Martin Hofmann's Contributions to Type Theory: Groupoids and Univalence

Thorsten Altenkirch

January 11, 2021

Abstract

My goal is to give an accessible introduction to Martin's work on the groupoid model and how it is related to the recent notion of univalence in Homotopy Type Theory while sharing some memories of Martin.

1 Meeting Martin

I met Martin Hofmann in the summer of 1991 in Erlangen, Germany. At this time I had already started my PhD in Edinburgh and I was spending the summer working for Siemens in Munich who were sponsoring my studies. I knew about Martin from Terry Stroup who was organising seminars and meetings of students at the University of Erlangen and who also visited Edinburgh frequently. One time he gave me a copy of Martin's diploma thesis saying that this was work by a very clever guy. I quickly realised that Martin had already solved some problems I was working on at the time in a very elegant way. This depressed me a bit because it showed my own incompetence but when I was in Munich I decided to take the bull by its horns and visit Martin. Martin invited me to come to Erlangen where I met him and his wife Annette and we had a pleasant day together with many discussions about Computer Science and Mathematics. At this time Martin already had an offer for a PhD position at Passau with Martin Wirsing but he had also been contacted by Don Sanella who wanted to recruit him for Edinburgh. Naturally, I tried to convince him to come to Edinburgh, which he did, starting his PhD in October of the same year. Even better, there was a space at my house, *Bath Street Housing Co-op* and we ended up living in the same house. We had countless discussions at breakfast on theoretical computer science, more specifically Type Theory and Category Theory, filling the communal blackboard with strange symbols incomprehensible to our housemates who were mostly musicians and artists.

2 The biggest problem in Type Theory

Martin and I were both interested in Type Theory, which had been invented by the Swedish philosopher and mathematician Per Martin-Löf and which can be at the same time viewed as a programming language and a system to represent mathematical constructions [3]. A version of Type Theory, the Calculus of Constructions had been implemented by Randy Pollack in Edinburgh [2]. Randy called his system LEGO because *it is fun to do constructions* [1] . Martin had already used the LEGO system for his diploma thesis and we were both keen to use the system but also to understand it better and develop the theory behind it further.

One central idea in Type Theory is the *propositions as types* principle. For any mathematical statement – these are called propositions – we associate a type of evidence. So for example the mathematical statement *There are infinitely many prime numbers* is translated into a type so that once we can exhibit a program of this type we know that the proposition is true. Given a proposition P we write $p : P$ to mean that the program p is evidence for the proposition P.

An example for propositions are equalities $a = b$ where $a, b : A$ – meaning that they are elements of some type A (for example numbers) – and we were concerned with evidence for equality, that is $p : a = b$. Since equality in type theory can be applied to elements of any type we can ask the question, given two usch *proofs* of an equality $p, q : a = b$ are they equal, $p = q$? This seems maybe a strange question at first but it turns out that the answer to this question has very important consequences to how we do Mathematics. The question is called *uniqueness of identity proofs* or UIP.

The biggest open problem in Type Theory at the time was the question whether UIP was provable in Type Theory using the rules given by Martin-Löf. Obviously this was just the right sort of thing for Martin.

3 The J-rule

I can't give a detailed introduction to Type Theory here but I can try to illustrate the basic idea. Let's look at a simple type, e.g. the natural numbers $\mathbb{N} = \{0, 1, 2, 3, \dots\}$. [2] [3] These are the canonical natural numbers but we can also form expressions such as $3 + 4 : \mathbb{N}$ which are not canonical but they can be reduced to a canonical natural number, namely 7. Using variables we can also form expressions such as $x + y$ and we may want to answer the question whether $x + y = y + x$ for all natural numbers $x, y : \mathbb{N}$. Using the propositions as types principle this means that we have to write a program that when we feed it with any two canonical natural numbers, e.g. $x = 3$ and $y = 4$ it computes evidence that $x = y$ that is in this case evidence that $3 + 4 = 4 + 3$. But what is evidence

[1] This line came up each time you started LEGO.

[2] Computer scientists start counting at 0, mathematicians usually at 1

[3] The curly brackets $\{\dots\}$ come from set theory but they can be used in Type Theory also to denote *labelled sums*.

for an equality? According to Martin-Löf the canonical evidence for equality is reflexivity, that is, that identical things are equal: given $a : A$ we have a program $\mathrm{refl} : a = a$. Hence we can use refl to prove that $3 + 4 = 4 + 3$ since both sides reduce to 7. A program that shows $x + y = y + x$ will assign refl to any pair of numbers. [4]

What can we do with equality? That is given we know an equality what can we prove from it? Martin-Löf introduced a rule, called the J-rule which basically says if you want to prove anything about equality (that is given $x, y : A$ and $p : x = y$) it is enough if you do it for refl (that is for $p = \mathrm{refl}$ which also forces us to identify x and y). For example if we want to show in general that equality is symmetric, that is that given $p : x = y$ we can derive $\mathrm{sym}(p) : y = x$ then we can use the J-rule to show this because if we assume that $x = y$ was proven by refl, then we know that x and y are identical and then we can also use refl to prove $y = x$.

Hence the question is can we prove UIP from the J-rule? At first glance it seems obvious, since reflexivity is the only proof of an equality, any two proofs must be equal. However, when we actually try to prove this, e.g. with the LEGO system, it doesn't work, we get stuck. The problem is that to prove that $p = q$ it is sufficient to show this for p being refl using the J-rule, that is to show $\mathrm{refl} = q$. However, the problem is that q is now evidence for $x = x$ and the J-rule only works for equalities of the general form $x = y$. This failure is not enough to show it is impossible to prove this; there may be a different way.

4 Groupoids

It was clear that to show that it is impossible to derive UIP we needed to construct a counter model: that is, an interpretation of type theory which validates all rules including the J-rule but which invalidates UIP. Thomas Streicher had investigated models of the Calculus of Constructions already in his dissertation and had shown certain non-provability results [4]. However, all the models Thomas and other people had considered validated UIP.

It was at the TYPES meeting in Nijmegen in 1993 when Martin had the essential idea: groupoids. Basically Martin observed that equality in Type Theory behaves like a group, e.g. like the integers with addition and negation.

Integers	Equality
$m + n$	transitivity (if $p : x = y$ and $q : y = z$ then $\mathrm{trans}(p, q) : x = z$)
0	reflexivity ($\mathrm{refl} : x = x$)
$-m$	symmetry (if $p : x = y$ then $\mathrm{sym}(p) : y = x$)

We can use the J-rule to verify the algebraic properties of equality corresponding to equations on integers we know:

[4] However, it is not so easy to write the program in a way that the computer can recognize that it is correct.

Integers	Equality
$m + 0 = m$	$\text{trans}(p, \text{refl}) = p$
$0 + m = m$	$\text{trans}(\text{refl}, p) = p$
$(l + m) + n = l + (m + n)$	$\text{trans}(\text{trans}(p, q), r) = \text{trans}(p, \text{trans}(q, r))$
$m + -m = -m + m = 0$	$\text{trans}(p, \text{sym}(p)) = \text{trans}(\text{sym}(p), p) = \text{refl}$

We call a structure like the integers a *group*, while the equality proofs form a *groupoid*. The difference is that in the case of the integers you only have one type (namely $\mathbb{Z}$) whereas in the case of equality you have a type $(x = y)$ for any pair of elements $x, y : A$ for some type A. [5]

Together with Thomas Streicher he worked out the details, that the groupoid interpretation validates all rules of type theory [1]. On the other hand it is clear that there are groupoids which have more than one element, indeed since every group is also a groupoid, the integers serve as a counterexample to UIP because there is more than one integer.

5 Univalence

After Martin and Thomas had cracked that nut what next? I thought, ok we still want to have UIP since the only proof of an equality is refl, hence any two proofs should be equal. We knew that this was actually easy to fix: we just had to add another rule to Type Theory which Thomas called the K-rule (because K is the next letter after J). The K-rule allows us to show properties of equality proofs of the form $q : x = x$ by just showing them for $q = \text{refl}$ and so we could finish the failed proof of UIP. Hence to me the proof that UIP was not derivable from the J-rule just meant that we do need the K-rule.

But Martin already had other ideas: he pointed out that there are some interesting principles which hold in the groupoid model and which we could use to enhance type theory. In particular he suggested that we could interpret equality of *types* in a novel way which allows us to identify types whose elements are in a one-to-one correspondence. Take for example the integers and the natural numbers. Naively you would expect that there are more integers then natural numbers but this is not the case because we can construct the following identification:

$\mathbb{N}$(natural numbers)	0	1	2	3	4	5	6	...
$\mathbb{Z}$(integers)	0	1	-1	2	-2	3	-3	...

Now in Martin's theory we would have $\mathbb{N} = \mathbb{Z}$. This may seem surprising at first but the point is that in Type Theory we cannot look into a type, we cannot

[5] You may have noticed that I left out one important law namely $x + y = y + x$ which witnesses that the integers are a *commutative group*. However, this doesn't make much sense for groupoids because given $p : x = y$ and $q : y = z$ to form $\text{trans}(p, q)$ we can't in general form $\text{trans}(q, p)$, unless x and z are identical and we need all of them to be identical to be allowed to write $\text{trans}(p, q) = \text{trans}(q, p)$.

see its representation. And this means that we cannot distinguish the natural numbers and the integers from outside. [6]

Martin exploited that the structure of one-to-one correspondences of types, which are called isomorphisms, have the structure of a groupoid. Usually we would think of an equality like $2 = 3$ as a proposition. In conventional mathematics propositions are identified with the booleans [7] , i.e. true or false, but in Type Theory we use the type of evidence. However, not every type corresponds to a proposition, for example the natural numbers or the integers aren't just a proposition —- there is more to them than just having an element. In Type Theory we call a type propositional if it has at most one element. Hence UIP can be translated as asking whether equalities are always propositions. As we have seen this is not the case for equalities like $\mathbb{N} = \mathbb{Z}$ since there are many possible identifications of integers and natural numbers. Martin's observation that types and their equality can be viewed as groupoids means that we can have equalities which are types that do not correspond to propositions.

At the time I was unconvinced. Equality to me needed to be propositional and Martin's explanation seemed a bit too exotic. I was quite happy to add the K-rule and not have $\mathbb{N} = \mathbb{Z}$.

Now forward to around 2008. Vladimir Voevodsky, a Field medallist [8] got interested in Type Theory. Vladimir was working in algebraic topology, which can best be described as very abstract geometry. He noticed that concepts from Homotopy Theory can be useful when talking about mathematical objects instead of sets. He also realised that Type Theory may be a good framework to express his ideas. In particular he formulated the univalence principle which says that equivalent types are equal. And in the case of simple types like $\mathbb{N}$ and $\mathbb{Z}$ equivalence means that there is a one-to-one correspondence. That is in Vladimir's theory we also have $\mathbb{N} = \mathbb{Z}$.

When Vladimir posted an early paper (which I didn't understand) about his ideas to a mailing list, I started a discussion with him. I mentioned that some of his ideas seemed to be related to Martin's and Thomas' groupoid model. "Indeed", he replied, "but I am using higher groupoids".

This was related to a shortcoming of the groupoid interpretation we had already noticed. While in the groupoid model there can be more than one element in an equality type, e.g. we can have $p, q : a = b$ which are not equal, there can be at most one element in an equality of equalities: that is given $r, s : p = q$ we always have that $r = s$. However, we cannot prove this in Type Theory, there is a mismatch between the groupoid model and the rules of Type Theory. We realised that we would need to understand higher groupoids to fix this but this was beyond us at the time.

Vladimir had the advantage that he came from a different background, inspired by geometry, and for him groupoids were just paths on a surface and

[6] Certainly there are things we can do with the integers we cannot do with the natural numbers, like subtraction. But here we don't just talk about the type but the type and some operations, e.g. addition.

[7] After George Boole, a 19th century English mathematician.

[8] This is the equivalent of a Nobel Prize in Mathematics.

higher groupoids were just paths on surfaces in higher dimensions. While for us higher dimensions may look scary this is something the mathematicians have already worked out very well.

Vladimir's ideas have created a renaissance of Type Theory, indeed a new Type Theory called Homotopy Type Theory [5]. We have now new implementations of Type Theory like cubical agda [6] which use Vladimir's ideas about how Type Theory should look like.

We should remember that it was Martin and Thomas who first formulated the univalence principle, even though not under this name.

Allowing types whose equalities are not propositions has a big impact on how we do mathematics. It allows us to do mathematics in a structural way, by identifying objects that behave the same from outside instead of refering to the way they are defined. Liberating our minds from the idea that equalities are just propositions enables us to understand mathematical concepts in a more generic, more abstract, way, allowing us to build higher and more beautiful towers of abstraction. Univalence may be one of the fundamental principles with the greatest impact for the mathematics of the future.

6 Missing Martin

Martin Hofmann died in January 2018 when hillwalking in Japan after getting lost in the mountains. [9]

Martin was a close friend and I have learnt a lot from him. Not only about mathematics, that too, but I also was impressed by his bravery in mathematics. He was rather unimpressed by apparently complicated constructions and would not rest until he had fully understood them and could use them in his own work, often in an improved or modified form.

Since working on Type Theory, Martin had moved on to other areas, in particular he had done important work on understanding the complexity of programs – that is how much time and space they need. He was using ideas from type theory to limit the resource use of programs by using types and other static properties.

Still, it was always a very rewarding experience to discuss questions related to type theory with him. Actually and questions unrelated to type theory or even to mathematics. I miss being able to do this: from time to time I think, I would like to discuss something with Martin. Most of all I miss him as a generous and very clever friend.

Achkowledgements

Thanks to Ana Ransom for proof reading and Thomas Streicher and Graham Hutton for comments on a draft. Thanks to the referee for careful reading and

[9] Vladimir Voevodsky also died just before this in September 2017 due to an unexpected health issue.

helpful corrections.

References

[1] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. *Twenty-five years of constructive type theory (Venice, 1995)*, 36:83–111, 1998.

[2] Zhaohui Luo and Robert Pollack. *LEGO proof development system: User's manual*. LFCS, Department of Computer Science, University of Edinburgh, 1992.

[3] Per Martin-Löf and Giovanni Sambin. *Intuitionistic type theory*, volume 9. Bibliopolis Naples, 1984.

[4] Thomas Streicher. *Semantics of type theory: correctness, completeness and independence results*. Springer Science & Business Media, 2012.

[5] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.

[6] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical agda: a dependently typed programming language with univalence and higher inductive types. *Proceedings of the ACM on Programming Languages*, 3(ICFP):1–29, 2019.