Type systems for dynamic instances for algebraic effects and handlers

Albert ten Napel

1 Introduction

In this thesis we will devise a type and effect systems that can type some programs that use dynamic instances for algebraic effects and handlers.

1.1 Problem statement

1.2 Proposed solution

1.3 Thesis structure

2 Background

2.1 Algebraic effects

Algebraic effects and handlers are a way of treating computational effects that is modular and compositional.

theory? Category theory, algebras forget about equations

With algebraic effects impure behavior is modeled using operations. For example a mutable store has get and put operations, exceptions have a throw operation and console input/output has read and print operations. Handlers of algebraic effects generalize handlers of exceptions by not only catching called operations but also adding the ability to resume where the operation was called. While not all monads can be written in terms of algebraic effects, for example the continuation monad, in practice most useful computation effects can be modeled this way.

For example we can model stateful computations that mutate an integer by defining the following algebraic effect signature:

$$\mathbf{State} := \{\mathbf{Get} : () \to \mathbf{Int}, \mathbf{Put} : \mathbf{Int} \to ()\}$$

State is an effect that has two operations **Get** and **Put**. **Get** takes unit has its parameter type and returns an integer value, **Put** takes an integer value and returns unit.
We can then use the **State** operations in a program:

$$inc\ () := x \leftarrow \mathbf{Get}\ ();\ \mathbf{Put}\ (x + 1)$$

The program **inc** uses the **Get** and **Put**, but these operations are abstract. Handlers are used to give the abstract effects in a computation semantics.

2.2 Handlers

Exception handlers can be generalized by also supplying a continuation to the programmer. The programmer can then decide to continue at the point where the exception was thrown. These generalized exception handlers were further generalized by Plotkin and Pretnar to allow for many different effects.

For example the following handler gives the **Get** and **Put** the usual function-passing style state semantics:

$$state\ := \mathbf{handler}\ \{\ \mathbf{return}\ v \to \lambda s \to v, \mathbf{Get}\ ()\ k \to \lambda s \to k\ s\ s, \mathbf{Put}\ s\ k \to \lambda s' \to k\ ()\ s, \}$$

We are able to give different interpretations of a computation by using different handlers. We could for example think of a transaction state interpretation where changed to the state are only applied at the end if the computation succeeds.

citations example of interpretation more examples? (effects)

2.3 Instances
2.4 Dynamic instances
2.5 Resources
2.6 Effect systems for algebraic effects
3 Type system
4 Formalization
5 Related work
6 Conclusion and future work
Bibliography