Expander2, a Haskell-based prover and rewriter

Peter Padawitz
TU Dortmund, Germany

Expander2 is a flexible multi-purpose workbench for interactive term rewriting, graph transformation, theorem proving, constraint solving, flow graph analysis and other procedures that build up proofs or other rewrite sequences. Moreover, tailor-made interpreters display terms as two-dimensional structures ranging from trees and rooted graphs to a variety of pictorial representations that include tables, matrices, alignments, partitions, fractals and various tree-like or rectangular graph layouts.

An Expander2 specification consists of a signature with functions, predicates, axioms, theorems and conjectures (terms to be rewritten or formulas to be solved or proved). It describes a set of algebraic (constructor-based) and/or coalgebraic (destructor-based) types (formerly called *swinging types*). Syntactically, it follows Haskell (for presenting functions) and usual mathematical notations (for presenting relations and propositional, predicate-logic, modal or temporal operators). Predicates are interpreted as the least or greatest solutions of their Horn resp. co-Horn axioms.

The user interacts with the system at three levels of control over proofs and computations. At the top level, rules like Noetherian induction and incremental fixpoint co/induction are applied locally and step by step. At the medium level, goals are co/resolved or narrowed, i.e, axioms are applied exhaustively and iteratively. At the bottom level, built-in rules (some of them executing Haskell programs) simplify, i.e., (partially) evaluate terms and formulas, and thus hide routine steps of a proof or computation. Simplifications may be executed automatically after each step performed at the top or medium level.

As the co/Horn axioms co/resolved or narrowed upon are part of the user-defined specification, so additional simplification rules (equations or equivalences) may be entered into the specification. Recently, functional and logical fixpoint operators have been integrated into the simplifier along with corresponding (non-incremental) co/induction rules. Proofs and other rewrite sequences are automatically translated into proof terms that can be evaluated and modified later. Of course, a textual record listing all elements of the sequence and the rules producing them is also generated.

Expander2 has been written in O'Haskell, an extension of Haskell with object-oriented features for reactive programming and with an easy-to-use interface to Tcl/Tk. O'Haskell did not only allow us to develop a comfortable GUI for Expander2, but also to meet the other design goals of the system like the integration of testing, proving and visualizing deductive methods, working at several levels of interaction (see above) and keeping it open for extensions and adaptations of individual components to changing demands.

The talk will start out from a small demonstration of the various ways terms and formulas can be edited, represented and simplified with Expander2. Then we give a short overview of the system components, the interfaces to each other, their interaction with the user via GUI events and their implementation in terms of O'Haskell's templates (monadic types involving methods that access and modify global variables concurrently). The third part of the talk introduces into one of the main uses of Expander2, namely the generation of proofs or solutions of predicate-logic formulas with co/resolution, narrowing and co/induction. Roughly said, these rules generalize classical model checking from Kripke structures to arbitrary initial or final models. We will present examples of both kinds and carry out sample proofs with Expander2.

References

[1] P. Padawitz, *Computing in Horn Clause Theories*, Springer 1988

[2] P. Padawitz, *Deduction and Declarative Programming*, Cambridge University Press 1992

[3] P. Padawitz, *Inductive Theorem Proving for Design Specifications*, J. Symbolic Computation 21 (1996) 41-99

[4] P. Padawitz, *Proof in Flat Specifications*, in: E. Astesiano et al., eds., *Algebraic Foundations of Systems Specification*, IFIP State-of-the-Art Report, Springer (1999) 321-384

[5] P. Padawitz, *Swinging Types = Functions + Relations + Transition Systems*, Theoretical Computer Science 243 (2000) 93-165

[6] P. Padawitz, *Expander2: A Formal Methods Presenter and Animator*, *link*

[7] P. Padawitz, *Expander2: Towards a Workbench for Interactive Formal Reasoning*, in: H.-J. Kreowski et al., *Formal Methods in Software and Systems Modeling*, Essays Dedicated to Hartmut Ehrig, Springer LNCS 3393 (2005) 236-258

[8] P. Padawitz, *Expander2: Program verification between interaction and automation*, Proc. WFLP 2006, Elsevier ENTCS 177 (2007) 35-57

[9] P. Padawitz, *Expander2 as a Prover and Rewriter*, *link*

[10] P. Padawitz, *Algebraic Model Checking and more*, Electronic Communications of the EASST 26 (2010), *link*; slides with more examples: *link*

[11] P. Padawitz, *From grammars and automata to algebras and coalgebras*, Proc. CAI 2011, Springer LNCS 6742 (2011) 21-43, revised version: *link*

[12] P. Padawitz, *From fixpoint to predicate co/induction and its use in standard models*, TU Dortmund 2011, *link*

[13] P. Padawitz, *Dialgebraic Specification and Modeling*, slides in preparation, TU Dortmund 2011, *link*