Programming with Objects in Theorem Provers based on Martin Löf Type Theory

Anton Setzer

Dept of Computer Science, Swansea University, Singleton Park, Swansea, UK,
a.g.setzer@swan.ac.uk

In programming, the main programming paradigm used is object-orientation. The reason is that it supports well the hiding of implementation details from the outside and the change of implementations details of units without affecting the whole code. The question is whether it is possible to apply a similar approach in the area of proofs, and get similar benefits. Interactive proofs typically use a huge amount of lemmas and definitions, and techniques from programming might help to organise them in a better way.

One way of addressing this question is by looking at Martin-Löf type theory (MLTT), where proofs and programs are the same. This allows it to transfer concepts between programming and proving in a direct way.

As a first step we will look at how to represent the notion of an object in MLTT, extended by coinductively defined sets [AAS16]. We follow the approach described in [Set12,APTS13], where coinductively defined sets will be given by their observations or elimination rules. An object is essentially determined by its methods', and therefore one can represent objects as elements of coalgebras.

We will introduce a notion of extending an object. A novelty in the context of MLTT is that we obtain indexed state-dependent coalgebras, and correspondingly state dependent objects. In state dependent objects, the methods available change after applying method calls. An example is a safe stack, for which the the pop method is only available if the stack is non-empty. We will as well combine objects with state-dependent interactive programs.

If time permits we will introduce related research on representing processes in MLTT [IS16] (joint work with Bashar Igried). Processes will be again represented coalgebraically as non-well-founded trees, with branching over the transitions a process can make.

References

[AAS16] Andreas Abel, Stephan Adelsberger, and Anton Setzer. *Interactive programming in Agda – objects and graphical user interfaces.* To appear in Journal of Functional Programming. Preprint available at http://www.cs.swan.ac.uk/∼csetzer/articles/ooAgda.pdf, 2016.

[APTS13] Andreas Abel, Brigitte Pientka, David Thibodeau, and Anton Setzer. *Copatterns: Programming infinite structures by observations.* In Roberto Giacobazzi and Radhia Cousot, editors, Proceedings of the 40th annual ACM SIGPLAN-SIGACT Symposium on Principles of programming languages, POPL '13, ACM, New York, USA, 2013, pp. 27–38.

[IS16] Bashar Igried and Anton Setzer. *Programming with monadic CSP-style processes in dependent type theory.* To appear in proceedings of TyDe 2016, Type-driven Development, preprint available from http://www.cs.swan.ac.uk/∼csetzer/articles/TyDe2016.pdf, 2016.

[Set12] Anton Setzer. *Coalgebras as types determined by their elimination rules.* In P. Dybjer, Sten Lindström, Erik Palmgren, and G. Sundholm, editors, Epistemology versus Ontology, volume 27 of Logic, Epistemology, and the Unity of Science, Springer, Dordrecht, Heidelberg, New York, London, 2012, pp. 351–369.