Egbert Rijke
Department of Philosophy
Carnegie Mellon University, Pittsburgh, USA
erijke@andrew.cmu.edu

Bas Spitters
Department of Computer Science
Aarhus University, Denmark
spitters@cs.au.dk

Homotopy type theory and the formalization of mathematics

Egbert Rijke is a PhD student at Carnegie Mellon University, working on homotopy type theory. Bas Spitters is associate professor at Aarhus university, working on type theory, topos theory and proof assistants. Both authors participated in the Univalent Foundations Program.

Formalization of mathematics, that is the complete reduction of mathematical arguments to the axioms, has become more and more feasible due to the development of better computer implementations and advances in languages and algorithms. In this article we focus on one such language, homotopy type theory, which Voevodsky has proposed as a new foundation for all of mathematics. It extends the usual set theoretic foundations with a computational meaning, while at the same time taking the more general notion of space as primitive.

Computer formalization of mathematics

Reasons for formalization

The present refereeing process for mathematical articles, based on social consensus by a small group of experts, does not guarantee that all proofs are flawless. Fields medalist Voevodsky courageously points to an error [23] which existed for a decade in one of his own fundamental papers. One reason for this is that checking proofs for correctness can be boring or hard, and may require a lot of effort and expertise. These issues are addressed by software called 'proof assistants', in which proofs can be programmed and their correctness is verified algorithmically.

Moreover, formalization of mathematics in proof assistants provides new insights and conveys new mathematical structures. It is also a source of pleasure, a computer game, for mathematicians with an appreciation of detail. Finally, it gives peace of mind when ones arguments have been checked very carefully with the aid of a computer.

Figure 1 Cover image of the HoTT book.

Great formalization feats include: a formal proof of the Kepler conjecture [10], the four color theorem [8] and the Feit–Thompson theorem [9] (the beginning of the classification of finite simple groups). Notably, the former two are huge proofs which crucially depend on computer computations which due to their sheer size cannot be checked as a whole by humans.

Importance for computer science

In computer science, ideally one would want the behavior of software to be specified and we would like a formal guarantee that it actually sticks to that specification. Such formal verification has become practical for large software products. Examples of formally verified software include a verified microkernel called seL4 [14], a verified compiler for the C language [17], and on top of the latter, parts of the SSL internet protocol [5]. Another example is a verified compiler [13] for the ML functional programming language.

Foundations: sets, categories, types

The *material* conception of the mathematical notion of set conceives sets as being build up iteratively from the empty set. For instance, the disjoint union of sets A, B may be encoded as

$$\{(\emptyset, a) \mid a \in A\} \cup \{(\{\emptyset\}, b) \mid b \in B\}$$

The natural numbers can be encoded as $0 = \emptyset$, $1 = \{0\}$, $2 = \{0, 1\}$, et cetera. In contrast, Lawvere's elementary theory of the category of sets avoids this encoding by providing a *structural* account using categorical methods [16]. Category theory considers not only objects, but also the relations between them. These are captured using arrows (abstract functions).

Many familiar objects of set theory, such as singletons, the disjoint union, the set of natural numbers or the power set, are introduced in the elementary theory of the category of sets by their *universal property*. The disjoint union will be specified as the least object which allows embeddings from A, B. The natural numbers are captured as the least object that has a 0 and a successor function S. Here we use 'least' informally, and indeed the formal way to express this is the universal property. For instance, the universal property of $\mathbb{N}$ is the familiar property that, for every set X, a base point $x_0 \in X$ and a map $f : X \to X$ *uniquely determines* a sequence $(x_n)_n$ of elements of X, satisfying $x_{n+1} = f(x_n)$.

In the following, we shall see that type theory shares many of the structural aspects of the theory of sets. In particular, many types are specified by a type theoretic analogue of the universal property: the induction principle.

Practical foundations

Presently, foundations for computer formalization roughly fall in two classes, both have a *structural* flavor. They are either based on higher order logic (HOL), or on dependent type theory. The former is a little simpler, but the latter is a more expressive language for mathematics and at the same time provides a functional programming language, like Haskell or OCaml.

Type systems are used in most modern programming languages to avoid programming errors, with richer type systems allowing one to capture more errors. For example, adding a function $\mathbb{R} \to \mathbb{R}$ to a 2×2-matrix would raise a type error. This is similar to the unit tests in physics which warn against adding meters to kilos. In an untyped language like set theory, such errors are more difficult to catch. For example, one could ask for the elements of the real number π, although this question has little meaning. This is the reason that even the systems which are based on set theory have a weak type system build on top of them.

Type theory shares much of the structural properties of category theory, including the specification of types by universal properties. A large class of types specified in this way are the *inductive types*. For example, the type of natural numbers is a type with $0 : \mathbb{N}$ and a successor function $\mathsf{S} : \mathbb{N} \to \mathbb{N}$ which adhere to an *induction principle*. We will describe the type theoretical induction principle of the natural numbers, but before we do so we need to explain some of the concepts of dependent type theory.

First of all, in dependent type theory, there is a notion of *type families* (or, dependent types). A type family B indexed by a type A, consists of a type $B(a)$ varying over a variable a of type A. As an example, there is a type $\mathsf{list}_A(n)$, depending on $n : \mathbb{N}$, which gives for each n the type of lists with n elements from a type A. We often write $B : A \to \mathsf{Type}$ if B is a type family indexed by a type A.

Like a section of a family of sets, a section b of a type family B indexed by A consists of a term $b(a) : B(a)$, depending on a variable $a : A$. The sections of the type family B are themselves terms of a type: the *dependent product type* $\Pi_{(a:A)} B(a)$. Note that there is a universal quantification involved in the dependent product type: a term f of $\Pi_{(a:A)} B(a)$ assigns to *every* $a : A$, a term $f(a) : B(a)$.

Now we can state the induction principle of the natural numbers. The induction principle tells us how to show $\Pi_{(n:\mathbb{N})} P(n)$, for an arbitrary type family $P : \mathbb{N} \to \mathsf{Type}$. In other words, it tells us how to prove a universal quantification over the natural numbers, just as the induction principle you might be familiar with.

Induction principle of the natural numbers: *Let* $P : \mathbb{N} \to \mathsf{Type}$ *be a family of types over the natural numbers. Given*

$$\begin{aligned} &a_0 : P(0) \\ &a_{\mathsf{S}} : \Pi_{(n:\mathbb{N})} P(n) \to P(n+1), \end{aligned}$$

there is a section $f : \Pi_{(n:\mathbb{N})} P(n)$. *Moreover, we can* compute *with* f *as follows*:

$$\begin{aligned} f(0) &:= a_0 \\ f(\mathsf{S}(n)) &:= a_{\mathsf{S}}(n, f(n)) . \end{aligned}$$

This allows us to compute, say, $f(2)$ recursively:

$$\begin{aligned} f(\mathsf{SS}0) &= a_{\mathsf{S}}(1, f(\mathsf{S}0)) \\ &= a_{\mathsf{S}}(1, a_{\mathsf{S}}(0, f(0))) \\ &= a_{\mathsf{S}}(1, a_{\mathsf{S}}(0, a_0)) . \end{aligned}$$

Computation is an important aspect of any formalization. Fortunately, it is safely supported by most modern proof assistants. Computation is crucial to formalize proofs efficiently. Already, Russell and Whitehead used hundreds of pages in their book *Principia Mathematica*, to arrive at their proof that $1 + 1 = 2$. With modern implementations such proofs are automatic. This is necessary as, even with this support, the formalization of the Feit–Thompson theorem, for example, still has 170.000 lines of code. Just by their sheer size, such formalization projects need to be treated as software projects. Without verified computation those projects would simply be unfeasible.

The alert reader might have observed that the induction principle of the natural numbers only asserts the *existence* of a section when certain conditions are met. Nothing is said about the uniqueness, whereas the universal property is about existence *and* uniqueness. Of course, being unique depends on the notion of equality. The notion of equality in dependent type theory is inductively defined, just as the natural numbers. Martin-Löf, who first wrote down their inductive definition, called these equality types *identity types*. The identity type $a =_A b$ is a family of types which depends on two variables a, b of type A, and every type is equipped with an identity type. The identity type is generated by a term $\mathsf{refl}(a) : a =_A a$, for every $a : A$.

Since the identity types are themselves type families, one might anticipate that it is possible to use the induction principle again, to show the uniqueness of the sections that are obtained by the induction principle. This is indeed the case, but we will not go into that here. We only note that this derivation requires function extensionality. Function extensionality is the principle that two sections $f, g : \Pi_{(a:A)} B(a)$ are equal precisely when they take equal values. Since we take identity types as our notion of equality, f and g take equal values if there is a section of type $\Pi_{(a:A)} f(a) =_{B(a)} g(a)$. Function extensionality cannot be proved directly in dependent type theory. However, in homotopy type theory we can prove function extensionality from Voevodsky's univalence axiom which we will discuss below.

Martin-Löf's dependent type theory
Foundations based on dependent type theory are characteristically different from the set theoretic foundations, in that the set theoretic foundations have two layers (one for first-order logic, and one for the theory of sets), whereas in dependent type theory there is only one such layer. Predicates of first order logic are replaced by dependent types. Notably, we have a *dependent type* $x =_A y$ of equalities in A instead of an equality predicate.

Apart from identity types, we have types such as the empty type, a type with one element and a type $\mathbb{N}$ of natural numbers. Given two types A and B, we can form their cartesian product $A \times B$ and their disjoint union $A + B$. More generally, given a family B of types indexed by a type A, we can form their dependent product $\Pi_{(a:A)} B(a)$, and we can also form their dependent sum $\Sigma_{(a:A)} B(a)$.

There is a tight connection between logic and type theory, which is illustrated by the Curry–Howard correspondence, as shown in Table 1. Apart from these logical connectives, type theory also possesses types which are not directly linked to any logical connective, including a type $\mathbb{N}$ of natural numbers and a so-called *universe* $\mathcal{U}$. Famously, by the Russell paradox there cannot be a set of all sets. So, one usually

Logical connective	**Type operation**
truth	$\mathbf{1}$
falsehood	$\mathbf{0}$
conjunction	$A \times B$
disjunction	$A + B$
universal quantification	$\Pi_{(x:A)} P(x)$
implication	$A \to B$
existential quantification	$\Sigma_{(x:A)} P(x)$
equality	$x =_A y$

Table 1 Curry–Howard correspondence

distinguishes between small sets and large sets, sometimes called sets and classes. In type theory, a universe is a type which contains the 'small' types as its terms, and is closed under the mentioned constructions.

Notably, there is no untyped elementhood predicate ($\in$) in type theory, as there is in set theory. A term always comes with a specified type. There are rules for how to derive a term of a given type, and for how to use them to derive other things. From one point of view, these rules are much like the rules for how to derive a proposition of a certain form. From another point of view, these rules are a version of the universal property of a particular type constructor. We have seen an instance of such rules in the case of the type $\mathbb{N}$, with its induction principle.

Homotopical models

Homotopy type theory refers to the use of homotopy theory to study models of dependent type theory or the use of ideas and constructions from homotopy theory to prove theorems in type theory. So, homotopy type theory is both a theory of homotopy types and a homotopical perspective of type theory:

$$\text{(homotopy type) theory} \\ = \\ \text{homotopy (type theory).}$$

Voevodsky proposes homotopy type theory together with his univalence axiom as a new foundation for mathematics, as we will now explain.

The set model
The collection of all sets forms a model of type theory. In this model, a type is interpreted as a set, and a family of types over a base type X is simply an indexed collection $(Y_i)_{i \in X}$ of sets. Universes in type theory can be modeled by so-called Grothendieck universes in set theory. In this way, types can be used to talk about sets in a structural way. In this model, any two terms of the same type can be equal in at most one way. This leads to a variant of the Curry–Howard correspondence where propositions are truncated. The (-1)-truncation $\|A\|$ of a type A identifies all its terms. Such truncated types are also referred to as mere propositions. Importantly, the (dependent) sum of propositions need not be a proposition, so we need to truncate it to soundly interpret first order logic. Having such propositional identity types seems natural, but models without this extra requirement are mathematically relevant too.

The groupoid model
In the groupoid model of type theory equality of terms is modeled by isomorphism. A groupoid is a category in which all morphisms are invertible. Groupoids are fundamental in homotopy theory. Given a topological space one defines the fundamental groupoid, the objects of which are the points of the space, and the arrows are the (homotopy equivalence classes of) paths between them. The identity arrow is the path that stands still, composition of arrows is composition of paths, the inverse of an arrow is the path in the opposite direction.

A homotopy between two paths:

$$\begin{tikzcd} \bullet \arrow[r, bend left, ""{name=U, below}] \arrow[r, bend right, ""{name=D, above}] & \bullet \arrow[Rightarrow, from=U, to=D] \end{tikzcd}$$

Hofmann and Streicher [11] showed that many of the ideas from the set theoretical model can be extended to the groupoid model. Importantly, the groupoid model also contains a universe of all small sets, namely the groupoid of all sets in which the arrows are the bijections between sets. Since the equality type is modeled by isomorphism in the groupoid model, we see that the groupoid model gives a precise sense in which isomorphic sets may be identified. This is one of the early instances of the univalence axiom.

The way in which equality is modeled in the groupoid model gives a fundamental distinction between the set model and the groupoid model: in the set model there are many distinguishable natural numbers objects (e.g. $\mathbb{N}$, $2\mathbb{N}$), while in the groupoid model all of them are equal, because they are all isomorphic. This fits with our usual phrase: *the* natural numbers.

The homotopy interpretation of type theory
Again, it turns out to be impossible to show that any type is a groupoid, because it is not provable that any two proofs of equality are themselves equal in at most one way. Sets (discrete spaces) and groupoids (spaces for which all higher homotopy groups are trivial) are just two levels of a hierarchy of more and more complex structures, indeed: homotopy types.

Just as every space gives rise to its fundamental groupoid, we can associate to every space its fundamental ∞-groupoid. Whereas in the fundamental groupoid of a space we identified all the homotopic paths, in the fundamental ∞-groupoid these homotopies between paths will be the arrows at the next level.

This suggests a homotopy interpretation of type theory, in which identities are interpreted as paths in a space, identities between identities as homotopies between paths, and so on. This idea has been made precise by Voevodsky [12, 21] who showed the Kan simplicial sets form a model of type theory. Intuitively, simplicial sets are collections of triangulations. We will say more about them shortly. Independently, Awodey and Warren [2] showed that a relaxed version of identity types can be interpreted in an arbitrary Quillen model category (an abstract setting for homotopy theory). The original identity types can be interpreted in *cloven* weak factorization systems [4].

In the homotopy interpretation of type theory, types are interpreted by spaces, and dependent types are interpreted by fibrations. Terms of a type are points in a space, and a term of a dependent type is a section of the corresponding fibration.

A proof of equality is a point in the path space, so we see that a proof of equality between two equalities gets interpreted as a homotopy between paths. Higher equalities are interpreted accordingly by higher homotopies.

The simplicial set model

Voevodsky constructed, moreover, a universe in simplicial sets which has a very nice property: the path space between two Kan simplicial sets in the universe is weakly equivalent to the Kan simplicial set of weak equivalences between them. This property goes under the name *univalence*: for any two types X and Y in a universe U, we have a canonical (i.e. specified) equivalence

$$(X =_U Y) \simeq (X \simeq Y).$$

Grothendieck's *homotopy hypothesis* states that ∞-groupoids are a model for homotopy types, topological spaces up to homotopy. The hypothesis itself is very delicate due to the difficulty of giving a precise definition of ∞-groupoid. ∞-groupoids in turn can be modeled by a so-called model structure on simplicial sets. Just like sets, we can take products and sums (disjoint unions, i.e. coproducts) of such ∞-groupoids. Moreover, we can even take dependent (co)products. This makes ∞-groupoids (Kan simplicial sets) into a model of type theory.

Motivated by Grothendieck's insight that ∞-groupoids are ubiquitous in mathematics, and by the foundational role of type theory, Voevodsky proposes homotopy type theory as a new 'univalent' foundation for all of mathematics. The connection between types and ∞-groupoids is further strengthened by a theorem of Van den Berg, Garner and Lumsdaine, that types in Martin-Löf type theory indeed exhibit the structure of an ∞-groupoid [3, 15].

In the simplicial set model, we can find the set model and the groupoid model, and indeed a whole hierarchy of n-groupoid models. The 0-groupoids, which are also known as the (homotopy) discrete spaces, are our ordinary sets [19].

Improving set theory

Zermelo–Fraenkel set theory is often considered to be 'the' foundation of mathematics. This was introduced in the beginning of the previous century. There have been several proposals to update this foundation to more modern mathematical insights. As already discussed, one such proposal is Lawvere's structural set theory. This arose from the development of Grothendieck's topos theory, a vast generalization of topology applicable to a wide range of fields including algebraic geometry. Lawvere and Tierney showed that toposes can be described by a (structural) set theory. A prime example of a topos is the topos of sets. Moreover, for a general topological space, the variable sets, the so-called sheaves, over this space form again a topos. Another class of toposes is connected to the theory of computation. In other words, toposes are everywhere.

Since there are many toposes, and therefore many interpretations of the *structural* axioms of set theory, one might consider the axioms of set theory similar to the axioms of a group (in fact, both are essentially algebraic theories). By adding axioms to structural set theory, one reduces the class of models, just as one reduces the class of groups by adding the axiom of commutativity.

In the case of structural set theory, we can add for example the axiom of choice. Not every topos satisfies this property, as the logic of the set theory a topos adheres to is not classical logic, but Heyting's intuitionistic logic. Nevertheless, we can restrict our attention to the toposes in which classical logic is satisfied, by adding the axiom of choice. In this sense, topos theory *subsumes* classical set theory.

Similarly, type theory is by its computational nature an intuitionistic theory. But just as in toposes, homotopy type theory subsumes classical reasoning, in the sense that the axiom of choice can be added. The axiom of choice is satisfied, for instance, in the model of Kan simplicial sets.

Universes play a key part in the univalent foundations. The univalence axiom characterizes the identity types on the universe. In fact, by characterizing the identity types on the universe, one describes indirectly a universal property of the universe. This universal property is a homotopical analogue of the notion of a subobject classifier in a topos. In a topos, the subobject classifier can be seen as the object of all propositions in the logic of that topos. In the case of the sheaves over a topological space, this subobject classifier just consists of the opens of the topological space. Hence, this is an important object in the theory of toposes. By the propositions-as-types [22] paradigm (i.e. the Curry–Howard correspondence), the universe plays the same role for general types. Another way to express this is that the univalence axiom connects univalent type theory with higher topos theory. Higher topos theory is a joint generalization of topos theory and the abstract theory of homotopy types [18}.

Consequences homotopy interpretation

Structuralism

Another application of the univalence axiom is the structure invariance principle between general (algebraic) structures. For instance, any two isomorphic groups may be identified. This idea is prominent in Bourbaki's vision of mathematics as the science of abstract structures, where structures are important only up to isomorphism [20]. Univalent type theory satisfies the principle of structuralism [1]: that isomorphic structures may be identified. In set theoretic foundations for mathematics, the principle of structuralism is simply false.

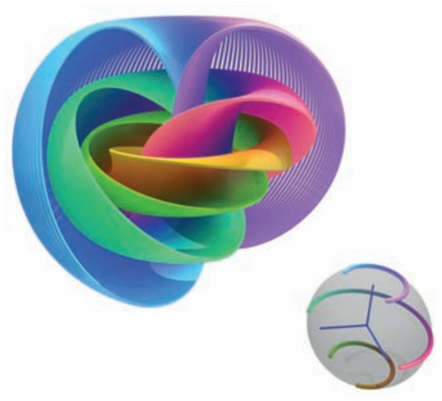

Figure 2 Hopf fibration.

Axiomatic reasoning

Since homotopy type theory can be interpreted in ∞-groupoids, we can actually use it to reason axiomatically about them. This may be compared to two ways of reasoning in geometry: analogous to traditional homotopy theory we have analytic reasoning using coordinates, and analogous to reasoning in homotopy type theory we have synthetic reasoning using axioms about points and lines and construction methods such as ruler and compass. Just like many theorems in geometry have beautiful synthetic proofs, likewise, many theorems in algebraic topology have elegant and very elementary proofs in homotopy type theory. One of the first examples of such reasoning, and indeed one of the first non-trivial applications of the univalence axiom, is Shulman's proof that the fundamental group of the circle is equivalent to the integers.

Many of the classical constructions of CW-complexes can be specified axiomatically in homotopy type theory by generalizing the construction of inductive types by specifying not only its points, but also (some) equalities. We call these constructions *higher* inductive types. The interval consists of two points and an equality (i.e., path) between them. A circle is a point with a loop, (using univalence one shows that in the free type with a point and a loop, the loop is distinguishable from the trivial path refl). The circle is also obtained as a special instance of suspensions of types, and by iterated suspensions one gets the n-sphere for any $n:\mathbb{N}$. Many other familiar spaces, including the torus, can be defined in this way.

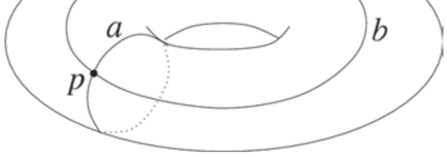

Figure 3 The fundamental group of the torus with base point p has two generators a and b.

Many theorems about fundamental groups of spheres have short new proofs in the language of homotopy type theory using reasoning principles from type theory. Also, many of the classic constructions are possible. For instance, once the spheres are defined, one may use the univalence axiom to construct the Hopf fibration.

One of the big accomplishments of homotopy type theory is Brunerie's proof [6] that the fourth homotopy group $\pi_4(\mathbb{S}^3)$ of the 3-sphere is $\mathbb{Z}/2\mathbb{Z}$. The first part of his proof shows that there is a natural number n for which $\pi_4(\mathbb{S}^3)$ is $\mathbb{Z}/n\mathbb{Z}$. He then proceeds by proving that this number is 2.

Cubical proof assistant

In principle one should be able to compute the number n, which was constructed in the first half of the proof, thus replacing tens of pages of advanced mathematics. However, since the univalence axiom is added as an axiom to Martin-Löf's dependent type theory, the computation may get stuck on applications of the univalence axiom. In other words, a *computational interpretation* of the univalence axiom has to be provided before the algorithms for computation can be extended to compute with applications of the univalence axiom.

Fortunately, there is now a fully constructive model of the univalence axiom [7], which provides the desired computational interpretation. This is a model of cubical sets, instead of simplicial sets. In the cubical sets, spaces are built up from squares instead of triangles. This slight modification alters the algebraic structure of spaces slightly, making it possible to give a fully constructive proof that the univalence axiom is satisfied in this model, whereas Voevodsky's original proof makes use of non-computational reasoning principles in an essential way.

The cubical model of type theory turns out to provide a convenient language, cubical type theory, to reason about (homotopical) constructions. This type theory has been implemented so that one can now compute with univalence. The term for Brunerie's number n which we introduced above has been defined in cubical type theory. Unfortunately, the type checker is currently too slow to compute the answer. However, many possibilities for improvements exists, and we expect more such applications in the future.

Axiomatic physics

Schreiber and Shulman [24] use homotopy type theory to provide an axiomatic treatment of modern physics, more precisely of local higher gauge quantum field theory. To do so they greatly generalize Lawvere's axioms for synthetic differential geometry, which uses so-called cohesive toposes, by providing a modal type theory which can be modeled in cohesive *higher* toposes.

Open science

Voevodsky proposed the thematic year (2012–2013) about univalent foundations of mathematics at the Institute for Advanced Study to bring together researchers from a wide variety of backgrounds to develop mathematics based on his univalence axiom. He organized the year with Awodey and Coquand. During this year there was a pressure cooker full of ideas from researchers coming from different directions such as type theory and proof assistants, category theory and abstract homotopy theory. It was decided to write a book [20] as a means to collect the ideas and take a snapshot of the state of the field at the end of the year. As a proof of concept, the book develops a substantial amount of mathematics within the univalent foundations. This includes homotopy theory and algebraic topology (using synthetic reasoning), but also set theory, category theory and various constructions of the real numbers. Part of the book 'unformalizes' code that existed only in the computer before. This prominently includes both the 'Foundations' library by Voevodsky and the substantial work on using higher inductive types for synthetic reasoning. This was a new experience. Before, the usual procedure was to go from paper proofs in mathematics to computer formalization, but this time the formalization came first and the natural language presentation afterwards. In these formal proofs, the proof assistant is very helpful in developing the proofs, both for proofs by computation and in order to keep the bookkeeping manageable.

Participants of the Univalent Foundations Program

The book is self-published under a permissive creative commons licence. One of the advantages of this process is that the book is still being updated regularly with additions and clarifications from readers. Often this is done by means of a 'pull-request', the common way of contributing to an open source programming project. The book was created in a true open source spirit and was developed using open source tools such as git version control. The LaTeX-sources are freely available to everyone. It was a unique experience to write a book with such a large number of authors. The efficient software tools were important in this process, but working collectively from one place where we could regularly meet was still crucial.

Many parts of the book have been formalized a number of times, and also in different proof assistants such as Coq, Agda and Lean. While some details of the semantics remain to be worked out, we can still be very confident that all these theorems are correct. Conversely, in the cases where the formalization came afterwards, usually a number of bugs were found in the book. These bugs have been corrected. It is worth stressing that the book has been proof read very carefully, similar bugs showed up when other parts of mathematics were formalized.

The book was included on the 2013 list of notable items published in computing by Computing Reviews of the Association of Computing Machinery (ACM).

References

1 Steve Awodey, Structuralism, invariance, and univalence, *Philosophia Mathematica* 22(1) (2013), 1–11.

2 Steve Awodey and Michael Warren, Homotopy theoretic models of identity types, *Mathematical Proceedings of the Cambridge Philosophical Society* 146(1) (2009), 45–55.

3 Benno van den Berg and Richard Garner, Types are weak ω-groupoids, *Proceedings of the London Mathematical Society* 102(2) (2011), 370–394.

4 Benno van den Berg and Richard Garner, Topological and simplicial models of identity types, *ACM transactions on computational logic (TOCL)* 13(1) (2012), 3.

5 Lennart Beringer, Adam Petcher, Katherine Q. Ye and Andrew W. Appel, Verified correctness and security of OpenSSL HMAC, in *USENIX Security* 15, 2015, pp. 207–221.

6 Guillaume Brunerie, *On the Homotopy Groups of Spheres in Homotopy Type Theory*, PhD thesis, Nice, 2016, arXiv:1606.05916.

7 Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg, Cubical type theory: a constructive interpretation of the univalence axiom, 2016, http://www.cse.chalmers.se/~coquand/cubicaltt.pdf.

8 G. Gonthier, Formal proof — the four-color theorem, *Notices of the AMS* 55(11) (2008), 1382–1393.

9 G. Gonthier, A. Asperti, J. Avigad, Y. Bertot, C. Cohen, F. Garillot, S. Le Roux, A. Mahboubi, R. O'Connor, S.O. Biha, I. Pasca, L. Rideau, A. Solovyev, E. Tassi and L. Théry, A machine-checked proof of the odd order theorem, in *Interactive Theorem Proving (ITP)*, Lecture Notes in Computer Science, Vol. 7998, Springer, 2013, pp. 163–179.

10 Thomas Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, John Harrison, Truong Le Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, Truong Quang Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, An Hoai Thi Ta, Trung Nam Tran, Diep Thi Trieu, Josef Urban, Ky Khac Vu, Roland Zumkeller, A formal proof of the Kepler conjecture, 2015, arXiv:1501.02155.

11 Martin Hofmann and Thomas Streicher, The groupoid interpretation of type theory, in Sambin and Smith, eds., *Twenty-five Years of Constructive Type Theory*, Oxford Logic Guides, Vol. 36, OUP, 1998, pp. 83–111.

12 Chris Kapulkin and Peter Lumsdaine, The simplicial model of univalent foundations (after Voevodsky), 2012, arXiv:1211.2851.

13 Yong Kiam Tan, Magnus O. Myreen, Ramana Kumar, Anthony Fox, Scott Owens, and Michael Norrish, A new verified compiler backend for CakeML, in *ICFP16*, 2016.

14 Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch and Simon Winwood, seL4: Formal verification of an OS kernel, in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, ACM, 2009, pp. 207–220.

15 Peter LeFanu Lumsdaine, *Weak ω-Categories from Intensional Type Theory*, Springer, 2009, pp. 172–187.

16 Tom Leinster, Rethinking set theory, *American Mathematical Monthly* 121(5) (2014), 403–415.

17 X. Leroy, Formal certification of a compiler back-end or: programming a compiler with a proof assistant, in *POPL*, ACM, 2006, pp. 42–54.

18 J. Lurie, *Higher Topos Theory*, Vol. 170, Princeton, 2009.

19 Egbert Rijke and Bas Spitters, Sets in homotopy type theory, *MSCS* 25, Special Issue 05 (2015), 1172–1202.

20 The Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations for Mathematics,* Institute for Advanced Study, 2013, homotopytypetheory.org/book.

21 Vladimir Voevodsky, The equivalence axiom and univalent models of type theory, 2010, arXiv:1402.5556, to appear in *MSCS*.

22 Philip Wadler, Propositions as types, *Communications of the ACM* 58(12) (2015), 75–84.

23 https://www.math.ias.edu/vladimir/sites/math.ias.edu.vladimir/files/2014_08_ASC_lecture.pdf.

24 https://ncatlab.org/schreiber/show/Modern+Physics+formalized+in+Modal+Homotopy+Type+Theory.