}{A = C}$$

For instance, a detailed explanation of transitivity is: $a = b \in A$ means that a and b yield canonical elements d and e, respectively, and that $d = e \in A$. Similarly, if c yields f, $e = f \in A$. Since we assume transitivity for canonical elements, we obtain $d = f \in A$, which means that $a = c \in A$.

The meaning of $A = B$ is that

$$\frac{a \in A}{\;}$$
$$a \in B$$

and

$$\frac{a = b \in A}{\;}$$
$$a = b \in B$$

for a, b canonical elements of A and B. From the same for $B = C$, we also obtain

$$\frac{a \in A}{\;}$$
$$a \in C$$

and

$$\frac{a = b \in A}{\;}$$
$$a = b \in C$$

for a, b canonical elements, which is the meaning of $A = C$.

In the same evident way, the meaning of $A = B$ justifies the rules:

Equality of sets

$$\frac{a \in A \qquad A = B}{a \in B} \qquad \frac{a = b \in A \qquad A = B}{a = b \in B}$$

Hypothetical judgements and substitution rules

The four basic forms of judgement are generalized in order to express also hypothetical judgements, i.e. judgements which are made under assumptions. In this section, we treat the case of one such assumption. So assume that A is a set. The first form of judgement is generalized to the hypothetical form

(1) B(x) set (x ∈ A)

which says that B(x) is a set under the assumption x ∈ A, or, better, that B(x) is a family of sets over A. A more traditional notation is $\{B_x\}_{x \in A}$ or $\{B_x : x \in A\}$. The meaning of a judgement of the form (1) is that B(a) is a set whenever a is an element of A, and also that B(a) and B(c) are equal sets whenever a and c are equal elements of A. By virtue of this meaning, we immediately see that the following substitution rules are correct:

Substitution

$$\frac{a \in A \qquad \begin{matrix}(x \in A)\\ B(x)\ \text{set}\end{matrix}}{B(a)\ \text{set}} \qquad\qquad \frac{a = c \in A \qquad \begin{matrix}(x \in A)\\ B(x)\ \text{set}\end{matrix}}{B(a) = B(c)}$$

The notation

$$\begin{matrix}x \in A\\ B(x)\ \text{set}\end{matrix}$$

only recalls that we make (have a proof of) the judgement that B(x) is a set under the assumption x ∈ A, which does not mean that we must have a derivation within any particular formal system (like the one that we are in the process of building up). When an assumption x ∈ A is discharged by the application of a rule, we write it inside brackets.

The meaning of a hypothetical judgement of the form

(2) B(x) = D(x) (x ∈ A)

which says that B(x) and D(x) are equal families of sets over the set A, is that B(a) and D(a) are equal sets for any element a of A (so, in particular, B(x) and D(x) must be families of sets over A). Therefore the rule

Substitution

$$\frac{a \in A \qquad \begin{matrix}(x \in A)\\ B(x) = D(x)\end{matrix}}{B(a) = D(a)}$$

is correct. We can now derive the rule

$$\frac{a = c \in A \qquad \begin{matrix}(x \in A)\\ B(x) = D(x)\end{matrix}}{B(a) = D(c)}$$

from the above rules. In fact, from a = c ∈ A and B(x) set (x ∈ A), we obtain B(a) = B(c) by the second substitution rule, and from c ∈ A, which is implicit in a = c ∈ A, B(c) = D(c) by the third substitution rule. So B(a) = D(c) by transitivity.

A hypothetical judgement of the form

(3) $b(x) \in B(x)$ $(x \in A)$

means that we know $b(a)$ to be an element of the set $B(a)$ assuming we know a to be an element of the set A, and that $b(a) = b(c) \in B(a)$ whenever a and c are equal elements of A. In other words, $b(x)$ is an extensional function with domain A and range $B(x)$ depending on the argument x. Then the following rules are justified:

Substitution

$$\frac{a \in A \qquad \overset{(x \in A)}{b(x) \in B(x)}}{b(a) \in B(a)} \qquad\qquad \frac{a = c \in A \qquad \overset{(x \in A)}{b(x) \in B(x)}}{b(a) = b(c) \in B(a)}$$

Finally, a judgement of the form

(4) $b(x) = d(x) \in B(x)$ $(x \in A)$

means that $b(a)$ and $d(a)$ are equal elements of the set $B(a)$ for any element a of the set A. We then have

Substitution

$$\frac{a \in A \qquad \overset{(x \in A)}{b(x) = d(x) \in B(x)}}{b(a) = d(a) \in B(a)}$$

which is the last substitution rule.

Judgements with more than one assumption and contexts

We may now further generalize judgements to include hypothetical judgements with an arbitrary number n of assumptions. We explain their meaning by induction, that is, assuming we understand the meaning of judgements with n-1 assumptions. So assume we know that

A_1 is a set,

$A_2(x_1)$ is a family of sets over A_1,

$A_3(x_1,x_2)$ is a family of sets with two indices $x_1 \in A_1$ and $x_2 \in A_2(x_1)$,

...

$A_n(x_1,\ldots,x_{n-1})$ is a family of sets with n-1 indices $x_1 \in A_1$, $x_2 \in A_2(x_1)$, ..., $x_{n-1} \in A_{n-1}(x_1,\ldots,x_{n-2})$.

Then a judgement of the form

(1) $A(x_1,\ldots,x_n)$ set $(x_1 \in A_1,\ x_2 \in A_2(x_1),\ \ldots,\ x_n \in A_n(x_1,\ldots,x_{n-1}))$

means that $A(a_1,\ldots,a_n)$ is a set whenever $a_1 \in A_1$, $a_2 \in A_2(a_1)$, ..., $a_n \in A_n(a_1,\ldots,a_{n-1})$ and that $A(a_1,\ldots,a_n) = A(b_1,\ldots,b_n)$ whenever $a_1 = b_1 \in A_1$, ..., $a_n = b_n \in A_n(a_1,\ldots,a_{n-1})$. We say that $A(x_1,\ldots,x_n)$ is a family of sets with n indices. The n assumptions in a judgement of the form (1) constitute what we call the context, which plays a role analogous to the sets of formulae Γ, Δ (extra formulae) appearing in Gentzen sequents. Note also that any initial segment of a context is always a context. Because of the meaning of a hypothetical

judgement of the form (1), we see that the first two rules of substitution may be extended to the case of n assumptions, and we understand these extensions to be given.

It is by now clear how to explain the meaning of the remaining forms of hypothetical judgement:

(2) $A(x_1,\ldots,x_n) = B(x_1,\ldots,x_n)$ $(x_1 \in A_1, \ldots, x_n \in A_n(x_1,\ldots,x_{n-1}))$

(equal families of sets with n indices),

(3) $a(x_1,\ldots,x_n) \in A(x_1,\ldots,x_n)$ $(x_1 \in A_1, \ldots, x_n \in A_n(x_1,\ldots,x_{n-1}))$

(function with n arguments),

(4) $a(x_1,\ldots,x_n) = b(x_1,\ldots,x_n) \in A(x_1,\ldots,x_n)$
$(x_1 \in A_1, \ldots, x_n \in A_n(x_1,\ldots,x_{n-1}))$

(equal functions with n arguments),

and we assume the corresponding substitution rules to be given.

Sets and categories

A category is defined by explaining what an object of the category is and when two such objects are equal. A category need not be a set, since we can grasp what it means to be an object of a given category even without exhaustive rules for forming its objects. For instance, we now grasp what a set is and when two sets are equal, so we have defined the category of sets (and, by the same token, the category of propositions), but it is not a set. So far, we have defined several categories:

the category of sets (or propositions),

the category of elements of a given set (or proofs of a proposition),

the category of families of sets $B(x)$ $(x \in A)$ over a given set A,

the category of functions $b(x) \in B(x)$ $(x \in A)$, where A set, $B(x)$ set $(x \in A)$,

the category of families of sets $C(x,y)$ $(x \in A, y \in B(x))$, where A set, $B(x)$ set $(x \in A)$,

the category of functions $c(x,y) \in C(x,y)$ $(x \in A, y \in B(x))$, where A is a set, $B(x)$ $(x \in A)$ and $C(x,y)$ $(x \in A, y \in B(x))$ families of sets,

etc.

In addition to these, there are higher categories, like the category of binary functions which take two sets into another set. The function $\times$, which takes two sets A and B into their cartesian product $A \times B$,

is an example of an object of that category.

We will say object of a category but element of a set, which reflects the difference between categories and sets. To define a category it is not necessary to prescribe how its objects are formed, but just to grasp what an (arbitrary) object of the category is. Each set determines a category, namely the category of elements of the set, but not conversely: for instance, the category of sets and the category of propositions are not sets, since we cannot describe how all their elements are formed. We can now say that a judgement is a statement to the effect that something is an object of a category ($a \in A$, A set, ...) or that two objects of a category are equal ($a = b \in A$, $A = B$, ...).

What about the word type in the logical sense given to it by Russell with his ramified (resp. simple) theory of types? Is type synonymous with category or with set? In some cases with the one, it seems, and in other cases with the other. And it is this confusion of two different concepts which has led to the impredicativity of the simple theory of types. When a type is defined as the range of significance of a propositional function, so that types are what the quantifiers range over, then it seems that a type is the same thing as a set. On the other hand, when one speaks about the simple types of propositions, properties of individuals, relations between individuals etc., it seems as if types and categories are the same. The important difference between the ramified types of propositions, properties, relations etc. of some finite order and the simple types of all propositions, properties, relations etc. is precisely that the ramified types are (or can be understood as) sets, so that it makes sense to quantify over them, whereas the simple types are mere categories.

For example, B^A is a set, the set of functions from the set A to

the set B (B^A will be introduced as an abbreviation for $(\prod x \in A)B($ when B(x) is constantly equal to B). In particular, $\{0,1\}^A$ is a se but it is not the same thing as $\mathcal{P}(A)$, which is only a category. Th reason that B^A can be construed as a set is that we take the notion of function as primitive, instead of defining a function as a set o ordered pairs or a binary relation satisfying the usual existence a uniqueness conditions, which would make it a category (like $\mathcal{P}(A)$) instead of a set.

When one speaks about data types in computer science, one migh just as well say data sets. So here type is always synonymous with set and not with category.

General remarks on the rules

We now start to give the rules for the different symbols we use. We will follow a common pattern in giving them. For each operation we have four rules:

set formation,

introduction,

elimination,

equality.

The formation rule says that we can form a certain set (proposition) from certain other sets (propositions) or families of sets (propositional functions). The introduction rules say what are the canonical elements (and equal canonical elements) of the set, thus giving its meaning. The elimination rule shows how we may define functions on the set defined by the introduction rules. The equality rules relate the introduction and elimination rules by showing how a function defined by means of the elimination rule operates on the canonical elements of the set which are generated by the introduction rules.

In the interpretation of sets as propositions, the formation rules are used to form propositions, introduction and elimination rules are like those of Gentzen[9], and the equality rules correspond to the reduction rules of Prawitz[10].

[9] G. Gentzen, Untersuchungen über das logische Schliessen, Mathematische Zeitschrift, Vol. 39, 1934, pp .176-210 and 405-431.

[10] D. Prawitz, Natural Deduction, A Proof-Theoretical Study, Almqvist & Wiksell, Stockholm, 1965.

We remark here also that to each rule of set formation, introduction and elimination, there corresponds an equality rule, which allows us to substitute equals for equals.

The rules should be rules of immediate inference; we cannot further analyse them, but only explain them. However, in the end, no explanation can substitute each individual's understanding.

Cartesian product of a family of sets

Given a set A and a family of sets B(x) over the set A, we can form the product:

$\prod$-formation

$$\frac{A \text{ set} \qquad (x \in A)\ B(x) \text{ set}}{(\prod x \in A)B(x) \text{ set}} \qquad \frac{A = C \qquad (x \in A)\ B(x) = D(x)}{(\prod x \in A)B(x) = (\prod x \in C)D(x)}$$

The second rule says that from equal arguments we get equal values. The same holds for all other set forming operations, and we will never spell it out again. The conclusion of the first rule is that something is a set. To understand which set it is, we must know how its canonical elements and its equal canonical elements are formed. This is explained by the introduction rules:

$\prod$-introduction

$$\frac{(x \in A)\ b(x) \in B(x)}{(\lambda x)b(x) \in (\prod x \in A)B(x)}$$

$$\frac{(x \in A)\ b(x) = d(x) \in B(x)}{(\lambda x)b(x) = (\lambda x)d(x) \in (\prod x \in A)B(x)}$$

Note that these rules introduce canonical elements and equal canonical elements, even if b(a) is not a canonical element of B(a) for $a \in A$. Also, we assume that the usual variable restriction is met, i.e. that x does not appear free in any assumption except (those of the form) $x \in A$. Note that it is neccessary to understand that $b(x) \in B(x)$ $(x \in A)$ is a function to be able to form the canonical element $(\lambda x)b(x) \in (\prod x \in A)B(x)$; we could say that the latter is a name of the former. Since, in general, there are no exhaustive rules for generating all functions from one set to another, it follows that we cannot generate inductively all the elements of a set of the form $(\prod x \in A)B(x)$ (or, in particular, of the form B^A, like N^N).

We can now justify the second rule of set formation. So let $(\lambda x)b(x)$ be a canonical element of $(\prod x \in A)B(x)$. Then $b(x) \in B(x)$ $(x \in A)$. Therefore, assuming $x \in C$ we get $x \in A$ by symmetry and equality of sets from the premiss $A = C$, and hence $b(x) \in B(x)$. Now, from the premiss $B(x) = D(x)$ $(x \in A)$, again by equality of sets (which is assumed to hold also for families of sets), we obtain $b(x) \in D(x)$, and hence $(\lambda x)b(x) \in (\prod x \in C)D(x)$ by $\prod$-introduction. The other direction is similar.

$$\frac{\dfrac{\dfrac{A = C}{(x \in C)\quad C = A}}{x \in A} \qquad \dfrac{\dfrac{\dfrac{A = C}{(x \in C)\quad C = A}}{x \in A}}{B(x) = D(x)}}{b(x) \in B(x) \qquad\qquad b(x) \in D(x)}$$

$$(\lambda x)b(x) \in (\prod x \in C)D(x)$$

We remark that the above derivation cannot be considered as a formal proof of the second $\prod$-formation rule in type theory itself since there is no formal rule of proving an equality between two sets which corresponds directly to the explanation of what such an equality means. We also have to prove that

$$\frac{(\lambda x)b(x) = (\lambda x)d(x) \in (\prod x \in A)B(x)}{(\lambda x)b(x) = (\lambda x)d(x) \in (\prod x \in C)D(x)}$$

under the same assumptions. So let $(\lambda x)b(x)$ and $(\lambda x)d(x)$ be equal canonical elements of $(\prod x \in A)B(x)$. Then $b(x) = d(x) \in B(x)$ $(x \in A)$, and therefore the derivation

$$\dfrac{\dfrac{\dfrac{A = C}{(x \in C)\quad C = A}}{x \in A} \qquad \dfrac{\dfrac{A = C}{(x \in C)\quad C = A}}{x \in A}}{\dfrac{b(x) = d(x) \in B(x) \qquad B(x) = D(x)}{b(x) = d(x) \in D(x)}}{(\lambda x)b(x) = (\lambda x)d(x) \in (\prod x \in C)D(x)}$$

shows that $(\lambda x)b(x)$ and $(\lambda x)d(x)$ are equal canonical elements of $(\prod x \in C)D(x)$.

$\prod$-elimination

$$\frac{c \in (\prod x \in A)B(x) \qquad a \in A}{Ap(c,a) \in B(a)}$$

$$\frac{c = d \in (\prod x \in A)B(x) \qquad a = b \in A}{Ap(c,a) = Ap(d,b) \in B(a)}$$

We have to explain the meaning of the new constant Ap (Ap for Application). Ap(c,a) is a method of obtaining a canonical element of B(a), and we now explain how to execute it. We know that $c \in (\prod x \in A)B(x)$, that is, that c is a method which yields a canonical element $(\lambda x)b(x)$ of $(\prod x \in A)B(x)$ as result. Now take $a \in A$ and substitute it for x in b(x). Then $b(a) \in B(a)$. Calculating b(a), we obtain as result a canonical element of B(a), as required. Of course, in this explanation, no concrete computation is carried out; it has the character of a thought experiment (Ger. Gedankenexperiment). We use Ap(c,a) instead of the more common b(a) to distinguish the result of applying the binary application function Ap to the two arguments c and a from the result of applying b to a. Ap(c,a) corresponds to the application operation (ca) in combinatory logic. But recall that in combinatory logic there are no type restrictions, since one can always form (ca), for any c and a.

$\prod$-equality

$$\frac{\begin{array}{c}(x \in A)\\ a \in A \qquad b(x) \in B(x)\end{array}}{Ap((\lambda x)b(x),a) = b(a) \in B(a)}$$

$$\frac{c \in (\prod x \in A)B(x)}{c = (\lambda x)Ap(c,x) \in (\prod x \in A)B(x)}$$

The first equality rule shows how the new function Ap operates on the canonical elements of $(\prod x \in A)B(x)$. Think of $(\lambda x)b(x)$ as a name of the program b(x). Then the first rule says that applying the name of a program to an argument yields the same result as executing the program with that argument as input. Similarly, the second rule is needed

to obtain a notation, Ap(c,x), for a program of which we know only the name c. The second rule can be explained as follows. Recall that two elements are equal if they yield equal canonical elements as results. So suppose c yields the result $(\lambda x)b(x)$, where $b(x) \in B(x)$ $(x \in A)$. Since $(\lambda x)Ap(c,x)$ is canonical, what we want to prove is

$$(\lambda x)b(x) = (\lambda x)Ap(c,x) \in B(x) \quad (x \in A)$$

By the rule of Π-introduction for equal elements, we need $b(x) = Ap(c,x) \in B(x)$ $(x \in A)$. This means $b(a) = Ap(c,a) \in B(a)$ provided $a \in A$. But this is true, since c yields $(\lambda x)b(x)$ and hence $Ap(c,a)$ yields the same value as $b(a)$.

The rules for products contain the rules for B^A, which is the set of functions from the set A to the set B. In fact, we take B^A to be $(\Pi x \in A)B$, where B does not depend on x. Here the concept of definitional equality is useful.

Definitional equality

Definitional equality is intensional equality, or equality of meaning (synonymy). We use the symbol $\equiv$ or $=_{def.}$ (which was first introduced by Burali-Forti). Definitional equality $\equiv$ is a relation between linguistic expressions; it should not be confused with equality between objects (sets, elements of a set etc.) which we denote by =. Definitional equality is the equivalence relation generated by abbreviatory definitions, changes of bound variables and the principle of substituting equals for equals. Therefore it is decidable, but not in the sense that $a \equiv b \vee \neg (a \equiv b)$ holds, simply because $a \equiv b$ is not a proposition in the sense of the present theory. Definitional equality can be used to rewrite expressions, in which case its decidability is essential in checking the formal correctness of a proof. In fact, to check the correctness of an inference like

$$\frac{A \text{ true} \qquad B \text{ true}}{A \,\&\, B \text{ true}}$$

for instance, we must in particular make sure that the occurrences of the expressions A and B above the line and the corresponding occurrences below are the same, that is, that they are definitionally equal. Note that the rewriting of an expression is not counted as a formal inference.

Applications of the cartesian product

First, using definitional equality, we can now define B^A by putting

$$B^A \equiv A \to B \equiv (\prod x \in A)B,$$

provided B does not depend on x. We next consider the $\prod$-rules in the interpretation of propositions as sets. If, in the first rule, $\prod$-formation, we think of B(x) as a proposition instead of a set, then, after the definition

$$(\forall x \in A)B(x) \equiv (\prod x \in A)B(x),$$

it becomes the rule

$\forall$-formation

$$\frac{\begin{array}{c}(x \in A)\\ A \text{ set} \qquad B(x) \text{ prop.}\end{array}}{(\forall x \in A)B(x) \text{ prop.}}$$

which says that universal quantification forms propositions. A set merely says that the domain over which the universal quantifier ranges is a set and this is why we do not change it into A prop. Note that the rule of $\forall$-formation is just an instance of $\prod$-formation. We similarly have

$\forall$-introduction

$$\frac{\begin{array}{c}(x \in A)\\ B(x) \text{ true}\end{array}}{(\forall x \in A)B(x) \text{ true}}$$

which is obtained from the rule of $\prod$-introduction by suppressing the proof b(x). Namely, we write in general A true instead of $a \in A$ for some a, when A is thought of as a proposition and we don't care about what its proof (construction) is.

More generally, we can suppress proofs as follows. Suppose that

$$a(x_1,\ldots,x_n) \in A(x_1,\ldots,x_m)\ (x_1 \in A_1,\ \ldots,\ x_m \in A_m(x_1,\ldots,x_{m-1}),\ x_{m+1} \in A_{m+1}(x_1,\ldots,x_m),\ \ldots,\ x_n \in A_n(x_1,\ldots,x_m))$$

namely, suppose that A_{m+1} up to A_n and A depend only on $x_1, \ldots, x_m$. Then, if we are merely interested in the truth of $A(x_1,\ldots,x_m)$, it is inessential to write explicit symbols for the elements of $A_{m+1}, \ldots, A_n$; so we abbreviate it with

$$A(x_1,\ldots,x_m) \text{ true } (x_1 \in A_1,\ \ldots,\ x_m \in A_m(x_1,\ldots,x_{m-1}),\ A_{m+1}(x_1,\ldots,x_m) \text{ true},\ \ldots,\ A_n(x_1,\ldots,x_m) \text{ true}).$$

Similarly, we write

$$A(x_1,\ldots,x_m) \text{ prop. } (x_1 \in A_1,\ \ldots,\ x_m \in A_m(x_1,\ldots,x_{m-1}),\ A_{m+1}(x_1,\ldots,x_m) \text{ true},\ \ldots,\ A_n(x_1,\ldots,x_m) \text{ true})$$

that is, $A(x_1,\ldots,x_m)$ is a proposition provided $x_1 \in A_1, \ldots, x_m \in A_m(x_1,\ldots,x_{m-1})$ and $A_{m+1}(x_1,\ldots,x_m), \ldots, A_n(x_1,\ldots,x_m)$

are all true, as an abbreviation of

$$A(x_1,\ldots,x_m) \text{ prop. } (x_1 \in A_1,\ \ldots,\ x_m \in A_m(x_1,\ldots,x_{m-1}),$$
$$x_{m+1} \in A_{m+1}(x_1,\ldots,x_m),\ \ldots,\ x_n \in A_n(x_1,\ldots,x_m).$$

Turning back to the $\forall$-rules, from the rule of $\prod$-elimination, we have in particular

$\forall$-elimination

$$\frac{a \in A \qquad (\forall x \in A)B(x) \text{ true}}{B(a) \text{ true}}$$

Restoring proofs, we see that, if c is a proof of $(\forall x \in A)B(x)$, then Ap(c,a) is a proof of B(a); so a proof of $(\forall x \in A)B(x)$ is a method which takes an arbitrary element of A into a proof of B(a), in agreement with the intuitionistic interpretation of the universal quantifier.

If we now define

$$A \supset B \equiv A \to B \equiv B^A \equiv (\prod x \in A)B,$$

where B does not depend on x, we obtain from the $\prod$-rules the rules for implication. From the rule of $\prod$-formation, assuming B does not depend on x, we obtain

$\supset$-formation

$$\frac{\begin{array}{c}(A \text{ true}) \\ A \text{ prop.} \qquad B \text{ prop.}\end{array}}{A \supset B \text{ prop.}}$$

which is a generalization of the usual rule of forming $A \supset B$, since we may also use the assumption A true to prove B prop. This generalization is perhaps more evident in the Kolmogorov interpretation, where we might be in the position to judge B to be a problem only under the assumption that the problem A can be solved, which is clearly sufficient for the problem $A \supset B$, that is, the problem of solving B provided that A can be solved, to make sense. The inference rules for $\supset$ are:

$\supset$-introduction

$$\frac{\begin{array}{c}(A \text{ true}) \\ B \text{ true}\end{array}}{A \supset B \text{ true}}$$

which comes from the rule of $\prod$-introduction by suppressing proofs, and

$\supset$-elimination

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

which is obtained from the rule of $\prod$-elimination by the same process.

Example (the combinator I). Assume A set and $x \in A$. Then, by $\prod$-introduction, we obtain $(\lambda x)x \in A \to A$, and therefore, for any proposition A, $A \supset A$ true. This expresses the fact that a proof of $A \supset A$ is the method: take the same proof (construction). We can define the combinator I putting $I \equiv (\lambda x)x$. Note that the same I belongs to any set of the form $A \to A$, since we do not have different variables for different types.

Example (the combinator K). Assume A set, B(x) set ($x \in A$) and let $x \in A$, $y \in B(x)$. Then, by λ-abstraction on y, we obtain $(\lambda y)x \in B(x) \to A$, and, by λ-abstraction on x, $(\lambda x)(\lambda y)x \in (\prod x \in A)(B(x) \to A)$. We can define the combinator K putting $K \equiv (\lambda x)(\lambda y)x$. If we think of A and B as propositions, where B does not depend on x, K appears as a proof of $A \supset (B \supset A)$; so $A \supset (B \supset A)$ is true. K expresses the method: given any proof x of A, take the function from B to A which is constantly x for any proof y of B.

Example (the combinator S). Assume A set, B(x) set ($x \in A$), C(x,y) set ($x \in A$, $y \in B(x)$) and let $x \in A$, $f \in (\prod x \in A)B(x)$ and $g \in (\prod x \in A)(\prod y \in B(x))C(x,y)$. Then $Ap(f,x) \in B(x)$ and $Ap(g,x) \in (\prod y \in B(x))C(x,y)$ by $\prod$-elimination. So, again by $\prod$-elimination,

$$Ap(Ap(g,x),Ap(f,x)) \in C(x,Ap(f,x)).$$

Now, by λ-abstraction on x, we obtain

$$(\lambda x)Ap(Ap(g,x),Ap(f,x)) \in (\prod x \in A)C(x,Ap(f,x)),$$

and, by λ-abstraction on f,

$$(\lambda f)(\lambda x)Ap(Ap(g,x),Ap(f,x)) \in (\prod f \in (\prod x \in A)B(x))(\prod x \in A)C(x,Ap(f,x)).$$

Since the set to the right does not depend on g, abstracting on g, we obtain

$$(\lambda g)(\lambda f)(\lambda x)Ap(Ap(g,x),Ap(f,x)) \in (\prod x \in A)(\prod y \in B(x))C(x,y) \to (\prod f \in (\prod x \in A)B(x))(\prod x \in A)C(x,Ap(f,x)).$$

We may now put

$$S \equiv (\lambda g)(\lambda f)(\lambda x)Ap(Ap(g,x),Ap(f,x))$$

which is the usual combinator S, denoted by $\lambda gfx.gx(fx)$ in combinatory logic. In this way, we have assigned a type (set) to the combinator S. Now think of C(x,y) as a propositional function. Then we have proved

$$(\forall x \in A)(\forall y \in B(x))C(x,y) \supset (\forall f \in (\prod x \in A)B(x))(\forall x \in A)C(x,Ap(f,x)) \text{ true}$$

which is traditionally written

$$(\forall x \in A)(\forall y \in B(x))C(x,y) \supset (\forall f \in \prod_{x \in A} B_x)(\forall x \in A)C(x,f(x)).$$

If we assume that C(x,y) does not depend on y, then $(\prod y \in B(x))C(x,y) \equiv B(x) \to C(x)$ and therefore

$$S \in (\prod x \in A)(B(x) \to C(x)) \to ((\prod x \in A)B(x) \to (\prod x \in A)C(x)).$$

So, if we think of B(x) and C(x) as propositions, we have

$$(\forall x \in A)(B(x) \supset C(x)) \supset ((\forall x \in A)B(x) \supset (\forall x \in A)C(x)) \text{ true}.$$

Now assume that B(x) does not depend on x and that C(x,y) does not depend on x and y. Then we obtain

$$S \in (A \to (B \to C)) \to ((A \to B) \to (A \to C)),$$

that is, in the logical interpretation,

$$(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C)) \text{ true}.$$

This is just the second axiom of the Hilbert style propositional calculus. In this last case, the proof above, when written in treeform,

becomes:

$$\dfrac{\dfrac{\dfrac{\dfrac{(x \in A) \quad (f \in A \to B)}{Ap(f,x) \in B} \qquad \dfrac{(x \in A) \quad (g \in A \to (B \to C))}{Ap(g,x) \in B \to C}}{Ap(Ap(g,x),Ap(f,x)) \in C}}{(\lambda x)Ap(Ap(g,x),Ap(f,x)) \in A \to C}}{(\lambda f)(\lambda x)Ap(Ap(g,x),Ap(f,x)) \in (A \to B) \to (A \to C)}$$

$$(\lambda g)(\lambda f)(\lambda x)Ap(Ap(g,x),Ap(f,x)) \in (A \to (B \to C)) \to ((A \to B) \to (A \to C))$$

Disjoint union of a family of sets

The second group of rules is about the disjoint union of a family of sets.

Σ-formation

$$\dfrac{A \text{ set} \qquad \dfrac{(x \in A)}{B(x) \text{ set}}}{(\Sigma x \in A)B(x) \text{ set}}$$

A more traditional notation for $(\Sigma x \in A)B(x)$ would be $\sum_{x \in A} B_x$ ($\coprod_{x \in A} B_x$ or $\bigcup_{x \in A} B_x$). We now explain what set $(\Sigma x \in A)B(x)$ is by prescribing how its canonical elements are formed. This we do with the rule:

Σ-introduction

$$\dfrac{a \in A \qquad b \in B(a)}{(a,b) \in (\Sigma x \in A)B(x)}$$

We can now justify the equality rule associated with Σ-formation:

$$\dfrac{A = C \qquad \dfrac{(x \in A)}{B(x) = D(x)}}{(\Sigma x \in A)B(x) = (\Sigma x \in C)D(x)}$$

In fact, any canonical element of $(\Sigma x \in A)B(x)$ is of the form (a,b) with $a \in A$ and $b \in B(a)$ by Σ-introduction. But then we also have

$a \in C$ and $b \in D(a)$ by equality of sets and substitution. Hence $(a,b) \in (\Sigma x \in C)D(x)$ by Σ-introduction. The other direction is similar.

Σ-elimination

$$\frac{\begin{array}{cc} & (x \in A,\ y \in B(x)) \\ c \in (\Sigma x \in A)B(x) & d(x,y) \in C((x,y)) \end{array}}{E(c,(x,y)d(x,y)) \in C(c)}$$

where we presuppose the premiss C(z) set $(z \in (\Sigma x \in A)B(x))$, although it is not written out explicitly. (To be precise, we should also write out the premisses A set and B(x) set $(x \in A)$.) We explain the rule of Σ-elimination by showing how the new constant E operates on its arguments. So assume we know the premisses. Then we execute E(c,(x,y)d(x,y)) as follows. First execute c, which yields a canonical element of the form (a,b) with $a \in A$ and $b \in B(a)$. Now substitute a and b for x and y, respectively, in the right premiss, obtaining $d(a,b) \in C((a,b))$. Executing d(a,b) we obtain a canonical element e of C((a,b)). We now want to show that e is also a canonical element of C(c). It is a general fact that, if $a \in A$ and a has value b, then $a = b \in A$ (note, however, that this does not mean that $a = b \in A$ is necessarily formally derivable by some particular set of formal rules). In our case, $c = (a,b) \in (\Sigma x \in A)B(x)$ and hence, by substitution, C(c) = C((a,b)). Remembering what it means for two sets to be equal, we conclude from the fact that e is a canonical element of C((a,b)) that e is also a canonical element of C(c).

Another notation for E(c,(x,y)d(x,y)) could be (Ex,y)(c,d(x,y)), but we prefer the first since it shows more clearly that x and y become bound only in d(x,y).

Σ-equality

$$\frac{\begin{array}{ccc} & & (x \in A,\ y \in B(x)) \\ a \in A & b \in B(a) & d(x,y) \in C((x,y)) \end{array}}{E((a,b),(x,y)d(x,y)) = d(a,b) \in C((a,b))}$$

(Here, as in Σ-elimination, C(z) set $(z \in (\Sigma x \in A)B(x))$ is an implicit premiss.) Assuming that we know the premisses, the conclusion is justified by imagining E((a,b),(x,y)d(x,y)) to be executed. In fact, we first execute (a,b), which yields (a,b) itself as result; then we substitute a, b for x, y in d(x,y), obtaining $d(a,b) \in C((a,b))$, and execute d(a,b) until we obtain a canonical element $e \in C((a,b))$. The same canonical element is produced by d(a,b), and thus the conclusion is correct.

A second rule of Σ-equality, analogous to the second rule of Π-equality, is now derivable, as we shall see later.

Applications of the disjoint union

As we have already done with the cartesian product, we shall now see what are the logical interpretations of the disjoint union. If we put

$$(\exists x \in A)B(x) \equiv (\Sigma x \in A)B(x),$$

then, from the Σ-rules, interpreting B(x) as a propositional function over A, we obtain as particular cases:

$\exists$-formation

$$\frac{\begin{array}{c}(x \in A)\\ A \text{ set} \qquad B(x) \text{ prop.}\end{array}}{(\exists x \in A)B(x) \text{ prop.}}$$

$\exists$-introduction

$$\frac{a \in A \qquad B(a) \text{ true}}{(\exists x \in A)B(x) \text{ true}}$$

In accordance with the intuitionistic interpretation of the existential quantifier, the rule of Σ-introduction may be interpreted as saying that a (canonical) proof of $(\exists x \in A)B(x)$ is a pair (a,b), where b is a proof of the fact that a satisfies B. Suppressing proofs, we obtain the rule of $\exists$-introduction, in which, however, the first premiss $a \in A$ is usually not made explicit.

$\exists$-elimination

$$\frac{(\exists x \in A)B(x) \text{ true} \qquad \begin{array}{c}(x \in A,\ B(x) \text{ true})\\ C \text{ true}\end{array}}{C \text{ true}}$$

Here, as usual, no assumptions, except those explicitly written out, may depend on the variable x. The rule of Σ-elimination is stronger than the $\exists$-elimination rule, which is obtained from it by suppressing proofs, since we take into consideration also proofs (constructions), which is not possible within the language of first order predicate logic. This additional strength will be visible when treating the left and right projections below.

The rules of disjoint union deliver also the usual rules of conjunction and the usual properties of the cartesian product of two sets if we define

$$A \,\&\, B \equiv A \times B \equiv (\Sigma x \in A)B,$$

where B does not depend on x. We derive here only the rules of conjunction.

&-formation

$$\frac{\begin{array}{c}(A \text{ true})\\ A \text{ prop.} \qquad B \text{ prop.}\end{array}}{A \,\&\, B \text{ prop.}}$$

This rule is an instance of Σ-formation and a generalization of the usual rule of forming propositions of the form A & B, since we may know that B is a proposition only under the assumption that A is true.

&-introduction

$$\frac{A \text{ true} \qquad B \text{ true}}{A \,\&\, B \text{ true}}$$

Restoring proofs, we see that a (canonical) proof of A & B is pair (a,b), where a and b are given proofs of A and B respectively.

&-elimination

$$\frac{\begin{array}{c}(A \text{ true, } B \text{ true}) \\ A \,\&\, B \text{ true} \qquad C \text{ true}\end{array}}{C \text{ true}}$$

From this rule of &-elimination, we obtain the standard &-elimination rules by choosing C to be A and B themselves:

$$\frac{A \,\&\, B \text{ true} \qquad (A \text{ true})}{A \text{ true}} \qquad \frac{A \,\&\, B \text{ true} \qquad (B \text{ true})}{B \text{ true}}$$

Example (left projection). We define

$$p(c) \equiv E(c,(x,y)x)$$

and call it the left projection of c since it is a method of obtaining the value of the first (left) coordinate of the pair produced by an arbitrary element c of $(\Sigma x \in A)B(x)$. In fact, if we take the term d(x,y) in the explanation of Σ-elimination to be x, then we see that to execute p(c) we first obtain the pair (a,b) with $a \in A$ and $b \in B(a)$ which is the value of c, and then substitute a, b for x, y in x, obtaining a, which is executed to yield a canonical element of A. Therefore, taking C(z) to be A and d(x,y) to be x in the rules of Σ-elimination and Σ-equality, we obtain as derived rules:

Left projection

$$\frac{c \in (\Sigma x \in A)B(x)}{p(c) \in A} \qquad \frac{a \in A \qquad b \in B(a)}{p((a,b)) = a \in A}$$

If we now turn to the logical interpretation, we see that

$$\frac{c \in (\exists x \in A)B(x)}{p(c) \in A}$$

holds, which means that from a proof of $(\exists x \in A)B(x)$ we can obtain an element of A for which the property B holds. So we have no need of the description operator $(\imath x)B(x)$ (the x such that B(x) holds) or the choice operator $(\epsilon x)B(x)$ (an x such that B(x) holds), since, from the intuitionistic point of view, $(\exists x \in A)B(x)$ is true when we have a proof of it. The difficulty with an epsilon term $(\epsilon x)B(x)$ is that it is construed as a function of the property B(x) itself and not of the proof of $(\exists x)B(x)$. This is why Hilbert had to postulate both a rule of the form

$$\frac{(\exists x)B(x) \text{ true}}{(\epsilon x)B(x) \text{ individual}}$$

a counterpart of which we have just proved, and a rule of the form

$$\frac{(\exists x)B(x) \text{ true}}{B((\epsilon x)B(x)) \text{ true}}$$

which has a counterpart in the first of the rules of right projection that we shall see in the next example.

Example (right projection). We define

$$q(c) \equiv E(c,(x,y)y).$$

Take $d(x,y)$ to be y in the rule of Σ-elimination. From $x \in A$, $y \in B(x)$ we obtain $p((x,y)) = x \in A$ by left projection, and therefore $B(x) = B(p((x,y)))$. So, by the rule of equality of sets, $y \in B(p((x,y)))$. Now choose $C(z)$ set $(z \in (\Sigma x \in A)B(x))$ to be the family $B(p(z))$ set $(z \in (\Sigma x \in A)B(x))$. Then the rule of Σ-elimination gives $q(c) \in B(p(c))$. More formally:

$$\dfrac{c \in (\Sigma x \in A)B(x) \qquad \dfrac{(y \in B(x)) \qquad \dfrac{\dfrac{\dfrac{(x \in A) \quad (y \in B(x))}{p((x,y)) = x \in A}}{x = p((x,y)) \in A}}{B(x) = B(p((x,y)))}}{y \in B(p((x,y)))}}{q(c) \equiv E(c,(x,y)y) \in B(p(c))}$$

So we have:

Right projection

$$\dfrac{c \in (\Sigma x \in A)B(x)}{q(c) \in B(p(c))} \qquad \dfrac{a \in A \quad b \in B(a)}{q((a,b)) = b \in B(a)}$$

The second of these rules is derived by Σ-equality in much the same way as the first was derived by Σ-elimination.

When $B(x)$ is thought of as a propositional function, the first rule of right projection says that, if c is a construction of $(\exists x \in A)B(x)$, then $q(c)$ is a construction of $B(p(c))$, where, by left projection, $p(c) \in A$. Thus, suppressing the construction in the conclusion, $B(p(c))$ is true. Note, however, that, in case $B(x)$ depends on x, it is impossible to suppress the construction in the premiss, since the conclusion depends on it.

Finally, when $B(x)$ does not depend on x, so that we may write it simply as B, and both A and B are thought of as propositions, the first rule of right projection reduces to

&-elimination

$$\dfrac{A \ \& \ B \text{ true}}{B \text{ true}}$$

by suppressing the constructions in both the premiss and the conclusion.

Example (axioms of conjunction). We first derive $A \supset (B \supset (A \ \& \ B))$ true, which is the axiom corresponding to the rule of &-introduction. Assume A set, $B(x)$ set $(x \in A)$ and let $x \in A$, $y \in B(x)$. Then $(x,y) \in (\Sigma x \in A)B(x)$ by Σ-introduction, and, by $\prod$-introduction, $(\lambda y)(x,y) \in B(x) \to (\Sigma x \in A)B(x)$ (note that $(\Sigma x \in A)B(x)$ does not depend on y) and $(\lambda x)(\lambda y)(x,y) \in (\prod x \in A)(B(x) \to (\Sigma x \in A)B(x))$. The logical reading is then

$$(\forall x \in A)(B(x) \supset (\exists x \in A)B(x)) \text{ true},$$

from which, in particular, when B does not depend on x,

$$A \supset (B \supset (A \ \& \ B)) \text{ true}.$$

We now use the left and right projections to derive $A \ \& \ B \supset A$ true and $A \ \& \ B \supset B$ true. To obtain the first, assume $z \in (\Sigma x \in A)B(x)$.

Then $p(z) \in A$ by left projection, and, by λ-abstraction on z,

$$(\lambda z)p(z) \in (\Sigma x \in A)B(x) \to A.$$

In particular, when B(x) does not depend on x, we obtain

$$A \ \& \ B \supset A \text{ true}.$$

To obtain the second, from $z \in (\Sigma x \in A)B(x)$, we have $q(z) \in B(p(z))$ by right projection, and hence, by λ-abstraction,

$$(\lambda z)q(z) \in (\Pi z \in (\Sigma x \in A)B(x))B(p(z))$$

(note that B(p(z)) depends on z). In particular, when B(x) does not depend on x, we obtain

$$A \ \& \ B \supset B \text{ true}.$$

Example (another application of the disjoint union). The rule of Σ-elimination says that any function d(x,y) with arguments in A and B(x) gives also a function (with the same values, by Σ-equality) with a pair in $(\Sigma x \in A)B(x)$ as single argument. What we now prove is an axiom corresponding to this rule. So, assume A set, B(x) set $(x \in A)$, C(z) set $(z \in (\Sigma x \in A)B(x))$ and let $f \in (\Pi x \in A)(\Pi y \in B(x))C((x,y))$. We want to find an element of

$$(\Pi x \in A)(\Pi y \in B(x))C((x,y)) \to (\Pi z \in (\Sigma x \in A)B(x))C(z).$$

We define $Ap(f,x,y) \equiv Ap(Ap(f,x),y)$ for convenience. Then Ap(f,x,y) is a ternary function, and $Ap(f,x,y) \in C((x,y))$ $(x \in A, y \in B(x))$. So, assuming $z \in (\Sigma x \in A)B(x)$, by Σ-elimination, we obtain $E(z,(x,y)Ap(f,x,y)) \in C(z)$ (discharging $x \in A$ and $y \in B(x)$), and, by λ-abstraction on z, we obtain the function

$$(\lambda z)E(z,(x,y)Ap(f,x,y)) \in (\Pi z \in (\Sigma x \in A)B(x))C(z)$$

with argument f. So we still have the assumption

$$f \in (\Pi x \in A)(\Pi y \in B(x))C(x,y),$$

which we discharge by λ-abstraction, obtaining

$$(\lambda f)(\lambda z)E(z,(x,y)Ap(f,x,y)) \in$$
$$(\Pi x \in A)(\Pi y \in B(x))C((x,y)) \to (\Pi z \in (\Sigma x \in A)B(x))C(z).$$

In the logical reading, we have

$$(\forall x \in A)(\forall y \in B(x))C((x,y)) \supset (\forall z \in (\Sigma x \in A)B(x))C(z) \text{ true},$$

which reduces to the common

$$(\forall x \in A)(B(x) \supset C) \supset ((\exists x \in A)B(x) \supset C) \text{ true}$$

when C does not depend on z, and to

$$(A \supset (B \supset C)) \supset ((A \ \& \ B) \supset C) \text{ true}$$

when, in addition, B is independent of x.

The axiom of choice

We now show that, with the rules introduced so far, we can give a proof of the axiom of choice, which in our symbolism reads:

$$(\forall x \in A)(\exists y \in B(x))C(x,y)$$
$$\supset (\exists f \in (\prod x \in A)B(x))(\forall x \in A)C(x,Ap(f,x)) \text{ true.}$$

The usual argument in intuitionistic mathematics, based on the intuitionistic interpretation of the logical constants, is roughly as follows: to prove $(\forall x)(\exists y)C(x,y) \supset (\exists f)(\forall x)C(x,f(x))$, assume that we have a proof of the antecedent. This means that we have a method which, applied to an arbitrary x, yields a proof of $(\exists y)C(x,y)$, that is, a pair consisting of an element y and a proof of C(x,y). Let f be the method which, to an arbitrarily given x, assigns the first component of this pair. Then C(x,f(x)) holds for an arbitrary x, and hence so does the consequent. The same idea can be put into symbols, getting a formal proof in intuitionistic type theory. Let A set, B(x) set $(x \in A)$, C(x,y) set $(x \in A, y \in B(x))$, and assume $z \in (\prod x \in A)(\sum y \in B(x))C(x,y)$. If x is an arbitrary element of A, i.e. $x \in A$, then, by $\prod$-elimination, we obtain

$$Ap(z,x) \in (\sum y \in B(x))C(x,y).$$

We now apply left projection to obtain

$$p(Ap(z,x)) \in B(x)$$

and right projection to obtain

$$q(Ap(z,x)) \in C(x,p(Ap(z,x))).$$

By λ-abstraction on x (or $\prod$-introduction), discharging $x \in A$, we have

$$(\lambda x)p(Ap(z,x)) \in (\prod x \in A)B(x),$$

and, by $\prod$-equality,

$$Ap((\lambda x)p(Ap(z,x)),x) = p(Ap(z,x)) \in B(x).$$

By substitution, we get

$$C(x,Ap((\lambda x)p(Ap(z,x)),x)) = C(x,p(Ap(z,x)))$$

and hence, by equality of sets,

$$q(Ap(z,x)) \in C(x,Ap((\lambda x)p(Ap(z,x)),x))$$

where $(\lambda x)p(Ap(z,x))$ is independent of x. By abstraction on x,

$$(\lambda x)q(Ap(z,x)) \in (\prod x \in A)C(x,Ap((\lambda x)p(Ap(z,x)),x)).$$

We now use the rule of pairing (that is, $\sum$-introduction) to get

$$((\lambda x)p(Ap(z,x)),(\lambda x)q(Ap(z,x))) \in$$
$$(\sum f \in (\prod x \in A)B(x))(\prod x \in A)C(x,Ap(f,x))$$

(note that, in the last step, the new variable f is introduced and substituted for $(\lambda x)p(Ap(z,x))$ in the right member). Finally, by abstraction on z, we obtain

$$(\lambda z)((\lambda x)p(Ap(z,x)),(\lambda x)q(Ap(z,x))) \in (\prod x \in A)(\sum y \in B(x))C(x,y)$$
$$\supset (\sum f \in (\prod x \in A)B(x))(\prod x \in A)C(x,Ap(f,x)).$$

In Zermelo-Fraenkel set theory, there is no proof of the axiom of choice, so it must be taken as an axiom, for which, however, it seems to be difficult to claim self-evidence. Here a detailed

justification of the axiom of choice has been provided in the form of the above proof. In many sorted languages, the axiom of choice is expressible but there is no mechanism to prove it. For instance, in Heyting arithmetic of finite type, it must be taken as an axiom. The need for the axiom of choice is clear when developing intuitionistic mathematics at depth, for instance, in finding the limit of a sequence of reals or a partial inverse of a surjective function.

The notion of such that

In addition to disjoint union, existential quantification, cartesian product $A \times B$ and conjunction $A \& B$, the operation Σ has a fifth interpretation: the set of all $a \in A$ such that $B(a)$ holds. Let A be a set and $B(x)$ a proposition for $x \in A$. We want to define the set of all $a \in A$ such that $B(a)$ holds (which is usually written $\{x \in A: B(x)\}$). To have an element $a \in A$ such that $B(a)$ holds means to have an element $a \in A$ together with a proof of $B(a)$, namely an element $b \in B(a)$. So the elements of the set of all elements of A satisfying $B(x)$ are pairs (a,b) with $b \in B(a)$, i.e. elements of $(\Sigma x \in A)B(x)$. Then the Σ-rules play the role of the comprehension axiom (or the separation principle in ZF). The information given by $b \in B(a)$ is called the witnessing information by Feferman[11]. A typical application is the following.

Example (the reals as Cauchy sequences).

$$R \equiv (\Sigma x \in N \to Q)\mathrm{Cauchy}(x)$$

is the definition of the reals as the set of sequences of rational numbers satisfying the Cauchy condition,

$$\mathrm{Cauchy}(a) \equiv (\forall e \in Q)(e > 0 \supset (\exists m \in N)(\forall n \in N)(|a_{m+n} - a_m| \leqslant e),$$

where a is the sequence $a_0, a_1, \ldots$ In this way, a real number is a sequence of rational numbers toghether with a proof that it satisfies the Cauchy condition. So, assuming $c \in R$, $e \in Q$ and $d \in (e > 0)$ (in

[11] S. Feferman, Constructive theories of functions and classes, Logic Colloquium 78, Edited by M. Boffa, D. van Dalen and K. McAloon, North-Holland, Amsterdam, 1979, pp. 159-224.

other words, d is a proof of the proposition e > 0), then, by means of the projections, we obtain p(c) ∈ N → Q and q(c) ∈ Cauchy(p(c)). Then

$$Ap(q(c),e) \in (e > 0 \supset (\exists m \in N)(\forall n \in N)(|a_{m+n}-a_m| \leq e)$$

and

$$Ap(Ap(q(c),e),d) \in (\exists m \in N)(\forall n \in N)(|a_{m+n}-a_m| \leq e).$$

Applying left projection, we obtain the m we need, i.e.

$$p(Ap(Ap(q(c),e),d)) \in N,$$

and we now obtain a_m by applying p(c) to it,

$$Ap(p(c),p(Ap(Ap(q(c),e),d))) \in Q.$$

Only by means of the proof q(c) do we know how far to go for the approximation desired.

Disjoint union of two sets

We now give the rules for the sum (disjoint union or coproduct) of two sets.

+-formation

$$\frac{A \text{ set} \qquad B \text{ set}}{A + B \text{ set}}$$

The canonical elements of A + B are formed using:

+-introduction

$$\frac{a \in A}{i(a) \in A + B} \qquad \frac{b \in B}{j(b) \in A + B}$$

where i and j are two new primitive constants; their use is to give the information that an element of A + B comes from A or B, and which of the two is the case. It goes without saying that we also have the rules of +-introduction for equal elements:

$$\frac{a = c \in A}{i(a) = i(c) \in A + B} \qquad \frac{b = d \in B}{j(b) = j(d) \in A + B}$$

Since an arbitrary element c of A + B yields a canonical element of the form i(a) or j(b), knowing c ∈ A + B means that we also can determine from which of the two sets A and B the element c comes.

+-elimination

$$\frac{c \in A + B \qquad \begin{matrix}(x \in A)\\ d(x) \in C(i(x))\end{matrix} \qquad \begin{matrix}(y \in B)\\ e(y) \in C(j(y))\end{matrix}}{D(c,(x)d(x),(y)e(y)) \in C(c)}$$

where the premisses A set, B set and C(z) set (z ∈ A + B) are presupposed, although not explicitly written out. We must now explain how to execute a program of the new form D(c,(x)d(x),(y)e(y)). Assume we know c ∈ A + B. Then c will yield a canonical element i(a) with a ∈ A or j(b) with b ∈ B. In the first case, substitute a for x in d(x), obtaining d(a), and execute it. By the second premiss, d(a) ∈ C(i(a)), so d(a) yields a canonical element of C(i(a)). Similarly, in the second case, e(y) instead of d(x) must be used to obtain e(b), which produces a canonical element of C(j(b)). In either case, we obtain a canonical element of C(c), since, if c has value i(a), then c = i(a) ∈ A + B and hence C(c) = C(i(a)), and, if c has value j(b), then c = j(b) ∈ A + B and hence C(c) = C(j(b)). From this explanation of the meaning of D, the equality rules:

+-equality

$$\frac{a \in A \qquad \begin{matrix}(x \in A)\\ d(x) \in C(i(x))\end{matrix} \qquad \begin{matrix}(y \in B)\\ e(y) \in C(j(y))\end{matrix}}{D(i(a),(x)d(x),(y)e(y)) = d(a) \in C(i(a))}$$

$$\frac{b \in B \qquad \begin{matrix}(x \in A)\\ d(x) \in C(i(x))\end{matrix} \qquad \begin{matrix}(y \in B)\\ e(y) \in C(j(y))\end{matrix}}{D(j(b),(x)d(x),(y)e(y)) = e(b) \in C(j(b))}$$

become evident.

The disjunction of two propositions is now interpreted as the sum of two sets. We therefore put:

$$A \vee B \equiv A + B.$$

From the formation and introduction rules for +, we then obtain the corresponding rules for ∨ :

∨-formation

$$\frac{A \text{ prop.} \qquad B \text{ prop.}}{A \vee B \text{ prop.}}$$

∨-introduction

$$\frac{A \text{ true}}{A \vee B \text{ true}} \qquad \frac{B \text{ true}}{A \vee B \text{ true}}$$

Note that, if a is a proof of A, then i(a) is a (canonical) proof of A ∨ B, and similarly for B.

∨-elimination

$$\frac{A \vee B \text{ true} \qquad \begin{matrix}(A \text{ true})\\ C \text{ true}\end{matrix} \qquad \begin{matrix}(B \text{ true})\\ C \text{ true}\end{matrix}}{C \text{ true}}$$

follows from the rule of +-elimination by choosing a family C ≡ C(z) (z ∈ A + B) which does not depend on z and suppressing proofs (constructions) both in the premisses, including the assumptions, and the conclusion.

Example (introductory axioms of disjunction). Assume A set, B set and let $x \in A$. Then $i(x) \in A + B$ by +-introduction, and hence $(\lambda x)i(x) \in A \to A + B$ by λ-abstraction on x. If A and B are propositions, we have $A \supset A \vee B$ true. In the same way, $(\lambda y)j(y) \in B \to A + B$, and hence $B \supset A \vee B$ true.

Example (eliminatory axiom of disjunction). Assume A set, B set, C(z) set ($z \in A + B$) and let $f \in (\prod x \in A)C(i(x))$, $g \in (\prod y \in B)C(j(y))$ and $z \in A + B$. Then, by $\prod$-elimination, from $x \in A$, we have $Ap(f,x) \in C(i(x))$, and, from $y \in B$, we have $Ap(g,y) \in C(j(y))$. So, using $z \in A + B$, we can apply +-elimination to obtain $D(z,(x)Ap(f,x),(y)Ap(g,y)) \in C(z)$, thereby discharging $x \in A$ and $y \in B$. By λ-abstraction on z, g, f in that order, we get

$$(\lambda f)(\lambda g)(\lambda z)D(z,(x)Ap(f,x),(y)Ap(g,y)) \in (\prod x \in A)C(i(x)) \to ((\prod y \in B)C(j(y)) \to (\prod z \in A + B)C(z)).$$

This, when C(z) is thought of as a proposition, gives

$$(\forall x \in A)C(i(x)) \supset ((\forall y \in B)C(j(y)) \supset (\forall z \in A + B)C(z)) \text{ true.}$$

If, moreover, C(z) does not depend on z and A, B are propositions as well, we have

$$(A \supset C) \supset ((B \supset C) \supset (A \vee B \supset C)) \text{ true.}$$

Propositional equality

We now turn to the axioms for equality. It is a tradition (deriving its origin from Principia Mathematica) to call equality in predicate logic identity. However, the word identity is more properly used for definitional equality, $\equiv$ or $=_{def.}$, discussed above. In fact, an equality statement, for instance, $2^2 = 2+2$ in arithmetic, does not mean that the two members are the same, but merely that they have the same value. Equality in predicate logic, however, is also different from our equality $a = b \in A$, because the former is a proposition, while the latter is a judgement. A form of propositional equality is nevertheless indispensable: we want an equality I(A,a,b), which asserts that a and b are equal elements of the set A, but on which we can operate with the logical operations (recall that e.g. the negation or quantification of a judgement does not make sense). In a certain sense, I(A,a,b) is an internal form of =. We then have four kinds of equality:

(1) $\equiv$ or $=_{def.}$,

(2) $A = B$,

(3) $a = b \in A$,

(4) I(A,a,b).

Equality between objects is expressed in a judgement and must be defined separately for each category, like the category sets, as in (2), or the category of elements of a set, as in (3); (4) is a proposition, whereas (1) is a mere stipulation, a relation between linguistic expressions. Note however that I(A,a,b) true <u>is</u> a judgement, which will turn out to be equivalent to $a = b \in A$ (which is not to say

that it has the same sense). (1) is intensional (sameness of meaning), while (2), (3) and (4) are extensional (equality between objects). As for Frege, elements a, b may have different meanings, or be different methods, but have the same value. For instance, we certainly have $2^2 = 2+2 \in N$, but not $2^2 \equiv 2+2$.

I-formation

$$\frac{A \text{ set} \qquad a \in A \qquad b \in A}{I(A,a,b) \text{ set}}$$

We now have to explain how to form canonical elements of I(A,a,b). The standard way to know that I(A,a,b) is true is to have $a = b \in A$. Thus the introduction rule is simply: if $a = b \in A$, then there is a canonical proof r of I(A,a,b). Here r does not depend on a, b or A; it does not matter what canonical element I(A,a,b) has when $a = b \in A$, as long as it has one.

I-introduction

$$\frac{a = b \in A}{r \in I(A,a,b)}$$

Also, note that the rule for introducing equal elements of I(A,a,b) is the trivial one:

$$\frac{a = b \in A}{r = r \in I(A,a,b)}$$

We could now adopt elimination and equality rules for I in the same style as for Π, Σ, +, namely introducing a new eliminatory operator.

We would then derive the following rules, which we here take instead as primitive:

I-elimination

$$\frac{c \in I(A,a,b)}{a = b \in A}$$

I-equality

$$\frac{c \in I(A,a,b)}{c = r \in I(A,a,b)}$$

Finally, note that I-formation is the only rule up to now which permits the formation of families of sets. If only the operations Π, Σ, +, N_n, N, W were allowed, we would only get constant sets.

Example (introductory axiom of identity). Assume A set and let $x \in A$. Then $x = x \in A$, and, by I-introduction, $r \in I(A,x,x)$. By abstraction on x, $(\lambda x)r \in (\forall x \in A)I(A,x,x)$. Therefore $(\lambda x)r$ is a canonical proof of the law of identity on A.

$$\frac{\dfrac{\dfrac{(x \in A)}{x = x \in A}}{r \in I(A,x,x)}}{(\lambda x)r \in (\forall x \in A)I(A,x,x)}$$

Example (eliminatory axiom of identity). Given a set A and a property B(x) prop. $(x \in A)$ over A, we claim that the law of equality corresponding to Leibniz's principle of indiscernibility holds, namely that equal elements satisfy the same properties,

$$(\forall x \in A)(\forall y \in A)(I(A,x,y) \supset (B(x) \supset B(y))) \text{ true.}$$

To prove it, assume $x \in A$, $y \in A$ and $z \in I(A,x,y)$. Then $x = y \in A$ and hence $B(x) = B(y)$ by substitution. So, assuming $w \in B(x)$, by equality of sets, we obtain $w \in B(y)$. Now, by abstraction on w, z, y, x in that order, we obtain a proof of the claim:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(z \in I(A,x,y)) \quad (x \in A)}{x = y \in A} \quad B(x) \text{ set}}{(w \in B(x)) \quad B(x) = B(y)}}{w \in B(y)}}{(\lambda w)w \in B(x) \supset B(y)}}{(\lambda z)(\lambda w)w \in I(A,x,y) \supset (B(x) \supset B(y))}}{(\lambda x)(\lambda y)(\lambda z)(\lambda w)w \in (\forall x \in A)(\forall y \in A)(I(A,x,y) \supset (B(x) \supset B(y)))}$$

The same problem (of justifying Leibniz's principle) was solved in Principia by the use of impredicative second order quantification. There one defines

$$(a = b) \equiv (\forall X)(X(a) \supset X(b))$$

from which Leibniz's principle is obvious, since it is taken to define the meaning of identity. In the present language, quantification over properties is not possible, and hence the meaning of identity has to be defined in another way, without invalidating Leibniz's principle.

Example (proof of the converse of the projection laws). We can now prove that the inference rule

$$\dfrac{c \in (\Sigma x \in A)B(x)}{c = (p(c),q(c)) \in (\Sigma x \in A)B(x)}$$

is derivable. It is an analogue of the second Π-equality rule, which could also be derived, provided the Π-rules were formulated following the same pattern as the other rules. Assume $x \in A$, $y \in B(x)$. By the projection laws, $p((x,y)) = x \in A$ and $q((x,y)) = y \in B(x)$. Then, by Σ-introduction (equal elements form equal pairs),

$$(p((x,y)),q((x,y))) = (x,y) \in (\Sigma x \in A)B(x).$$

By I-introduction,

$$r \in I((\Sigma x \in A)B(x),(p((x,y)),q((x,y))),(x,y)).$$

Now take the family C(z) in the rule of Σ-elimination to be $I((\Sigma x \in A)B(x),(p(z),q(z)),z)$. Then we obtain

$$E(c,(x,y)r) \in I((\Sigma x \in A)B(x),(p(c),q(c)),c)$$

and hence, by I-elimination, $(p(c),q(c)) = c \in (\Sigma x \in A)B(x)$.

$$\dfrac{c \in (\Sigma x \in A)B(x) \qquad \dfrac{\dfrac{(x \in A) \quad (y \in B(x))}{p((x,y)) = x \in A} \quad \dfrac{(x \in A) \quad (y \in B(x))}{q((x,y)) = y \in B(x)}}{(p((x,y)),q((x,y))) = (x,y) \in (\Sigma x \in A)B(x)}}{r \in I((\Sigma x \in A)B(x),(p((x,y)),q((x,y))),(x,y))}$$

$$\dfrac{E(c,(x,y)r) \in I((\Sigma x \in A)B(x),(p(c),q(c)),c)}{(p(c),q(c)) = c \in (\Sigma x \in A)B(x)}$$

This example is typical. The I-rules are used systematically to show the uniqueness of a function, whose existence is given by an elimination rule, and whose properties are expressed by the associated equality rules.

Example (properties and indexed families of elements). There are two ways of looking at subsets of a set B:

(1) a subset of B is a propositional function (property) $C(y)$ $(y \in B)$;

(2) a subset of B is an indexed family of elements $b(x) \in B$ $(x \in A)$.

Using the identity rules, we can prove the equivalence of these two concepts. Given an indexed family as in (2), the corresponding property is

$$(\exists x \in A)I(B,b(x),y) \quad (y \in B),$$

and, conversely, given a property as in (1), the corresponding indexed family is

$$p(x) \in B \quad (x \in (\Sigma y \in B)C(y)).$$

Finite sets

Note that, up to now, we have no operations to build up sets from nothing, but only operations to obtain new sets from given ones (and from families of sets). We now introduce finite sets, which are given outright; hence their set formation rules will have no premisses. Actually, we have infinitely many rules, one group of rules for each $n = 0, 1, \ldots$

N_n-formation

$$N_n \text{ set}$$

N_n-introduction

$$m_n \in N_n \qquad (m = 0, 1, \ldots, n-1)$$

So we have the sets N_0 with no elements, N_1 with the single canonical element 0_1, N_2 with canonical elements 0_2, 1_2, etc.

N_n-elimination

$$\frac{c \in N_n \qquad c_m \in C(m_n). \quad (m = 0, 1, \ldots, n-1)}{R_n(c,c_0,\ldots,c_{n-1}) \in C(c)}$$

Here, as usual, the family of sets $C(z)$ set $(z \in N_n)$ may be interpreted as a property over N_n. Assuming we know the premisses, R_n is explained as follows: first execute c, whose result is m_n for some m between 0 and n-1. Select the corresponding element c_m of $C(m_n)$ and continue by executing it. The result is a canonical element $d \in C(c)$, since c has been seen to be equal to m_n and $c_m \in C(m_n)$ is a premiss. R_n is recursion over the finite set N_n; it is a kind of definition by cases.

From the meaning of R_n, given by the above explanation, we have the n rules (note that $m_n \in N_n$ by N_n-introduction):

N_n-equality

$$\frac{c_m \in C(m_n) \quad (m = 0, 1, \ldots, n-1)}{R_n(m_n, c_0, \ldots, c_{n-1}) = c_m \in C(m_n)}$$

(one such rule for each choice of $m = 0, 1, \ldots, n-1$ in the conclusion). An alternative approach would be to postulate the rules for n equal to 0 and 1 only, define $N_2 \equiv N_1 + N_1$, $N_3 \equiv N_1 + N_2$ etc., and then derive all other rules.

Example (about N_0). N_0 has no introduction rule and hence no elements; it is thus natural to put

$$\bot \equiv \emptyset \equiv N_0.$$

The elimination rule becomes simply:

N_0-elimination

$$\frac{c \in N_0}{R_0(c) \in C(c)}$$

The explanation of the rule is that we understand that we shall never get an element $c \in N_0$, so that we shall never have to execute $R_0(c)$. Thus the set of instructions for executing a program of the form $R_0(c)$ is vacuous. It is similar to the programming statement abort introduced by Dijkstra[12].

[12] See note 2.

When C(z) does not depend on z, it is possible to suppress the proof (construction) not only in the conclusion but also in the premiss. We then arrive at the logical inference rule

$\bot$-elimination

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

traditionally called ex falso quodlibet. This rule is often used in ordinary mathematics, but in the form

$$\frac{A \vee B \text{ true} \qquad \begin{matrix}(B \text{ true}) \\ \bot \text{ true}\end{matrix}}{A \text{ true}}$$

which is easily seen to be equivalent to the form above.

Example (about N_1). We define

$$\top \equiv N_1.$$

Then 0_1 is a (canonical) proof of $\top$, since $0_1 \in N_1$ by N_1-introduction. So $\top$ is true. We now want to prove that 0_1 is in fact the only element of N_1, that is, that the rule

$$\frac{c \in N_1}{c = 0_1 \in N_1}$$

is derivable. In fact, from $0_1 \in N_1$, we get $0_1 = 0_1 \in N_1$, and hence $r \in I(N_1, 0_1, 0_1)$. Now apply N_1-elimination with $I(N_1, z, 0_1)$ $(z \in N_1)$ for the family of sets $C(z)$ $(z \in N_1)$. Using the assumption $c \in N_1$, we get $R_1(c,r) \in I(N_1, c, 0_1)$, and hence $c = 0_1 \in N_1$.

Conversely, by making the definition $R_1(c,c_0) \equiv c_0$, the rule of N_1-elimination is derivable from the rule

$$\frac{c \in N_1}{c = 0_1 \in N_1}$$

and the rule of N_1-equality trivializes. Thus the operation R_1 can be dispensed with.

Example (about N_2). We make the definition

$$\text{Boolean} \equiv N_2.$$

Boolean is the type used in programming which consists of the two truth values true, false. So we could put true $\equiv 0_2$ and false $\equiv 1_2$. Then we can define *if* c *then* c_0 *else* c_1 $\quad R_2(c,c_0,c_1)$ because, if c is true, which means that c yields 0_2, then $R_2(c,c_0,c_1)$ has the same value as c_0; otherwise c yields 1_2 and $R_2(c,c_0,c_1)$ has the same value as c_1.

As for N_1 above, we can prove that any element of N_2 is either 0_2 or 1_2, but obviously only in the propositional form

$$\frac{c \in N_2}{I(N_2,c,0_2) \vee I(N_2,c,1_2) \text{ true}}$$

Example (negation). If we put

$$\sim A \equiv \neg A \equiv -A \equiv A \to N_0$$

we can easily derive all the usual rules of negation.

Consistency

What can we say about the consistency of our system of rules? We can understand consistency in two different ways:

(1) Metamathematical consistency. Then, to prove mathematically the consistency of a theory T, we consider another theory T', which contains codes for propositions of the original theory T and a predicate Der such that Der('A') expresses the fact that the proposition A with code 'A' is derivable in T. Then we define Cons $\equiv$ $\neg$ Der('$\bot$') $\equiv$ Der('$\bot$') $\supset \bot$ and (try to) prove that Cons is true in T'. This method is the only one applicable when, like Hilbert, we give up the hope of a semantical justification of the axioms and rules of inference; it could be followed, with success, also for intuitionistic type theory, but, since we have been as meticulous about its semantics as about its syntax, we have no need of it. Instead, we convince ourselves directly of its consistency in the following simple minded way.

(2) Simple minded consistency. This means simply that $\bot$ cannot be proved, or that we shall never have the right to judge $\bot$ true (which, unlike the proposition Cons above, is not a mathematical proposition). To convince ourselves of this, we argue as follows: if $c \in \bot$ would hold for some element (construction) c, then c would yield a canonical element $d \in \bot$; but this is impossible since $\bot$ has no canonical element by definiton (recall that we defined $\bot \equiv N_0$). Thus $\bot$ true cannot be proved by means of a system of correct rules. So, in case we hit upon a proof of $\bot$ true, we would know that there must be an error somewhere in the proof; and, if a formal proof of $\bot$ true is found, then at least one of the formal rules used in it is not correct. Reflecting on the meaning of each of the rules of

intuitionistic type theory, we eventually convince ourselves that they are correct; therefore we will never find a proof of $\bot$ true using them.

Finally, note that, in any case, we must rely on the simple minded consistency of at least the theory T' in which Cons is proved in order to obtain the simple minded consistency (which is the form of consistency we really care about) from the metamathematical consistency of the original theory T. In fact, once $c \in Cons$ for some c is proved, one must argue as follows: if T were not consistent, we would have a proof in T of $\bot$ true, or $a \in N_0$ for some a. By coding, this would give $'a' \in Der('\bot')$; then we would obtain $Ap(c,'a') \in \bot$, i.e. that $\bot$ true is derivable in T'. At this point, to conclude that $\bot$ true is not provable in T, we must be convinced that $\bot$ true is not provable in T'.

Natural numbers

So far, we have no means of constructing an infinite set. We now introduce the simplest one, namely the set of natural numbers, by the rules:

N-formation

$$N \text{ set}$$

N-introduction

$$0 \in N \qquad \frac{a \in N}{a' \in N}$$

Note that, as is the case with any other introduction rule, $a' \in N$ is always canonical, whatever element a is. Thus $a \in N$ means that a has value either 0 or a_1', where a_1 has value either 0 or a_2', etc., until, eventually, we reach an element a_n which has value 0.

N-elimination

$$\frac{c \in N \qquad d \in C(0) \qquad \begin{array}{c}(x \in N,\ y \in C(x)) \\ e(x,y) \in C(x')\end{array}}{R(c,d,(x,y)e(x,y)) \in C(c)}$$

where C(z) set $(z \in N)$. R(c,d,(x,y)e(x,y)) is explained as follows: first execute c, getting a canonical element of N, which is either 0 or a' for some $a \in N$. In the first case, continue by executing d, which yields a canonical element $f \in C(0)$; but, since $c = 0 \in N$ in this case, f is also a canonical element of C(c) = C(0). In the second case, substitute a for x and R(a,d,(x,y)e(x,y)) (namely, the

preceding value) for y in e(x,y) so as to get e(a,R(a,d,(x,y)e(x,y))). Executing it, we get a canonical f which, by the right premiss, is in C(a') (and hence in C(c) since c = a' ∈ N) under the assumption R(a,d,(x,y)e(x,y)) ∈ C(a). If a has value 0, then R(a,d,(x,y)e(x,y)) is in C(a) by the first case. Otherwise, continue as in the second case, until we eventually reach the value 0. This explanation of the elimination rule also makes the equality rules

N-equality

$$\frac{d \in C(0) \qquad e(x,y) \in C(x') \quad (x \in A,\ y \in C(x))}{R(0,d,(x,y)e(x,y)) = d \in C(0)}$$

$$\frac{a \in N \qquad d \in C(0) \qquad e(x,y) \in C(x') \quad (x \in N,\ y \in C(x))}{R(a',d,(x,y)e(x,y)) = e(a,R(a,d,(x,y)e(x,y))) \in C(a')}$$

evident. Thinking of C(z) (z ∈ N) as a propositional function (property) and suppressing the proofs (constructions) in the second and third premisses and in the conclusion of the rule of N-elimination, we arrive at

Mathematical induction

$$\frac{c \in N \qquad C(0)\ \text{true} \qquad C(x')\ \text{true} \quad (x \in N,\ C(x)\ \text{true})}{C(c)\ \text{true}}$$

If we explicitly write out the proof (construction) of C(c), we see that it is obtained by recursion. So recursion and induction turn out to be the same concept when propositions are interpreted as sets.

Example (the predecessor function). We put

$$pd(a) \equiv R(a,0,(x,y)x).$$

This definition is justified by computing R(a,0,(x,y)x): if a yields 0, then pd(a) also yields 0, and, if a yields b', then pd(a) yields the same value as R(b',0,(x,y)x), which, in turn, yields the same value as b. So we have pd(0) = 0 and pd(a') = a, which is the usual definition, but here these equalities are not definitional. More precisely, we have

$$\frac{a \in N}{pd(a) \in N}$$

which is an instance of N-elimination, and

$$\begin{cases} pd(0) = 0 \in N, \\ pd(a') = a \in N, \end{cases}$$

which we obtain by N-equality.

Using pd, we can derive the third Peano axiom

$$\frac{a' = b' \in N}{a = b \in N}$$

Indeed, from a' = b' ∈ N, we obtain pd(a') = pd(b') ∈ N which, together with pd(a') = a ∈ N and pd(b') = b ∈ N, yields a = b ∈ N by symmetry and transitivity. We can also obtain it in the usual form $(\forall x,y)(x' = y' \supset x = y)$, that is, in the present symbolism,

$$(\forall x \in N)(\forall y \in N)(I(N,x',y') \supset I(N,x,y))\ \text{true}.$$

In fact, assume $x \in N$, $y \in N$ and $z \in I(N,x',y')$. By I-elimination, $x' = y' \in N$; hence $x = y \in N$, from which $r \in I(N,x,y)$ by I-introduction. Then, by λ-abstraction, we obtain that $(\lambda x)(\lambda y)(\lambda z)r$ is a proof (construction) of the claim.

Example (addition). We define

$$a + b \equiv R(b,a,(x,y)y').$$

The meaning of $a + b$ is to perform b times the successor operation on a. Then one easily derives the rules:

$$\frac{a \in N \qquad b \in N}{a + b \in N}$$

$$\frac{a \in N}{a + 0 = a \in N} \qquad \frac{a \in N \qquad b \in N}{a + b' = (a + b)' \in N}$$

from which we can also derive the corresponding axioms of first order arithmetic, like in the preceding example. Note again that the equality here is not definitional.

Example (multiplication). We define

$$a \cdot b \equiv R(b,0,(x,y)(y + a)).$$

Usual properties of the product $a \cdot b$ can then easily be derived.

Example (the bounded μ-operator). We want to solve the problem: given a boolean function f on natural numbers, i.e. $f \in N \to N_2$, find the least argument, under the bound $a \in N$, for which the value of f is true. The solution will be a function $\mu(x,f) \in N$ $(x \in N,\ f \in N \to N_2)$ satisfying:

$$\mu(a,f) = \begin{cases} \text{the least } b \leq a \text{ such that } Ap(f,b) = 0_2 \in N_2, \\ \text{if such } b \text{ exists,} \\ \\ a, \text{ otherwise.} \end{cases}$$

Such a function will be obtained by solving the recursion equations:

$$\begin{cases} \mu(0,f) = 0 \in N, \\ \mu(a',f) = R_2(Ap(f,0),0,\mu(a,\overleftarrow{f})') \in N, \end{cases}$$

where $\overleftarrow{f} \equiv (\lambda x)Ap(f,x')$ is f shifted one step to the left, i.e. $Ap(\overleftarrow{f},x) = Ap(f,x') \in N_2$ $(x \in N)$. In fact, in case the bound is zero, $\mu(0,f) = 0 \in N$, irrespective of what function f is. When the bound has successor form, $\mu(a',f) = \mu(a,\overleftarrow{f})' \in N$, provided that $f(0) = \text{false} \equiv 1_2 \in N_2$; otherwise, $\mu(a',f) = 0 \in N$. Therefore to compute $\mu(a,f)$, we can shift f until the bound is 0, but checking each time if the value at 0 is true $\equiv 0_2$ or false $\equiv 1_2$. Even if it admits of a primitive recursive solution, the problem is most easily solved through higher types, as we shall now see in detail. We want to find a function $\mu(x) \in (N \to N_2) \to N$ $(x \in N)$ such that

$$\begin{cases} \mu(0) = (\lambda f)0 \in (N \to N_2) \to N, \\ \mu(a') = (\lambda f)R_2(Ap(f,0),0,Ap(\mu(a),\overleftarrow{f})') \in (N \to N_2) \to N, \end{cases}$$

so that we can define the function $\mu(a,f)$ we are looking for by putting $\mu(a,f) \equiv Ap(\mu(a),f)$. The requirements on $\mu(a)$ may be satisfied through an ordinary primitive recursion, but on a higher type; this task is fulfilled by the rule of N-elimination. We obtain

$$\mu(a) \equiv R(a,(\lambda f)0,(x,y)(\lambda f)R_2(Ap(f,0),0,Ap(y,\overleftarrow{f})')) \in (N \to N_2) \to N$$

under the premisses $a \in N$ and $f \in N \to N_2$, and hence

$$\mu(x,f) \in N \ (x \in N,\ f \in N \to N_2).$$

Written out in tree form the above proof of $\mu(a,f) \in N$ looks as follows:

$$\dfrac{\dfrac{a \in N \quad \dfrac{0 \in N}{(\lambda f)0 \in (N \to N_2) \to N} \quad \dfrac{\dfrac{\dfrac{(f \in N \to N_2) \quad 0 \in N}{Ap(f,0) \in N_2} \quad 0 \in N \quad \dfrac{\dfrac{(y \in (N \to N_2) \to N) \quad \overleftarrow{f} \in N \to N_2}{Ap(y,\overleftarrow{f}) \in N}}{Ap(y,\overleftarrow{f})' \in N}}{R_2(Ap(f,0),0,Ap(y,\overleftarrow{f})') \in N}}{(\lambda f)R_2(Ap(f,0),0,Ap(y,\overleftarrow{f})') \in (N \to N_2) \to N}}{\mu(a) \equiv R(a,(\lambda f)0,(x,y)(\lambda f)R_2(Ap(f,0),0,Ap(y,\overleftarrow{f})')) \in (N \to N_2) \to N} \quad f \in N \to N_2}{\mu(a,f) \equiv Ap(\mu(a),f) \in N}$$

Observe how the evaluation of $\mu(a,f) \equiv Ap(\mu(a),f)$ $Ap(R(a,(\lambda f)0,(x,y)(\lambda f)R_2(Ap(f,0),0,Ap(y,\overleftarrow{f})')),f)$ proceeds. First, a is evaluated. If the value of a is 0, the value of $\mu(a,f)$ equals the value of $Ap((\lambda f)0,f)$, which is 0. If, on the other hand, the value of a is b', the value of $\mu(a,f)$ equals the value of

$$Ap((\lambda f)R_2(Ap(f,0),0,\mu(b,\overleftarrow{f})'),f),$$

which, in turn, equals the value of

$$R_2(Ap(f,0),0,\mu(b,\overleftarrow{f})').$$

Next, $Ap(f,0)$ is evaluated. If the value of $Ap(f,0)$ is true $\equiv 0_2$, then the value of $\mu(a,f)$ is 0. If, on the other hand, the value of $Ap(f,0)$ is false $\equiv 1_2$, then the value of $\mu(a,f)$ equals the value of $\mu(b,\overleftarrow{f})'$.

Lists

We can follow the same pattern used to define natural numbers to introduce other inductively defined sets. We see here the example of lists.

List-formation

$$\dfrac{A \text{ set}}{List(A) \text{ set}}$$

where the intuitive explanation is: List(A) is the set of lists of elements of the set A (finite sequences of elements of A).

List-introduction

$$nil \in List(A) \qquad \dfrac{a \in A \quad b \in List(A)}{(a.b) \in List(A)}$$

where we may also use the notation $() \equiv nil$.

List-elimination

$$\dfrac{c \in List(A) \quad d \in C(nil) \quad e(x,y,z) \in C((x.y)) \quad (x \in A,\ y \in List(A),\ z \in C(y))}{listrec(c,d,(x,y,z)e(x,y,z)) \in C(c)}$$

where C(z) ($z \in List(A)$) is a family of sets. The instructions to execute listrec are: first execute c, which yields either nil, in which case continue by executing d and obtain $f \in C(nil) = C(c)$, or (a.b) with $a \in A$ and $b \in List(A)$; in this case, execute $e(a,b,listrec(b,d,(x,y,z)e(x,y,z)))$ which yields a canonical element

$f \in C((a.b)) = C(c)$. If we put $g(c) \equiv$ listrec(c,d,(x,y,z)e(x,y,z)), then f is the value of e(a,b,g(b)).

List-equality

$$\frac{d \in C(\mathrm{nil}) \qquad e(x,y,z) \in C((x.y)) \quad (x \in A,\ y \in \mathrm{List}(A),\ z \in C(y))}{\mathrm{listrec}(\mathrm{nil},d,(x,y,z)e(x,y,z)) = d \in C(\mathrm{nil})}$$

$$\frac{a \in A \qquad b \in \mathrm{List}(A) \qquad d \in C(\mathrm{nil}) \qquad e(x,y,z) \in C((x.y)) \quad (x \in A,\ y \in \mathrm{List}(A),\ z \in C(y))}{\mathrm{listrec}((a.b),d,(x,y,z)e(x,y,z)) = e(a,b,\mathrm{listrec}(b,d,(x,y,z)e(x,y,z))) \in C((a.b))}$$

Similar rules could be given for finite trees and other inductively defined concepts.

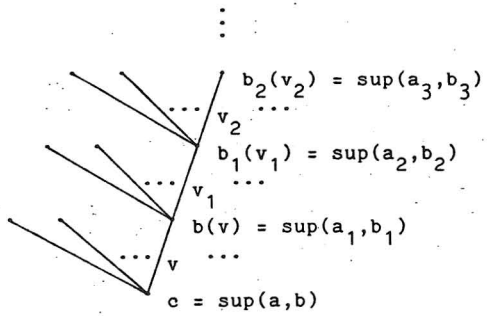
Wellorderings

The concept of wellordering and the principle of transfinite induction were first introduced by Cantor. Once they had been formulated in ZF, however, they lost their original computational content. We can construct ordinals intuitionistically as wellfounded trees, which means that they are no longer totally ordered.

W-formation

$$\frac{A \text{ set} \qquad B(x) \text{ set} \quad (x \in A)}{(Wx \in A)B(x) \text{ set}}$$

What does it mean for c to be an element of $(Wx \in A)B(x)$? It means that, when calculated, c yields a value of the form sup(a,b) for some a and b, where $a \in A$ and b is a function such that, for any choice of an element $v \in B(a)$, b applied to v yields a value $\sup(a_1,b_1)$, where $a_1 \in A$ and b_1 is a function such that, for any choice of v_1 in $B(a_1)$, b_1 applied to v_1 has a value $\sup(a_2,b_2)$, etc., until in any case (i.e. however the successive choices are made) we eventually reach a bottom element of the form $\sup(a_n,b_n)$, where $B(a_n)$ is empty, so that no choice of an element in $B(a_n)$ is possible. The following picture, in which we loosely write b(v) for Ap(b,v), can help (look at it from bottom to top):



By the preceding explanation, the following rule for introducing canonical elements is justified:

W-introduction

$$\frac{a \in A \qquad b \in B(a) \to (Wx \in A)B(x)}{\sup(a,b) \in (Wx \in A)B(x)}$$

Think of sup(a,b) as the supremum (least ordinal greater than all) of the ordinals b(v), where v ranges over B(a).

We might also have a bottom clause, $0 \in (Wx \in A)B(x)$ for instance, but we obtain 0 by taking one set in B(x) set $(x \in A)$ to be the empty set: if $a_0 \in A$ and $B(a_0) = N_0$, then $R_0(y) \in (Wx \in A)B(x)$ $(y \in B(a_0))$ so that $\sup(a_0,(\lambda y)R_0(y)) \in (Wx \in A)B(x)$ is a bottom element.

From the explanation of what an element of $(Wx \in A)B(x)$ is, we see the correctness of the elimination rule, which is at the same time transfinite induction and transfinite recursion. The appropriate principle of transfinite induction is: if the property $C(w)$ $(w \in (Wx \in A)B(x))$ is inductive (i.e. if it holds for all predecessors $Ap(b,v) \in (Wx \in A)B(x)$ $(v \in B(a))$ of an element sup(a,b), then it holds for sup(a,b) itself), then C(c) holds for an arbitrary element $c \in (Wx \in A)B(x)$. A bit more formally,

$$\frac{c \in (Wx \in A)B(x) \qquad \begin{array}{c}(\forall x \in A)(\forall y \in B(x) \to (Wx \in A)B(x)) \\ ((\forall v \in B(x))C(Ap(y,v)) \supset C(\sup(x,y))) \text{ true}\end{array}}{C(c) \text{ true}}$$

Now we resolve this, obtaining the W-elimination rule. One of the premisses is that C(sup(x,y)) is true, provided that $x \in A$, $y \in B(x) \to (Wx \in A)B(x)$ and $(\forall v \in B(x))C(Ap(y,v))$ is true. Letting d(x,y,z) be the function which gives the proof of C(sup(x,y)) in terms of $x \in A$, $y \in B(x) \to (Wx \in A)B(x)$ and the proof z of $(\forall v \in B(x))C(Ap(y,v))$, we arrive at the rule

W-elimination

$$\frac{c \in (Wx \in A)B(x) \qquad \begin{array}{c}(x \in A,\ y \in B(x) \to (Wx \in A)B(x),\ z \in (\prod v \in B(x))C(Ap(y,v))) \\ d(x,y,z) \in C(\sup(x,y))\end{array}}{T(c,(x,y,z)d(x,y,z)) \in C(c)}$$

where T(c,(x,y,z)d(x,y,z)) is executed as follows. First execute c, which yields sup(a,b), where $a \in A$ and $b \in B(a) \to (Wx \in A)B(x)$. Select the components a and b and substitute them for x and y in d, obtaining d(a,b,z). We must now substitute for z the whole sequence of previous function values. This sequence is $(\lambda v)T(Ap(b,v),(x,y,z)d(x,y,z))$, because $Ap(b,v) \in (Wx \in A)B(x)$ $(v \in B(a))$ is the function which enumerates the subtrees (predecessors) of sup(a,b). Then $d(a,b,(\lambda v)T(Ap(b,v),(x,y,z)d(x,y,z)))$ yields a canonical element $e \in C(c)$ as value under the assumption that $T(Ap(b,v),(x,y,z)d(x,y,z)) \in C(Ap(b,v))$ $(v \in B(a))$.

If we write $f(c) \equiv T(c,(x,y,z)d(x,y,z))$, then, when c yields $\sup(a,b)$, f(c) yields the same value as $d(a,b,(\lambda v)f(Ap(b,v)))$. This explanation also shows that the rule

W-equality

$$(x \in A,\ y \in B(x) \to (Wx \in A)B(x),\ z \in (\prod v \in B(x))C(Ap(y,v)))$$

$$\frac{a \in A \quad b \in B(a) \to (Wx \in A)B(x) \qquad d(x,y,z) \in C(\sup(x,y))}{T(\sup(a,b),(x,y,z)d(x,y,z)) = d(a,b,(\lambda v)T(Ap(b,v),(x,y,z)d(x,y,z))) \in C(\sup(a,b))}$$

is correct.

Example (the first number class). Having access to the W-operation and a family of sets $B(x)$ $(x \in N_2)$ such that $B(0_2) = N_0$ and $B(1_2) = N_1$, we may define the first number class as $(Wx \in N_2)B(x)$ instead of taking it as primitive.

Example (the second number class). We give here the rules for a simple set of ordinals, namely the set $\mathcal{O}$ of all ordinals of the second number class, and show how they are obtained as instances of the general rules for wellorderings.

$\mathcal{O}$-formation

$$\mathcal{O}\ \text{set}$$

Cantor generated the second number class from the initial ordinal 0 by applying the following two principles:

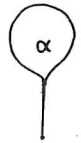
(1) given $\alpha \in \mathcal{O}$, form the successor $\alpha' \in \mathcal{O}$;

(2) given a sequence of ordinals $\alpha_0, \alpha_1, \alpha_2, \ldots$ in $\mathcal{O}$, form the least ordinal in $\mathcal{O}$ greater than each element of the sequence.

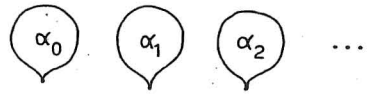
We can give pictures:

(1) if

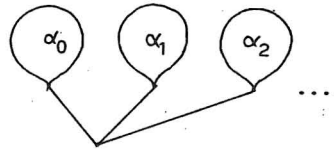
is in $\mathcal{O}$, then we can build the successor α':



(2) if



is a sequence of ordinals in $\mathcal{O}$, then we can build the supremum $\sup_n(\alpha_n)$:



So $\mathcal{O}$ will be inductively defined by the three rules:

$\mathcal{O}$-introduction

$$0 \in \mathcal{O} \qquad \frac{a \in \mathcal{O}}{a' \in \mathcal{O}} \qquad \frac{b \in N \to \mathcal{O}}{\sup(b) \in \mathcal{O}}$$

Transfinite induction over $\mathcal{O}$ is evident, and it is given by

$$\frac{c \in \mathcal{O} \quad C(0)\ \text{true} \quad (x \in \mathcal{O},\ C(x)\ \text{true}) \quad C(x')\ \text{true} \quad (z \in N \to \mathcal{O},\ (\forall n \in N)C(Ap(z,n))\ \text{true}) \quad C(\sup(z))\ \text{true}}{C(c)\ \text{true}}$$

where $C(z)$ $(z \in \mathcal{O})$ is a property over $\mathcal{O}$. Writing it with proofs, we obtain

$\mathcal{O}$-elimination

$$\frac{c \in \mathcal{O} \quad d \in C(0) \quad e(x,y) \in C(x') \quad (x \in \mathcal{O},\ y \in C(x)) \quad f(z,w) \in C(\sup(z)) \quad (z \in N \to \mathcal{O},\ w \in (\prod n \in N)C(Ap(z,n))}{T(c,d,(x,y)e(x,y),(z,w)f(z,w)) \in C(c)}$$

where the transfinite recursion operator T is executed as follows. First, execute c. We distinguish the three possible cases:

if we get $0 \in \mathcal{O}$, the value of $T(c,d,(x,y)e(x,y),(z,w)f(z,w))$ is the value of $d \in C(0)$;

if we get a', then the value is the value of $e(a,T(a,d,(x,y)e(x,y),(z,w)f(z,w)))$;

if we get $\sup(b)$, we continue by executing $f(b,(\lambda x)T(Ap(b,x),d,(x,y)e(x,y),(z,w)f(z,w)))$.

In any case, we obtain a canonical element of $C(c)$ as result.

It is now immediate to check that we can obtain all $\mathcal{O}$-rules (including $\mathcal{O}$-equality, which has not been spelled out) as instances of the W-rules if we put

$$\mathcal{O} \equiv (Wx \in N_3)B(x),$$

where $B(x)$ $(x \in N_3)$ is a family of sets such that $B(0_3) = N_0$, $B(1_3) = N_1$ and $B(2_3) = N$. Such a family can be constructed by means of the universe rules.

Example (initial elements of wellorderings). We want to show that, if at least one index set is empty, then the wellordering $(Wx \in A)B(x)$ is nonempty. Recall that we want to do it intuitionistically, and recall that A true is equivalent to A nonempty, so that $\neg A$ true is equivalent to A empty. So our claim is:

$$(\exists x \in A)\ \neg B(x) \to (Wx \in A)B(x)\ \text{true}.$$

To see this, assume $x \in A$, $y \in \neg B(x)$ and $v \in B(x)$. Then $Ap(y,v) \in N_0 \equiv \bot$ and hence $R_0(Ap(y,v)) \in (Wx \in A)B(x)$, applying the rule of N_0-elimination. We now abstract on v to get $(\lambda v)R_0(Ap(y,v)) \in B(x) \to (Wx \in A)B(x)$ and, by W-introduction, $\sup(x,(\lambda v)R_0(Ap(y,v))) \in (Wx \in A)B(x)$. Assuming $z \in (\Sigma x \in A)\neg B(x)$, by Σ-elimination, we have

$$E(z,(x,y)\sup(x,(\lambda v)R_0(Ap(y,v)))) \in (Wx \in A)B(x),$$

from which, by λ-abstraction on z,

$$(\lambda z)E(z,(x,y)\sup(x,(\lambda v)R_0(Ap(y,v)))) \in (\Sigma x \in A)\neg B(x) \to (Wx \in A)B(x).$$

We now want to show a converse. However, note that we cannot have $(Wx \in A)B(x) \to (\exists x \in A)\neg B(x)$ true, because of the intuitionistic meaning of the existential quantifier. But we do have:

$$(Wx \in A)B(x) \to \neg(\forall x \in A)B(x)\ \text{true}.$$

Assume $x \in A$, $y \in B(x) \to (Wx \in A)B(x)$ and $z \in B(x) \to N_0$. Note that

$B(x) \to N_0 \equiv (\prod v \in B(x))C(Ap(y,v))$ for $C(w) \equiv N_0$, so that we can apply the rule of W-elimination. Assuming $f \in (\prod x \in A)B(x)$, we have $Ap(f,x) \in B(x)$, and hence also $Ap(z,Ap(f,x)) \in N_0$. $Ap(z,Ap(f,x))$ takes the role of $d(x,y,z)$ in the rule of W-elimination. So, if we assume $w \in (Wx \in A)B(x)$, we obtain $T(w,(x,y,z)Ap(z,Ap(f,x))) \in N_0$. Abstracting on f, we have

$$(\lambda f)T(w,(x,y,z)Ap(z,Ap(f,x))) \in \neg(\forall x \in A)B(x),$$

and, abstracting on w, we have

$$(\lambda w)(\lambda f)T(w,(x,y,z)Ap(z,Ap(f,x))) \in (Wx \in A)B(x) \to \neg(\forall x \in A)B(x).$$

Universes

So far, we only have a structure of finite types, because we can only iterate the given set forming operations starting from $I(A,a,b)$, N_0, N_1, ... and N a finite number of times. To strengthen the language, we can add transfinite types, which in our language are obtained by introducing universes. Recall that there can be no set of all sets, because we are not able to exhibit once and for all all possible set forming operations. (The set of all sets would have to be defined by prescribing how to form its canonical elements, i.e. sets. But this is impossible, since we can always perfectly well describe new sets, for instance, the set of all sets itself.) However, we need sets of sets, for instance, in category theory. The idea is to define a universe as the least set closed under certain specified set forming operations. The operations we have been using so far are:

$$\frac{A \text{ set} \quad B(x) \text{ set}\ (x \in A)}{(\prod x \in A)B(x) \text{ set}} \qquad \frac{A \text{ set} \quad B(x) \text{ set}\ (x \in A)}{(\sum x \in A)B(x) \text{ set}} \qquad \frac{A \text{ set} \quad B \text{ set}}{A + B \text{ set}}$$

$$\frac{A \text{ set} \quad b, c \in A}{I(A,b,c) \text{ set}} \qquad N_0 \text{ set} \quad N_1 \text{ set} \ldots N \text{ set} \qquad \frac{A \text{ set} \quad B(x) \text{ set}\ (x \in A)}{(Wx \in A)B(x) \text{ set}}$$

There are two possible ways of building a universe, i.e. to obtain closure under possibly transfinite iterations of such operations.

Formulation à la Russell. Consider $\prod$, $\sum$, ... both as set forming operations and as operations to form canonical elements of

the set U, the universe. This is like in ramified type theory.

Formulation à la Tarski. So called because of the similarity between the family $T(x)(x \in U)$ below and Tarski's truth definition. We use new symbols, mirroring (reflecting) Π, Σ , ..., to build the canonical elements of U. Then U consists of indices of sets (like in recursion theory). So we will have the rules:

U-formation

$$U \text{ set} \qquad \frac{a \in U}{T(a) \text{ set}}$$

U and $T(x)(x \in U)$ are defined by a simultaneous transfinite induction, which, as usual, can be read off the following introduction rules:

U-introduction

$$\frac{a \in U \qquad (x \in T(a))\ b(x) \in U}{\pi(a,(x)b(x)) \in U} \qquad \frac{a \in U \qquad (x \in T(a))\ b(x) \in U}{T(\pi(a,(x)b(x))) = (\Pi x \in T(a))T(b(x))}$$

$$\frac{a \in U \qquad (x \in T(a))\ b(x) \in U}{\sigma(a,(x)b(x)) \in U} \qquad \frac{a \in U \qquad (x \in T(a))\ b(x) \in U}{T(\sigma(a,(x)b(x))) = (\Sigma x \in T(a))T(b(x))}$$

$$\frac{a \in U \qquad b \in U}{a + b \in U} \qquad \frac{a \in U \qquad b \in U}{T(a + b) = T(a) + T(b)}$$

$$\frac{a \in U \qquad b \in T(a) \qquad c \in T(a)}{i(a,b,c) \in U} \qquad \frac{a \in U \qquad b \in T(a) \qquad c \in T(a)}{T(i(a,b,c)) = I(T(a),b,c)}$$

$$n_0 \in U \qquad n_1 \in U \qquad \ldots \qquad T(n_0) = N_0 \qquad T(n_1) = N_1 \qquad \ldots$$

$$n \in U \qquad T(n) = N$$

$$\frac{a \in U \qquad (x \in T(a))\ b(x) \in U}{w(a,(x)b(x)) \in U} \qquad \frac{a \in U \qquad (x \in T(a))\ b(x) \in U}{T(w(a,(x)b(x))) = (Wx \in T(a))T(b(x))}$$

We could at this point iterate the process, obtaining a second universe U' with the two new introduction rules:

$$u \in U' \qquad T'(u) = U$$

$$\frac{a \in U}{t(a) \in U'} \qquad \frac{a \in U}{T'(t(a)) = T(a)}$$

then a third universe U", and so on.

In the formulation à la Russell, T disappears and we only use capital letters. So the above rules are turned into:

U-formation

$$U \text{ set} \qquad \frac{A \in U}{A \text{ set}}$$

U-introduction

$$\frac{A \in U \qquad B(x) \in U \quad (x \in A)}{(\Pi x \in A)B(x) \in U} \qquad \frac{A \in U \qquad B(x) \in U \quad (x \in A)}{(\Sigma x \in A)B(x) \in U}$$

$$\frac{A \in U \qquad B \in U}{A + B \in U} \qquad \frac{A \in U \qquad b, c \in A}{I(A,b,c) \in U}$$

$$N_0 \in U \qquad N_1 \in U \qquad \ldots \qquad N \in U$$

$$\frac{A \in U \qquad B(x) \in U \quad (x \in A)}{(Wx \in A)B(x) \in U}$$

However, U itself is not an elemnt of U. In fact, the axiom $U \in U$ leads to a contradiction (Girard's paradox[13]). We say that a set A is small, or a U-set, if it has a code $a \in U$, that is, if there is an element $a \in U$ such that $T(a) = A$. More generally, a family $A(x_1,\ldots,x_n)$ $(x_1 \in A_1, \ldots, x_n \in A_n(x_1,\ldots,x_{n-1}))$ is said to be small provided $A(x_1,\ldots,x_n) = T(a(x_1,\ldots,x_n))$ $(x_1 \in A_1, \ldots, x_n \in A_n(x_1,\ldots,x_{n-1}))$ for some indexing function $a(x_1,\ldots,x_n) \in U$ $(x_1 \in A_1, \ldots, x_n \in A_n(x_1,\ldots,x_{n-1}))$. So the category of small sets is closed under the operations Σ, Π, etc. U is a perfectly good set,

[13] J. Y. Girard, Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur, Thèse, Université Paris VII, 1972.

but it is not small. Using U, we can form transfinite types (using a recursion with value in U, for instance).

The set $V \equiv (Wx \in U)T(x)$ (or, in the formulation à la Russell, simply $(WX \in U)X$) has been used by Aczel[14] to give meaning to a constructive version of Zermelo-Fraenkel set theory via intuitionistic type theory.

Example (fourth Peano axiom). We now want to prove the fourth Peano axiom, which is the only one not trivially derivable from our rules. So the claim is:

$$(\forall x \in N) \neg I(N,0,x') \text{ true.}$$

We use U-rules in the proof; it is probably not possible to prove it otherwise. From N set, $0 \in N$, $x \in N$ we have $x' \in N$ and $I(N,0,x')$ set. Now assume $y \in I(N,0,x')$. Then, by I-elimination, $0 = x' \in N$. By U-introduction, $n_0 \in U$ and $n_1 \in U$. Then we define $f(a) \equiv R(a,n_0,(x,y)n_1)$, so that $f(0) = n_0 \in U$ and $f(a') = n_1 \in U$ provided that $a \in N$. From $0 = x' \in N$, we get, by the equality part of the N-elimination rule, $R(0,n_0,(x,y)n_1) = R(x',n_0,(x,y)n_1) \in U$. But $R(0,n_0,(x,y)n_1) = n_0 \in U$ and $R(x',n_0,(x,y)n_1) = n_1 \in U$ by the rule of N-equality. So, by symmetry and transitivity, $n_0 = n_1 \in U$. By the (implicitly given) equality part of the U-formation rule, $T(n_0) = T(n_1)$. Hence, from $T(n_0) = N_0$ and $T(n_1) = N_1$, $N_0 = N_1$. Since $0_1 \in N_1$, we also have $0_1 \in N_0$. So $(\lambda y)0_1 \in I(N,0,x') \to N_0$ and $(\lambda x)(\lambda y)0_1 \in (\forall x \in N) \neg I(N,0,x')$.

We remark that, while it is obvious (by reflecting on its meaning) that $0 = a' \in N$ is not provable, a proof of $\neg I(N,0,a')$ true seems to involve treating sets as elements in order to define a propositional function which is $\bot$ on 0 and $\top$ on a'.

[14] P. Aczel, The type theoretic interpretation of constructive set theory, Logic Colloquium 77, Edited by A. Macintyre, L. Pacholski and J. Paris, North-Holland, Amsterdam, 1978, pp. 55-66.