Thorsten Altenkirch
Conor McBride (Eds.)

LNCS 4502

Types for Proofs and Programs

International Workshop, TYPES 2006
Nottingham, UK, April 2006
Revised Selected Papers

Springer

TP311.1-53
T??
2006

Thorsten Altenkirch   Conor McBride (Eds.)

Types for Proofs and Programs

International Workshop, TYPES 2006
Nottingham, UK, April 18-21, 2006
Revised Selected Papers

Springer

Volume Editors

Thorsten Altenkirch
Conor McBride
University of Nottingham
School of Computer Science and Information Technology
Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK
E-mail: {txa, ctm}@cs.nott.ac.uk

Library of Congress Control Number: 2007936170

CR Subject Classification (1998): F.3.1, F.4.1, D.3.3, I.2.3

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN	0302-9743
ISBN-10	3-540-74463-0 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-74463-4 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper   SPIN: 12111886   06/3180   5 4 3 2 1 0

Preface

These proceedings contain a selection of refereed papers presented at or related to the Annual Workshop of the TYPES project (EU coordination action 510996), which was held April 18–21, 2006 at the University of Nottingham, UK.

The topic of this workshop was formal reasoning and computer programming based on type theory: languages and computerized tools for reasoning, and applications in several domains such as analysis of programming languages, certified software, formalization of mathematics and mathematics education.

The workshop was attended by more than 100 researchers and included more than 60 presentations. We also had the pleasure of three invited lectures, from Bart Jacobs (University of Nijmegen), Hongwei Xi (Boston University) and Simon Peyton Jones (Microsoft Research). Simon Peyton Jones spoke in a joint session with the workshop on Trends in Functional Programming (TFP), which was co-located with the TYPES conference.

From 29 submitted papers, 17 were selected after a reviewing process. The final decisions were made by the editors.

This workshop followed a series of meetings of the TYPES working group funded by the European Union (IST project 29001, ESPRIT Working Group 21900, ESPRIT BRA 6435). The proceedings of these workshop were published in the LNCS series:

TYPES 1993	Nijmegen, The Netherlands, LNCS 806
TYPES 1994	Båstad, Sweden, LNCS 996
TYPES 1995	Turin, Italy, LNCS 1158
TYPES 1996	Aussois, France, LNCS 1512
TYPES 1998	Kloster Irsee, Germany, LNCS 1657
TYPES 1999	Lökeborg, Sweden, LNCS 1956
TYPES 2000	Durham, UK, LNCS 2277
TYPES 2002	Berg en Dal, The Netherlands, LNCS 2646
TYPES 2003	Turin, Italy, LNCS 3085
TYPES 2004	Jouy-en-Josas, France, LNCS 3839

ESPRIT BRA 6453 was a continuation of ESPRIT Action 3245, Logical Frameworks: Design, Implementation and Experiments. Proceedings for annual meetings under that action were published by Cambridge University Press in the books *Logical Frameworks* and *Logical Environments*, edited by Gérard Huet and Gordon Plotkin.

We are grateful for the support of the School of Computer Science and Information Technology at the University of Nottingham in organizing the meeting. We should like to thank James Chapman, Wouter Swierstra and Peter Morris, who helped with the administration and coordination of the meeting. We are also grateful to Peter Morris for help in the preparation of the volume.

March 2007

Thorsten Altenkirch
Conor McBride

Referees

A. Abel
P. Aczel
R. Adams
R. Atkey
S. van Bakel
C. Ballarin
S. Berardi
Y. Bertot
A. Bove
E. Brady
P. Callaghan
J. Carlstrom
J. Cheney
J. Chrząszcz
M. Coppo
J. Courant
C. Coquand
T. Coquand
R. Crole
R. Davies
J. Despeyroux
L. Dixon
G. Dowek
R. Dychhoff
M. Escardo
J-C. Filliâtre
M. Fluet
P. Fontaine
N. Gambino
H. Geuvers
N. Ghani
A. Gordon
B. Grégoire
P. Hancock
J. Harrison
M. Huisman
B. Jacobs
S. Jost
T. Kelsey
J. Lipton
Z. Luo
M.E. Maietti
J. McKinna
M. Miculan
A. Miquel
P. Morris
S. Negri
M. Oostdijk
R. Paterson
D. Pattinson
R. Pollack
T. Ridge
G. Sambin
T. Streicher
C. Urban
D. Walukiewicz-Chrząszcz
S. Weirich
A. Weiermann
B. Werner
F. Wiedijk

¥484.00元

Table of Contents

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.

Keywords: 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.

These applications of type theory 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 to what degree 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.

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

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

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 further contend that it is worth developing and studying 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 [BC04] provide its foundations.
- **Predicative classical mathematics.** This was the approach taken by Weyl in his influential monograph of 1918, *Das Kontinuum* [Weyl18].
- **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; namely the predicative, classical foundational system of mathematics developed by Weyl in his monograph *Das Kontinuum* [Weyl18]. 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

Hermann Weyl (1885–1955) contributed to many branches of mathematics in his lifetime. His greatest contribution to the foundations of mathematics was the book *Das Kontinuum* [Weyl18] in 1918, in which he presented a predicative foundation which he showed was adequate for a large body of mathematics.

The concept of predicativity originated with Poincaré [Poi06], 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.

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

2.1 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;

[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].

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 *predicative* definitions, and definition by recursion over the natural numbers.
3. Statements about these objects 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 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 are basic categories, including the 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 $(A_1 \times \cdots \times A_m) \rightarrow \mathrm{Set}\,(B_1 \times \cdots \times B_n)$ of functions of m arguments that take objects of categories $A_1, \ldots, A_m$, and return sets of n-tuples of

[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.

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 $A \rightarrow B$ 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 $A \rightarrow \mathrm{Set}\,(B \times C)$, 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' (Sxy) between natural numbers;
   - the relation '$x = y$' between objects of any *basic* category;
   - the relation $\langle y_1, \ldots, y_n \rangle \in F(x_1, \ldots, x_m)$ where F is of category $(A_1 \times \cdots \times A_m) \rightarrow \mathrm{Set}\,(B_1 \times \cdots \times B_n)$, x_i of category A_i and y_i of 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 $\rightarrow$, 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 $(A_1 \times \cdots \times A_m) \rightarrow \mathrm{Set}\,(B_1 \times \cdots \times B_n)$ by declaring

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

Making this declaration has the effect of introducing the axiom

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

Principle of Iteration. This principle allows us to define functions by recursion over the natural numbers; given a function F from a category $S \rightarrow S$, we can form a function G of category $S \times \mathbb{N} \rightarrow S$ 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 $\mathrm{Set}\,(B_1 \times \cdots \times B_n)$. Given an object F of category $(A_1 \times \cdots \times A_m \times S) \rightarrow S$, we may introduce an object G of category $(A_1 \times \cdots \times A_m \times S \times \mathbb{N}) \rightarrow S$ 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

$$\begin{gathered}\forall \boldsymbol{x}, \boldsymbol{y}(\boldsymbol{y} \in G(\boldsymbol{x}, X, 0) \leftrightarrow \boldsymbol{y} \in X) \\ \forall \boldsymbol{x}, \boldsymbol{y}, a, b(Sab \rightarrow (\boldsymbol{y} \in G(\boldsymbol{x}, X, b) \leftrightarrow \boldsymbol{y} \in F(\boldsymbol{x}, G(\boldsymbol{x}, X, a)))\end{gathered} \tag{5}$$

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 $C = (A_1 \times \cdots \times A_m) \rightarrow \mathrm{Set}\,(B_1 \times \cdots \times B_n)$; 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 $F(\boldsymbol{x}) = \{\boldsymbol{y} \mid \phi[\boldsymbol{x}, \boldsymbol{y}]\}$.

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 $A \rightarrow B$ of functions from A to B for all categories (not only the case where B has the form $\mathrm{Set}\,(\cdots)$).
   $A \rightarrow B$ shall always be an ideal category. For the system to be predicative, quantification over functions must not be allowed in small propositions; quantifying over $A \rightarrow \mathbb{N}$, for example, would provide an effective means of quantifying over $\mathrm{Set}\,(A)$. (Recall that, classically, the power set of 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 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'. 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]. 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. 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, 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 $\{x \mid \phi[x]\}$ 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; 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.

[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].