Importing Agda Proofs in Logipedia

Frédéric Blanqui & Guillaume Genestier
frederic.blanqui@inria.fr

2019-2020

Location

Deducteam, Laboratoire Spécification et Vérification (LSV), ENS Paris-Saclay. The ENS is currently located at Cachan but will move in April 2020 to Gif-sur-Yvette, rue Noetzlin, near PCRI, IUT Orsay and Centrale Supélec.

1 Context of the Internship

The Logipedia project

Logipedia [1] is an online encyclopedia of formal proofs, shared between several proof assistants (Coq, HOL Light, Isabelle/HOL, Lean, Matita, PVS). All the proofs are expressed in the logical framework Dedukti [2, 5], which implements an extension of the Edinburgh Logical Framework LF [6] with rewriting rules. To enrich the encyclopedia, proofs are first developed in a proof assistant and then translated to Dedukti.

The whole community of proof assistant users will take advantage of such an encyclopedia. Indeed, it avoids the community to rebuild existing work, since it allows the users to import the theorems they need in their favorite proof assistant, no matter the system the development was originally made in.

Weak theories are well-suited for sharing

Most proof assistants implement strong logic, in which it is possible to prove many theorems. But taking naively the union of two such strong theories leads to an inconsistent logic.

On the other hand, a weak theory allows to prove less theorems. In return, one can hope to encode it in every proof assistant. This strategy is the one adopted by Logipedia.

François's Thiré presents in [7] a protocol to translate arithmetic proofs coming from Matita to Simple Type Theory with Prenex Polymorphism, a weak theory included in most existing proof assistants.

Agda

Agda [3] is a dependently typed functional programming language developed at Chalmers University (Sweden). Thanks to Curry-Howard correspondence, it can be seen (and is often used) as a proof assistant.

2 Objectives of the Internship

Importing proofs made in Agda in Logipedia requires to have a translator from Agda to Dedukti. Likewise, importing proofs available in Logipedia into Agda requires to have a translator from Dedukti to Agda.

The aim of this internship, would be to develop such a bidirectional translator.

A prototypical translator from Agda to Dedukti [4], which covers only part of the features offered by Agda, has been developed. For instance, translation of inductive types is performed only in a peculiar simple case, and translation of co-inductive types is not performed at all.

Depending on the taste of the candidate, two subgoals can be adressed during this internship:

1. Today Logipedia contains hundreds of proofs expressed in Simple Type Theory and shared between six proof assistants.

   But, those proofs are not yet accessible to the Agda users. The aim of this internship could be to provide a version of these proofs in Agda, by developing a translator from Dedukti to Agda.

   However, among the targeted systems, Agda will be the first predicative one. Hence, it is probable that some proofs in Simple Type Theory with Prenex Polymorphism cannot be directly translated to Agda. Among those, some might really need impredicativity and cannot be expressed in Agda, but, in an arithmetic library, most of the uses of impredicativity should be deletable.

   Developing techniques and tools to detect which proofs use impredicativity, and eliminate it (when possible) will probably be the first achievement of this internship.

2. Today, the translator from Agda to Dedukti outputs proofs in a partial encoding of Agda's logic in Dedukti.

   However most of the exported proofs do not use the full expressiveness power of this logic, giving us great hopes to be able to translate the majority of this library to various proof assistants like Coq, HOL Light, Matita, Lean...

   A possible aim of the internship could hence be to define a (necessarily partial) translation from the encoding of the logic of Agda to those proofs assistants.

   An analog translation has been performed by François Thiré [7]. The strategy he adopted was to first translate proofs to the encoding in Dedukti of an as weak as possible logic, namely Simple Type Theory with Prenex Polymorphism, and then to export it.

   Hence, the aim is to define a (necessarily partial) translation from the encoding of the logic of Agda to the Simple Type Theory with Prenex Polymorphism. Then, we could reuse the exporter developed by François Thiré.

3 Requirements

Some familiarity with functional programming.

References

[1] Logipedia. http://logipedia.science

[2] Dedukti. https://deducteam.github.io/

[3] Agda. https://agda.readthedocs.io/en/v2.5.4.2/index.html

[4] Agda2Dedukti. https://github.com/GuillaumeGen/Agda2Dedukti

[5] A. Assaf, G. Burel, R. Cauderlier, D. Delahaye, G. Dowek, C. Dubois, F. Gilbert, P. Halmagrand, O. Hermant, R. Saillard. Dedukti: a Logical Framework based on the $\lambda\Pi$-Calculus Modulo Theory. 2019

[6] R. Harper, F. Honsell, G. Plotkin. A framework for defining logics. *JACM* 40(1):143–184, 1993.

[7] F. Thiré. Sharing a library between proof assistants: Reaching out the HOL family. LFMTP, 2018