Toward a minimalist foundation for constructive mathematics

Maria Emilia Maietti and Giovanni Sambin
Dipartimento di Matematica Pura ed Applicata Università di Padova

CLARENDON PRESS · OXFORD
2005

Abstract

The two main views in modern constructive mathematics, usually associated with constructive type theory and topos theory, are compatible with the classical view, but they are incompatible with each other, in a sense explained by some specific results which we briefly review. So it is desirable to design a common core which is compatible with all the theories in which mathematics has been developed, like Zermelo-Fraenkel set theory, topos theory, Martin-Löf's type theory, etc. and can be understood as it is by any mathematician, whatever foundation is adopted.

A requirement with increasing importance is that of developing mathematics in such a way that it can be formalized on a computer. This theoretically means that the foundation should obey the proofs-as-programs paradigm.

We claim that to satisfy both requirements it is necessary to use a minimal type theory mTT, which is obtained from Martin-Löf's type theory by relaxing the identification of propositions with sets. This ground type theory mTT is intensional and is needed for formalization. A "toolbox" of extensional concepts, needed to do mathematics, is built on it. The common core is obtained at this level, by subtraction.

The underlying conceptual novelty is that one should give up to the expectation of an all-embracing foundation, also in the concrete sense that one needs a formal system living at two different levels of abstraction.

After abandoning the classical view and all its limits, a prospective constructivist is faced with the problem of choosing among a variety of views. They generally share the choice for intuitionistic logic, but they differ in mathematical principles.

The principal distinction is between two views. One maintains that the meaning of mathematics lies in its computational content, and thus keeps its formalization in a computer language in mind. It is usually associated with type theory; the axiom of choice then turns out to be valid and (hence) the powerset axiom is not accepted. The other favours the mathematical structure beyond its particular presentations. It is usually expressed through category theory and often identified with topos theory. Extensionality is an essential feature, the powerset axiom is mostly accepted and hence the axiom of choice is not accepted as valid.

Both views are reasonable, well motivated and apparently cannot be dispensed with. It is however a matter of facts that they are incompatible, in the sense that by accepting both one is forced back to the classical view.

The same tension is revealed also as a technological challenge: while implementation of mathematics in a computer is intrinsically intensional (just think of the fact that a computer handles only expressions and no objects), mathematics itself needs to deal with objects independently of their presentations, and hence is extensional. Because of the first purpose, intensional type theory cannot be given up (it is actually used in most implementations), while an extensional foundation, like set theory or the theory of a generic topos, seems to be essential for the second (extensionality is assumed in all the actual approaches to math-

ematics). So again both visions seem to be indispensable; and yet they remain incompatible.

In our opinion, this incompatibility has to be faced openly. Communication between different approaches to mathematics clearly cannot be dismissed; and the urge to implement mathematics will certainly increase in the future. So it is necessary to identify a new foundational theory, which on the one hand permits to keep communication alive and on the other hand can be used as a convenient basis to address the problem of implementation. Because of the incompatibility between the two general views, this theory is not to be obtained by union but by subtraction, that is by finding a common kernel. Thus we are pushed to abandon the widespread expectation of a single all-embracing foundation, which is good for all purposes. Actually, this is true in two different ways.

First, the purpose of implementing mathematics requires to keep a clear-cut distinction between a ground level with an intensional type theory, which is necessary for implementation and actually is essentially an abstract functional programming language, and a more abstract level, in which one has only extensional concepts and constructs which are needed to develop mathematics. These extensional tools can be put on top of type theory in the spirit of "toolbox" (see the preface of [36] and section 3.2.1 of [34]). Thus any tool should be obtained according to the forget-restore principle: to reach extensionality, it is necessary to *forget* some information, like the algorithm to compute a function or the specific proof of a proposition; but to obtain implementation, one has to be able to *restore* such information at will. So the formal system must include independent treatment of two different levels of abstraction, as well as a systematic way to connect them. Typically, the axiom of choice is valid at the lower level, while it fails at the extensional level. Moreover, to be able to implement toolbox, it is necessary to introduce proof-irrelevance for propositions. Thus propositions must be kept well distinct from sets. So a key step is to give a justification of logic independently of set theory, and we do this via the principle of reflection [35].

Second, keeping communication between different views alive requires the design of a common theory in which all mathematicians can believe, while leaving them free to keep their intended semantics in mind. By leaving out both the axiom of choice and the powerset axiom at the level of toolbox, one obtains a common basis which is immediately understood as it is (that is, with no translations) and believed in by any mathematician. Thus, contrary to what is usually required on a foundational theory, our minimal theory cannot be the best possible description of a single semantics. Rather, it is designed to admit different, and actually mutually incompatible interpretations, or extensions.

After illustrating these technical and conceptual reasons, we present a specific formal system, called mTT. It is only a first proposal, and we are aware that it could be improved; still it works as a precise basis for a minimalist foundation. In fact, the toolbox developed over Martin-Löf type theory in [36] actually has our present mTT as the intended ground theory; to confirm this, one can observe that no real use is made of the axiom of choice. So, for example, since its beginning

[33] all of formal topology, as long as it uses only extensional notions defined as "tools" over type theory, has actually adopted - with informal rigour - the minimalist approach described here.

Awareness of the fact that all this could be expressed in a formal way, and moreover that it brings to compatibility with topos theory, comes from [22]. The two authors have discussed these topics since then; an anticipation of the conceptual content of the present paper was given in section 3.1.2 of [34]. As a final remark, we recall that a minimalist foundation has aided the emergence of new and deeper structures, like those in the basic picture [37].

Acknowledgements. The first author thanks Enrico Martino for fruitful discussions on the definition of a proofs-as-programs theory. We thank Thierry Coquand, Giovanni Curi, Ferruccio Guidi, Per Martin–Löf, Eike Ritter, Bas Spitters, Thomas Streicher, Silvio Valentini for their comments on the topics of this paper.

0.1 Arguments for a minimal constructive type theory

Here we present and discuss our arguments to show that some natural expectations are satisfied only by a minimalist foundation of constructive mathematics. In the next section, we propose a formal system which satisfies such expectations and we briefly discuss its connections with the literature.

0.1.1 *The computational view*

In the conception of Bishop [5–7] and of Martin-Löf [25], the meaning of constructive mathematics is given by its computational content. So every mathematical entity should be computable: sets must be inductively generated, functions must be given by an effective algorithm,... In other terms, every object of mathematical reasoning must be expressible in and computable by a computer, which theoretically means a Turing machine. This can be done by defining a concept of computation for sets and functions beside that for number-theoretic functions. Then the mathematical reasoning itself is required to preserve the computational content. So the logic must be intuitionistic, and this is usually explained by saying that it satisfies Heyting semantics (or the so-called BHK interpretation) of propositions and of logical constants: every proposition can be identified with the set of its proofs, and every proof gives an effective method. We call this the computational view of constructivism.

The mathematics developed in such a computational way is by definition formalizable on a computer, in the strict sense that all the constructions and functions are computed by a computer. So one needs a foundational theory satisfying what we call the proofs-as-programs paradigm:

Proofs-as-programs. *A theory satisfies the paradigm* proofs-as-programs *if i.) all its set-theoretic constructors preserve computability, in the sense that their element constructors are given by effective methods so that all effective methods between natural numbers are computed by machine programs, ii.) its underlying logic satisfies Heyting semantics.*

This is a strict formulation of what it means for a foundation to be constructive in the computational sense. In the same time it is a way to express theoretically the requirement that the mathematics developed on it can be implemented on a computer. Even if we do not commit ourselves to any definition of computation between arbitrary sets (like in Martin-Löf type theory, Coq, PCF,...), this means that all mathematical concepts and proofs must admit a realizability interpretation, where the realizers are just the real programs of the Turing machine.

We now can see that if a theory satisfies proofs-as-programs and it is formulated as a many-sorted logic whose sorts include finite types like the type of functions $A \to B$ and whose logic includes first order logic, then it is consistent with the axiom of choice and formal Church thesis. By the axiom of choice we mean the statement

$$AC: \quad (\forall x \in A)\ (\exists y \in B)\ R(x,y) \longrightarrow (\exists f \in A \to B)\ (\forall x \in A)\ R(x, f(x))$$

expressing the idea that from a proof of a specification $(\forall x \in A)\ (\exists y \in B)\ R(x,y)$ one can extract a function (or program) f which on the input $x \in A$ computes the output $f(x) \in B$ satisfying $R(x, f(x))$.

By the *formal* Church thesis we mean the statement expressing that all number-theoretic functions are programs:

$$CT: \quad (\forall f \in \mathbb{N} \to \mathbb{N})\ (\exists e \in \mathbb{N})\ (\forall x \in \mathbb{N})\ (\exists y \in \mathbb{N})\ (T(e,x,y)\,\&\,U(y) = f(x))$$

where $T(e,x,y)$ is the Kleene predicate expressing that y is the computation executed by the program with code number e on the input x and $U(y)$ is output of the computation y.

Now, since a proofs-as-programs theory enjoys Heyting semantics, then it must be consistent with AC, since Heyting semantics validates AC because of the meaning given to the existential quantifier. Moreover, since we required that all effective methods between natural numbers are computed by programs, we deduce that the considered theory is also consistent with CT. Therefore, we conclude that *a many sorted theory satisfying proofs-as-programs must be consistent with the axiom of choice and formal Church thesis, that is with AC+CT.*

A natural question to ask is what theories apt to develop mathematics satisfy the proofs-as-programs paradigm, and if there are any. First of all, one can observe immediately that the classical theory of sets ZF does not satisfy proofs-as-programs, given that not all number-theoretic functions are recursive. But not many other classical theories can satisfy the paradigm, since even many-sorted classical arithmetic fails to do it, as proved by the following (see for example [11]):

Proposition 0.1 *Let $\mathcal{S}$ be a many sorted theory of first-order classical logic, whose sorts include types of simply typed lambda calculus [13] like that of natural numbers $\mathbb{N}$, the product type $A \times B$ and arrow types like $A \to B$. If $\mathcal{S}$ satisfies Peano axioms, formal Church thesis and the axiom of choice (written by means of the arrow type), then it is inconsistent.*

Proof. Consider any predicate $P(x)$ and observe that

$$\forall x \in \mathbb{N}\ \exists y \in \mathbb{N}\ \ P(x) \longleftrightarrow y = 0$$

is valid thanks to the principle of excluded middle and recalling that $0 \neq 1$ is one of Peano axioms: if $P(x)$ holds put $y = 0$ and, if not, put $y = 1$. Then, by applying CT to it we obtain that for all $x \in \mathbb{N}$ there exists a recursive function f such that $P(x)$ iff $f(x) = 0$; that is, all predicates are recursive, which is not possible.

This proof reveals that, in order to keep arithmetic and to realize the proofs-as-programs paradigm, and hence to be consistent with AC and CT, one possibility is to discharge the principle of excluded middle, as it seems to be the main cause of the inconsistency in the above proof. Therefore, from now on we only consider intuitionistic theories. Among them, we have intuitionistic set theories formulated in the style of Zermelo-Fraenkel set theories like IZF [38], Myhill's CST [31] or Aczel's CZF [2]. In the approach of category theory, we could also consider a generic elementary topos as a universe of sets where to develop intuitionistic mathematics.

However, none of these proposals satisfies the proofs-as-programs paradigm. The main reason is that they validate extensional principles like the principle that two functions are equal if they have equal values. This can be seen well in the following proposition, where we prove that also extensional Martin-Löf's type theory does not satisfy proofs-as-programs. The proof is obtained by adapting that in [41] (page 498) of the fact that Heyting arithmetic with finite types HA^{ω} is inconsistent with CT+AC and function extensionality:

Proposition 0.2 *Martin-Löf's extensional type theory [26], which validates the axiom of choice, is inconsistent with formal Church thesis.*

Proof. By applying AC on CT one gets a function $h \in (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ such that

$$(*) \qquad (\forall f \in \mathbb{N} \to \mathbb{N})\ (\forall x \in \mathbb{N})\ \ (\exists y \in \mathbb{N}) \quad (T(h(f), x, y)\,\&\,U(y) = f(x))$$

By extensionality of functions, that is the fact that $\lambda x.f(x) =_{\mathbb{N}\to\mathbb{N}} \lambda x.g(x)$ is derivable if and only if $\forall x \in \mathbb{N}\ f(x) =_{\mathbb{N}} g(x)$, one can prove that h is injective: from $(*)$ and $h(\lambda x.f(x)) =_{\mathbb{N}} h(\lambda x.g(x))$ one has $\forall x \in \mathbb{N} f(x) =_{\mathbb{N}} g(x)$ and hence by extensionality also $\lambda x.f(x) =_{\mathbb{N}\to\mathbb{N}} \lambda x.g(x)$. Since $\lambda x.f(x) =_{\mathbb{N}\to\mathbb{N}} \lambda x.g(x)$ obviously implies $h(\lambda x.f(x)) = h(\lambda x.g(x))$ because h is a function, $\lambda x.f(x) = \lambda x.g(x)$ turns out to be equivalent to $h(\lambda x.f(x)) = h(\lambda x.g(x))$. Since equality of natural numbers is decidable, then extensional equality of recursive functions would also be decidable, which is not. Hence, the theory is inconsistent.

The main cause of the inconsistency in the above proof seems to be the principle of extensionality of functions, which is obtained by means of the extensional equality. So to keep consistency with AC+CT one possibility is to consider a type theory with an *intensional* equality, like Martin-Löf's intensional type theory in [32, 28]. From now on we call it M-L type theory.

It is proposed as a full scale approach to a constructive (intuitionistic and predicative) mathematics. It satisfies the proofs-as-programs paradigm since it is a functional programming language which satisfies all the desirable properties, like type-checking, needed for computation. Moreover, it can be seen as a rigorous and general description of the identification of propositions with sets (propositions-as-sets interpretation), of which Heyting semantics is a consequence. So validity of AC follows from the interpretation of quantifiers.

All of this is very satisfactory. So why should one look for something different? There are some good reasons to do that.

0.1.2 *Intrinsic reasons: intensionality vs. extensionality*

Even assuming wholeheartedly the computational view, there are some good reasons to look for a modification of M-L type theory. We begin here with some intrinsic reasons.

Of course, our type theory should be sufficient for the working mathematician to develop mathematics on it, and not only to formalize it. A peculiarity which is common to all mathematics is its extensionality: sets and subsets with the same elements are equal, functions giving the same values are equal,... However, as seen in proposition 0.2, adding extensionality to a theory makes it inconsistent with proof-as-programs, and hence unsuitable to formalize mathematics. This is a matter of facts that cannot be ignored. Extensional aspects, like extensional equality as in [26], simply cannot be required directly at the level of the ground type theory, if this has to satisfy the proofs-as-programs paradigm.

There is a way out, and it seems unavoidable to us: to develop mathematics in a type theory, one must have both a ground intensional level, which satisfies proofs-as-programs and where formalization of mathematics is implemented, and an extensional level, where mathematics is actually developed. These two levels must be kept carefully distinct, but of course they must also be linked in a clear and systematic way. We do not see them as two theories, but as two connected levels of abstraction in the same theory, as we now explain.

Mathematics is commonly conceived as dealing with "objects", which are independent of their different presentations. This is just a way to say that it is extensional. So the higher, extensional level should be obtained from the lower, intensional one by leaving out those details of the presentations which are not necessary to determine a mathematical "object", that is by "forgetting" some information. For instance, as most often in mathematics, one is interested only in the provability or truth of a proposition, and not in the specific form of its proofs (proof-irrelevance). Therefore it is convenient to use a judgement of the form $A\ true$ (see [26]) for a proposition A with the meaning that A is provable. This requirement implies that in the ground type theory, on which we abstract, propositions should be formally well distinct from sets. In fact, proof-irrelevance must act on only propositions; it simply would make no sense on sets.

The design of the extensional level over type theory has to be done by following a certain discipline. We propose that it should be achieved by building

a suitable collection of (extensional) mathematical concepts, or tools; from now on, we call it the *toolbox*, as in [36].

It is then in toolbox that mathematics can be developed. But one of our requirement was that mathematics should be formalizable in a computer. To this aim, one usually devises a computational model of the extensional theory. We don't need to do this for our toolbox of extensional concepts. In fact, if all tools are introduced according to what has been called the *forget-restore* principle (for a detailed explanation, see the preface of [36] and section 3.2.1 of [34]), then their implementability at the ground level is automatically preserved. In fact, *restore* says that the computational content is fully under control (and thus *forget* was correct) since it can be restored whenever wished. Thus the tool can be brought back to the ground theory. In other words, the forget-restore principle says that only those tools can be introduced for which implementation is known.

Another example of extensional theory apt to develop mathematics, and that admits a constructive interpretation in type theory, is Aczel's system CZF of (constructive) set theory. However, its interpretation in M-L type theory is global and static (technically, it uses type theory as a metalanguage to construct a model of CZF, and this requires W-types and universes). So implementation has to be studied case by case. Instead, in our approach, the link between toolbox and our ground type theory is local and dynamic (it can be seen in the same moment mathematics is done), and the computational content is systematically preserved (which is the reason for keeping proof-terms of propositions). A precise and practical difference is hence that all the mathematics which is developed in toolbox is automatically formalizable.

The fact that to formalize mathematics a theory with two different levels is needed helps to clarify the debate about the validity of the axiom of choice. As recently emphasized by Martin-Löf, one must distinguish between an "intensional" AC, which holds constructively simply by the BHK interpretation of quantifiers, and an "extensional" AC, which simply fails (constructively).[1]

So also when developing mathematics on the basis of type theory, either one specifies that mathematics is meant to be "intensional" (which is not done in practice), or one carefully justifies each application of AC, by showing that the choice function respects the equality involved (there are sets, like the natural numbers, in which the generation of elements automatically produces "objects", so intensional and extensional equality coincide and AC is justified). In other words, indiscriminate use of AC is admissible only in a classical axiomatic set theory, like ZFC.

Using toolbox guarantees that this problem is solved. In fact, because of proposition 0.2 (and the wish of compatibility, see next subsection), we need to

[1] In his talk entitled *100 years of Zermelo's axiom of choice: what was the problem with it?* given at TYPES '04, Jouy-en-Josas (France), 15-18 Dec. 2004, he also stressed that the debate among mathematicians about the acceptance of AC should be revisited with this distinction in mind.

block validity of AC at the level of toolbox; this is done by keeping propositions distinct from sets, and by adopting a weak notion of existential quantifier. Of course, it remains true that the corresponding formulation of AC for sets, which becomes just the distributivity of Σ over Π, is valid in the ground theory.

0.1.3 *Extrinsic reasons: compatibility*

There are also other good reasons to consider a variant of M-L type theory, and these are extrinsic. We firmly believe that even if one decides to develop mathematics in the computational way, one should not ignore all the mathematics already developed and the important results achieved in other conceptions, like Zermelo-Fraenkel set theory and topos theory. It would be very strange to develop mathematics in different ways and with no communication between them.

Reaching mutual communication is not just an ideal wish, there are strong and concrete reasons that one can experience in the actual development of mathematics. A common base would allow for a more systematic transfer of techniques and results, than what is possible today, between two approaches to the same topic (for example, pointfree topology), one developed in topos theory (theory of locales) and the other in type theory (formal topology). Ideally, one would like to develop constructive mathematics in a theory which is compatible with existing foundations and can be understood and used by most mathematicians.

To reach mutual communication with existing relevant theories, besides proofs-as-programs, we add also the following condition to our desiderata:

Compatibility. *The theory should be* compatible *with known theories such as classical set theory and the theory of a generic topos [20,19], Martin-Löf's intensional type theory [32] and the Calculus of Construction [9], and hence it should be* minimal *with respect to such more expressive existing theories.*

Note that the above condition implies the compatibility also with all the set theories that are interpretable in M-L type theory, as CZF [2].

To reach compatibility with classical set theory ZFC we certainly need to abandon the internal validity of formal Church thesis, whilst we can allow the internal validity of the axiom of choice. Giving up CT is not so painful since CT is not necessarily valid in Heyting semantics.

However, we also have to give up the internal validity of the axiom of choice if we want to reach compatibility also with the theory of a generic topos. The reason is that a topos satisfying the internal axiom of choice is boolean [12], that is it enjoys classical logic. Hence, a topos with a natural numbers object is inconsistent with AC+CT, because classical arithmetic is already inconsistent with such principles. The same applies to intuitionistic set theories in the style of Zermelo-Fraenkel like IZF [38], Myhill's CST [31] or Aczel's CZF [2].

The fact that topos theory is inconsistent with AC+CT is certainly due to AC, given that there are examples of toposes, like the effective topos [16], that validate CT. The reason of such incompatibility is well visible in the proof of the following proposition (first given in [14]) in the context of a set theory validating the axiom of choice and some other simple axioms (see also [3, 4]):

Proposition 0.3 *Let $\mathcal{S}$ be a set theory in a many sorted language with the empty set axiom, pair set axiom, comprehension axiom, set extensionality axioms and the axiom of choice. Then $\mathcal{S}$ validates the principle of excluded middle.*

Proof. Let P any formula of the set theory. Let $0 \equiv \emptyset$ and $1 \equiv \{\emptyset\}$. Consider the sets

$$V_o \equiv \{x \in \{0,1\} \mid x = 0 \vee P\} \qquad V_1 \equiv \{x \in \{0,1\} \mid x = 1 \vee P\}$$

Note that we can apply the axiom of choice on

$$(\forall z \in \{V_o, V_1\})\ (\exists y \in \{0,1\})\ (y \in z)$$

which is always valid, to get a function f from $\{V_o, V_1\}$ to $\{0,1\}$. Through this function, which must give equal values on extensionally equal sets, we can decide whether P holds or not by comparing the values $f(V_o)$ and $f(V_1)$.

This proof reveals that extensionality is constructively incompatible with the axiom of choice. Indeed, the choice function does not need to respect set extensionality since a proof of $(\forall x \in A)\,(\exists y \in B)\,R(x,y)$ can be done *intensionally* by associating a b in B to any $a \in A$ without necessarily respecting a given equivalence relation on A.

Extensionality is an intrinsic aspect of topos theory. Indeed, the internal type theory of a topos formulated in the style of Martin-Löf's type theory is extensional because of the presence both of the extensional propositional equality and of extensional powersets [23]. This makes the axiom of choice become what in type theory - or better in the extensional theory of setoids built upon type theory - is recognized as the *extensional* axiom of choice, since the choice function must be extensional by definition.

Hence a type theorist to be understood by a topos theorist has to give up the axiom of choice that the topos theorist automatically understands as the extensional one. This difficulty of expressing intensional concepts in topos theory is indeed a limit and a possible source of misunderstanding, given that the extensional axiom of choice is by no means valid in Heyting semantics (and in general), contrary to the intensional axiom of choice which certainly is.

Finally, to make our proofs-as-programs theory compatible with M-L type theory we must certainly give up extensional powersets. In fact dropping extensional propositional equality is not enough to keep constructive compatibility with AC if we keep extensional powersets as described in [24]. [2]

Therefore we conclude that to satisfy the compatibility condition our proofs-as-programs theory has to avoid the internal validity of CT, of AC and the presence of extensional powersets.

[2] There it is proved that an extension of M-L type theory [32,28] with *extensional* powersets, in which subsets are represented by propositional functions, necessarily validates the principle of excluded middle.

0.1.4 *Conceptual reasons: against reductionism*

Two further reasons of more conceptual nature speak in favour of a minimalist approach, since it avoids the reduction of logic and of geometrical intuition to computation.

The explanation of logic via the propositions-as-sets interpretation evidently makes logic strongly dependent on set theory. But while it seems legitimate to reduce mathematics to computation, it is hard to see reasons why this should apply also to logic. In fact, it seems clear that logic should be applicable also to fields not (yet) treated mathematically, and thus it is quite natural that its justification should not require any prior justification of set theory. So one reason to modify M-L type theory is to obtain a distinction of logic from set theory, that is of deductions from computations.

A second reason is that the computational view underlying M-L type theory is so strict that it clashes with some kind of spatial or geometric intuition. For instance, one would like to find a precise formulation of Brouwer's notion of choice sequence [8], but this appears to be impossible in M-L type theory unless it is modified to reach control over the use of AC (see section 3.2.6 of [34]).

0.2 A proposal for a minimal type theory

After giving some good motivations in favour, we now have to show that a minimal theory is possible. We agree with Bishop and Martin-Löf that AC is a direct consequence of the explanation of intuitionistic logic as in the BHK interpretation, or in the propositions-as-sets view. Given that AC is to be abandoned, we have to abandon propositions-as-sets too and thus we must explain logic in a different way, independently of proof-terms and of set theory in general.

We believe that logical constants (connectives and quantifiers) can be explained in a satisfactory way via the *principle of reflection* (see [35]), which relies only on the notion of truth of a proposition, possibly depending on an arbitrary element of a domain. In this way logic becomes independent of full set theory. This is a delicate point. For instance, also the explanation of logical constants given by Martin-Löf in [27], following Prawitz's inversion principle, does not make explicit use of proof-terms. But one central idea there is that introduction rules define the meaning, and elimination rules are completely determined by them. In our opinion, this idea has to be abandoned too, otherwise the propositions-as-sets interpretation is implicitly present (a fact which goes together with validity of AC, which in our approach fails).

0.2.1 *Logic explained via the principle of reflection*

The principle of reflection says that the meaning of a logical constant is given by its definitional equation, which requires the logical constant to be the reflection in the object language of a link between judgements (assertions of truth of a proposition) at the metalanguage. The formal rules of inference, defining the logical constant explicitly, are derived from its definitional equation. This is done for all connectives in [35], where it is shown that they are all explained by using

only two metalinguistic links, *and* and *yields* (corresponding to comma , and turnstile $\vdash$ in a sequent calculus). Moreover, the derivations of inference rules all follow the same pattern. We here show that also the quantifiers $\forall$ and $\exists$ can be explained in the same way. With respect to connectives, one needs the additional notion of propositional function depending on an element of a domain.

The informal version of the definitional equation for $\forall$ is: $\Gamma \vdash (\forall x \in D)A(x)$ iff *for every* $d \in D, \Gamma \vdash A(d)$. A more rigorous formulation is obtained by replacing the right member with $\Gamma, z \in D \vdash A(z)$, on the assumption that Γ does not depend on z:

$$\Gamma \vdash (\forall x \in D)A(x) \quad \text{iff} \quad \Gamma, z \in D \vdash A(z)$$

So to explain the meaning of $\forall$ one first has to understand the meaning of $\Gamma, z \in D \vdash A(z)$, that is, the fact that one can replace z with any element d of D and obtain that Γ true yields $A(d)$ true. In the explanation of $\forall$ given in the BHK interpretation, one has to understand this and *moreover* that a method is given to transform d and a proof of Γ into a proof of $A(d)$. We do not need here the addition of the method, because only the truth of propositions (with no proof-term) is involved.

Of course, knowing that D is a set in the sense of type theory, that is, with rules to generate all its elements, is certainly sufficient to give a clear explanation of the meaning of $\Gamma, z \in D \vdash A(z)$. It does not seem, however, to be also necessary; in fact, it seems that one can be in the position of knowing $\Gamma \vdash A(d)$ whenever $d \in D$ is known, however this information may be obtained. For example, knowing that d is a human being, one understands that d is mortal, even if one is not able to give fixed rules to generate all human beings as a set.

The solution of the definitional equation for quantifiers follows the same pattern as for connectives. We prefer to skip the details in the case of $\forall$ and give them only for $\exists$, because it is a bit less intuitive. The informal version of the definitional equation for $\exists$ is: $\Gamma, (\exists x \in D)A(x) \vdash \Delta$ iff *for every* $d \in D, \Gamma, A(d) \vdash \Delta$. It is a bit puzzling at first that $\exists$ at object level is explained as the reflection of what looks as a universal quantification at the metalevel, but actually this is explained by observing that *for every* $d \in D$ is reflected at the left of $\vdash$, that is on assumptions. Then the definitional equation can be seen as an expression of the standard meaning of $\exists$: to know that from the pure existence of an element satisfying A one can conclude Δ means precisely to know that one can conclude Δ from $A(d)$, whatever the element d is such that $A(d)$ holds.

Here too, the precise formulation of the definitional equation for $\exists$ is obtained by replacing *for every* $d \in D$, $\Gamma, A(z) \vdash \Delta$ with $\Gamma, z \in D, A(z) \vdash \Delta$, under the condition that Γ, Δ do not depend on z:

$$\Gamma, (\exists x \in D)A(x) \vdash \Delta \quad \text{iff} \quad \Gamma, z \in D, A(z) \vdash \Delta$$

The formation rule is just one direction of the definitional equation:

$\exists$-formation

$$\frac{\Gamma, z \in D, A(z) \vdash \Delta}{\Gamma, (\exists x \in D)A(x) \vdash \Delta}$$

under the condition that Γ, Δ do not depend on z. The other direction can also be written as a rule; it is called "implicit reflection" since it can be seen as information on $\exists$ given implicitly:

$\exists$-implicit reflection:

$$\frac{\Gamma, (\exists x \in D)A(x) \vdash \Delta}{\Gamma, z \in D, A(z) \vdash \Delta}$$

One can think of it as a wish which is to be satisfied by finding a proper rule (that is, one in which $\exists$ does not appear in the premises) which is equivalent to it. This is achieved as follows. First, the premise of $\exists$-implicit reflection is made trivially valid by choosing $\Gamma = \emptyset$ and $\Delta = (\exists x \in D)A(x)$. So one obtains:

$\exists$-axiom

$$z \in D, A(z) \vdash (\exists x \in D)A(x)$$

This is closer to the standard explanation of $\exists$, since it says that if one has any element of D on which A holds, then one can conclude $(\exists x \in D)A(x)$. The proper rule of reflection is now obtained by composing the axiom with derivations of the two components $z \in D$ and $A(z)$:

$\exists$-explicit reflection:

$$\frac{\Gamma \vdash z \in D \quad \Gamma' \vdash A(z)}{\Gamma, \Gamma' \vdash (\exists x \in D)A(x)}$$

In detail, from the premise $\Gamma \vdash z \in D$ and the $\exists$-axiom, by cut one obtains $\Gamma, A(z) \vdash (\exists x \in D)A(x)$ and hence by composing also with $\Gamma' \vdash A(z)$ one obtains the conclusion $\Gamma, \Gamma' \vdash (\exists x \in D)A(x)$.

Now it is easy to prove that actually explicit reflection is equivalent to implicit reflection. In fact, choosing $\Gamma = z \in D$ and $\Gamma' = A(z)$, $\exists$-explicit reflection gives the $\exists$-axiom, and $\exists$-implicit reflection is obtained from the $\exists$-axiom by cutting $(\exists x \in D)A(x)$. Hence $\exists$-formation and $\exists$-explicit reflection together are equivalent to the definitional equation, which is thus solved.

It is still an open problem to see how the principle of reflection should be extended to include a treatment of proof-terms.

0.2.2 *A formal system for minimal type theory.*

After explaining logic by means of the reflection principle, we want to embed it into a set theory that satisfies our requirements. We make a proposal of such a theory and we call it mTT, for Minimal Type Theory. Here, we just give a brief description of its rules and the motivations behind its design, and we collect in the appendix all the formal rules.

Our type theory is obtained by embedding intuitionistic predicate logic into a constructive theory of sets as M-L type theory [32]. However we do not want to do that by a meta-translation of propositions into sets, but we simply add

rules forming propositions to those forming sets. In particular, to build sets we use the four kinds of judgements in [32]:

$$A\ set\ [\Gamma] \qquad A = B\ set\ [\Gamma] \qquad a \in A\ set\ [\Gamma] \qquad a = b \in A\ set\ [\Gamma]$$

that is the judgements about set formation and their terms, equality between sets and equality between terms of the same set. As specific set constructors we take those of [32], like the empty set, lists, dependent products, disjoint sums and strong indexed sums.

Then, to build propositions we use new kinds of judgement saying that something is a proposition and when two propositions are equal:

$$A\ prop\ [\Gamma] \qquad A = B\ prop\ [\Gamma]$$

mTT includes the formation of propositions like falsum, universal and existential quantifications, implication, disjunction, conjunction and an intensional propositional equality, with their terms and equalities. Their elimination rules act only on propositions and hence they are more restrictive than the corresponding rules in M-L type theory.

Considering that the rules of logic are obtained by following the reflection principle in a sequent calculus style, the best would be to formulate all the system in a sequent calculus style, including set theory. But, being more familiar with type theories formulated in natural deduction style, we leave this problem to future work and we express logic too in the natural deduction style.

Since we want our theory to satisfy the proofs-as-programs paradigm, we must give all the inference rules for propositions with explicit treatment of how proofs are constructed; formally, we must provide the rules with proof-terms. In other terms, we identify a proposition with the set if its proofs. This might at first seem to contradict our view that logic is to be independent of set theory. However, this choice is just forced on us if we want to keep proofs as programs: the only way we have to "teach a computer" how to deal with deductions is to identify them with some kind of computations. From the conceptual point of view, this move is less artificial than it looks at first: while it certainly remains odd to assume that a proposition like "All human beings are mortal" is identified with the set of its proofs, this is justified when we restrict our attention to mathematical propositions, and moreover we want to express them and their proofs within a formal system. After all, requiring that also propositions are defined by inductive rules is a way of expressing internally how the proofs in a formal system, like a sequent calculus LJ for intuitionistic logic, are defined inductively.

To express formally that a proposition is identified with the set of its proofs, and so that it is a set, we add the rule:

prop-into-set

$$ps)\ \frac{A\ prop}{A\ set}$$

We also need to add the rule saying that if $A = B\ prop$ then $A = B\ set$. Since

we think of propositions as special sets, like small sets in M-L type theory, we do not introduce new kinds of judgements to express the formation of a proof of a proposition and the definitional equality between proofs, like for example $a \in A\ prop\ [\Gamma]$ and $a = b \in A\ prop\ [\Gamma]$. If we did that, then the *prop-into-set* rule should be enriched with the rule saying that if $a \in A\ prop\ [\Gamma]$ then $a \in A\ set\ [\Gamma]$ and also conversely if $A\ prop\ [\Gamma]$ and $a \in A\ set\ [\Gamma]$ then $a \in A\ prop\ [\Gamma]$, where the converse expresses the fact that the elements of propositions are only their proof-terms. Since we choose to avoid these extra kinds of judgement about proof-terms, then we can reduce contexts to lists of assumptions of variables varying on sets. Therefore, the rules to form contexts in mTT are simply the usual ones of M-L type theory:

$$1C)\ \frac{}{\emptyset\ \ cont} \qquad 2C)\ \frac{\Gamma\ \ cont \qquad A\ set\ [\Gamma]}{\Gamma, x \in A\ \ cont}\ (x \in A \notin \Gamma)$$

Then, to express the fact that equality is an equivalence relation, we add all the inference rules that express reflexivity, symmetry and transitivity of equality between sets, propositions and their terms. Moreover, to express the fact that equality preserves the typing of a term we add the set equality rule

$$seteq)\ \frac{a \in A\ [\Gamma] \quad A = B\ set\ [\Gamma]}{a \in B\ [\Gamma]}$$

Of course, we add the rule of assumption of variables

$$var)\ \frac{\Gamma, x \in A, \Delta\ \ cont}{x \in A\ [\Gamma, x \in A, \Delta]}$$

saying that if $\Gamma, x \in A, \Delta$ is a context then x is an element of A under that context. By this formulation of variable assumption the structural rules of weakening, substitution and of a suitable exchange turn out to be derivable.

The formulation of the *prop-into-set* rule we present resembles the formation of universes à la Russell (see [26]). This observation reminds us that we could also formulate the *prop-into-set* rule à la Tarski, saying that if A is a proposition then $T(A)$ is its set of proofs together with an encoding of proofs of the proposition A into proof-terms of $T(A)$, under the encoding of the propositional context, and with also a decoding of the proof-terms of $T(A)$ into proofs of A, under the corresponding contexts.

The formulation à la Tarski of the *prop-into-set* rule corresponds to the rule in the version of the Calculus of Constructions in [9] expressing that a proposition $\phi : Prop$ can be turned into the type of its proofs $T(\phi)\ type$ by associating the proofs of the dependent product type to those of the universal quantification. Then, the main difference between our type theory and the Calculus of Constructions is that our type theory is predicative also on propositions and hence that propositions are defined inductively with proof-terms.

The presence of proof-terms is necessary from a computational point of view to get a type theory for which type-checking can be performed.

Proof-terms are also crucial to implement a toolbox to deal with extensional concepts, as that in [36]. For example, proof-terms are needed to implement the notion of function between subsets. Recall that a subset is represented in [36] by a propositional function $U(a)$ *prop* $[a \in S]$. Then the difficulty of implementing a function between subsets is due to the fact that it is a partial function between the underlying sets, and a partial function is not definable directly as a proof-term in a type theory where all functions are total. However, we can solve this difficulty by using the indexed sum of a set with a propositional function, namely $\Sigma_{a \in S} U(a)$ for the propositional function $U(a)$ *prop* $[a \in S]$. And it is the *prop-into-set* rule that makes the set $\Sigma_{a \in S} U(a)$ well formed by turning $U(a)$ *prop* $[a \in S]$ into $U(a)$ *set* $[a \in S]$.

Another important role of the *prop-into-set* rule is to allow proofs by induction of a proposition depending on an inductive set. For example, consider a proposition $A(x)$ *prop* $[x \in \mathbb{N}]$ depending on the set of natural numbers (see the appendix for the rules[3]). Since the proposition $A(x)$ *prop* $[x \in \mathbb{N}]$ is also a family of sets $A(x)$ *set* $[x \in \mathbb{N}]$, by means of the *prop-into-set* rule we can use the elimination rule of the natural numbers to derive $El_{List}(a, l, n) \in A(n)$ given the proof-terms $a \in A(0)$ and $l(x, y) \in A(s(x))$ $[x \in \mathbb{N}, y \in A(x)]$. Hence, we do not need to add the specific induction principles for propositions singularly, as in [1].

Considering that in developing mathematics one is interested only in the provability of a propositions and not in its proofs, one could be unsatisfied with the introduction of proof-terms for propositions. This is certainly a legitimate request, but to be implemented not at the level of type theory but at the level of toolbox. This is why in toolbox to talk about the validity of a proposition one can use an additional kind of judgement (introduced in [26]) of the form A *true* for a proposition A, with the meaning that the proposition A is true. However, according to the introduction in [36] and section 3.2.1 in [34], one must be careful when using proof-irrelevance of propositions since this is a derived property depending on the underlying type theory. Indeed, it is acceptable to forget the information of proof-terms that is necessary to give a presentation of an extensional object, like for example a function between subsets, only when we do not loose the possibility of restoring it, that is when we do not destroy the computation necessary to implement the concept itself.

Whilst a proposition is identified with the set of its proofs, in the resulting type theory the separation between logic and set theory remains clearly visible. In fact, while the elimination rules of sets follow the inversion principle (see principle (1) on page 8 of [26]), the elimination rules of propositions do not. Indeed, the sets of proofs of mathematical propositions originate from the reflection principle

[3]Note that we can represent $\mathbb{N} = List(N_1)$ as the set of lists on the singleton set which is in turn represented as $N_1 \equiv List(N_0)$.

and hence the elimination rule of a proposition refers only to propositions. In other words to generate the proofs of propositions we do not follow the principle that the introduction rules determine the elimination rules, but we simply give a different justification of the rules of logic with respect to those of set theory.

To clarify this point, recall that also Martin-Löf in [27] gives an explanation of logic independently from its interpretation in set theory, but still using Heyting semantics. Then, to develop mathematics he interprets logic into set theory; it is here that he interprets the quantifier $\exists$ into the set-theoretic constructor Σ, probably - our guess - because they have the same introduction rules. But we may conceive of different indexed sum constructors with the same introduction rules and different elimination rules, such as the strong indexed sum in [32, 28] and Howard's weak indexed sum in [15], which we report here:

Weak indexed sum set

$$\text{F-}\Sigma^w)\ \frac{C(x)\ set\ \ [x \in B]}{\Sigma^w_{x\in B}C(x)\ set} \qquad \text{I-}\Sigma^w)\ \frac{b \in B \quad c \in C(b)}{< b,_w c > \in \Sigma^w_{x\in B}C(x)}$$

$$\text{E-}\Sigma^w)\ \frac{\begin{array}{l} M\ set \\ d \in \Sigma^w_{x\in B}C(x) \quad m(x,y) \in M\ [x \in B, y \in C(x)] \end{array}}{El_{\Sigma^w}(d,m) \in M}$$

$$\text{C-}\Sigma^w)\ \frac{\begin{array}{l} M\ set \\ b \in B \quad c \in C(b) \quad m(x,y) \in M\ [x \in B, y \in C(x)] \end{array}}{El_{\Sigma^w}(< b,_w c >, m) = m(b,c) \in M}$$

Observe that the elimination rule of the weak indexed sum is restricted to sets not depending on the weak indexed sum itself. Hence, the weak indexed sum solves a definitional equation of weaker complexity than the one of the strong indexed sum.

Our rules for the existential quantifier in mTT (see the appendix) resemble those of Howard's weak indexed sum, but they are applied only to propositions. This is a crucial restriction to get a type theory where the axiom of choice (and probably also the axiom of unique choice) is no longer valid. In fact, at a first glance it might appear that an alternative proposal for a minimal type theory could be the extension of M-L type theory with Howard's weak indexed sum, to be used to interpret the existential quantification. But it turns out that , when considered as propositions, weak indexed sum sets are provably equivalent to strong ones, as first noted by Luo [21]. Hence, interpreting the existential quantifier as the weak indexed sum would not change the theorems of the system and in particular the validity of choice principles.

In mTT we are able to avoid such an equivalence between the strong indexed sum set and the existential quantifier since the existential quantifier solves an even weaker definitional equation than the one for Howard's weak indexed sum, because, after distinguishing propositions and sets, its elimination rule can be applied only to propositions and not to generic sets.

An analysis of the connections between the sets of propositional proofs and the set-theoretic operations on sets of proofs is left to future research (for example, for A, B propositions we have that if $A \to B$ is provable then $\Pi_A B$ is not empty and conversely, but this connection does not seem to follow between $A \vee B$ and $A + B$ as well as between $\exists_{x \in A} B$ and $\Sigma_{x \in A} B$ with A simply a set, as just observed).

Whilst mTT satisfies proofs-as-programs informally, we think that a mathematical model of this type theory where AC+CT are valid could be built starting from the results in [17].

A categorical semantics of mTT, for which mTT provides an internal language, is an open question because of its intensional aspect. Certainly we can design extensional categorical models, considering that for many extensional type theories we know categorical semantics having them as an internal language [23].

Thanks to its genesis, mTT can be seen easily embeddable in M-L type theory in [32, 28] and in the theory of a generic topos, simply because they are obtained by the addition of some further principle. In this strong sense, mTT is compatible with both, and hence also with classical set theory, and this explains our adjective minimal.

Translation of mTT into Martin-Löf's set theory. To translate mTT into M-L type theory it is enough to translate propositions into the corresponding sets following Martin-Löf's propositions-as-sets translation, since all the proofs made with propositions become proofs of the corresponding sets. This also shows that the lambda calculus underlying mTT is strongly normalizing as the one underlying M-L type theory.

Translation of mTT in the internal theory of a topos. Consider the typed calculus of toposes with a natural numbers object introduced in [22, 23] written in the style of extensional Martin-Löf's type theory in [26]. In such a calculus we can also define the falsum type, the disjoint sum type and the quotient type on any equivalence relation. In particular any type can be turned into a type with at most one proof, called *mono type*, by quotienting it over the total relation.

Then, we translate all the sets of mTT into the corresponding types of the internal typed calculus of toposes and the propositions into mono types by induction. In particular, the intensional equality proposition is translated into the extensional equality type, which is mono. Note that the *prop-into-set* rule is valid by definition since propositions are translated as special types.

Translation of mTT into the Calculus of Constructions. We could translate mTT in a version of the Calculus of Constructions [9] where we assume that the types include the inductive sets of M-L type theory. Then, we translate sets into the corresponding types and propositions into the small types of proofs of the form $T(\phi)$ for a proposition ϕ by induction.

Toolbox for mTT. A toolbox for subsets over mTT is already ready in [36]. Whilst this was built for M-L type theory, it can be thought of as built for mTT, since it is based on truth judgements $A\ true$ for propositions whose logical

rules coincide with those of mTT. This might be surprising considering that the mTT existential quantifier is no longer identified with the strong indexed sum and it does not make AC valid. But note that if one consider the rules for the existential quantifier at the level of truth judgements, one can see that the same rules can be obtained also starting from different type constructors. Indeed, the strong indexed sum and Howard's weak one (via propositions-as-sets interpretation) produce the same rules after suppressing proof-terms, namely those of the existential quantifier of intuitionistic logic which also coincide with the rules of mTT existential quantifier. Hence all these constructors can not be distinguished at the pure level of logical judgements. As a consequence, the axiom of choice does not follow from the meaning of the logical constants expressed in terms of truth judgements (see [40,39]). Also the validity of the axiom of unique choice is not automatic in toolbox since one can distinguish a single valued relation from a program that computes it. Several other tools, like setoids, finite subsets, etc. are also needed, but are not yet formally completed.

Related work. The logic-enriched type theory by P. Aczel and N. Gambino in [1] is a theory for constructive mathematics compatible with topos theory and classical set theory. However this is closer to a many-sorted logic approach, while the one we propose here is closer to the type-theoretic approach, since we have proof-terms for propositions in order to be able to perform type-checking. Moreover, by our embedding of propositions into sets we have induction principles for propositions on the various set-theoretic constructions built into it.

In order to reproduce topos theory within M-L type theory, in [18] a predicative notion of topos is introduced. This notion has clearly an extensional character, as that of topos, since it is based on a pretopos whose internal dependent type theory is extensional (see [23]) in the sense of extensional type theory [26].

A comparison between the weak and strong existential quantifiers was also considered in the framework of the Calculus of Constructions in [10, 9] or to represent data types in programming languages in [30].

0.3 Some visible benefits

It is encouraging to observe that the foundational attitude underlying minimal type theory, which apparently was born for reasons of compatibility and communication, actually has a specific identity and an intrinsic mathematical interest. Our claim that minimal type theory, together with toolbox, can be used to develop constructive mathematics is proved by the fact that most of formal topology has been already developed over it, though informally (necessarily so, since the underlying formal system is given here for the first time). In fact, formal topology is developed basing on toolbox in [36], which is meaningful only if propositions are kept distinct from sets and which can be implemented in mTT; this is confirmed also by the fact that in most of formal topology no use is made of choice principles.

In our opinion, some other benefits of the minimalistic attitude are even more interesting. In mathematics, the idea of dealing at the same time with a variety of

different interpretations is not new. It is just the attitude of abstract algebra, and of the modern axiomatic method in general. What we propose is to do the same at the level of foundations. Like it happened with abstract algebra, adopting weaker foundational assumptions has stimulated the emergence of new, stronger structures. Actually, the real and deep reason for choosing a new foundation is that by using it some *new* mathematics is developed (see [34], section 1.3.2). The whole new field called basic picture [34, 37], a new mathematical structure which underlies and at the same time generalizes constructive topology, has sprung out of an informal but rigorous adherence to a minimalist foundation.

The basic picture is patently a piece of mathematics, since it is extensional, it is very simple technically and can be understood as it is (wih no translation) both by a type theorist and by a topos theorist. This means that any mathematician can understand and take advantage of the new mathematical concepts arising in constructive mathematics. Its main conceptual novelty lies in a specific mathematical treatment of notions with existential character. So the well known notions of inclusion $\subseteq$ between subsets, of open subset, of cover $\lhd$ generated by induction,... are now accompanied by their dual notions of overlap $\between$ between subsets, of closed subset, of positivity relation $\ltimes$ generated by co-induction [29],...

Finally, since the axiom of choice does not hold, the notion of formal point is not reduced to be lawlike, and thus the possibility of conceiving choice sequences as formal points remains open (see [34], sect. 3.2.6).

0.4 Open questions and further work

A natural question is to ask what known type theories satisfy our proofs-as-programs paradigm as we specified. In particular we have such open questions:

Is there a realizability model testifying that Martin-Löf's intensional type theory in [32, 28] is consistent with CT? We strongly believe that Martin-Löf's intensional type theory is consistent with CT and on the other hand we hope to see a realizability model of it to get a mathematical proof of that belief. This model would also be a model of mTT and hence a proof of its consistency with AC+CT.

Is second order many-sorted (on finite types including function types) intuitionistic logic, or the Calculus of Constructions consistent with propositional AC+CT? In other terms, does some version of the Calculus of Constructions satisfy our proofs-as-programs paradigm? We are very curious to know a proof of its consistency with AC+CT.

0.5 Appendix: The system mTT

We present here the inference rules for sets and propositions in mTT. For brevity, in presenting formal rules we omit the corresponding equality rules, as in [26]. Moreover, the piece of context common to all judgements involved in a rule is omitted. The typed variables appearing in a context are meant to be added to the implicit context as the last one. We also recall that the contexts are made of assumptions on sets only, and that we have the rule:

Prop-into-set

$$ps)\ \frac{A\ prop}{A\ set}$$

The underlying set theory of mTT is Martin-Löf's constructive set theory [32]. Hence, it includes the following sets:

Empty set

F-Em) $N_0\ set$ E-Em) $\dfrac{a \in N_0 \quad A(x)\ set[x \in N_0]}{\mathsf{emp}_o(a) \in A(a)}$

Strong Indexed Sum set

$F-\Sigma)\ \dfrac{C(x)\ set\ \ [x \in B]}{\Sigma_{x \in B} C(x)\ set}$ I-Σ) $\dfrac{b \in B \quad c \in C(b)}{< b, c > \in \Sigma_{x \in B} C(x)}$

E-Σ) $\dfrac{\begin{array}{l} M(z)\ set\ [z \in \Sigma_{x \in B} C(x)] \\ d \in \Sigma_{x \in B} C(x) \quad m(x,y) \in M(< x, y >)\ [x \in B, y \in C(x)] \end{array}}{El_\Sigma(d, m) \in M(d)}$

C-Σ) $\dfrac{\begin{array}{l} M(z)\ set\ [z \in \Sigma_{x \in B} C(x)] \\ b \in B \quad c \in C(b) \quad m(x,y) \in M(< x, y >)\ [x \in B, y \in C(x)] \end{array}}{El_\Sigma(< b, c >, m) = m(b, c) \in M(< b, c >)}$

Disjoint Sum set

$F-+)\ \dfrac{B\ set \quad C\ set}{B + C\ set}$ I_1-+) $\dfrac{b \in B}{\mathsf{inl}(b) \in B + C}$ I_2-+) $\dfrac{c \in C}{\mathsf{inr}(c) \in B + C}$

E-+) $\dfrac{\begin{array}{l} A(z)\ set\ [z \in B + C] \\ w \in B + C \quad a_B(x) \in A(\mathsf{inl}(x))\ [x \in B] \quad a_C(y) \in A(\mathsf{inr}(y))\ [y \in C] \end{array}}{El_+(w, a_B, a_C) \in A(w)}$

C_1-+) $\dfrac{\begin{array}{l} A(z)\ set\ [z \in B + C] \\ b \in B \quad a_B(x) \in A(\mathsf{inl}(x))\ [x \in B] \quad a_C(y) \in A(\mathsf{inr}(y))\ [y \in C] \end{array}}{El_+(\mathsf{inl}(b), a_B, a_C) = a_B(b) \in A(\mathsf{inl}(c))}$

C_2-+) $\dfrac{\begin{array}{l} A(z)\ set\ [z \in B + C] \\ c \in C \quad a_B(x) \in A(\mathsf{inl}(x))\ [x \in B] \quad a_C(y) \in A(\mathsf{inr}(y))\ [y \in C] \end{array}}{El_+(\mathsf{inr}(c), a_B, a_C) = a_C(c) \in A(\mathsf{inr}(c))}$

List set

F-list) $\dfrac{C\ set}{List(C)\ set}$ I_1-list) $\epsilon \in List(C)$ I_2-list) $\dfrac{s \in List(C) \quad c \in C}{\mathsf{cons}(s, c) \in List(C)}$

$$\text{E-list)}\ \frac{L(z)\ set\ [z \in List(C)] \quad s \in List(C) \quad a \in L(\epsilon) \quad l(x,y,z) \in L(\mathsf{cons}(x,y))\ [x \in List(C), y \in C, z \in L(x)]}{El_{List}(a,l,s) \in L(s)}$$

$$\text{C}_1\text{-list)}\ \frac{L(z)\ set\ [z \in List(C)] \quad s \in List(C) \quad a \in L(\epsilon) \quad l(x,y,z) \in L(\mathsf{cons}(x,y))\ [x \in List(C), y \in C, z \in L(x)]}{El_{List}(a,l,\epsilon) = a \in L(\epsilon)}$$

$$\text{C}_2\text{-list)}\ \frac{L(z)\ set\ [z \in List(C)] \quad s \in List(C) \quad c \in C \quad a \in L(\epsilon) \quad l(x,y,z) \in L(\mathsf{cons}(x,y))\ [x \in List(C), y \in C, z \in L(x)]}{El_{List}(a,l,\mathsf{cons}(s,c)) = l(s,c,El_{List}(a,l,s)) \in L(\mathsf{cons}(s,c))}$$

Dependent Product set

$$\text{F-}\Pi\ \frac{C(x)\ set\ [x \in B]}{\Pi_{x \in B} C(x)\ set} \qquad \text{I-}\Pi\ \frac{c \in C(x)[x \in B]}{\lambda x^B.c \in \Pi_{x \in B} C(x)}$$

$$\text{E-}\Pi\ \frac{b \in B \quad f \in \Pi_{x \in B} C(x)}{\mathsf{Ap}(f,b) \in C(b)} \qquad \beta\text{C-}\Pi\ \frac{b \in B \quad c(x) \in C(x)[x \in B]}{\mathsf{Ap}(\lambda x^B.c(x), b) = c(b) \in C(b)}$$

Then, mTT includes the following propositions:

Falsum

$$\text{F-Fs)}\ \bot\ prop \qquad \text{E-Fs)}\ \frac{a \in \bot \quad A\ prop}{\mathsf{r}_0(a) \in A}$$

Existential quantification

$$F-\exists)\ \frac{C(x)\ prop\ \ [x \in B]}{\exists_{x \in B} C(x)\ prop} \qquad \text{I-}\exists)\ \frac{b \in B \quad c \in C(b)}{< b,_\exists\, c > \in \exists_{x \in B} C(x)}$$

$$\text{E-}\exists)\ \frac{M\ prop \quad d \in \exists_{x \in B} C(x) \quad m(x,y) \in M\ [x \in B, y \in C(x)]}{El_\exists(d,m) \in M}$$

$$\text{C-}\exists)\ \frac{M\ prop \quad b \in B \quad c \in C(b) \quad m(x,y) \in M\ [x \in B, y \in C(x)]}{El_\exists(< b,_\exists\, c >, m) = m(b,c) \in M}$$

Disjunction

$$F-\vee)\ \frac{B\ prop \quad C\ prop}{B \vee V\ prop} \qquad \text{I}_1\text{-}\vee)\ \frac{b \in B}{\mathsf{inl}_\vee(b) \in B \vee C} \qquad \text{I}_2\text{-}+)\ \frac{c \in C}{\mathsf{inr}_\vee(c) \in B \vee C}$$

$$\text{E-}\vee)\ \frac{A\ prop \quad w \in B \vee C \quad a_B(x) \in A\ [x \in B] \quad a_C(y) \in A\ [y \in C]}{El_\vee(w, a_B, a_C) \in A}$$

$$\text{C}_1\text{-}\vee)\ \frac{A\ prop \quad b \in B \quad a_B(x) \in A\ [x \in B] \quad a_C(y) \in A\ [y \in C]}{El_\vee(\mathsf{inl}_\vee(b), a_B, a_C) = a_B(b) \in A}$$

$$\text{C}_2\text{-}\vee) \quad \frac{\begin{array}{c} A\ prop \\ c \in C \quad a_B(x) \in A\ [x \in B] \quad a_C(y) \in A\ [y \in C] \end{array}}{El_\vee(\mathsf{inr}_\vee(c), a_B, a_C) = a_C(c) \in A}$$

Implication

$$\text{F-}\to \quad \frac{B\ prop \qquad C\ prop}{B \to C\ prop}$$

$$\text{I-}\to \quad \frac{B\ prop \qquad c(x) \in C\ [x \in B]}{\lambda_\to x^B.c(x) \in B \to C} \qquad \text{E-}\to \quad \frac{b \in B \quad f \in B \to C}{\mathsf{Ap}_\to(f, b) \in C}$$

$$\beta\text{C-}\to \quad \frac{B\ prop \qquad b \in B \quad c \in C\ [x \in B]}{\mathsf{Ap}_\to(\lambda_\to x^B.c, b) = c(b) \in C}$$

Universal quantification

$$\text{F-}\forall \quad \frac{C(x)\ prop\ [x \in B]}{\forall_{x \in B} C(x)\ prop} \qquad \text{I-}\forall \quad \frac{c(x) \in C(x)\ [x \in B]}{\lambda_\forall x^B.c(x) \in \forall_{x \in B} C(x)}$$

$$\text{E-}\forall \quad \frac{b \in B \quad f \in \forall_{x \in B} C(x)}{\mathsf{Ap}_\forall(f, b) \in C(b)} \qquad \beta\text{C-}\forall \quad \frac{b \in B \quad c(x) \in C(x)\ [x \in B]}{\mathsf{Ap}_\forall(\lambda_\forall x^B.c(x), b) = c(b) \in C(b)}$$

Propositional equality

$$\text{Id}) \quad \frac{A\ set \quad a \in A \quad b \in A}{\mathsf{Id}(A, a, b)\ prop} \qquad \text{I-Eq}) \quad \frac{a \in A}{\mathsf{id}_\mathsf{A}(a) \in \mathsf{Id}(A, a, a)}$$

$$\text{E-Eq}) \quad \frac{\begin{array}{c} C(x, y)\ prop\ [x : A, y \in A] \\ a \in A \quad b \in A \quad p \in \mathsf{Id}(A, a, b) \quad c(x) \in C(x, x)\ [x \in A] \end{array}}{El_{\mathsf{Id}}(p, (x)c(x)) \in C(a, b)}$$

$$\text{C-Eq}) \quad \frac{\begin{array}{c} C(x, y)\ prop\ [x : A, y \in A] \\ a \in A \quad c(x) \in C(x, x)\ [x \in A] \end{array}}{El_{\mathsf{Id}}(\mathsf{id}_A(a), (x)c(x)) = c(a) \in C(a, a)}$$

Conjunction

$$F - \wedge) \quad \frac{B\ prop \qquad C\ prop}{B \wedge C\ prop} \qquad \text{I-}\wedge) \quad \frac{b \in B \quad c \in C}{< b, _\wedge\ c > \in B \wedge C}$$

$$\text{E}_\mathbf{1}\text{-}\wedge \quad \frac{d \in B \wedge C}{\pi_1^B(d) \in B} \qquad \text{E}_\mathbf{2}\text{-}\wedge \quad \frac{d \in B \wedge C}{\pi_2^C(d) \in C}$$

$$\beta_1\ \text{C-}\wedge \quad \frac{b \in B \quad c \in C}{\pi_1^B(< b, _\wedge\ c >) = b \in B} \qquad \beta_2\ \text{C-}\wedge \frac{b \in B \quad c \in C}{\pi_2^C(< b, _\wedge\ c >) = c \in C}$$

REFERENCES

[1] Aczel, P. and Gambino, N. (2002). Collection principles in dependent type theory. In *Types for proofs and programs (Durham, 2000)*, Volume 2277 of *Lecture Notes in Comput. Sci.*, pp. 1–23. Springer, Berlin.

[2] Aczel, P. and Rathjen, M. Notes on constructive set theory. Mittag-Leffler Technical Report No.40, 2000/2001.

[3] Bell, J.L. (1988). *Toposes and Local Set Theories: an introduction.* Claredon Press, Oxford.

[4] Bell, J. (1997). Zorn's lemma and complete Boolean algebras in intuitionistic type theories. *J. of Symbolic Logic*, **62**, 1265–1279.

[5] Bishop, E. (1967). *Foundations of constructive analysis.* McGraw-Hill Book Co.

[6] Bishop, E. (1985). Schizophrenia in contemporary mathematics. In *Errett Bishop: reflections on him and his research (San Diego, Calif., 1983)* (ed. M. Rosenblatt), Volume 39 of *Contemporary Mathematics*, pp. 1–32. Amer. Math. Soc.

[7] Bishop, E. and Bridges, D. (1985). *Constructive analysis. Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*, Volume 279. Springer-Verlag, Berlin.

[8] Brouwer, L. E. J. (1975). *Collected Works I.* North Holland.

[9] Coquand, T. (1990). Metamathematical investigation of a calculus of constructions. In *Logic in Computer Science* (ed. P. Odifreddi), pp. 91–122. Academic Press.

[10] Coquand, T. (1990). On the analogy between propositions and types. In *Logical Foundations of Functional Programming.*, University of Texas at Austin Year of Programming., pp. 393. Addison-Wesley, Reading, MA. originally in LNCS 242.

[11] Coquand, T. (1997). Computational content of classical logic. In *Semantics and logics of computation (Cambridge, 1995)*, Volume 14 of *Publ. Newton Inst.*, pp. 33–78. Cambridge University Press, Cambridge.

[12] Diaconescu, R. (1975). Axiom of choice and complementation. *Proc. Amer. Math. Soc.*, **51**, 176–178.

[13] Girard (with the collaboration of Y. Lafont and P. Taylor), J. Y. (1989). *Proofs and types.*, Volume 7 of *Cambridge tracts in Theoretical Computer Science.* Cambridge University Press.

[14] Goodman, N. and Myhill, J. (1978). Choice implies excluded middle. *Z. Math. Logik Grundlag. Math.*, **24**, 461.

[15] Howard, W. A. (1980). The formulae-as-types notion of construction. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*,

pp. 480–490. Academic Press, London-New York.

[16] Hyland, J. M. E. (1982). The effective topos. In *The L.E.J. Brouwer Centenary Symposium (Noordwijkerhout, 1981)*, Volume 110 of *Stud. Logic Foundations Math.*, pp. 165–216. North-Holland, Amsterdam-New York,.

[17] Hyland, J. M. E. (2002). Variations on realizability: realizing the propositional axiom of choice. *Math. Structures Comput. Sci.*, **12**, 295–317.

[18] I.Moerdijk and Palmgren, E. (2002). Type theories, toposes and constructive set theory: predicative aspects of ast. *Annals of Pure and Applied Logic 114* (1-3), 155–201.

[19] Johnstone, P. T. (2002*a*). *Sketches of an elephant: a topos theory compendium. Vol. 1.*, Volume 43 of *Oxford Logic Guides*. The Clarendon Press, Oxford University Press, New York,.

[20] Johnstone, P. T. (2002*b*). *Sketches of an elephant: a topos theory compendium. Vol. 2.*, Volume 44 of *Oxford Logic Guides*. The Clarendon Press, Oxford University Press, New York,.

[21] Luo, Z. (1994). *Computation and reasoning. A type theory for computer science.*, Volume 11 of *International Series of Monographs on Computer Science*. The Clarendon Press, Oxford University Press, New York.

[22] Maietti, M.E. (1998, February). *The type theory of categorical universes.* Ph. D. thesis, University of Padova.

[23] Maietti, M.E. (2001). Modular correspondence between dependent type theories and categorical universes. *Mittag-Leffler Preprint Series*, **44**.

[24] Maietti, M.E. and Valentini, S. (1999). Can you add powersets to Martin-Löf intuitionistic type theory? *Mathematical Logic Quarterly*, **45**, 521–532.

[25] Martin-Löf, P. (1970). *Notes on Constructive Mathematics*. Almqvist & Wiksell.

[26] Martin-Löf, P. (1984). *Intuitionistic Type Theory. Notes by G. Sambin of a series of lectures given in Padua, June 1980.* Bibliopolis, Naples.

[27] Martin-Löf, P. (1985). On the meanings of the logical constants and the justifications of the logical laws. In *Proceedings of the conference on mathematical logic (Siena, 1983/1984)*, Volume 2, pp. 203–281. reprinted in: Nordic J. Philosophical Logic 1 (1996), no. 1, pages 11–60.

[28] Martin-Löf, P. (1998). An intuitionistic theory of types. In *Twenty five years of Constructive Type Theory* (ed. G. Sambin and J. Smith), pp. 127–172. Oxford Science Publications.

[29] Martin-Löf, P. and Sambin, G. (2005). Generating positivity by coinduction. to appear.

[30] Mitchell, J.C. and Plotkin, G.D. (1988). Abstract types have existential type. *ACM Transactions Programming Languages and Systems 10*(3), 470–502.

[31] Myhill, J. (1975). Constructive set theory. *J. Symbol. Logic*, **40**, 347–383.

[32] Nordström, B., Peterson, K., and Smith, J. (1990). *Programming in Martin Löf's Type Theory.* Clarendon Press, Oxford.

[33] Sambin, G. (1987). Intuitionistic formal spaces - a first communication.

Mathematical logic and its applications, 187–204.

[34] Sambin, G. (2003). Some points in formal topology. *Theoretical Computer Science 305*(1-3), 347–408.

[35] Sambin, G., Battilotti, G., and Faggian, C. (2000). Basic logic: reflection, symmetry, visibility. *J. Symbolic Logic 65*(3), 979–1013.

[36] Sambin, G. and Valentini, S. (1998). Building up a toolbox for Martin-Löf's type theory: subset theory. In *Twenty-five years of constructive type theory, Proceedings of a Congress held in Venice, October 1995* (ed. G. Sambin and J. Smith), pp. 221–244. Oxford U. P.

[37] Sambin (with the collaboration of S. Gebellato, P. Martin-Löf and V. Capretta), G. (2003, May). The basic picture. Preprint n. 08, Dipartimento di Matematica, Università di Padova.

[38] Scedrov, A. (1985). Intuitionistic set theory. In *Harvey Friedman's research on the foundations of mathematics*, Volume 117 of *Stud. Logic Found. Math.*, pp. 257–284. North-Holland, Amsterdam.

[39] Swaen, M. D. G. (1991). The logic of first order intuitionistic type theory with weak sigma-elimination. *J. Symbolic Logic*, **56**, 467–483.

[40] Swaen, M. D. G. (1992). A characterization of ML in many-sorted arithmetic with conditional application. *J. Symbolic Logic*, **57**, 924–953.

[41] Troelstra, A. and Dalen, D. van (1988). *Constructivism in mathematics. Vol. II An Introduction.*, Volume 123 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co. (Amsterdam).