Abstract

Parametric polymorphism in functional programming languages with explicit polymorphism is the property that polymorphic programs behave the same way at all type instantiations. This can be formulated more precisely using Reynold's notion of relational parametricity, which states that polymorphic functions should preserve relations. It has been known for a long time that parametric polymorphism can be used to encode inductive and coinductive data types, and this has been shown in a logic for parametricity suggested by Abadi and Plotkin.

In this dissertation we propose new category theoretic formulations of parametricity for models of the second-order lambda-calculus and models of a polymorphic lambda-calculus with linear function types and fixed points. These parametric models are models of Abadi and Plotkin's logic for parametricity, called parametric APL-structures and LAPL-structures, respectively. We show how that the encodings of inductive and coinductive types using parametric polymorphism give rise to initial algebras and final coalgebras in APL- and LAPL-structures and, using Plotkin's encodings, we show how to solve recursive domain equations in LAPL-structures.

Moreover, we show that the notions of APL- and LAPL-structures are general by constructing different examples. We construct a parametric APL-structure based on the per-model and a domain-theoretic parametric LAPL-structure. Based on recent work by Simpson and Rosolini we show how to construct parametric LAPL-structures using synthetic domain theory, and we device general ways of constructing parametric LAPL- and APL-structures using parametric completion processes.

Using the LAPL-structure constructed using synthetic domain theory we prove consequences of parametricity for a variant of the Lily programming language.