Programming languages - final project

Konrad Werbliński - 291878

January 7, 2020

1 Introduction

Algebraic effects have received lot of recognition in the functional programming community in recent years. They can be viewed as resumable exceptions, providing more composable approach for handling side effects than monads. We introduce Kwoka - stripped down and simplified version of the Koka language [1]. In this report we describe our work on implementation of the type inference for algebraic effects introduced in the article by Daan Leijen [2]. We follow his approach of using extensible effect rows with scoped labels for inference of

In this report we specify our variant of a language. We describe the details of the implemented type system. Then, we give an overview of the implementation and instructions for building and running Kwoka. Finally we discuss further work on this project.

2 Kwoka language description

Kwoka syntax is based on the syntax of the original Koka language and Rust.

```
fn fib(n)
{
  if n == 0 || n == 1 then
    n
  else
    fib(n - 1) + fib(n - 2)
}
```

Kwoka supports higher order and polymorphic functions.

```
fn compose(f, g)
{
  \x => f(g(x))
}
```

Something about algebraic effects

```
fn makeGreeting(name)
{
  Hello() ^ " " ^ name
}

effect Hello
{
  Hello() :: String
}

fn main()
{
  handle<Hello>(makeGreeting("General Kenobi."))
  {
    return(x) => (),
    Hello() => resume("Hello there!")
  }
}
```

Popular example of usage of the algebraic effects is using them to express exceptions.

```
effect Exc
{
  Raise(String)
}

fn saveDiv(n, m)
{
  if m == 0 then
    Raise("Division by zero")
  else
    n / m
}

fn main()
{
  handle<Exc>(saveDiv(4, 5))
  {
    return(x) => x,
    Raise(s) => 0
  }
}
```

3 Type Inference

4 Implementation

5 Installing and running Kwoka

Kwoka requires stack for building.

- Building: `stack build`
- Running: `stack run yourSourceFile.kwoka`
- Running unit tests: `stack test`

6 Further work

References

[1] Daan Leijen. Koka: Programming with row polymorphic effect types. *Electronic Proceedings in Theoretical Computer Science*, 153, 06 2014.

[2] Daan Leijen. Type directed compilation of row-typed algebraic effects. In *Proceedings of Principles of Programming Languages (POPL'17), Paris, France*, January 2017.