

Constructive notions of set
Part I
Sets in Martin–Löf type theory

LAURA CROSILLA

This is the first of two articles dedicated to the notion of constructive set. In them we attempt a comparison between two different notions of set which occur in the context of the foundations for constructive mathematics. We also put them under perspective by stressing analogies and differences with the notion of set as codified in the classical theory Zermelo–Fraenkel. In the current article we illustrate in some detail the notion of set as expressed in Martin–Löf type theory and present the essential characters of this theory. In a second article we shall explore a distinct notion of set, as arising in the context of intuitionistic versions of Zermelo–Fraenkel set theory. The theory we shall analyse there is Aczel’s **CZF** (Constructive Zermelo–Fraenkel) and we shall supplement its exposition by a succinct account of Aczel’s interpretation of **CZF** in type theory. This will enable us to compare the two notions in a more precise sense.

1. *Introduction*

Constructive type theory has been founded by Per Martin–Löf «with the philosophical motive of clarifying the syntax and semantics of intuitionistic mathematics»([22]). It is a rich and expressive

Received by the editors: November 2005.

This article collects material from lectures presented by the author at the Logic seminar of the Philosophy Department, Florence University, led by A. Cantini and P. Minari. The work has been accomplished while the author was supported by a Research Grant of the University of Florence and MIUR (for the years 2003-2005).

The rules of type theory have been written using the macro *bussproofs.sty* by S. Buss, University of California, San Diego; thanks to the author for making available to others his macro.

Thanks to P. Minari for kindly providing a customised version of the amsart class file.

theory whose fundamental concept is that of type. An essential part of type theory is constituted by a theory of sets, intended as specific kinds of types whose equality and membership can be asserted in an exhaustive way. A set theory in this sense is the main topic of the monograph [23].

Aim of this article is a presentation as simple as possible of the basic ideas of the theory of sets in [23]. We believe that a more widespread familiarity with type theory will be beneficial to the discussion on the foundations of mathematics as well as to the understanding of constructivism. One of the greatest benefits of type theory, in fact, is that the examination of the logical notions is not isolated but on the contrary carried out as part of the more general task of clarifying the foundations of constructive mathematics and in particular the notion of set. This seems very much in agreement with Brouwer's view of a precedence of mathematics over logic. The logical rules are elegantly obtained from the more general rules for sets by means of the Curry–Howard correspondence and stand out for clarity if compared for example with the rather mysterious justification of the intuitionistic logical operators usually referred to as *BHK-interpretation*.

In a second article we shall explore the relationship between the notion of set in type theory and that underlying a specific intuitionistic version of Zermelo–Fraenkel axiomatic set theory. The theory we shall analyse there is Aczel's **CZF**, Constructive Zermelo–Fraenkel (see for example [2] and [3]). It is characterised by the simplicity of its language, which is the same as that of classical Zermelo–Fraenkel set theory, of which it is in fact a subsystem. Hence in developing constructive mathematics, one can make use of the usual set–theoretic intuitions, the only and essential difference with classical set theory being the requirement to avoid arguments of a non–constructive character. The relationship between the two theories will be clarified by recalling Aczel's interpretation of **CZF** in a corresponding system of Martin–Löf type theory. In view of the fact that we shall here mainly present the fragment of type theory which relates with sets, we shall distinguish between these two notions of set by referring to Martin–Löf's theory of sets as *constructive type theory* and to Aczel's as *(axiomatic) constructive set theory*.

1.1. *Bishop style constructive mathematics.* Constructive type theory aims at a codification of a genuinely constructive notion of set which is sufficiently expressive to allow for a full formalisation of constructive mathematics. It also pursues the aim of expressing mathematical statements in a way which is in time with the technological changes which have modified the world surrounding us. In fact, its formalism also represents a very abstract and powerful programming language.

When referring to constructive mathematics, we here have in mind the constructive mathematics which has been and is developed by E. Bishop and his school.¹ The distinguished American mathematician published his *Foundations of constructive analysis* in 1967 [6]. The book as well as subsequent work by the Bishop school proved the possibility to develop considerable parts of mathematics on the basis of intuitionistic logic.² More importantly, it also showed that this can be accomplished without contradicting classical mathematics. The essential advantage with respect to classical mathematics is that the results now have a distinctive algorithmic flavour. In fact, in a constructive context the computational content of a mathematical theorem arises directly from its proof. More specifically, a constructive proof of a statement of the form ‘for all objects of type A there exists an object of type B such that ...’ in principle provides us with an algorithm which given an object of A produces a witness in B of the truth of this statement. It should also be noted that proofs in constructive mathematics may be harder and more complex, but are certainly more informative. Clearly, one can not expect to obtain the same scope of results which have honoured the classical tradition. From a constructive point of view, however, a classical theorem not amenable of constructive treatment simply is not justified or even meaningful.

From an historical perspective, it is worth observing that Bishop presented his new approach to mathematics in an informal style, as close as possible to the classical one. He thus made it accessible to

¹ It seems now a practice to refer to mathematicians working in constructive mathematics in the style of Bishop as the *Bishop school*. From an historical point of view, it should be stressed that some of the recognised representative of the ‘school’ like, for example, D. Bridges were not students of Bishop.

² For a comprehensive treatment of constructive analysis see Bishop and Bridges [7].

the wider community of mathematicians, crossing the boundary of the limited circle of partisans of intuitionism.³

Bishop's book of 1967 had an enduring impact on the logical community. Logicians like S. Feferman ([12]), J. Myhill ([24]), H. Friedmann [15] immediately felt the need to produce formal systems which could provide a clear foundation for Bishop's analysis, which had been presented by its author in an informal way.⁴

1.2. *Martin-Löf type theory.* Martin-Löf type theory also arose in the late sixties as a system in which to formalise Bishop style mathematics. As an element of distinction it retains from the very beginning a more philosophical inclination, if compared with other foundational systems. The theory seems to have arisen from a reflection on the essence of constructive reasoning and especially on the notion of constructive set, the syntactical details not preceding but following the semantical considerations. In developing his type theory, Martin-Löf had numerous sources of inspiration. B. Russell founded type theory at the beginning of the last century as a reaction to the paradoxes which had plagued the new-born foundational systems for mathematics. A. Church also introduced a theory of types in the context of his lambda calculus. He thus amended similar difficulties which had arisen in following the temptation of erecting a comprehensive foundational theory on the basis of a calculus whose primordial aim was to formalise and explicate the notion of function. Martin-Löf adopted the fundamental Russellian idea for which a type is a domain of quantification. He also combined this with ideas from the typed lambda calculus, of which his type theory can be seen as a particular version (characterised by predicativity

³ The so called Bishop school in general has on our view the merit of emphasising the relevance *per se* of constructive results as well as the computational impact of their approach and the benefits for the understanding. Another aspect which seems in general to characterise the approach of the Bishop school is that questions of a more philosophical nature, for example related to the specific constructive method or the ontological status of the mathematical object, do not concern the constructive mathematician. In the author's view this is rather a limitation than a merit of the school, whose technical contributions are undoubtedly far reaching.

⁴ See Feferman's [13] for a clear account of different approaches to constructive mathematics and foundations. Beeson's [5] is also a useful source of information. The monograph [8] by Bridges and Richman focuses instead on different varieties of constructive mathematics.

and the presence of dependent types). In adapting the notion of type to a constructive context the Swedish mathematician also benefitted from the reflections by D. Scott ([26]), G. Kreisel and N. D. Goodman ([18], [19], [17]).⁵ It is apparent from the way constructive type theory is formulated and especially from the relevance that meaning explanations have in the presentation and justification of the system, that G. Gentzen’s natural deduction in the form adopted by D. Prawitz must have also played a significant role. We also assume that Martin–Löf was aware of the research completed within the AUTOMATH project, led by N. G. de Bruijn (see for example [11]). Here a significant part of mathematics was proof-checked by a machine and wide use of the so called Curry–Howard correspondence was made for the first time (see section 1.4). The correspondence is doubtlessly one of the fundamental constituents of constructive type theory.⁶

1.3. *Two approaches to the notion of set.* As already mentioned, the monograph [23] is mainly committed to spelling out a constructive notion of set. We are thus induced to try and elucidate which relationship links this notion of set with the concept which arises in Zermelo–Fraenkel set theory. The following is simply an attempt to discuss this relationship and does not claim to be in any way a compelling and comprehensive analysis. On the contrary, we believe that an accurate investigation ought to be carried out to better clarify this point.⁷

From a purely syntactical point of view, there is an obvious difference between axiomatic set theory and type theory with regard

⁵ See [21] for a brief historical account.

⁶ A note to recall that the original version of constructive type theory (never come to print) was shown inconsistent by J. Y. Girard ([16]). The assumption which made it inconsistent was that of the existence of a type of all types. It appears that Martin–Löf was led to this assumption by the intent to allow for a formalisation inside type theory of category theory. Martin–Löf promptly corrected the mistake by replacing the heavily impredicative notion of a type of all types by a hierarchy of stronger and stronger universes, each one reflecting all the universes occurring ‘earlier’ in the hierarchy. (See section 4.12 for the notion of universe.)

⁷ I am grateful to Giovanni Sambin for a discussion we had at Venice international University in occasion of a Springschule organised by the GKLI München in April 2004. He brought to my attention the need to deepen our understanding of the relationship between the two notions of set as they occur in type theory and in constructive Zermelo–Fraenkel set theory.

to the two fundamental relations of equality and membership. In axiomatic set theory sets may be viewed as if they were all ‘at the same level’, the equality and membership relations connecting sets with sets. In type theory membership relates an element with a set, not a set with a set and equality may only hold of elements of the same set (that is, it is relativised to a specific set) or it holds of two sets. This syntactical distinctness underlines a deep conceptual difference between the two notions of set. We could attempt to clarify this point by imagining the universe described by the axiomatic theories as an unstratified uniform collection of objects essentially of the same kind and nested one inside the other according to the axioms of the theory. A hierarchical structure is imposed ‘a posteriori’ on the universe by assigning a rank level (an ordinal) to each set (via the foundation axiom). The type-theoretic universe, instead, is made of sets and these of elements, but the latter can not be in turn considered to be sets. One could also say that the type-theoretic universe is more rigid in structure and represents a stratified notion of set, while the Cantorian notion of set is essentially iterative. A paradigmatic example is given by the way the fundamental concept of natural number is represented in the two theories. Type theory represents natural numbers as distinct, primitive objects, elements of a set \mathbf{N} , which is inductively generated by taking first 0 and then repeatedly applying the successor function. Zermelo–Fraenkel set theory in its usual formulation, obtains the natural numbers by an iterated application of the successor operation to the empty set.⁸ The successor operation is a set-theoretic operation, which allows us to move from sets to sets and it is formulated so to ensure that the natural numbers satisfy the fundamental property of transitivity.⁹

Before proceeding with our discussion a remark is mandatory. In the second part of this project we shall see that an appropriate version of type theory allows for a natural interpretation of constructive Zermelo–Fraenkel set theory. In fact, as Aczel [1] has shown, the iterative concept of set can be captured within Martin–Löf type theory by a specific type construction: sets are seen as elements of a particular inductive type. Here, though, we attempt a comparison

⁸ Note that there are also formulations of set theory (for example Zermelo’s) which allow for urelements as well as sets.

⁹ A set is transitive if its elements are subsets of it.

between the iterative notion of set and the more general notion advanced in Martin–Löf’s [23] (in other terms sets as elements of the first universe rather than sets as elements of an inductive type built over the universe).

Another important characteristic of type theory which distinguishes it from other constructive and classical theories is the fact that it is ‘logic free’. While Zermelo–Fraenkel set theory is based on the first order predicate calculus, type theory is defined by a series of rules of set formation which do not presuppose logic in any way. Indeed, by simply modifying the reading of the notion of set, and seeing it as expressing a notion of proposition, specific instances of the rules yield the intuitionistic calculus in natural deduction form. This is essentially a consequence of embracing the Curry–Howard correspondence. As we shall see the propositions–as–types paradigm has far reaching consequences for the theory, among which the validity of a form of choice principle.

The previous considerations compare axiomatic set theory with type theory and apply to classical as well as intuitionistic versions of set theory. We believe that there is another significant difference between *classical* set theory (at least in the way it is often presented) and the approach proposed by Martin–Löf and also adopted in constructive versions of Zermelo–Fraenkel set theory. The dissimilarity concerns the status of the universe of sets. Typically the classical set theorist sees our mathematical activity as the gradual disclosure of properties of *the* universe of sets. By considering stronger principles as part of the stock of axioms of our set theory, we can in general discover properties of a wider fragment of this universe. In a constructive context, instead, according to the intuitionistic view which ascribes a more relevant role to the thinking subject, the universe of sets can not be considered as ‘given’ to us in any way. It is rather a mental construction and thus possesses the features of an open concept, it is a universe ‘in fieri’. New stronger principles determine a new form of universe which exists by the same act of postulating the new principles.

An essential feature of constructive type theory, also shared by constructive set theory, is its commitment to predicativity.¹⁰ These

¹⁰ See Myhill [24] for a discussion on the issue of predicativity for constructive set theory. For a deeper and more general account of mathematics developed in a predicative context see for example Feferman [14].

theories are in fact formulated so to prevent any use of strongly impredicative operations on sets such as powerset and unrestricted comprehension. The reason advanced for avoiding these operations is simply that they can not be justified by the bottom up approach to the notion of set which follows from the insight on what constitutes a constructive set. It is also believed that no essentially impredicative notion is needed for the development of constructive mathematics.

Finally, we have already mentioned that the computational reading has a key role for type theory: sets correspond in fact to data types and type theory may be seen as a very abstract programming language. This clearly distinguishes the notion of set of type theory from the ‘Cantorian’ notion, which was developed well before the advent of computers and seems to retain a purely mathematical scope.¹¹

1.4. *Curry–Howard isomorphism.* Type theory makes essential use of the so called Curry–Howard correspondence (also known as ‘propositions–as–types’ or Curry–Howard isomorphism). This is a correspondence between propositions and proofs on the one side and types and elements on the other (see for example [28] for a clear presentation and an historical account of the correspondence). The isomorphism becomes clear if we compare the notions of truth which characterise classical and intuitionistic logic. Classically a proposition is either true or false; truth therefore coincides, in ‘Fregean’ spirit, with admitting values in a two elements set $\{t, f\}$. Intuitionistically, instead, a proposition is true if it is provable, that is if we can, at least in principle, provide a proof of it (not necessarily in a formal system). The Curry–Howard correspondence is obtained by keeping this latter notion of truth in mind and identifying a proposition with the collection (or type) of its proofs.

For a more concrete example, let us consider the introduction rule for implication:

$$\frac{\begin{array}{c} [A] \\ \vdots \\ B \end{array}}{A \supset B}$$

¹¹ Krivine [20] has recently addressed the question of whether it is possible to extract computational content from classical set theory.

The rule allows us to derive an implication of the form $A \supset B$, whenever from a proof of A we can obtain a proof of B . This can also be expressed by saying that a proof of $A \supset B$ is a ‘function’ which transforms any proof a of A in a proof $b(a)$ of B . Utilising the notation of the Lambda calculus we can write this as $\lambda x : A. b(x) : B$, where $x : A$ denotes that x is an object of type A . The correspondence then associates to the proposition $A \supset B$ the set $A \rightarrow B$ of functions from A to B , the elements of which can also be seen as proofs of the proposition $A \supset B$.

1.5. *Type theory as a programming language.* In this section we briefly expand on the view for which type theory can be thought of as a rather general and expressive programming language. For a monograph on type theory which takes this perspective see [25].

We observe that the link with programming languages which is characteristic of type theory makes it a particularly suitable foundation for constructive mathematics à la Bishop. In this variety of constructive mathematics, in fact, *intuitive* computability constitutes proof of existence. In his ‘constructive manifesto’ [6] Bishop writes:

Our program is simple: to give numerical meaning to as much as possible of classical abstract analysis. Our motivation is the well-known scandal, exposed by Brouwer (and others) in great detail, that classical mathematics is deficient in numerical meaning.

On a more practical ground, the link with programming has probably determined most of recent interest in type theory. The theory has in the last 20 years been implemented in various proof development systems. In particular, the system NuPRL (R. Constable et al., Cornell University, see for example [10] and [9]) implements a theory inspired by Martin–Löf type theory. The family of systems known under the acronym ALF (developed in Göteborg, see [4]) is directly motivated by constructive type theory.¹²

The correspondence between constructive theories and programming can be rapidly hinted at as follows. Let us consider a statement

¹² NuPRL abbreviates ‘New Proof Refinement Logic’, while ALF stands for ‘Another Logical Framework’. Alfa is presently the proof assistant of the ALF family, while Agda is its proof checker. The web pages of the two groups provide full information.

of the form:

$$(1) \quad \forall x \in A \exists y \in B P(x, y)$$

By Curry–Howard we can associate to it a set, whose elements are lambda terms representing proofs of the statement. More precisely, a proof of (1) is a function which to each element a of A assigns a proof $b(a)$ of $\exists y \in B P(a, y)$. In turn, $b(a)$ can be thought of as a pair whose first component, d , is an element of B (the witness of the existential quantifier) while the second component is a proof of $P(a, d)$. One can now read the lambda term as a program (for example by applying an automatic translation of it in the favourite programming language). Then (1) will be a *specification* for this program while A represents the set of inputs and B the set of outputs. It is important to note that by this process we not only obtain a lambda term (or program) from a proof of a statement, but we also gain a verification that the program does what it is expected to do. This has led to the idea of semi-automatising the process of proof as well as automatising the process of deriving the corresponding term (and consequently the program as written in a specific language). Ideally, in a correct system, programs obtained from proofs will have the advantage of not requiring any verification.

The descriptions and comparisons until now put forward of the notion of set in type theory and in other theories will become more perspicuous by the following concise account of the theory itself.

2. Propositions and Judgments

We are now heading to a presentation of the theory of sets of constructive type theory. We first of all expand on the observation we already made that in the general context of type theory the notion of set is only a particular kind of type, characterised by the fact that we can exhaust its elements by means of a finite number of rules. An example of set is the collection of all natural numbers. Its elements are inductively specified as follows: 0 is a natural number and every successor of a natural number is a natural number. In addition, we can establish when two natural numbers are equal: they are equal if they are either both 0 or successors of equal natural numbers.¹³ A type in general might not be amenable of such a description: the notion of type is much more vague and open than

¹³ \mathbf{N} is in fact a particular kind of set as its rules have an inductive character.

that of set. It amounts essentially to a collection furnished with an equivalence relation on it, by means of which we compare elements of the type. As an example of type, we can consider the collection of all sets. This is not itself a set, as we have in general no rule which allows us to describe when something is a set. The distinction between types and sets resembles that between classes and sets in axiomatic set theory.¹⁴

2.1. *Judgments.* Constructive type theory is formulated by a series of rules. A rule enables us to make a *judgment* usually from one or more hypothesis. By means of a judgment, on the other hand, we assert properties of sets or, by virtue of the Curry–Howard correspondence, of *propositions*. The distinction between judgments and propositions is of essential importance to the understanding of type theory. As an example, the judgments ‘*A set*’ and ‘*A prop*’ state the fact that the collection *A* is a set and that *A* is a proposition, respectively. A proposition is here the object of a judgment. It should be stressed that this method of defining a theory is quite different from the axiomatic approach. In the latter we single out some *propositions*, the axioms, and use them in conjunction with the derivation rules to obtain new propositions. Here we derive judgments which are made out of propositions. Furthermore, the same concept of proposition is open-ended, as the collection of propositions does not form a set but a type.

The definition of the theory by means of rules manipulating judgments seems to harmonise with Martin–Löf’s clarifying the semantics of the rules prior to introducing their syntax. The author always ensures the meaning of each new set or constructor is explained in full and hence justified, thus conforming to the intuitionistic conception of what constitutes mathematical activity.

2.2. *Categorical judgments.* We shall now introduce the 4 fundamental forms of categorical judgments, that is those judgments not depending on assumptions. In the next section we shall consider the corresponding hypothetical judgments. The categorical judgments are as follows:

- (a) *A set*, that is *A* is a set;
- (b) $A = B$, that is *A* and *B* are equal sets;

¹⁴ A presentation of a type theory with types as formal objects may be found in [25].

- (c) $a \in A$, that is a is an element of the set A ;
 (d) $a = b \in A$, that is a and b are equal elements of the set A .

Each form of judgment can be read in different ways. By Curry–Howard we may for example read them as assertions on propositions. The first and the third judgments can then be expressed as: ‘ A is a proposition’ (or simply A *prop*) and ‘ a is a proof of the proposition A ’. In agreement with the intuitionistic concept of truth, for which truth is provability, the judgment $a \in A$ can also be restated as ‘ A true’. The proposition A , in fact, is true provided that it has a proof; when viewed as a set, this means that it is inhabited or, equivalently, it has some element. Observe that we here deliberately omit the information on which element of A we take as witness of the truth of A . We nonetheless assume that we can at any time recover all information regarding the witness for a judgment of the form A true; in fact the justification of this form of judgment lies in this same possibility.

One can read categorical judgments also in a more intensional manner: a proposition could be interpreted as a problem we want to solve or as a task to perform.¹⁵ In this case $a \in A$ could be read as a is a method for solving problem A or for accomplishing task A .

Each form of judgment can be explained as follows.

(a) The first form of judgment allows us to state what constitutes a set in our theory. It answers the question: ‘*what is a set?*’ The answer to this question is very much in the style of Bishop: a set is determined by its elements and each set comes equipped with an equivalence relation which establishes which of its elements are equal. More precisely:

We define a set A by prescribing:

- (i) *how to form a canonical element of A ,*
 (ii) *how to form two equal canonical elements of A .*

Paradigmatic example is the set of natural numbers, \mathbf{N} . This is defined first of all by postulating the introduction rules:

$$\begin{array}{cc}
 0 \in \mathbf{N} & 0 = 0 \in \mathbf{N} \\
 \\
 \frac{a \in \mathbf{N}}{\text{suc}(a) \in \mathbf{N}} & \frac{a = b \in \mathbf{N}}{\text{suc}(a) = \text{suc}(b) \in \mathbf{N}}
 \end{array}$$

¹⁵ This is reminiscent of Kolmogorov’s description of propositions.

The rules specify the *canonical elements* of \mathbf{N} , that is the 0 and all the objects of the form $\text{suc}(a)$ for a in \mathbf{N} , and they also establish when two such elements are equal.

A set in general will contain other elements a part from the canonical ones. In the case of the natural numbers, the *non-canonical elements* are those elements of \mathbf{N} which can be reduced to its canonical elements by a computational or reduction process. For example $2 + 2$ represents a non-canonical element of \mathbf{N} . Its membership in \mathbf{N} is justified by the fact that applying the computational rules defining $+$ to $2 + 2$ one obtains a canonical element of \mathbf{N} .

The computation rules for operations like $+$ are introduced by the elimination rule for \mathbf{N} . Elimination rules also enable us to specify that no other canonical elements a part from those arising from the introduction rules are in the specific set.¹⁶

(b) For the second kind of judgment we notice first of all that we assume that A and B are sets, hence in fact we presuppose a judgment of the first kind. Then we state that $A = B$ whenever the two sets are extensionally equal, that is if for any $a \in A$ we also have $a \in B$, and *viceversa*. A set is described not only by defining its elements but also by establishing when two of its elements are equal. Therefore for A and B to be equal we also have to ensure that for any two elements a, b of A , if $a = b \in A$ then also $a = b \in B$, and *viceversa*. All this can be expressed in a more compact way by stating:

$A = B$ if and only if

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

The double line here indicates the possibility to go from the judgment above the line to the one below and *viceversa*.

(c) The third kind of judgment allows us to assert when an object is an element of a set. Martin–Löf explains this form of judgment in an evocative way by appealing to the alternative readings of the notion of set previously recalled.

¹⁶ Note that in the case of inductive sets like the natural numbers, the specific form of the elimination rules can have an impact on the proof-theoretic strength of the resulting theory.

An element a of a set A is a **method** or program which when executed produces a canonical element of A as its result.

(d) The third kind of judgment is explained as follows: Two elements a and b of a set A are equal if once executed they produce equal canonical elements of A as their result.

2.3. *Hypothetical judgments.* The forms of judgment introduced until now are independent from any assumption, they are categorical. We consider next hypothetical judgments. These are essential if we want to represent mathematical reasoning in type theory.

For simplicity, we shall start by considering the case of a particular form of judgment depending only on one hypothesis and generalise later on to the case of judgments depending on many hypothesis. This simple kind of judgement has the form:

$$B(x) \text{ set } (x \in A).$$

We assert that $B(x)$ is a set under the hypothesis $x \in A$. We also say that $B(x)$ is a *family of sets depending on A* . The judgment yields that $B(a)$ is a set whenever a is an element of A , and that if a and b are equal elements of A , then $B(a)$ and $B(b)$ are equal.

Note that for typographical reasons in the following we shall make use of two different notations for hypothetical judgments:

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

or

$$(x \in A)$$

$$B(x) \text{ set}$$

Let us now see the case of n premises.

(i) Assume the following judgments:

$$A_1 \text{ set},$$

$$A_2(x_1) \text{ set } (x_1 \in A_1),$$

$$A_3(x_1, x_2) \text{ set } (x_1 \in A_1, x_2 \in A_2(x_1)),$$

etc.

$$A_n(x_1, \dots, x_{n-1}) \text{ set}$$

under the hypothesis

$$(x_1 \in A_1, x_2 \in A_2(x_1), \dots, x_{n-1} \in A_{n-1}(x_1, \dots, x_{n-2})).$$

We can then formulate the new judgment:

$$A(x_1, \dots, x_n) \text{ set}$$

under the hypothesis:

$$(x_1 \in A_1, x_2 \in A_2(x_1), \dots, x_n \in A_n(x_1, \dots, x_{n-1})).$$

$A_2(x_1)$ set $(x_1 \in A_1)$ expresses the fact that $A_2(x_1)$ is a set under the hypothesis $x_1 \in A_1$. This means that $A_2(a)$ is a set whenever $a \in A_1$ and also that $A_2(a) = A_2(b)$ if $a = b \in A_1$. Similarly $A_3(x_1, x_2)$ set $(x_1 \in A_1, x_2 \in A_2(x_1))$ states the fact that $A_3(x_1, x_2)$ is a family of sets depending on the sets A_1 and $A_2(x_1)$. The other judgments can be justified in a similar way.

An implicit convention applies for which A_i may (but does not need to) depend from x_j with $j < i$, and it does not depend from x_i , and so also A may depend from x_i but not from x .

The n assumptions in a judgment constitute the *context*, often written Γ, Δ, \dots . Similarly to the case of natural deduction, also in type theory some of the rules allow for the discharge of assumptions (an assumption is discharged if it appears in the premise but not in the conclusion of a rule). Clearly care is required in respecting the order of the assumptions to be discharged, especially in the case of a rather complex context.

It should now be clear how to justify the following judgments (written in the second notation):

(ii)

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

$$A(x_1, \dots, x_n) = B(x_1, \dots, x_n).$$

(iii)

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

$$a(x_1, \dots, x_n) \in A(x_1, \dots, x_n).$$

(iv)

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

$$a(x_1, \dots, x_n) = b(x_1, \dots, x_n) \in A(x_1, \dots, x_n).$$

3. Assumption, equality and substitution rules

3.1. *Conventions.* Before introducing the rules which describe what amounts to a set and when a new set can be formed from given ones, we need to clarify the structure of the theory and introduce some rules of a syntactical nature, for example those regulating substitution. But first of all some conventions are in order, which are in use to facilitate the reading and the understanding of the rules.

- (i) An assumption which occurs both in the premise and in the conclusion of a rule is not written out.
- (ii) A discharged assumption which is stated in the premise may be unutilised in the actual derivation of a judgment.
- (iii) In case of rules with conclusion of the form $a \in A$ or $a = b \in A$, we only explicitly recall the assumptions which have the same form as the conclusion.

3.2. *Assumption rule.* This is the only rule which introduces new assumptions which do not already occur in the hypothesis. For clarity we shall write it in its complete form, without making use of the conventions just stated. It reads:

$$\frac{A \text{ set } (x_1 \in A_1, \dots, x_n \in A_n)}{x \in A \quad (x_1 \in A_1, \dots, x_n \in A_n, x \in A)}$$

3.3. *Note on equality.* In type theory there are three distinct notions of equality which should be carefully kept distinct. The first is at the level of judgements, the second is a way to postulate syntactical conventions and the third is at the level of propositions. We observe that the way the first and the third kind of equality interact has a strong impact on the properties of the resulting theory, in particular with respect to its intensionality or extensionality (see section 4.7).

- (1) *Judgemental equality.* We have already seen the equality which occurs in judgments. We can assert that:
 - (i) two sets are equal;
 - (ii) two elements in a set are equal.

- (2) *Definitional equality.* Definitional equality, here written $:=$, is utilised to postulate syntactical conventions or rewriting rules.
- (3) *Propositional equality.* This notion of equality allows us to reflect judgmental equality at the level of propositions. In section 4.7 we introduce a set $\mathbf{I}(A, a, b)$ which is inhabited whenever $a = b \in A$ is a valid judgment.

3.4. *Equality rules.* The equality rules are divided into rules for elements and rules for sets. They ensure that judgmental equality satisfies the usual properties of an equivalence relation: *reflexivity*, *symmetry* and *transitivity*. In the case of sets, we also postulate a rule for which equal sets have the same elements (and equal elements).

We believe these rules are particularly clear, therefore we only state them without any explanation (see Martin–Löf [23]).

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:

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

Equality rule for sets:

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

3.5. *Substitution rules.* The following substitution rules are justified by simply inspecting the meaning of hypothetical judgments as explained in section 2.3. They allow us to make substitutions in sets and in elements.

Substitution in sets:

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

Substitution in equal sets:

$$\frac{(x \in A) \quad a \in A \quad B(x) = D(x)}{B(a) = D(a)}$$

Substitution in elements:

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

Substitution in equal elements:

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

4. Rules for sets

4.1. *Kinds of rules.* We shall now present the four kinds of rules which allow us to introduce basic sets and to form new sets from given ones. Departing from the presentation in [23] we shall here define first of all the sets of natural numbers and of finite sets as they are very perspicuous. We shall subsequently define new sets from given ones by means of fundamental operations like product, sum, etc.

All the rules we shall introduce have a common structure, as they fall under one of the following four kinds:

- (i) Formation
- (ii) Introduction
- (iii) Elimination
- (iv) Equality

(i) The first kind of rules tells us when we can form a new set, often from given ones or from a family of sets. It also establishes when we can form equal sets. Alternatively, these rules can be seen as describing how to obtain new propositions (from given ones or from propositional functions) and how to compare propositions.

(ii) The second kind of rules enables us to establish which are the canonical elements of a set and when they are equal. As we have already seen, a set is defined by postulating which canonical elements belong to it and when they are equal. These rules thus assign a meaning to the new set. They sometimes introduce new constants, like for example 0, and usually describe the behaviour of new constructors, like *suc*, λ , etc.

(iii) The third kind of rules allows us to make use of the elements of the set. The elimination rules often constitute a form of structural induction: to establish that a property holds of an arbitrary element of a set, it is enough to prove that it inductively holds of the canonical elements of the same set. This kind of rule introduces a new elimination constant (or destructor): when applied to arguments representing the validity of the inductive hypothesis, it allows us to obtain a proof of a proposition which expresses the validity of the given property for an arbitrary element. In the case of inductive types like \mathbf{N} , the elimination rule also provides recursion (as a

proof or element of the induction principle expressed by the elimination rule). They therefore explain how to ‘compute’ non-canonical elements.

(iv) The last kind of rules relates introduction and elimination rules by showing how an elimination constant operates on *canonical* elements of the set.

4.2. *Natural numbers.* The existence of a completed set of natural numbers is rarely disputed by mathematicians, logicians or philosophers of mathematics. Predicativists like Poincaré or Weyl assumed its existence. The existence of a set of natural numbers is in particular asserted by the intuitionists. It is indeed the most paradigmatic example of infinite set which emerges in the realm of mathematics. Doubtlessly, for the intuitionist the fundamental characteristic of mathematics is having infinity as its object. The departure from classical logic is determined exactly by the fact that mathematics deals with the infinity; classical logic is on the contrary modeled after finite domains and is therefore only adequate for them.

In type theory the set of natural numbers, \mathbf{N} , is inductively defined as follows: we first establish that 0 is a natural number and then that for each natural number a its successor, $suc(a)$, is also a natural number. Simultaneously, we establish when two elements of \mathbf{N} are equal.

Given the set of natural numbers, we clearly aim at utilising it for arithmetical computations. We then need to define the ordinary operations of addition, multiplication etc on natural numbers. For this purpose we introduce a new constant, \mathbf{R} , in the elimination rule and uniformly define by means of it all primitive recursive functions. The specific behaviour of the ‘destructor’ \mathbf{R} will be clarified only after introducing the elimination rule.

In the case of \mathbf{N} we shall indicate the rules in full, that is, also with their corresponding rules for equality. The latter will instead be omitted for the other sets (with the only exception of the product).¹⁷

(i) \mathbf{N} -Formation

$$\mathbf{N} \text{ set} \qquad \mathbf{N} = \mathbf{N}$$

¹⁷ See for example Beeson [5] for a schematic summary of all rules.

(ii) **N-Introduction**

$$0 \in \mathbf{N} \qquad 0 = 0 \in \mathbf{N}$$

$$\frac{a \in \mathbf{N}}{\text{suc}(a) \in \mathbf{N}} \qquad \frac{a = b \in \mathbf{N}}{\text{suc}(a) = \text{suc}(b) \in \mathbf{N}}$$

(iii) **N-Elimination**

$$\frac{\begin{array}{c} (x \in \mathbf{N}, y \in C(x)) \\ c \in \mathbf{N} \quad d \in C(0) \quad e(x, y) \in C(\text{suc}(x)) \end{array}}{\mathbf{R}(c, d, e) \in C(c)}$$

$$\frac{\begin{array}{c} (x \in \mathbf{N}, y \in C(x)) \\ c = f \in \mathbf{N} \quad d = g \in C(0) \quad e(x, y) = h(x, y) \in C(\text{suc}(x)) \end{array}}{\mathbf{R}(c, d, e) = \mathbf{R}(f, g, h) \in C(c)}$$

(iv) **N-Equality**

$$\frac{\begin{array}{c} (x \in \mathbf{N}, y \in C(x)) \\ d \in C(0) \quad e(x, y) \in C(\text{suc}(x)) \end{array}}{\mathbf{R}(0, d, e) = d \in C(0)}$$

$$\frac{\begin{array}{c} (x \in \mathbf{N}, y \in C(x)) \\ a \in \mathbf{N} \quad d \in C(0) \quad e(x, y) \in C(\text{suc}(x)) \end{array}}{\mathbf{R}(\text{suc}(a), d, e) = e(a, \mathbf{R}(a, d, e)) \in C(\text{suc}(a))}$$

In the last two rules $C(z)$ is a family of sets depending on \mathbf{N} .

The only rule which requires explanation is the elimination rule. It can be clarified as follows.

Given an arbitrary element, c , of \mathbf{N} (that is an element which is possibly non-canonical), we can read $C(c)$ as a proposition for which we require a proof. The rule then enables us to prove $C(c)$ by induction. We form first of all a proof d of $C(0)$. Then provided that for $x \in \mathbf{N}$, y is a proof of $C(x)$, we give a proof $e(x, y)$ of $C(\text{suc}(x))$.

The rule gives us a proof $\mathbf{R}(c, d, e)$ (depending from c , d and e), of the proposition $C(c)$.

More precisely, $\mathbf{R}(c, d, e)$ is computed as follows.

- (1) Take an arbitrary element, c , of \mathbf{N} and compute its canonical value;
- (2) if the result of the computation is $c = 0 \in \mathbf{N}$, then compute $d \in C(0)$, hence obtaining a new canonical element f of $C(0)$. Note that $c = 0 \in \mathbf{N}$, so that $C(c) = C(0)$ and so f will be a canonical element also of $C(c)$;
- (3) if instead the computation produces an element of \mathbf{N} of the form $\text{suc}(a)$ for $a \in \mathbf{N}$, then proceed as follows: substitute a for x and $\mathbf{R}(a, d, e)$ for y in e , hence obtaining $e(a, \mathbf{R}(a, d, e)) \in C(\text{suc}(a))$. Note that $C(\text{suc}(a)) = C(c)$, so that $e(a, \mathbf{R}(a, d, e)) \in C(c)$. Compute the latter, thus obtaining a canonical element g , of $C(c)$;
- (4) if a has value 0, then $\mathbf{R}(a, d, e) \in C(c)$ by (2); otherwise proceed again as in (3).

As an example we define the following operations of predecessor, addition and multiplication, respectively.

$$\text{pred}(x) := \mathbf{R}(x, 0, (x, y).x),$$

$$+(x, y) := \mathbf{R}(x, y, (z, w).\text{suc}(w)),$$

$$*(x, y) := \mathbf{R}(x, 0, (z, w).+(y, w)).$$

We write $(x, y).t$ to denote that x, y are the substitution variables from t . Parenthesis are here used in a similar way to the usual λ notation from lambda calculus. We use parenthesis because λ is a reserved character in type theory representing a primitive constant which characterises canonical elements of the cartesian product.

4.3. Finite sets. We now aim at introducing finite sets. For each natural number k we define a set \mathbf{N}_k , which has k elements. In the usual set-theoretic notation, this would be represented as the set $\{0_k, 1_k, \dots, (k-1)_k\}$. Note however that the objects of \mathbf{N}_k do not coincide with the first k elements of \mathbf{N} : for example, 0 is to be considered as distinct from 0_k .

For each natural number k , we have the 4 rules of \mathbf{N}_k -formation, \mathbf{N}_k -introduction, \mathbf{N}_k -elimination and \mathbf{N}_k -equality (each with its corresponding equality rule).¹⁸

(i) \mathbf{N}_k -Formation

$$\mathbf{N}_k \text{ set}$$

(ii) \mathbf{N}_k -Introduction

$$m_k \in \mathbf{N}_k \quad (m = 0, 1, \dots, k - 1)$$

(iii) \mathbf{N}_k -Elimination

$$\frac{c \in \mathbf{N}_k \quad c_m \in C(m_k) \quad (m = 0, 1, \dots, k - 1)}{\mathbf{R}_k(c, c_0, \dots, c_{k-1}) \in C(c)}$$

(iv) \mathbf{N}_k -Equality

$$\frac{c_m \in C(m_k) \quad (m = 0, 1, \dots, k - 1)}{\mathbf{R}_k(m_k, c_0, \dots, c_{k-1}) = c_m \in C(m_k)}$$

Once more $C(x)$ ($x \in \mathbf{N}_k$) is a family of sets and can also be seen as a propositional function asserting a property of elements of \mathbf{N}_k . \mathbf{R}_k represents a form of definition by cases.

We can explain the elimination rule as follows.

- (1) Execute c , so to obtain as result a canonical element m_k of \mathbf{N}_k with $(m = 0, 1, \dots, k - 1)$;
- (2) choose the corresponding element, c_m , of $C(m_k)$ and execute it. One obtains as a result a canonical element $d \in C(m_k)$. Since $c = m_k \in \mathbf{N}_k$, we have that $C(c) = C(m_k)$ and so $d \in C(c)$.

Note that the set \mathbf{N}_0 represents the empty set, as it has no elements. In this case the introduction rule is void. The elimination rule becomes instead:

¹⁸ The presence of an infinite number of rules like in the case of the finite sets can be problematic when defining for example an elimination rule for a reflecting universe (see section 4.12). One can in this case rather introduce only two sets \mathbf{N}_0 and \mathbf{N}_1 , and utilise the disjoint sum, $+$, to produce any finite set.

\mathbf{N}_0 -Elimination

$$\frac{c \in \mathbf{N}_0}{\mathbf{R}_0(c) \in C(c)}$$

The set \mathbf{N}_1 has instead only one element, namely 0_1 .¹⁹

4.3.1. *True, false, booleans.* If looking at finite sets as propositions, then \mathbf{N}_0 coincides with falsity, not possessing any element (that is, proof). It is also indicated by \perp . The set \mathbf{N}_1 coincides with truth, being always inhabited by a canonical element, 0_1 . The set \mathbf{N}_2 , whose elements are 0_2 and 1_2 , may be seen as representing the booleans, which are ubiquitous in programming languages.

When one reads \mathbf{N}_0 as \perp , the elimination rule becomes the well known natural deduction rule *ex falso quodlibet*:

\perp -Elimination

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

Note that in this case we are free to eliminate the information concerning the witness both in the premise and in the conclusion of the rule.

As an example of how to utilise type-theoretic constructs to define notions commonly used in programming languages, one can see that $\mathbf{R}_2(c, d, f)$ allows us to represent ‘if c then d else f ’.

The rules presented until now allow us to introduce new sets like \mathbf{N} or \mathbf{N}_k (for any k). We shall now see how to introduce new sets from given ones or from families of sets. From a purely syntactical point of view, the following introduction rules have premises.

4.4. *Generalised (cartesian) product.* Given a set A and a family of sets $B(x) (x \in A)$, we aim at forming a new set whose elements represent functions with arguments in A and values in the given family of sets.

We observe that the concept of function here represented is more general than the ordinary one between sets A and B . A function is here a correspondence which takes an element a of A to an element $b(a)$ of a set $B(a)$ depending from a . The codomain of the function, in other terms, is not a fixed set but it depends from the specific argument of the function. This is the first example we encounter of

¹⁹ This needs to be proved. Since it requires constructors, like propositional identity, yet not introduced, we omit its proof. The reader may wish to consult [23].

a dependent type. The importance of dependent types lies in the expressiveness they confer to the theory without for this enforcing the step into impredicativity. As an example, the generalised cartesian product allows us to express universal quantification at the level of propositions.

The canonical elements of $\Pi(A, B)$ are functions obtained by λ abstraction: from an element $b(x)$ of $B(x)$ we abstract b and we apply to it λ thus obtaining $\lambda(b)$, which is a canonical element of the product. The basic idea here is that $b(x) \in B(x)(x \in A)$ is a function of which $\lambda(b)$ is a name or representative.

Equality for canonical elements of the product is derived from equality in the family of sets $B(x)(x \in A)$. In fact, two canonical elements $\lambda(b)$ and $\lambda(d)$ are equal if for $x \in A$ we have $b(x) = d(x) \in B(x)$ (equality rule for Π -introduction). The notion of function is hence *extensional*.

By $\Pi(A, B)$ or $(\Pi x \in A)B(x)$ we denote the product of A times the family B (depending on A), and by $\lambda(b)$ a canonical element of the product. An explanation of the use of the elimination constant **Ap** will be given after introducing all the rules.

In the case of the product, to increase insight in the theory and single out extensionality, we recall also the rules for equality.

(i) Π - Formation

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

(ii) Π - Introduction

$$\frac{(x \in A) \quad b(x) \in B(x)}{\lambda(b) \in \Pi(A, B)} \qquad \frac{(x \in A) \quad b(x) = d(x) \in B(x)}{\lambda(b) = \lambda(d) \in \Pi(A, B)}$$

(iii) Π - Elimination

$$\frac{c \in \Pi(A, B) \quad a \in A}{\mathbf{Ap}(c, a) \in B(a)} \qquad \frac{c = f \in \Pi(A, B) \quad a = d \in A}{\mathbf{Ap}(c, a) = \mathbf{Ap}(f, d) \in B(a)}$$

(iv) Π - Equality

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

We can now explain the use of the elimination constant \mathbf{Ap} .

In general, $\mathbf{Ap}(c, a)$ is a method which allows us to obtain elements of $B(a)$. It can be used as follows.

- (1) Calculate $c \in \Pi(A, B)$, obtaining as a result a canonical element, $\lambda(b)$, for $b(x) \in B(x)(x \in A)$;
- (2) given $a \in A$ substitute it for x in b . We thus obtain $b(a) \in B(a)$;
- (3) calculate $b(a)$, to produce a canonical element of $B(a)$.

The elimination rule clearly represents an analogue of β -reduction in the lambda calculus. Alternatively, one could opt for an elimination rule representing a structural induction principle (see for example [25] pages 51 and 52).

4.4.1. *Universal quantifier.* The rules for the product may also be read as referring to propositions, in which case they represent the usual natural deduction rules for the universal quantifier. In the following, obvious restrictions on the free variables apply.

 \forall - Formation

$$\frac{(x \in A) \quad \frac{A \text{ set} \quad B(x) \text{ prop}}{(\forall x \in A)B(x) \text{ prop}}}{(\forall x \in A)B(x) \text{ prop}}$$

Note that the judgment $A \text{ set}$ is unchanged in the propositional reading, as A represents the domain of quantification.

 \forall - Introduction

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

\forall - Elimination

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

As in the propositional reading of the Π -rules the witnesses are not explicitly presented, there is here no corresponding rule for equality.

4.4.2. *Function space and implication.* We obtain a particular case of the product $\Pi(A, B)$ when the family B does not depend on the set A . The product then represents the function space from the set A to the set B , and is denoted by $A \rightarrow B$ or B^A .

It is important to remark here a difference between axiomatic set theory and type theory. The notion of function is in fact represented in Zermelo–Fraenkel set theory by that of ordered pair. Here it is instead a primitive notion not reducible to that of set.

When reading A and B as propositions, $A \rightarrow B$ represents the implication $A \supset B$. We here recall the derived rules.

\supset - Formation

$$\frac{(A \text{ true}) \quad (A \text{ prop}) \quad (B \text{ prop})}{A \supset B \text{ prop}}$$

\supset - Introduction

$$\frac{(A \text{ true}) \quad (B \text{ true})}{A \supset B \text{ true}}$$

\supset - Elimination

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

Note that the introduction and elimination rules for \supset correspond exactly to the respective rules of natural deduction. The formation rule has the peculiarity of allowing us to make use of the premise $A \text{ true}$ to prove that B is a proposition.

4.4.3. *Negation.* As in the usual intuitionistic calculus, negation is defined in terms of \supset and \perp . We let $\neg A := A \supset \perp$.

4.5. *Generalised sum.* Given a set A and a family $B(x)(x \in A)$ depending on A , we aim at forming a set whose elements are pairs, the first component of which is an element of A and the second an element of B . In other terms we aim at a generalisation of the usual cartesian product to the case in which B is not just a set but a family of sets. This will be represented by an operator Σ applied to A and the family B , written $\Sigma(A, B)$ or $(\Sigma x \in A)B(x)$.

(i) Σ - Formation

$$\frac{(x \in A) \quad \begin{array}{c} A \text{ set} \quad B(x) \text{ set} \end{array}}{\Sigma(A, B) \text{ set}}$$

(ii) Σ - Introduction

$$\frac{a \in A \quad b \in B(a)}{\langle a, b \rangle \in \Sigma(A, B)}$$

(iii) Σ - Elimination

$$\frac{(x \in A, y \in B(x)) \quad \begin{array}{c} c \in \Sigma(A, B) \quad d(x, y) \in C(\langle x, y \rangle) \end{array}}{\mathbf{E}(c, d) \in C(c)}$$

(iv) Σ - Equality

$$\frac{(x \in A, y \in B(x)) \quad \begin{array}{c} a \in A \quad b \in B(a) \quad d(x, y) \in C(\langle x, y \rangle) \end{array}}{\mathbf{E}(\langle a, b \rangle, d) = d(a, b) \in C(\langle a, b \rangle)}$$

In the last two rules C is a family of sets depending from $\Sigma(A, B)$.

We can explain $\mathbf{E}(c, d)$ as follows.

- (i) Execute c so to obtain a canonical element $\langle a, b \rangle$, of $\Sigma(A, B)$;
- (ii) substitute $a \in A$ to x and $b \in B(a)$ to y in the second premise, so to obtain $d(a, b) \in C(\langle a, b \rangle)$;
- (iii) execute $d(a, b)$ to produce a canonical element e of $C(\langle a, b \rangle)$. Note that $c = \langle a, b \rangle \in \Sigma(A, B)$, hence $C(c) = C(\langle a, b \rangle)$, and so e is a canonical element of $C(c)$.

4.5.1. *Existential quantifier.* The propositional mode yields the rules for the existential quantifier.

\exists - Formation

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

\exists - Introduction

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

\exists - Elimination

$$\frac{(x \in A, B(x) \text{ true}) \quad \frac{(\exists x \in A)B(x) \text{ true} \quad C \text{ true}}{C \text{ true}}}{C \text{ true}}$$

4.5.2. *Cartesian product and conjunction.* Whenever B does not depend on A we obtain as a particular case of $\Sigma(A, B)$ the usual cartesian product of two sets. The corresponding propositional reading instead gives us $\&$.

$\&$ - Formation

$$\frac{(A \text{ true}) \quad \frac{A \text{ prop} \quad B \text{ prop}}{A\&B \text{ prop}}}{A\&B \text{ prop}}$$

& - Introduction

$$\frac{A \text{ true} \quad B \text{ true}}{A \& B \text{ true}}$$

& - Elimination

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

The formation rule for & follows the pattern already seen for implication, as it allows us to derive $B \text{ prop}$ from the judgment $A \text{ true}$. The introduction rule is exactly the rule usually found in natural deduction. The elimination rule, instead, is a generalisation of the two natural deduction rules and it elegantly comprehends both of them. The usual rules can in fact be easily obtained from this rule by instantiating C with A or B , respectively.

4.6. *Disjunct sum of two sets.* We now aim at representing the disjunct sum of two sets. Note that also in this case one could opt for a non primitive sum, defined by Σ . We nonetheless prefer to take $+$ as primitive, also taking into account the fact that in its propositional interpretation it represents disjunction, usually considered as primitive in intuitionistic systems.

Following Martin–Löf we use an infix notation for $+$.

The canonical elements of $A+B$ have the form $\mathbf{inl}(a)$ for $a \in A$ or the form $\mathbf{inr}(b)$ for $b \in B$; the constants \mathbf{inl} and \mathbf{inr} indicate if an element of $A+B$ is justified by an element in A or in B . Note that given a non-canonical element c of $A+B$, we can establish if c is element of A or of B , as by calculating it one obtains a canonical element of the form $\mathbf{inl}(a)$ or $\mathbf{inr}(b)$.

(i) $+$ - Formation

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

(ii) + - Introduction

$$\frac{a \in A}{\mathbf{inl}(a) \in A+B} \qquad \frac{b \in B}{\mathbf{inr}(b) \in A+B}$$

(iii) + - Elimination

$$\frac{\begin{array}{ccc} (x \in A) & & (y \in B) \\ c \in A+B & d(x) \in C(\mathbf{inl}(x)) & e(y) \in C(\mathbf{inr}(y)) \end{array}}{\mathbf{D}(c, d, e) \in C(c)}$$

(iv) + - Equality

$$\frac{\begin{array}{ccc} (x \in A) & & (y \in B) \\ a \in A & d(x) \in C(\mathbf{inl}(x)) & e(y) \in C(\mathbf{inr}(y)) \end{array}}{\mathbf{D}(\mathbf{inl}(a), d, e) = d(a) \in C(\mathbf{inl}(a))}$$

$$\frac{\begin{array}{ccc} (x \in A) & & (y \in B) \\ b \in A & d(x) \in C(\mathbf{inl}(x)) & e(y) \in C(\mathbf{inr}(y)) \end{array}}{\mathbf{D}(\mathbf{inr}(b), d, e) = e(b) \in C(\mathbf{inr}(b))}$$

In the last rules C is a family of sets depending from $A+B$.

We explain $\mathbf{D}(c, d, e)$ as follows.

- (i) Calculate $c \in A+B$, so to obtain as a value a canonical element of $A+B$ of the form $\mathbf{inl}(a)$ for $a \in A$ or $\mathbf{inr}(b)$ for $b \in B$;
- (ii) in the first case substitute a for x in $d(x)$, so to obtain $d(a)$ and calculate it. The second premise allows us to conclude that $d(a) \in C(\mathbf{inl}(a))$, thus calculating $d(a)$ we obtain a canonical element of $C(\mathbf{inl}(a))$;
- (iii) in the second case substitute b to y in $e(y)$ and obtain $e(b) \in C(\mathbf{inr}(b))$. Calculate its value so to produce a canonical element of $C(\mathbf{inr}(b))$;
- (iv) clearly in both cases we produce a canonical element of $C(c)$. In fact, if $c = \mathbf{inl}(a) \in A+B$, then $C(c) = C(\mathbf{inl}(a))$, while if $c = \mathbf{inr}(b) \in A+B$, then $C(c) = C(\mathbf{inr}(b))$.

4.6.1. *Disjunction.* The propositional interpretation of $+$ is disjunction. We here recall its rules, which faithfully reproduce the natural deduction's rules.

\vee - Formation

$$\frac{A \text{ prop} \quad B \text{ prop}}{A \vee B \text{ prop}}$$

\vee - Introduction

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

\vee - Elimination

$$\frac{A \vee B \text{ true} \quad \begin{array}{l} A \text{ true} \quad B \text{ true} \\ C \text{ true} \quad C \text{ true} \end{array}}{C \text{ true}}$$

4.7. *Propositional equality.* As already observed there are three distinct notions of equality in type theory: judgemental equality, definitional equality and propositional equality. The latter is introduced in type theory to reflect, at the propositional level, judgments asserting the equality of two elements of a set.

The rules for propositional equality might be formulated in different ways depending on whether we intend to capture an intensional or an extensional notion of equality. The question is relevant as the two kinds of theories represent different perspectives and are aimed at accomplishing distinct purposes. Martin–Löf has presented intensional (see for example [21]) as well as extensional versions of type theory. The book [23] presents an extensional theory and we shall here take the same perspective. Extensional type theory is clearly an appropriate and elegant setting for developing constructive mathematics. Nonetheless, if one is interested for example in the theory as a programming language, then the intensional approach is advisable, since extensionality brings into the theory an undecidability of the membership relation which affects type checking.²⁰

²⁰ See for example [25] for a treatment of both notions.

(i) **I** - Formation

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

(ii) **I** - Introduction

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

(iii) **I** - Elimination

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

(iv) **I** - Equality

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

Observe that in the **I**-introduction rule the element r does not depend on a, b and A . This is a characteristic of extensional equality. In the case of intensional rules one would instead introduce canonical elements depending on a, b and A . Another characteristic of the extensional elimination rule is that it allows us to move from the propositional level back into the judgmental through the elimination rule.

As in the case of the product, also here we could introduce an elimination rule in the form of a structural induction principle and derive from it the present rule (see for example [25]).

We recall that propositional equality introduces families of sets depending from a given set. It is clearly useful in proofs of uniqueness, as can be seen for example in the case of the uniqueness of the element 0_1 of $\mathbf{N}_{\mathbf{k}}$.

4.8. *Adequacy for constructive mathematics.* Once we have developed the basics of constructive type theory it is quite natural to question whether it indeed provides us with a foundation for constructive mathematics. A detailed answer to this question would bring us too far. We simply give a very basic example of how to formalise the fundamental notion of real number (à la Cauchy) in

type theory. We first briefly discuss the status of the axiom of choice in type theory.

4.9. *The axiom of choice.* The axiom of choice, **AC**, is one of the most controversial principles formulated in the context of set theory.

In the case of classical Zermelo–Fraenkel set theory **AC** is well known (by fundamental work of Gödel and Cohen) to be consistent with but independent from the other axioms of set theory.

In the case of constructive versions of Zermelo–Fraenkel set theory the combination of the axiom of extensionality with the other principles of the theory produce the derivability of constructively unacceptable instances of excluded middle.

It might then be surprising that the axiom of choice does hold in the case of constructive type theory. This is to be ascribed to the meaning the quantifiers have in type theory (Curry–Howard correspondence) and to the fact that even in its extensional versions type theory has an intensional character. Extensionality is, so to speak, introduced on top of an intensional theory. The persistence of an intensional character of the theory can be seen in the fact that the elements of a set can not be characterised independently from the set they belong to and always carry with them information on equality for the set they belong to.

The axiom of choice is the following principle.

$$\begin{aligned} &(\forall x \in A)(\exists y \in B(x))C(x, y) \\ &\supset (\exists f \in \Pi(A, B))(\forall x \in A)C(x, \mathbf{A}p(f, x)) \text{ true.} \end{aligned}$$

It is not difficult to see that this is indeed derivable in type theory. A full proof can be found in [23].²¹

4.10. *Separation.* One of the essential constructions in set theory is that of separation: given a set A we can form a subset of it whose elements are exactly those elements of the set which satisfy

²¹ We observe that though the full axiom of choice is not compatible with constructive Zermelo–Fraenkel set theory, countable choice, which is its instance in the case in which A is the set of natural numbers, is acceptable. Similarly the more general dependent choice principle is acceptable also in extensional contexts. These two principles have been extensively used by the Bishop school. We briefly also notice that if countable choice is available, then conceptually distinct notions of real number as that of Cauchy and Dedekind reals coincide (see for example Troelstra and van Dalen [29] for an exposition of the two notions of set). On the other end, the theory of Cauchy reals becomes particularly smooth in the presence of countable choice.

a certain property B . The resulting set is usually denoted in the intuitive curly bracket notation by $\{x \in A : B(x)\}$.

In type theory we can represent separation by use of Σ , by taking $\Sigma(A, B)$, where A is a set and B a propositional function depending on A . The representation of $\{x \in A : B(x)\}$ is then a set of pairs the first component of which is an element, a , of A , and the second component is a proof, $b(a)$, that $B(a)$ holds.

4.11. *Cauchy reals.* A paradigmatic example of the use of separation is the following definition by Martin–Löf of the Cauchy reals:

$$R := (\Sigma x \in \mathbf{N} \rightarrow \mathbf{Q})\text{Cauchy}(x)$$

where for $a \in \mathbf{N} \rightarrow \mathbf{Q}$,

$$\begin{aligned} \text{Cauchy}(a) := & (\forall r \in \mathbf{Q})(r > 0 \supset (\exists m \in \mathbf{N})(\forall n \in \mathbf{N}) \\ & (|a_{m+n} - a_m| \leq r)). \end{aligned}$$

4.12. *A reflection principle for type theory: the first universe.* In the foregoing we have introduced a set theory in which new sets are constructed starting from the sets \mathbf{N} of natural numbers and \mathbf{N}_k of finite sets, by successively applying the type constructors \mathbf{I} , Σ , Π etc.. This process gives rise to a finite type structure. The theory so obtained is often indicated by \mathbf{ML} or \mathbf{ML}_0 .

We now extend this structure to the infinite. One way of achieving this is by thinking of the collection of all the sets built until now as forming a new object of the extended theory. This is suggestively named *universe* and indicated by \mathbf{U} . Intuitively, the idea is that the universe is a new type of a higher level which contains as element a code for each set of the previous level. We observe that in order to avoid paradoxical situations, the universe can not be considered to be a set in the sense until now presented; that is, it can not be taken to belong to the same level as the sets whose codes it contains. It will be called a type or a ‘large set’, while the elements of the universe will now be called ‘small sets’. The resulting new theory is usually indicated by \mathbf{ML}_1 .

There are two distinct ways of presenting the universe. In a more informal treatment, named by Martin–Löf *à la Russell* due to similarities with ramified type theory, the universe contains as elements the sets until now introduced. In the second, more precise and preferable presentation, the universe contains codes for the sets.

It is therefore introduced together with a decoding function which assigns sets to codes in the universe. This second presentation is named *à la Tarski*, as the decoding function has similarities with Tarski's truth predicate.

From a logical point of view, the closure under the universe of sets may be seen as a reflection principle, that is as a principle which allows us to treat propositions of the theory as first class objects.

Note that the universe may be expressed without introducing an elimination rule, thus leaving its concept 'open'. An elimination rule in the form of a structural induction principle would indeed have the effect of fixing once and for all the basic types and also the constructors which are allowed in forming new sets from given ones. In the present case we do not need an elimination rule so that we formulate \mathbf{U} without specifying its corresponding elimination.

As we shall see in the second part, a universe reflecting propositions is needed for the interpretation of \mathbf{CZF} in type theory.

We now recall the rules for the universe 'Tarski style'. In the rules we shall make use of a decoding function \mathbf{T} as well as constants $\hat{\Pi}$, $\hat{\Sigma}$ etc. which represent codes for the corresponding operators.

- \mathbf{U} - Formation

$$\mathbf{U} \text{ set} \quad \frac{A \in \mathbf{U}}{\mathbf{T}A \text{ set}}$$

- \mathbf{U} - Introduction

$$\frac{(x \in \mathbf{T}A) \quad A \in \mathbf{U} \quad B(x) \in \mathbf{U}}{\hat{\Pi}(A, B) \in \mathbf{U}} \quad \frac{(x \in \mathbf{T}A) \quad A \in \mathbf{U} \quad B(x) \in \mathbf{U}}{\mathbf{T}(\hat{\Pi}(A, B)) = \Pi(\mathbf{T}A, \mathbf{T}B)}$$

$$\frac{(x \in \mathbf{T}A) \quad A \in \mathbf{U} \quad B(x) \in \mathbf{U}}{\hat{\Sigma}(A, B) \in \mathbf{U}} \quad \frac{(x \in \mathbf{T}A) \quad A \in \mathbf{U} \quad B(x) \in \mathbf{U}}{\mathbf{T}(\hat{\Sigma}(A, B)) = \Sigma(\mathbf{T}A, \mathbf{T}B)}$$

$$\frac{A \in \mathbf{U} \quad B \in \mathbf{U}}{A \hat{+} B \in \mathbf{U}} \quad \frac{A \in \mathbf{U} \quad B \in \mathbf{U}}{\mathbf{T}(A \hat{+} B) = \mathbf{T}A + \mathbf{T}B}$$

$$\frac{A \in \mathbf{U} \quad x, y \in \mathbf{TA}}{\hat{\mathbf{I}}(A, x, y) \in \mathbf{U}} \qquad \frac{A \in \mathbf{U} \quad x, y \in \mathbf{TA}}{\mathbf{T}(\hat{\mathbf{I}}(A, x, y)) = \mathbf{I}(\mathbf{TA}, x, y)}$$

$$\begin{array}{ccccccc} \hat{\mathbf{N}}_0 \in \mathbf{U} & \hat{\mathbf{N}}_1 \in \mathbf{U} & \dots & \mathbf{T}\hat{\mathbf{N}}_0 = \mathbf{N}_0 & \mathbf{T}\hat{\mathbf{N}}_1 = \mathbf{N}_1 & & \\ & & & \hat{\mathbf{N}} \in \mathbf{U} & \mathbf{T}\hat{\mathbf{N}} = \mathbf{N} & & \end{array}$$

Once the universe has been introduced type theory can express constructs of a higher level.

An interesting exercise is to show that the fourth Peano axiom, which is not derivable in type theory without universes, is now derivable in \mathbf{ML}_1 (see [27]).

4.13. *The hierarchy of universes.* We may repeat the operation of construction of the universe and build a new universe of a higher level. The new universe may be thought of as containing not only codes for the small sets but also a code for the first universe. This will now constitute a stronger reflection principle. This same procedure may be iterated further on, to obtain a hierarchy of universes: $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3, \dots$, each containing a copy of the basic sets as well as copies of all previous universes. Clearly we can not postulate a universe containing itself as an element, as this would produce a form of paradox. On the contrary each time we construct a new universe we need to make sure we ‘go up one level’ with respect to its elements.

We shall see in the second part how the first universe is utilised to construct on top of it a new large type which represents within type theory the iterative universe of sets of constructive Zermelo–Fraenkel set theory. The new type is a particular instance of application of a general construct for inductive types, called \mathbf{W} , which we shall now present.

4.14. *Inductive types or wellorderings.* The type constructor \mathbf{W} has rather complex rules, and might considerably increase the expressiveness as well as the strength of a theory when it is added to it. It provides means for expressing inductive definitions and inductive data types in constructive type theory.

In order to intuitively understand the action of this type constructor, we now consider a particular case of it, which arises by applying the constructor to the three elements set \mathbf{N}_3 . This gives

a representation of the second number class (that is the class of countable ordinals).

Let us see how the second number class could be defined in the context of type theory. After the rules for **W** are introduced, it should become clear how to rephrase the following rules in terms of **W** and \mathbf{N}_3 .

We need first of all to introduce a set, \mathcal{O} , and its canonical elements. These can be defined as follows:

- (i) 0 is in \mathcal{O} ;
- (ii) if α is in \mathcal{O} , then its successor α' is in \mathcal{O} ;
- (iii) for a sequence of ordinals $\alpha_0, \alpha_1, \dots$ in \mathcal{O} , we can form a new ordinal $\sup_n(\alpha_n)$, intuitively representing the least ordinal greater than each element in the sequence.

The elimination rule for the set \mathcal{O} will then be a rule expressing transfinite induction over the set.

- \mathcal{O} - Formation

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

- \mathcal{O} - Introduction

$$0 \in \mathcal{O} \qquad \frac{\alpha \in \mathcal{O}}{\alpha' \in \mathcal{O}} \qquad \frac{f \in \mathbf{N} \rightarrow \mathcal{O}}{\sup f \in \mathcal{O}}$$

- \mathcal{O} - Elimination

$$\frac{\begin{array}{c} (x \in \mathcal{O}, y \in C(x)) \qquad (z \in \mathbf{N} \rightarrow \mathcal{O}, \\ w \in \Pi(\mathbf{N}, (n).C(Ap(z, n)))) \\ c \in \mathcal{O} \quad d \in C(0) \quad e(x, y) \in C(x') \quad f(z, w) \in C(\sup(z)) \end{array}}{\mathbf{R}_{\mathcal{O}}(c, d, e, f) \in C(c)}$$

We can now generalise this construction to the case of an arbitrary set A and a family B of sets depending from A as follows.

- **W** - Formation

$$\frac{\begin{array}{c} (x \in A) \\ A \text{ set} \quad B(x) \text{ set} \end{array}}{\mathbf{W}(A, B) \text{ set}}$$

- **W** - Introduction

$$\frac{a \in A \quad b \in B(a) \rightarrow \mathbf{W}(A, B)}{\text{sup}(a, b) \in \mathbf{W}(A, B)}$$

- **W** - Elimination

$$\frac{\begin{array}{l} (x \in A, y \in B(x) \rightarrow \mathbf{W}(A, B), \\ z \in \Pi(B(x), (v).C(Ap(y, v)))) \\ c \in \mathbf{W}(A, B) \quad d(x, y, z) \in C(\text{sup}(x, y)) \end{array}}{\mathbf{R}_W(c, d) \in C(c)}$$

In the second part we shall see how to use the type constructor **W** to build a type which represents within type theory the universe of sets described by constructive axiomatic set theories like **CZF**. This will be a restricted form of **W** type, mainly **W(U, T)**. We shall first of all introduce the system **CZF** and then present the basic tools for its interpretation in type theory. By means of the interpretation the relationship between the two notions of sets will become clear.

References

- [1] P. Aczel, *The strength of Martin-Löf type theory with one universe*. Thec. Rep., Dept. of Philosophy, University of Helsinki 1977.
- [2] P. Aczel, *The Type Theoretic Interpretation of Constructive Set Theory*, in *Logic Colloquium '77*, ed. by A. MacIntyre, L. Pacholski, J. Paris, North-Holland, Amsterdam-New York 1978, pp. 55–66.
- [3] P. Aczel, M. Rathjen, *Notes on Constructive Set Theory*, Draft available from the Internet at the address: <http://www.cs.man.ac.uk/~petera/>.
- [4] L. Augustsson, T. Coquand, B. Nordström: *A short description of Another Logical Framework*, in *Proceedings of the First Workshop on Logical Frameworks*, Antibes 1990, pp. 39–42.
- [5] M. Beeson, *Foundations of Constructive Mathematics*, Springer Verlag, Berlin 1985.
- [6] E. Bishop, *Foundations of constructive analysis*, McGraw-Hill, New York 1967.
- [7] E. Bishop, D. Bridges, *Constructive Analysis*, Springer, Berlin-Heidelberg 1985.
- [8] D. Bridges, F. Richman, *Varieties of constructive mathematics*, London Math. Soc., *Lecture Notes 97*, Cambridge Univ. Press, 1987.

- [9] R. L. Constable, *Naïve computational type theory*, in *Proof and System-Reliability*, ed. by H. Schwichtenberg, R. Steinbrüggen, Kluwer Academic Publisher, Dordrecht 2002, pp. 213–259.
- [10] R. L. Constable et al., *Implementing Mathematics with the NuPRL Proof Development System*, Prentice-Hall, Englewood Cliffs NJ 1986.
- [11] N. G. de Bruijn, *A survey of the project AUTOMATH* in *To H. B. Curry: Essays on combinatory Logic, Lambda Calculus, and Formalism*, ed. by J. P. Seldin and J. R. Hindley, Academic Press, New York 1980, pp. 589–606.
- [12] S. Feferman, *A language and axioms for explicit mathematics* in *Algebra and Logic*, ed. by J. Crossley, *Lecture Notes in Mathematics*, vol 450, Springer, Berlin 1975, pp. 87–139.
- [13] S. Feferman, *Constructive theories of functions and classes*, in *Logic Colloquium '78*, ed. by M. Boffa, D. van Dalen, K. McAloon, North Holland, Amsterdam 1979, pp. 159–224.
- [14] S. Feferman, *Predicativity*, to appear in *Handbook of the Philosophy of Mathematics and Logic*, ed. by S. Shapiro.
- [15] H. Friedman, *Set theoretic foundations for constructive analysis*, *«Annals of Mathematics»*, 105, 1977, pp. 1–28.
- [16] J.-Y. Girard, *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur*, Thèse, Université Paris VII 1972.
- [17] N. D. Goodman, *A theory of constructions equivalent to arithmetic*, in *Intuitionism and Proof Theory*, ed. by J. Myhill, A. Kino and R. E. Vesley, North-Holland, Amsterdam 1970, pp. 101–120.
- [18] G. Kreisel, *Foundations of intuitionistic logic*, in *Logic, Methodology and philosophy of science*, ed. by E. Nagel, P. Suppes and A. Tarski, III Stanford Univeristy Press, Stanford California 1962, pp. 198–210.
- [19] G. Kreisel, *Mathematical logic*, in *Lectures on Modern Mathematics* ed. by T. L. Saaty, Vol III Wiley, New York 1965, pp. 95–195.
- [20] J. L. Krivine, *Typed lambda-calculus in classical Zermelo-Fraenkel set theory*, *«Archives Mathematical Logic»*, 40, 3, 2001, pp. 189–205.
- [21] P. Martin-Löf, *An intuitionistic Theory of types: predicative part*, in *Logic Colloquium 1973*, ed. by H. E. Rose and J. C Shepherdson, North Holland, Amsterdam 1975, pp. 73–118.
- [22] P. Martin-Löf, *Constructive Mathematics and Computer Programming*, in *Logic, Methodology, and Philosophy of Science VI*, ed by L. J. Choen, North Holland, Amsterdam 1982, pp. 153–175.
- [23] P. Martin-Löf, *Intuitionistic Type Theory*, Bibliopolis, Naples 1984.
- [24] J. Myhill, *Constructive Set Theory*, *«Journal of Symbolic Logic»*, 40, 1975, pp. 347–382.
- [25] B. Nordström, K. Petersson, J. Smith, *Programming in Martin-Löf's Type Theory. An introduction*, Oxford University Press, 1990.
- [26] D. Scott, *Constructive validity*, in *Symposium on Automatic Demonstration*, *Lecture Notes in Mathematics*, Vol. 125, Springer, Berlin 1970, pp. 237–275.

- [27] J. Smith, *The independence of Peano’s fourth axiom from Martin–Löf’s type theory without Universes*, «Journal of Symbolic Logic», 53(3), 1988.
- [28] A. S. Troelstra, *From constructivism to computer science*, «Theoretical Computer Science», 211, 1999, pp. 233–252.
- [29] A. S. Troelstra and D. van Dalen, *Constructivism in Mathematics: an Introduction*, volumes I and II, North–Holland, Amsterdam 1988.

