Greenlight Presentation Msc Thesis

Elaine: Elaborations of Higher-Order Effects as First-Class Language Feature

Terts Diepraam
Supervisors: Casper Bach Poulsen & Cas van der Rest
Thesis advisor: Matthijs Spaan

June 29, 2023

Effects

- Printing,
- State,
- Filesystem access,
- Coroutines,
- Exceptions,
- & everything else monads can do.

Effects

Many programming languages have limitations when dealing with effects:

- the effects are not reflected in the type system,
- the effects are not modular.

What they do have is limited to specific features (e.g. exceptions).

```
def foo(x: int) -> int: # <- does not include effect
    print(x)            # <- implementation is fixed
    return x * x
```

Algebraic Effects

In languages with algebraic effects, we can:

- declare custom effects,
- and define **handlers** for them.

Effects need to satisfy the *algebraicity property*.

Languages: Koka, Frank, Eff, and more.

Example effects: state, IO, iteration, nondeterminism

Higher-Order Effects

Higher-order effects: effect operations that take effectful computations as parameters.

They are not algebraic: we cannot make them modular.

Example effects: Reader monad with local, exception catching, lambda abstraction

Hefty Algebras

Higher-order effects are **elaborated** into expressions with only algebraic effects.

The elaboration provides modularity we want.

Elaine

Language designed in this thesis

Core idea: apply hefty algebras to PL design

Novel features:

- elaborations as language construct
- implicit elaboration resolution
- elaboration compilation