ANALYSIS AND SYNTHESIS OF INDUCTIVE FAMILIES

Hsiang-Shang Ko
University College, Oxford

D.Phil. thesis
Hilary 2014

Abstract

Based on a natural unification of logic and computation, Martin-Löf's *intuitionistic type theory* can be regarded simultaneously as a computationally meaningful higher-order logic system and an expressively typed functional programming language, in which proofs and programs are treated as the same entities. Two modes of programming can then be distinguished: in *externalism*, we construct a program separately from its correctness proof with respect to a given specification, whereas in *internalism*, we encode the specification in a sophisticated type such that any program inhabiting the type also encodes a correctness proof, and we can use type information as a guidance on program construction. Internalism is particularly effective in the presence of *inductive families*, whose design can have a strong influence on program structure. Techniques and mechanisms for facilitating internalist programming are still relatively lacking, however.

This dissertation proposes that internalist programming can be facilitated by exploiting an interconnection between internalism and externalism, expressed as isomorphisms between inductive families into which data structure invariants are encoded and their simpler variants paired with predicates expressing those invariants. The interconnection has two directions: one *analysing* inductive families into simpler variants and predicates, and the other *synthesising* inductive families from simpler variants and specific predicates. They respectively give rise to two applications, one achieving a modular structure of internalist libraries, and the other bridging internalist programming with relational specifications and program derivation. Theoretically, the key datatype-generic mechanisms supporting the applications — *parallel composition of ornaments* and *relational algebraic ornamentation* — are further characterised in terms of lightweight category theory. A significant number of the results are completely formalised in the Agda programming language.