Chapter7:Conclusions and future work

1 Conclusions

In this thesis, we have introduced the notion of algebraic design framework and after that, we have chosen a kernel one (ASL with refinement) to present an implementation strategy of such a framework reusing the current technology of therem provers for type theories with inductive and dependent types.

First, we have presented a new principle of encoding in an expressive type theory UTT with a higher-order intuitionistic logic and inductive and dependent types. We have shown that the new principle of encoding improves the one of LF and we have presented several examples of encoding of logical systems including one which is not possible to develop in LF.

Next, we have presented the ASL framework with refinement generalizing the semantics of the behavioural operators in an algebraic institution with arbitrary partial congruences and equivalence relations between algebras. We have also defined the notion of behavioural algebraic institution in order to define the normal form of the behavioural operators and to relate the generalised semantics with an alternative semantics presented in [HS96] for higher-order logic.

After that, we have presented some of the proof systems presented in [HWB97] and [Hen97] for the deduction of properties from ASL specifications and for the refinement of ASL specifications. We have redesigned a certain kind of non-structured proof systems, the ones that were inductively defined by specification expressions adding specific extra rules and axioms for each case, since it was not possible to give an adequate encoding in UTT because originally they were infinitary proof systems. We have presented them as finitary proof systems with first-order and higher-order logic with a concrete observational equality.

Finally, we have presented the encoding of some of the proof systems for deduction and refinement in UTT. Most of the encodings are adequate, in the sense that there exists a bijection between the derivations of the encoded proof systems and the normal forms of the inhabitants of the inductive relations which represent the proof systems. One exception is the proof system for refinement because proof obligations to represent side-conditions are used.

As we have mentioned several times in this thesis, the main goal of this work is to reuse current and future technology of theorem provers of expressive type theories to develop theorem provers for algebraic design frameworks. Today, we have the following proof support from the current proof checker of UTT [LP92]:

- Type inference. Given a term of the type theory, the system calculates the principal type of the term.
- Refinement of proofs. The basic idea of refinement of proofs is to prove a goal with the conclusion of a theorem which is already proved. In the case of the Lego proof assistant the procedure is as follows:
  - If the theorem and the goal unifies, the goal is proved.
  - If not, if the theorem is a Π-abstraction, subgoals with the type of the arguments of the theorem are generated till the resulting theorem,

after applying the proofs of the subgoals (represented as metavariables) to the original theorem, is unifiable with the original goal. See Lego's manual ([LP92]) for examples.

- Otherwise the refinement fails.

- Assistance in inductive reasoning. The main assistance of this issue is the automatic generation of the inductive principles and computational rules associated to an inductive type from the formation and introduction rules which defines the inductive type.

- Inversion tactics. These tactics consists basically of searching a proof of a goal which is an application of an inductive relation to a whole list of correct arguments. The searching procedures is to prove all the premises of the constructors which succesfully refine the current goal. See [McB] for details.

2 Future work

Some possible improvements of the presented framework which in most of the cases can also be applied to CASL are the following:

- Include functional and object-oriented programming languages in these frameworks. We consider the functional paradigm easier to begin with and we think that the work of Aspinall in $ASL+$ ([SST92], [Asp97]) is a good start. On the other hand, we think that further research is needed in developing proof systems for verification of functional programs and in adding new features to the functional programming language like for example laziness or control operators.

- Make more flexible the notion of refinement of the framework presented in this thesis in a way that the signature of the abstract and refined specification do not have to coincide, and introduce in the resulting frameworks higher-order parameterisation and modular facilities.

- Develop medium-size case studies like for example in the design of critical parts of compiler design or global computation.

References

[Asp97] David Aspinall. *Type Systems for Modular Programs and Specifications.* PhD thesis, University of Edinburgh, 1997.

[Hen97] Rolf Hennicker. *Structured Specifications with Behavioural Operators: Semantics, Proof Methods and Applications.* Habilitationsschrift, Institut für Informatik, Ludwig-Maximilians-Universität München, June 1997.

[HS96] Martin Hofmann and Donald Sannella. On behavioural abstraction and behavioural satisfaction in higher-order logic. *Theoretical Computer Science*, 167:3–45, 1996.

[HWB97] Rolf Hennicker, Martin Wirsing, and Michel Bidoit. Proof systems for structured specifications with observability operators. *Theoretical Computer Science*, 173, February 1997.

[LP92] Zhaohui Luo and Randy Pollack. LEGO proof development system: User's manual. Report ECS-LFCS-92-211, Department of Computer Science, University of Edinburgh, May 1992.

[McB] Conor McBride. Inverting inductively defined relations in LEGO. To appear in TYPES96.

[SST92] Donald Sanella, Stefan Sokolowski, and Andrzej Tarlecki. Toward formal development of programs from algebraic specifications: parameterisation revisited. *ECS-LFCS-92-222*, July 1992.