Scientific report

ERC_16_MOBIL 127740

Setoid type theory

Ambrus Kaposi

2018-04-15—2018-07-15

1 Introduction

I visited the Gallinette research group lead by Nicolas Tabareau in Nantes for three months with the help of the ERC_16_MOBIL grant. We worked together on the project of setoid type theory and I received useful information for preparing an ERC starting grant submission. This will help me to establish a type theory research group at Eötvös Loránd University. The visit resulted in a deeper understanding of the connections between models and syntactic translations and a definition of a setoid translation. It materialized in several Agda formalisations and a draft paper on setoid type theory which we plan to submit to the TYPES post-proceedings (submission deadline 2018-10-08) or CPP (deadline 2018-10-18). The formalisations are all available in public online repositories (see below) and the draft paper is available on my website [16].

2 Results

2.1 Models and syntactic translations

I investigated the relationship of models and syntactic translations together with Simon Boulier and Nicolas Tabareau. This resulted in the following.

If the metatheory is made explicit, a model of type theory (CwF [5] with extra structure) can be seen as syntax in a metatheory, where the definitional equality of the object theory is modelled by propositional equality of the metatheory. An observation based on this is that the set model (standard model) of type theory justifies equality reflection [15]. This might lead to a simplified presentation of Hofmann's conservativity result [6, 17].

A strict model is one where all the equalities are definitional in the metatheory. Such a model can be called a syntactic model since definitional equality of the object theory is modelled by definitional equality of the metatheory. Strict models are easy to formalise in intensional type theories as one does not need to transport along equalities. An example is the formalisation of the setoid model [13] in Agda using a definitionally proof-irrelevant universe of propositions.

A syntactic translation is a syntactic model where object theory contexts are modelled by target theory contexts (possibly with extra structure), types by types, terms by terms and substitutions with substitutions (each possibly with extra structure).

Syntactic translations can also be given in an extrinsic way [4], that is, by an operation on precontexts, pretypes, preterms. In this case one needs to prove that these operations preserve typing and definitional equality. This view has the advantage that it matches current implementations and formalisations such as Template Coq [2] and has more tool support. In contrast with this, intrinsic syntax needs some form of extensional type theory to deal with the "transport hell" [7]. The disadvantage of the extrinsic approach is its lower level of abstraction: redundancy and obfuscation. This is why we took the intrinsic approach.

Currently the only result showing the connection between extrinsic and intrinsic approaches is Streicher's initiality construction [18]. Together with Ambroise Lafont, I investigated how to formalise this result in type theory. We have partial results showing how to reduce certain examples of inductive-inductive types to indexed inductive types [8, 9].

There are syntactic translations where there is no corresponding strict model. An example is a display map variant of the graph model [10] as opposed to the indexed graph model [12]. A syntactic translation corresponding to the display map variant is the logical relation interpretation of dependent types [3]. There is no syntactic translation corresponding to the indexed variant.

2.2 Turning the setoid model into a syntactic translation

Using the equivalence of indexed families and display maps we turned Altenkirch's variant of the setoid model [1] into a syntactic translation [16, Sections 4 and 5]. We extended the translation with proofs that coercion respects reflexivity thus the translation justifies a definitional computation rule for the identity type. In the model, this property is only respected when the metatheory has functional extensionality [14]. However the translation justifies this since definitional equality supports functional extensionality by the η law.

The setoid translation was further turned into a setoid type theory [16, Section 6]. Apart from adding all the components of the translation as syntax to a plain type theory, we had to add a new congruence eliminator for the function space. This is in line with the setoid model where a function is modelled by a pair: a function and a proof that it respects equality.

2.3 An open universe of propositions

We were not able to find a way to make the setoid model justify an open universe of propositions where a proposition is a type where all the elements are (propositionally) equal. The problem is that once we add a universe of propositions, the equality of this (logical equivalence) is not a proposition anymore. If we truncate it to be a proposition, we are not able to define coercion for propositions. It seems that the solution is to use untruncated globular setoids instead of setoids. A globular setoid is a globular set where the relation space is propositional and an equivalence relation.

As a first step achieving this goal, we formalised the globular type model of type theory [11].

3 Summary

As a result of this project, we have a deeper understanding of the connections between models of type theory and syntactic translations, we defined a setoid translation and a setoid type theory and formalised the globular type model of type theory. Our research paves the way for converting more models of type theory into syntactic translations or new type theories.

The research resulted in a draft paper on setoid type theory [16] (to be submitted at the beginning of October 2018) and the Agda formalisations [8]–[15].

References

[1] Thorsten Altenkirch. Extensional equality in intensional type theory. In *14th Symposium on Logic in Computer Science*, pages 412 – 420, 1999.

[2] Abhishek Anand, Simon Boulier, Cyril Cohen, Matthieu Sozeau, and Nicolas Tabareau. Towards certified meta-programming with typed template-coq. In *ITP*, volume 10895 of *Lecture Notes in Computer Science*, pages 20–39. Springer, 2018.

[3] Jean-Philippe Bernardy, Patrik Jansson, and Ross Paterson. Proofs for free — parametricity for dependent types. *Journal of Functional Programming*, 22(02):107–152, 2012.

[4] Simon Boulier, Pierre-Marie Pédrot, and Nicolas Tabareau. The next 700 syntactical models of type theory. In *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs*, CPP 2017, pages 182–194, New York, NY, USA, 2017. ACM.

[5] Peter Dybjer. Internal type theory. In *Lecture Notes in Computer Science*, pages 120–134. Springer, 1996.

[6] Martin Hofmann. Conservativity of equality reflection over intensional type theory. In *TYPES 95*, pages 153–164, 1995.

[7] Ambrus Kaposi. *Type theory in a type theory with quotient inductive types*. PhD thesis, University of Nottingham, 2017.

[8] Ambrus Kaposi. Agda formalisation of reducing an inductive-inductive type to quotient inductive types. `https://bitbucket.org/akaposi/erc/tt-in-tt/src/HEAD/Extrinsic/TTsmallRec.agda`, 2018.

[9] Ambrus Kaposi. Agda formalisation of reducing inductive-inductive types to indexed inductive types. `https://bitbucket.org/akaposi/erc/elims/src/HEAD/agda/IIReduction/IIReductionExamples`, 2018.

[10] Ambrus Kaposi. Agda formalisation of the display map variant of the graph model. `https://bitbucket.org/akaposi/erc/tt-in-tt/src/HEAD/Graph/DisplayMap/Readme.agda`, 2018.

[11] Ambrus Kaposi. Agda formalisation of the globular types model of type theory. `https://bitbucket.org/akaposi/tt-in-tt/src/HEAD/Glob`, 2018.

[12] Ambrus Kaposi. Agda formalisation of the graph model. `https://bitbucket.org/akaposi/erc/tt-in-tt/src/HEAD/Graph/Readme.agda`, 2018.

[13] Ambrus Kaposi. Agda formalisation of the setoid model. `https://bitbucket.org/akaposi/tt-in-tt/src/HEAD/Setoid/Readme.agda`, 2018.

[14] Ambrus Kaposi. Agda formalisation of the setoid model with definitional computation rule for the identity type. `https://bitbucket.org/akaposi/tt-in-tt/src/HEAD/Setoid/SPropRefl/Func.agda`, 2018.

[15] Ambrus Kaposi. Agda formalisation showing that the set model of type theory supports equality reflection. `https://bitbucket.org/akaposi/tt-in-tt/src/HEAD/StandardModel/ExtIden.agda`, 2018.

[16] Ambrus Kaposi. Setoid type theory. `http://akaposi.github.io/brutal.pdf`, 2018.

[17] Nicolas Oury. *Extensionality in the calculus of constructions*, pages 278–293. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[18] Thomas Streicher. *Semantics of Type Theory: Correctness, Completeness, and Independence Results*. Birkhauser Boston Inc., Cambridge, MA, USA, 1991.