Weyl's Predicative Classical Mathematics as a Logic-Enriched Type Theory*

Robin Adams and Zhaohui Luo

Dept of Computer Science, Royal Holloway, Univ of London
{robin,zhaohui}@cs.rhul.ac.uk

Abstract. In *Das Kontinuum*, Weyl showed how a large body of classical mathematics could be developed on a purely predicative foundation. We present a logic-enriched type theory that corresponds to Weyl's foundational system. A large part of the mathematics in Weyl's book — including Weyl's definition of the cardinality of a set and several results from real analysis — has been formalised, using the proof assistant Plastic that implements a logical framework. This case study shows how type theory can be used to represent a non-constructive foundation for mathematics.

Key words: logic-enriched type theory, predicativism, formalisation

1 Introduction

Type theories have proven themselves remarkably successful in the formalisation of mathematical proofs. There are several features of type theory that are of particular benefit in such formalisations, including the fact that each object carries a type which gives information about that object, and the fact that the type theory itself has an inbuilt notion of computation.

Most of these applications of type theory have used the doctrine of *propositions-as-types*. They have proven particularly successful for the formalisation of intuitionistic, or constructive, proofs. The correspondence between terms of a type theory and intuitionistic proofs has been well studied. The degree to which type theory can be used for the formalisation of other notions of proof has been investigated to a much lesser degree.

There have been several formalisations of classical proofs by adapting a proof checker intended for intuitionistic mathematics, say by adding the principle of excluded middle as an axiom (such as [Gon05]). But the metatheoretic properties of the type theory thus obtained, and the degree to which that theory corresponds to the practice of classical mathematics, are not well known. For the more exotic schools of mathematics, such as predicativism, the situation is still worse.

* This work is partially supported by the UK EPSRC research grants GR/R84092 and GR/R72259 and EU TYPES grant 510996.

We contend that the intuitions behind type theory apply outside of intuitionistic mathematics, and that the advantages of type theory would prove beneficial when applied to other forms of proof. It is equally natural in classical mathematics to divide mathematical objects into types, and it would be of as much benefit to take advantage of the information provided by an object's type in a classical proof. The notion of computation is an important part of classical mathematics. When formally proving a property of a program, we may be perfectly satisfied with a classical proof, which could well be shorter or easier to find.

We contend that it would be better to develop and study type theories specifically designed for non-constructive mathematical foundations. For this purpose, the systems known as *logic-enriched type theories* (LTTs), proposed by Aczel and Gambino [AG02,GA06], would seem to be particularly appropriate.

LTTs can be considered in a uniform type-theoretic framework that supports formal reasoning with different logical foundations, as proposed in [Luo06]. In particular, this may offer a uniform setting for studying and comparing different mathematical foundations, in the way that predicate logic has in traditional mathematical logic research. For example, when building a foundational system for mathematics, we must decide whether the logic shall be *classical* or *constructive* and whether *impredicative* definitions are allowed, or only *predicative*. Each of the four possible combinations of these options has been advocated as a foundation for mathematics at some point in history. The four possibilities are:

- **Impredicative classical mathematics**. This is arguably the way in which the vast majority of practising mathematicians work (although much of their work can often also be done in the other settings). Zermelo-Fraenkel Set Theory (ZF) is one such foundation.
- **Impredicative constructive mathematics**. Impredicative types theories such as CC [CH88] and UTT [Luo94], or CIC `CITATION` provide its foundations.
- **Predicative classical mathematics**. This was the approach taken by Weyl in his influential monograph of 1918, *Das Kontinuum* [Wey18].
- **Predicative constructive mathematics**. Its foundations are provided, for example, by Martin-Löf's type theory. [NPS90,ML84].

Our type-theoretic framework provides a uniform setting for formalisation of these different mathematical foundations.

In this paper, we present a case study in the type-theoretic framework: to construct an LTT to represent a non-constructive approach to the foundation of mathematics. The LTT shall correspond to the predicative, classical foundational system of mathematics developed by Weyl in his monograph *Das Kontinuum* [Wey18]. We describe a formalisation in that LTT of several of the results proven in the book.

Weyl presents in his book a programme for the development of mathematics on a foundation that is *predicative*; that is, that avoids any definition which involves a 'vicious circle', where an object is defined in terms of a collection of which it is a member. The system presented in the book has attracted interest since, inspiring for example the second-order system ACA_0 [Fef00], which plays an important role in the project of Reverse Mathematics [Sim99]. It is a prominent example of a fully developed non-mainstream mathematical foundation, and so a formalisation should be of quite some interest.

We begin this paper describing in Section 2 in detail the version of Weyl's foundational system we shall be using. We then proceed in Section 3 to describe a logic-enriched type theory within a modified version of the logical framework LF[1] [Luo94]. We claim that this logic-enriched type theory faithfully corresponds to the system presented in Section 2. The formalisation itself was carried out in a modified version of the proof assistant Plastic [CL01], an implementation of LF. We describe the results proven in the formalisation in Section 4. The source code for the formalisation is available online at `http://www.cs.rhul.ac.uk/~robin/weyl`.

2 Weyl's Predicative Foundations for Mathematics

2.1 Weyl and predicativism: a historical account

Hermann Weyl (1885–1955) contributed to many branches of mathematics in his lifetime. His greatest contribution to the foundations of mathematics was the monograph *Das Kontinuum* [Wey18] in 1918, in which he presented a predicative foundation which he demonstrated was adequate for the development of a large body of mathematics.

The concept of predicativity originated with Poincaré [Poi05,Poi06a,Poi06b], who advocated the *vicious circle principle*: a definition of an object is illegitimate if it is defined by reference to a totality that contains the object itself. Thus, we may not quantify over all sets when defining a set (as with Russell's famous 'set of all sets that do not contain themselves'); we may not quantify over all real numbers when defining a real number (as with the least upper bound of a set of reals); and so forth. A definition which involves such a quantification is called *impredicative*; one which does not, *predicative*. The advocacy of the exclusion of impredicative definitions has been given the name *predicativism*.

However much philosophical sympathy we may feel with predicativism, we may worry that, since impredicative definitions are so common in mathematical practice, their exclusion may require us to abandon too much of the mathematical corpus. Weyl's book provides evidence that this is not necessarily the case. In it, he shows how many results that are usually proven impredicatively can be proven predicatively; and that, even for those results that cannot, one can often prove predicatively a weaker result which in practice is just as useful. He does this by laying out a predicative foundation for mathematics, and developing a fairly large body of mathematics on this foundation.

[1] The logical framework LF here is the typed version of Martin-Löf's logical framework [NPS90]. It is different from the Edinburgh LF [HHP93]: besides the formal differences, LF is intended to be used to specify computation rules and hence type theories such as Martin-Löf's type theory [NPS90] and UTT [Luo94]. A recent study of logical frameworks can be found in [Ada04].

A further discussion of the background to and content of Weyl's monograph can be found in Feferman [Fef98].

2.2 Weyl's Foundational System

We shall now present the version of Weyl's foundational system on which we based the formalisation. It differs from the semi-formal system described in *Das Kontinuum* in several details. In particular, we have extended Weyl's system with several features which are redundant in theory, but very convenient practically; these shall be described in the paragraphs headed 'Extensions to Weyl's system' below. Our notation in this section also differs considerably from Weyl's own.

Before turning to the formal details, we begin with a discussion of the intuitions behind Weyl's system, which is constructed following these principles:

1. The natural numbers are accepted as a primitive concept.
2. Sets and relations can be introduced by two methods: explicit definitions, which must be *predicative*; and definition by recursion over the natural numbers.
3. Statements about these objects have a definite truth value; they are either true or false.

Regarding point 2, we are going to provide ourselves with the ability to define sets by *abstraction*: given a formula $\phi[x]$ of the system, to form the set

$$S = \{x \mid \phi[x]\} \ . \tag{1}$$

In order to ensure that every such definition is predicative, we restrict which quantifiers can occur in the formula $\phi[x]$ that can appear in (1): we may quantify over natural numbers, but we may not quantify over sets or functions. In modern terminology, we would say that $\phi[x]$ must contain only first-order quantifiers.

Weyl divides the universe of mathematical objects into collections which he calls *categories*. These categories behave very similarly to the types of a modern type theory. (This is no coincidence: Weyl was influenced by many of the ideas in Russell's theory of types when constructing his system.) For example, there shall be the category of all natural numbers, and the category of all sets of natural numbers. We give a full list of the categories present in the system below.

The categories are divided into *basic* categories, those that may be quantified over in a definition of the form (1); and the *ideal* categories, those that may not. The category of natural numbers shall be a basic category; categories of sets and categories of functions shall be ideal categories. In modern terminology, the basic categories contain first-order objects, while the ideal categories contain second-, third- and higher-order objects.

He proceeds to divide the propositions of his system into the *small*[2] propositions, those which involve quantification over basic categories only, and so may

[2] Weyl chose the German word *finite*, which in other contexts is usually translated as 'finite'; however, we agree with Pollard and Bole [Wey87] that this would be misleading.

occur in a definition of the form (1); and the *large* propositions, those which involve quantification over one or more ideal category, and so may not.

In more detail, here is our version of Weyl's foundational system.

Categories There are a number of *basic categories* and a number of *ideal categories*, each of which has *objects*.

1. There is a basic category $\mathbb{N}$ of natural numbers.
2. Given any categories $A_1, \ldots, A_m$ and $B_1, \ldots, B_n$, we may form the ideal category $(A_1 \times \cdots \times A_m) \to \mathrm{Set}\,(B_1 \times \cdots \times B_n)$ of functions of m arguments that take objects of categories $A_1, \ldots, A_m$, and return sets of n-tuples of objects of categories $B_1, \ldots, B_n$. The number m may be zero here; the number n may not.

   (These were the only categories of functions present in *Das Kontinuum*. For the purposes of our formalisation, we have added categories of functions $A \to B$ for *any* categories A and B; see 'Extensions' below.)

   For example, taking $m = 0$ and $n = 1$ allows us to form the category $\mathrm{Set}\,(B)$, the category of all sets whose elements are of category B. Taking $m = 1$ and $n = 2$ allows us to form the category $A \to \mathrm{Set}\,(B \times C)$, the category of all functions which take an object from A and return a binary relation between the categories B and C.

Propositions

1. There are a number of *primitive relations* that hold between the objects of these categories:
   - the relation 'x is the successor of y' between natural numbers;
   - for any *basic* category, the relation '$x = y$' between objects of that category;
   - the relation $\langle y_1, \ldots, y_n \rangle \in F(x_1, \ldots, x_m)$ where F is of category $(A_1 \times \cdots \times A_m) \to \mathrm{Set}\,(B_1 \times \cdots \times B_n)$, x_i of category A_i and y_i of category B_i.
2. The *small* propositions are those that can be built up from the primitive relations using the operations of substituting objects of the appropriate category for variables, the propositional connectives $\neg$, $\wedge$, $\vee$ and $\to$, and the universal and existential quantifications over the *basic* categories.
3. The *propositions* are those that can be built up from the primitive relations using substitution of objects for variables, the propositional connectives and quantification over *any* categories.

Objects

- **Explicit Definition** Given any *small* proposition $\phi[x_1, \ldots, x_m, y_1, \ldots, y_n]$, we may introduce an object F of category $(A_1 \times \cdots \times A_m) \to \mathrm{Set}\,(B_1 \times \cdots \times B_n)$ by declaring

$$F(x_1, \ldots, x_m) = \{\langle y_1, \ldots, y_n \rangle \mid \phi[x_1, \ldots, x_m, y_1, \ldots, y_n]\} \qquad (2)$$

  Making this declaration has the effect of introducing the axiom

$$\forall \boldsymbol{x}, \boldsymbol{y}(\boldsymbol{y} \in F(\boldsymbol{x}) \leftrightarrow \phi[\boldsymbol{x}, \boldsymbol{y}]) \ . \qquad (3)$$

Principle of Iteration This principle allows us to define functions by recursion over the natural numbers; given a function F from a category $S \to S$, we can form a function G of category $S \times \mathbb{N} \to S$ by setting

$$G(X, n) = F^n(X) \ .$$

G is thus formed by *iterating* the function F.
More formally, let S be a category of the form Set $(B_1 \times \cdots \times B_n)$. Given an object F of category $(A_1 \times \cdots \times A_m \times S) \to S$, we may introduce an object G of category $(A_1 \times \cdots \times A_m \times S \times \mathbb{N}) \to S$ by declaring

$$\left.\begin{aligned} G(x_1, \ldots, x_m, X, 0) &= X \\ G(x_1, \ldots, x_m, X, k+1) &= F(x_1, \ldots, x_m, G(x_1, \ldots, x_m, X, k)) \end{aligned}\right\} \tag{4}$$

where x_i is affiliated with category A_i, X with S, and k with $\mathbb{N}$.
Making these declarations has the effect of introducing the axiom

$$\begin{aligned} &\forall \boldsymbol{x}, \boldsymbol{y}(\langle y_1, \ldots, y_n \rangle \in G(x_1, \ldots, x_m, X, 0) \leftrightarrow \langle y_1, \ldots, y_n \rangle \in X) \\ &\forall \boldsymbol{x}, \boldsymbol{y}, a, b(Sab \to (\langle y_1, \ldots, y_n \rangle \in G(x_1, \ldots, x_m, X, b) \\ &\qquad \leftrightarrow \langle y_1, \ldots, y_n \rangle \in F(x_1, \ldots, x_m, G(x_1, \ldots, x_m, X, a))) \end{aligned} \tag{5}$$

Axioms The theorems of Weyl's system are those that can be derived via *classical* predicate logic from the following axioms:
1. The axioms for the equality relation on the basic categories.
2. Peano's axioms for the natural numbers (including proof by induction).
3. The axioms (3) and (5) associated with any definitions (2) and (4) that have been introduced.

We note that there is a one-to-one correspondence, up to the appropriate equivalence relations, between:

- the objects of category $C = (A_1 \times \cdots \times A_m) \to$ Set $(B_1 \times \cdots \times B_n)$; and
- the small propositions $\phi[x_1, \ldots, x_m, y_1, \ldots, y_n]$, with distinguished free variables x_i of category A_i and y_i of category B_i.

Given any F of category C, the corresponding small proposition is $\boldsymbol{y} \in F(\boldsymbol{x})$. Conversely, given any small proposition $\phi[\boldsymbol{x}, \boldsymbol{y}]$, the corresponding object F of category C is the one introduced by the declaration

$$F(\boldsymbol{x}) = \{\boldsymbol{y} \mid \phi[\boldsymbol{x}, \boldsymbol{y}]\}$$

For F, G of category C, let us write $F \approx G$ for

$$\forall \boldsymbol{x}, \boldsymbol{y}(\boldsymbol{y} \in F(\boldsymbol{x}) \leftrightarrow \boldsymbol{y} \in G(\boldsymbol{x})) \ .$$

For small propositions $\phi[\boldsymbol{x}, \boldsymbol{y}]$, $\psi[\boldsymbol{x}, \boldsymbol{y}]$, let us write $\phi \simeq \psi$ for

$$\forall \boldsymbol{x}, \boldsymbol{y}(\phi[\boldsymbol{x}, \boldsymbol{y}] \leftrightarrow \psi[\boldsymbol{x}, \boldsymbol{y}]) \ .$$

Then we have

$$F \approx G \leftrightarrow (\boldsymbol{y} \in F(\boldsymbol{x})) \simeq (\boldsymbol{y} \in G(\boldsymbol{x})) \ .$$

Further, if we make the declarations

$$H(\boldsymbol{x}) = \{\boldsymbol{y} \mid \phi[\boldsymbol{x}, \boldsymbol{y}]\}, \qquad K(\boldsymbol{x}) = \{\boldsymbol{y} \mid \psi[\boldsymbol{x}, \boldsymbol{y}]\} \ ,$$

then we have

$$\phi \simeq \psi \leftrightarrow H \approx K \ .$$

Thus, the correspondence described is a bijection up to the equivalence relations $\approx$ and $\simeq$.

Extensions to Weyl's System For the purposes of this formalisation, we have added features which were not explicitly present in Weyl's system, but which can justifiably be seen as conservative extensions of the same. We shall allow ourselves the following.

1. We shall introduce a category $A \times B$ of *pairs* of objects, one from the category A and one from the category B. $A \times B$ shall be a basic category when A and B are both basic, and ideal otherwise. This shall allow us, for example, to talk directly about integers (which shall be pairs of natural numbers) and rationals (which shall be pairs of integers).
2. We shall introduce a category $A \to B$ of functions from A to B for all categories (not only the case where B has the form $\mathrm{Set}\,(\cdots)$).
   $A \to B$ shall always be an ideal category. For the system to be predicative, quantification over functions must not be allowed in small propositions; quantifying over $A \to \mathbb{N}$, for example, would provide us with an effective means of quantifying over $\mathrm{Set}\,(A)$. (Recall that, classically, the powersets of a set X and the functions from X to a two-element set are in one-to-one correspondence.)
   Weyl instead defined functions as particular sets of ordered pairs, and showed in detail how addition of natural numbers can be constructed. For the purposes of formalisation, it was much more convenient to provide ourselves with these categories of functions, and the ability to define functions by recursion, from the very beginning.

We shall permit ourselves to use a function symbol 's' for successor, rather than only the binary relation Sxy.

We also allow the definition of objects of any category (not just those of the form $\boldsymbol{A} \to \mathrm{Set}\,(\boldsymbol{B})$) by recursion over the natural numbers. (Weyl shows through several examples how this is possible, including a detailed construction of the addition of natural numbers.)

Minor Differences We have diverged from Weyl's system in two other, more minor, ways which should be noted. We choose to start the natural numbers at 0, whereas Weyl begins at 1; and, when we come to construct the real numbers, we follow the sequence of constructions $\mathbb{N} \longrightarrow \mathbb{Z} \longrightarrow \mathbb{Q} \longrightarrow \mathbb{R}$ rather than Weyl's $\mathbb{N} \longrightarrow \mathbb{Q}^+ \longrightarrow \mathbb{Q} \longrightarrow \mathbb{R}$.

3 Weyl's Foundation as a Logic-Enriched Type Theory

What immediately strikes a modern eye reading *Das Kontinuum* is how similar the system presented there is to what we now know as a type theory; almost the only change needed is to replace the word 'category' with 'type'. (This is no coincidence — Weyl was influenced by many of the ideas in Russell's theory of types when constructing his system.) In particular, Weyl's system is very similar to a *logic-enriched type theory* (LTT for short).

The concept of an LTT, an extension of the notion of type theory, was proposed by Aczel and Gambino in their study of type-theoretic interpretations of constructive set theory [AG02,GA06]. It is very close to the internal language of a topos. A type-theoretic framework, which formulates LTTs in a logical framework, has been proposed in [Luo06] to support formal reasoning with different logical foundations. In particular, it adequately supports classical inference with a notion of predicative set, as described below.

An LTT consists of a type theory augmented with a separate, primitive mechanism for forming and proving propositions. When working with a logic-enriched type theory, we thus do not use any version of the doctrine of propositions-as-types. Rather, we introduce a new syntactic class of *formulas*, and new judgement forms for a formula being a well-formed proposition, and for a proposition being provable from given hypotheses.

An LTT thus has two rigidly separated components or 'worlds': the *datatype* world of terms and types, and the *logical* world of proofs and propositions, which is used for describing and reasoning about the datatype world.[3] In particular, we can form propositions by *quantification* over a type; and prove propositions by *induction*.

In this work, we shall also allow the datatype world to depend on the logical world in just one way: by permitting the formation of *sets*. Given a proposition $\phi[x]$, we shall allow the construction of the set $\{x \mid \phi[x]\}$, which shall live in the datatype world; thus, a set shall be a term that depends on a proposition. (Note that these sets are not themselves types.) This shall be the only way in which the datatype world may depend on the logical world, however; in particular, no type may depend on a proposition, and no type, term or proposition may depend on a proof.

We start by extending the logical framework LF with a kind *Prop*, standing for the world of logical propositions. Then, we introduce a type for each category: a construction in *Prop* for each method of forming propositions; a type universe U of names of the basic categories; and a propositional universe *prop* of names of the small propositions. Thus constructed, the LTT with predicative sets corresponds extremely closely to Weyl's foundational system.

We have built *prop* as a universe à la Tarski; that is, its objects are *names* of small propositions. Plastic does not provide the necessary mechanism for defining

[3] This is very much in line with the idea that there should be a clear separation between logical propositions and data types, as advocated in the development of type theories ECC and UTT [Luo94].

1. Rules of Deduction for $Type$ and El

$$\frac{\Gamma \text{ valid}}{\Gamma \vdash Type \text{ kind}} \qquad \frac{\Gamma \vdash A : Type}{\Gamma \vdash El\,(A) \text{ kind}} \qquad \frac{\Gamma \vdash A = B : Type}{\Gamma \vdash El\,(A) = El\,(B)}$$

2. Rules of Deduction for $Prop$ and Prf

$$\frac{\Gamma \text{ valid}}{\Gamma \vdash Prop \text{ kind}} \qquad \frac{\Gamma \vdash P : Prop}{\Gamma \vdash Prf\,(P) \text{ kind}} \qquad \frac{\Gamma \vdash P = Q : Prop}{\Gamma \vdash Prf\,(P) = Prf\,(Q)}$$

Fig. 1: Kinds $Type$ and $Prop$ in LF$'$

prop as a universe à la Russell, where its objects would be the small propositions themselves. We suspect that the choice makes no practical difference.

3.1 Logic-Enriched Type Theories in Logical Frameworks

There exist today many *logical frameworks*, designed as systems for representing many different type theories. It requires only a small change to make a logical framework capable of representing LTTs as well.

For this work, we have used the logical framework LF [Luo94], which is the basis for the proof checker Plastic [CL01]. LF provides a kind $Type$ and a kind constructor El. To make LF capable of representing LTTs, we add a kind $Prop$ and a kind constructor Prf. We shall refer to this extended framework as LF$'$.

Recall that a logical framework, such as LF or LF$'$, is intended as a metalanguage for constructing various type theories, the *object systems*. The frameworks consist of *kinds* and *objects*. The object systems are constructed in the framework by representing their expressions by certain objects. An LTT consists of *terms* and *types* (in the datatype world), and *propositions* and *proofs* (in the logical world). We shall build an LTT in LF$'$by representing:

- the *types* by the objects of kind $Type$;
- the *terms* of type A by the objects of kind $El\,(A)$;
- the *propositions* by the objects of kind $Prop$;
- the *proofs* of the proposition ϕ by the objects of kind $Prf\,(\phi)$.

The rules of deduction for these new kinds $Prop$ and $Prf\,(\cdots)$ are given in Figure 1, along with the rules those for $Type$ and El, for comparison.

These new kinds allow us to form judgements of the following forms:

- $\Gamma \vdash \phi : Prop$, indicating that ϕ is a well-formed proposition;
- $\Gamma \vdash P : Prf\,(\phi)$, indicating that P is a proof of the proposition ϕ;
- $\Gamma, p_1 : Prf\,(\phi_1), \ldots, p_n : Prf\,(\phi_n) \vdash P : Prf\,(\psi)$, indicating that ψ is derivable from the hypotheses $\phi_1, \ldots, \phi_n$, with the object P encoding the derivation; this is the state of affairs that was denoted by $\Gamma \vdash \phi_1, \ldots, \phi_n \Rightarrow \psi$ in [AG02].

When formalizing a piece of mathematics using an LTT, the provable propositions are those ϕ for which $Prf(\phi)$ is inhabited. We state each theorem by forming the appropriate object ϕ of kind *Prop*, and then show that it is provable by constructing an object P of kind $Prf(\phi)$. (Due to its novelty, we shall *not* omit the constructor Prf in this paper.)

We also obtain judgements of the form $\Gamma \vdash \phi = \psi : Prop$. Judgements of this last form express that ϕ and ψ are *intensionally equal* propositions — that is, that ψ can be obtained from ϕ by a sequence of reductions and expansions of subterms. This is not to be confused with logical equivalence; intensional equality is a much stronger relation. The distinction is similar to that between judgemental equality (convertibility) and propositional equality between terms.

We recall that a type theory is specified in LF by declaring a number of *constants* with their kinds, and declaring several *computation rules* to hold between objects of some kinds of LF. An LTT can be specified in LF′ by making the above declarations for each constructor in its datatype component, and also declaring:

- for each logical constant (connective or quantifier) we wish to include, a constant of kind $(\cdots)Prop$;
- for each rule of deduction by which a proposition may be proved, a constant of kind $(\cdots)Prf(\phi)$
- a number of *computation rules for propositions*, of the form $(\cdots)(\phi = \psi : Prop)$

It was essential for this work that the logical framework we use be capable of representing computation rules. A framework such as Twelf `CITATION`, for example, would not be suitable for our purposes.

LTTs and Type Theories Compared When using a type theory for formalisation, we identify each proposition with a particular type, and show that a theorem is provable by constructing a term of the corresponding type. The way we prove propositions in an LTT by constructing an object of kind $Prf(\cdots)$ is very similar. However, there are two important differences to be noted:

- We have separated the datatypes from the propositions. This allows us to add axioms without changing the datatype world. We can, for example, add the axiom Pierce without thereby causing all the function types $((A \to B) \to A) \to A$ to be inhabited.
- We do not have any computation rules on proofs: only on terms, types and propositions. Further, a proof cannot occur inside a term, type or proposition. We can thus add whatever axioms we want to the logic: we know that, by adding the axiom Pierce (say), we shall not affect any of the properties of the reduction relation, such as decidability of convertibility or strong normalisation.

Natural Numbers

$$\begin{aligned}
\mathbb{N} &: Type \\
0 &: \mathbb{N} \\
\mathrm{s} &: (\mathbb{N})\mathbb{N} \\
\mathrm{E}_{\mathbb{N}} &: (C : (\mathbb{N})Type)(C0)((x : \mathbb{N})(Cx)C(\mathrm{s}x))(n : \mathbb{N})Cn \\
\mathrm{Ind}_{\mathbb{N}} &: (P : (\mathbb{N})Prop)(Prf\,(P0))((x : \mathbb{N})(Prf\,(Px))Prf\,(P(\mathrm{s}x)))(n : \mathbb{N})Prf\,(Pn)
\end{aligned}$$

$$\begin{aligned}
\mathrm{E}_{\mathbb{N}}\,C\,a\,b\,0 &= a : El\,(C0) \\
\mathrm{E}_{\mathbb{N}}\,C\,a\,b\,(\mathrm{s}n) &= b\,n\,(E_{\mathbb{N}}\,C\,a\,b\,n) : El\,(C(\mathrm{s}n))
\end{aligned}$$

Implication

$$\begin{aligned}
\supset &: (Prop)(Prop)Prop \\
\supset \mathrm{I} &: (P : Prop)(Q : Prop)((Prf\,(P))Prf\,(Q))Prf\,(\supset P\,Q) \\
\supset \mathrm{E} &: (P : Prop)(Q : Prop)(Prf\,(\supset P\,Q))(Prf\,(P))Prf\,(Q)
\end{aligned}$$

Pierce's Law

$$\mathrm{Pierce} : (P : Prop)(Q : Prop)(((Prf\,(P))Prf\,(Q))Prf\,(P))Prf\,(P)$$

Fig. 2: Declaration of an LTT in LF′

3.2 Natural Numbers, Products, Functions and Predicate Logic

We can now proceed to construct a logic-enriched type theory that corresponds to the foundational system Weyl presents in *Das Kontinuum.*

Our starting point is an LTT that contains, in its datatype component, a type $\mathbb{N}$ of natural numbers, as well as non-dependent product and function types $A \times B$ and $A \to B$; and, in its logical component, classical predicate logic. We present some of the declarations involved in its specification in Figure 2, namely those involving natural numbers (including $\mathrm{E}_{\mathbb{N}}$, which permits the definition of functions by recursion, and $\mathrm{Ind}_{\mathbb{N}}$, which permits propositions to be proven by induction) and implication. The other types and logical constants follow a similar pattern. We also include a version of Pierce's Law to ensure the logic is classical.

Note that we do not attach any computational content to Pierce's Law. We know that adding such constants as Pierce will not affect the reduction behaviour of the system, as there are no computation rules involving proofs. When working with LTTs, we can add whatever axioms we wish in the logical world without affecting such properties as decidability of judgemental equality and type checking, and strong normalisation.

3.3 Type Universes and Propositional Universes

We have now introduced our collection of categories: they are the objects of kind $Type$. We still however need to divide them into the basic and ideal categories.

The device we need to do this is one with which we are familiar: that of a *type universe*. A type universe U (à la Tarski) is a type whose objects are names of types. Intuitively, the types that have a name in U are often thought of as the 'small' types, and those that do not (such as U itself) as the 'large' types. Together with U, we introduce a constant T such that, for each name $a : U$, $T(a)$ is the type named by a.

For our system, we provide ourselves with a universe whose objects are the names of the *basic* categories. We thus need a universe U that contains a name for $\mathbb{N}$, and a method for constructing a name for $A \times B$ out of a name for A and a name for B. This is done in Figure 3(1). We also introduce a relation of equality for every basic category.

Now we need to divide our propositions into the small propositions and the large propositions. To do so, we use the notion in the logical world which is analagous to a type universe: a *propositional universe*.

We wish to introduce the collection *prop* of names of the *small* propositions; that is, the propositions that only involve quantification over small types. It is not immediately obvious where this collection should live. One the one hand, we could declare $prop : Type$; but this would violate the idea that the objects of $Type$ should be Weyl's categories and nothing more.

Instead, we declare $prop : Prop$. Now, it must be admitted that *prop* is not conceptually a proposition; it does not assert any relation to hold between any mathematical objects. However, it seems to make little practical difference which choice is made. Choosing to place *prop* in *Prop* provides a pleasing symmetry with U and $Type$, and *prop* seems to belong more to the logical world than the datatype world. Until more foundational work on LTTs has been done, we accept this compromise: *prop* is a 'proposition', each of whose 'proofs' is a name of a small proposition.[4]

As with the type universe, when we introduce a propositional universe *prop* we provide ourselves with a constant V such that, for each name $p : prop$, $V(p)$ is the proposition named by p. We also provide constants that reflect equality, the propositional connectives, and quantification over the basic types. The declarations are given in Figure 3(2). Note that the propositional universe provides us with our first examples of computation rules for propositions.

3.4 The Predicative and Impredicative Notion of Set

We now have all the machinery necessary to be able to introduce *typed sets*. For any type A, we wish to introduce the type $\mathrm{Set}\,(A)$ consisting of all the sets that can be formed, each of whose members is an object of type A. (Thus we do not have any sets of mixed type.) We take a set to be introduced by a *small predicate* over A; that is, an object of kind $(A)prop$, a function which takes objects of A and returns (a name of) a small proposition.

We therefore make the declarations given in Figure 4:

- Given any type A, we can form the type $\mathrm{Set}\,(A)$. The terms of $\mathrm{Set}\,(A)$ are all the sets that can be formed whose elements are terms of type A.

[4] Other alternatives would be to introduce a new top-kind to hold *prop*, or to make *prop* itself a top-kind. We do not discuss these here.

1. The Type Universe

$$\begin{aligned}
U &: Type\\
T &: (U)Type\\
\hat{\mathbb{N}} &: U\\
\hat{\times} &: (U)(U)U\\
T(\hat{\mathbb{N}}) = \mathbb{N} &: Type\\
T(\hat{\times} a\, b) = \times(Ta)\,(Tb) &: Type
\end{aligned}$$

Propositional Equality

$$\begin{aligned}
\simeq &: (A:U)(TA)(TA)Prop\\
\simeq I &: (A:U)(a:TA)\simeq A\,a\,a\\
\simeq E &: (A:U)(P:(TA)Prop)(a,b:TA)\\
&\quad (Prf\,(\simeq A\,a\,b))(Prf\,(Pa))Prf\,(Pb)
\end{aligned}$$

2. The Propositional Universe

$$\begin{aligned}
prop &: Prop\\
V &: (prop)Prop\\
\hat{\bot} &: prop\\
\hat{\supset} &: (prop)(prop)prop\\
\hat{\forall} &: (a:U)((Ta)prop)prop\\
\hat{\simeq} &: (a:U)(Ta)(Ta)prop\\
V(\hat{\bot}) = \bot &: Prop\\
V(\hat{\supset} p\, q) = \supset(Vp)\,(Vq) &: Prop\\
V(\hat{\forall} a\, p) = \forall(Ta)\,[x:Ta]V(px) &: Prop\\
V(\hat{\simeq} a\, s\, t) &= \simeq(Ta)\,s\,t
\end{aligned}$$

Fig. 3: A Type Universe and a Propositional Universe

$$\begin{aligned}
\mathrm{Set} &: (Type)Type\\
\mathrm{set} &: (A:Type)((A)prop)\mathrm{Set}\,(A)\\
\in &: (A:Type)(A)(\mathrm{Set}\,(A))prop\\
\in A\,a\,(\mathrm{set}\,A\,P) = Pa &: prop
\end{aligned}$$

Fig. 4: The Predicative Notion of Set

- Given a *small* proposition $\phi[x]$ with variable x of type A, we can form the set $\{x : \phi[x]\}$. Formally, given a name $p[x] : prop$ of a small proposition containing x, we can the set $\mathrm{set}\,A\,([x : A]p[x])$, which we shall write as $\{x : V(p[x])\}$.
- If $a : A$ and $X : \mathrm{Set}\,(A)$, we can form the proposition $\in A\,a\,X$, which we shall write as $a \in X$.
- Finally, we want to ensure that the elements of the set $\{x : \phi[x]\}$ are precisely the terms a such that $\phi[a]$ is true. This is achieved by adding our second example of a computation rule on propositions, the last line on Figure 4, which we may read as: $a \in \{x : \phi[x]\}$ computes to $\phi[a]$.

As $\mathrm{Set}\,(A)$ is always to be an ideal category, we do not provide any means for forming a name of $\mathrm{Set}\,(A)$ in U.

These sets are not themselves types; they are terms of the types $\mathrm{Set}\,(\cdots)$, and the membership condition $a \in A$ is a proposition, and not a typing judgement. In particular, we distinguish between a type A and the set $\{x : A \mid \top\}$ of type $\mathrm{Set}\,(A)$.

A similar construction could be carried out in an LTT if we wished to work in an impredicative setting, simply by replacing *prop* with *Prop* throughout Figure 4. This would allow us to form the set $\{x : \phi[x]\}$ for *any* proposition $\phi[x]$. (See [Luo06] for more details.) Thanks to the similarity of the two approaches, much if not all of the work done in the predicative system could be reused in the impredicative system. We shall return to this point in Section 4.2.

4 Formalisation in Plastic

We have formalised this work in a version of the proof assistant Plastic [CL01], modified by Paul Callaghan to be an implementation of LF′. We have produced a formalisation which includes all the definitions and proofs of several of the results from Weyl's book.

In Plastic, all lines that are to be parsed begin with the character `>`; any line that does not is a comment line. A constant c may be declared to have kind $(x_1 : K_1) \cdots (x_n : K_n)K$ by the input line

$$\texttt{> [}c\texttt{[}x_1\texttt{:}K_1\texttt{]} \cdots \texttt{[}x_n\texttt{:}K_n\texttt{]}\ \texttt{:}\ K\texttt{];}\ .$$

We can define the constant c to be the object $[x_1 : K_1] \cdots [x_n : K_n]k$ of kind $(x_1 : K_1) \cdots (x_n : K_n)K$ by writing

$$\texttt{> [}c\texttt{[}x_1\texttt{:}K_1\texttt{]} \cdots \texttt{[}x_n\texttt{:}K_n\texttt{]}\ \texttt{=}\ k\ \texttt{:}\ K\texttt{];}$$

In both these lines, the kind indicator $:K$ is optional, and in practice is usually omitted.

While Plastic at present does not allow implicit arguments, we can replace any object with a 'meta-variable' `?`, indicating that we wish Plastic to infer its value.

These are the only features of the syntax that we shall use in this paper.

4.1 Results Proven

Cardinality of Sets Weyl's system contains a category of natural numbers, and categories of sets. Weyl was able to define the predicate 'the set X has exactly n members' in the following manner, which shows the power of the principle of iteration (the ability to define functions into ideal categories by recursion).

Given a basic category A, define the function $K : \mathbb{N} \to \mathrm{Set}\,(\mathrm{Set}\,(A))$ by recursion as follows. The intention is that $K(n)$ is the set of all sets $X : \mathrm{Set}\,(A)$ that have at least n members.

$$\begin{aligned} K(0) &= \{X \mid \top\} \\ K(n+1) &= \{X \mid \exists a(a \in X \wedge X \setminus \{a\} \in K(n))\} \end{aligned}$$

In Plastic, this is done as follows:

```
> [at_least_set [tau : U] = E_Nat ([_ : Nat] Set (Set (T tau)))
>     (full (Set (T tau)))
>     [n : Nat] [Kn : Set (Set (T tau))] set (Set (T tau))
>        [X : Set (T tau)] ex tau [a : T tau]
>           and (in (T tau) a X) (in ? (setminus' tau X a) Kn)];
```

We can now define the proposition 'X has at least n members' to be $X \in K(n)$.

```
> [At_Least [tau : U] [X : Set (T tau)] [n : Nat]
>    = In ? X (at_least_set tau n)];
```

For n a natural number, define the *cardinal number* $\overline{n}$ to be $\{x \mid x < n\} : \mathrm{Set}\,(\mathbb{N})$.

```
> [card [n : Nat] = set Nat [x : Nat] lt x n];
```

Define the *cardinality* of a set A to be $|A| = \{n \mid A \text{ has at least } \mathrm{s}n \text{ members}\} : \mathrm{Set}\,(\mathbb{N})$.

```
> [cardinality [tau : U] [A : Set (T tau)]
>    = set Nat [n : Nat] at_least tau A (succ n)];
```

We can prove the following result:

> For any set $X : \mathrm{Set}\,(A)$, its cardinality $|X|$ is either $\{x \mid \top\}$ or $\overline{n}$ for some n.

We thus have two classes of cardinal numbers: $\overline{n}$, for measuring the size of finite sets, and $\{x \mid \top\}$, which we denote by ∞, for measuring the size of infinite sets. (There is thus only one infinite cardinality in *Das Kontinuum*.) We define 'X has exactly n members' to be $|X| \approx \overline{\mathrm{s}n}$, where $\approx$ denotes the following equivalence relation on sets:

$$X \approx Y \equiv \forall x(x \in X \leftrightarrow x \in Y) \ .$$

```
> [infty = full Nat];
> [Exactly [tau : U] [A : Set (T tau)] [n : Nat]
>    = Seteq Nat (cardinality tau A) (card (succ n))];
```

With these definitions, we can prove results such as the following:

1. If A has at least n elements and $m \leq n$, then A has at least m elements.
2. If A has exactly n elements, then $m \leq n$ iff A has at least m elements.
3. If A has exactly m elements, B has exactly n elements, and A and B are disjoint, then $A \cup B$ has exactly $m + n$ elements.

We have thus provided definitions of the concepts 'having at least n members' and 'having exactly n members' in such a way that the sets

$$\{X \mid X \text{ has at least } n \text{ members}\} \text{ and } \{X \mid X \text{ has exactly } n \text{ members}\}$$

are definable predicatively. This would not be possible if, for example, we defined 'X has exactly n elements' as the existence of a bijection between X and $\overline{\mathrm{s}n}$, as this would involve quantification over the ideal category $A \to \mathbb{N}$. It also cannot be done as directly in a predicative system of second order arithmetic such as ACA_0 [Sim99].

Construction of the Reals The set of real numbers is constructed by the following process. We first define the type of integers $\mathbb{Z}$, with a defined relation of equality $\approx_{\mathbb{Z}}$. We then define a *rational* to be a pair of integers, the second of which is non-zero. That is, for $q : \mathbb{Z} \times \mathbb{Z}$, we define the proposition 'q is rational' by

$$\langle x, y \rangle \text{ is rational} \equiv y \not\approx_{\mathbb{Z}} 0 \ .$$

We proceed to define equality of rationals $q \approx_{\mathbb{Q}} q'$, addition, multiplication and ordering on the rationals.

A *real* is a Dedekind cut of rationals; that is, an object R of the category $\mathrm{Set}\,(\mathbb{Z} \times \mathbb{Z})$ that:

- is a *domain of rationals*; if q and q' are rationals, $q \in R$, and $q \approx_{\mathbb{Q}} q'$, then $q' \in R$;
- is *closed downwards*; if q and q' are rationals, $q \in R$, and $q' < q$, then $q' \in R$;
- has no maximal element; for every rational $q \in R$, there exists a rational $q' \in R$ such that $q < q'$;
- and is neither empty nor full; there exists a rational q such that $q \in R$, and a rational q' such that $q' \notin R$.

Equality of reals is defined to be extensional equality restricted to the rationals:

$$R \approx_{\mathbb{R}} S \equiv \forall q (q \text{ is rational} \rightarrow (q \in R \leftrightarrow q \in S))$$

We note that, in this formalisation, there was no way to define the collection of rationals as a type, say as the 'sigma-type' '$(\Sigma q : \mathbb{Z} \times \mathbb{Z}) q$ is rational'. This is because our LTT offers no way to form a type from a type $\mathbb{Z} \times \mathbb{Z}$ and a proposition 'q is rational'.

Real Analysis Weyl was keen to show that his predicative system was strong enough to be used for mathematical work by demonstrating that, while several traditional theorems cannot be proven within it, we can usually prove a version of the theorem that is only slightly weaker.

For example, it seems not to be provable predicatively the *least upper bound principle*: that every set A of real numbers bounded above has a least upper bound l. Impredicatively, we would define l to be the union of A. This cannot be done predicatively, as it involves quantification over real numbers. However, we can prove the following two statements, one of which is usually enough for any practical purpose:

1. Every set S of *rational* numbers bounded above has a unique (real) least upper bound l. Take $l = \{q \in \mathbb{Q} \mid (\exists q' \in S) q < q'\}$.
2. Every *sequence* $r_1, r_2, \ldots$ of real numbers bounded above has a unique least upper bound l. Take $l = \{q \in \mathbb{Q} \mid (\exists n : \mathbb{N}) q \in r_n\}$.

These involve only quantification over the rationals and the natural numbers, respectively. (We note that either of these is equivalent to the least upper bound principle in an impredicative setting.)

The first of these is enough to prove the classical version of the Intermediate Value Theorem:

> If $f : \mathrm{Set}\,(\mathbb{Z} \times \mathbb{Z}) \to \mathrm{Set}\,(\mathbb{Z} \times \mathbb{Z})$ is a continuous function from the reals to the reals, and $f(a) < v < f(b)$ for some reals a, b, v with $a < b$, then there exists a real c such that $a < c < b$ and $f(c) = v$.

Weyl proves this proposition by taking c to be the least upper bound of the set of all rationals q such that $a < q < b$ and $f(q) < v$. For the formalisation, it was more convenient to define c directly:

$$c = \{q \in \mathbb{Q} \mid (\exists q' \in \mathbb{Q}) q < q' < b \wedge f(q') < v\} \ .$$

4.2 An Impredicative Development

As mentioned in Section 3.4, it would not be difficult to modify this formulation to get a development of the same theorems in an impredicative system. All we have to do is remove the distinction between large and small propositions.

Recall that our propositional universe was introduced by the constructors in Figure 3(2). In principle, we could simply replace these constant declarations with

$$\begin{array}{ll} prop = Prop & V = [x : Prop]Prop \\ \hat{\bot} = \bot & \hat{\supset} = \supset \\ \hat{\forall} = [A : U]\forall(TA) & \hat{\simeq} = \simeq \end{array}$$

However, at present plastic becomes unstable when definitions are made at the top-kind level such as $prop = Prop$.

Alternatively, we can add two impredicative quantifiers to *prop*, together with their computation rules:

$$\begin{array}{ll} \overline{\forall} : (A : Type)((A)prop)prop & V(\overline{\forall} AP) = \forall A([x : A]V(Px)) \\ \overline{\exists} : (A : Type)((A)prop)prop & V(\overline{\exists} AP) = \exists A([x : A]V(Px)) \end{array}$$

Now *prop*, which determines the collection of propositions over which sets can be formed, covers the whole of *Prop*. We can form the set $\{x \mid \phi[x]\}$ for any well-formed proposition $\phi[x]$. However, all our old proof files written in terms of *prop* and V still parse.[5]

Once this change has been made, we can go on to prove the statement that every set of real numbers bounded above has a least upper bound.

It would be interesting to carry this out to develop impredicative analysis in out setting on the one hand, and to study the reuse of proof development on the other.

[5] The same effect could also be made by changing the construction of U, making it to equal *Type*; or by making both these changes (to *prop* and U).

5 Conclusion

We have conducted a case study in Plastic of the use of a type-theoretic framework to construct a logic-enriched type theory as the basis for a formalisation of a non-constructive system of mathematical foundations, namely that presented in Weyl's *Das Kontinuum*. As a representation of Weyl's work, it is arguably better in some ways than such second-order systems as ACA_0 [Fef00], since we can form a definition of the cardinality of a set that is much closer to Weyl's own. The formalisation work required only a minor change to the existing logical framework implemented in Plastic, allowing us to preserve all the features of Plastic with which we were already familiar.

Future work includes comparison of the type-theoretic framework with other systems such as ACA_0. It would also be interesting to carry out the impredicative development of analysis in our setting, reusing the predicative proof development.

Further, it was easy to modify the development to produce an impredicative system, with the proof files produced in the predicative system being fully reusable.

Acknowledgements Thanks go to Paul Callaghan for his efforts in extending Plastic to implement the type-theoretic framework which has made the reported case study possible and Peter Aczel for his comments during his visit to Royal Holloway.

References

[Ada04] R. Adams. *A Modular Hierarchy of Logical Frameworks*. PhD thesis, University of Manchester, 2004.

[AG02] Peter Aczel and Nicola Gambino. Collection principles in dependent type theory. In Paul Callaghan, Zhaohui Luo, James McKinna, and Robert Pollack, editors, *Types for Proofs and Programs: International Workshop, TYPES 2000, Durham, UK, December 8–12, 2000. Selected Papers*, volume 2277 of *LNCS*, pages 1–23. Springer-Verlag, 2002.

[CH88] Th. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76(2/3), 1988.

[CL01] P. C. Callaghan and Z. Luo. An implementation of typed LF with coercive subtyping and universes. *J. of Automated Reasoning*, 27(1):3–27, 2001.

[Fef98] S. Feferman. Weyl vindicated. In *In the Light of Logic*, pages 249–283. Oxford University Press, 1998.

[Fef00] S. Feferman. The significance of Hermann Weyl's Das Kontinuum. In V. Hendricks et al, editor, *Proof Theory – Historical and Philosophical Significance, Vol 292 of Synthese Library*. 2000.

[GA06] N. Gambino and P. Aczel. The generalised type-theoretic interpretation of constructive set theory. *J. of Symbolic Logic*, 71(1):67–103, 2006.

[Gon05] G. Gonthier. A computer checked proof of the four colour theorem, 2005.

[HHP93] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, 1993.

[Luo94] Zhaohui Luo. *Computation and Reasoning: A Type Theory for Computer Science*. Number 11 in International Series of Monographs on Computer Science. Oxford University Press, 1994.

[Luo06] Z. Luo. A type-theoretic framework for formal reasoning with different logical foundations. In M. Okada and I. Satoh, editors, *Proc of the 11th Annual Asian Computing Science Conference*. Tokyo, 2006.

[ML84] Per Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

[NPS90] B. Nordström, K. Petersson, and J. Smith. *Programming in Martin-Löf's Type Theory: An Introduction*. Oxford University Press, 1990.

[Poi05] H. Poincaré. Les mathématiques et la logique. *Revue de Métaphysique et de Morale*, 13:815–35, 1905.

[Poi06a] H. Poincaré. Les mathématiques et la logique. *Revue de Métaphysique et de Morale*, 14:17–34, 1906.

[Poi06b] H. Poincaré. Les mathématiques et la logique. *Revue de Métaphysique et de Morale*, 14:294–317, 1906.

[Sim99] S. Simpson. *Subsystems of Second-Order Arithmetic*. Springer-Verlag, 1999.

[Wey18] Hermann Weyl. *Das Kontinuum*. 1918. Translated as [Wey87].

[Wey87] Hermann Weyl. *The Continuum: A Critical Examination of the Foundation of Analysis*. Thomas Jefferson University Press, Kirksville, Missouri, 1987. Translated by Stephen Pollard and Thomas Bole.