Effect Systems in Haskell - Part II

Sanchayan Maity

Agenda

- Cover two papers on Effect Systems.
  - Generalized Evidence Passing for Effect Handlers[1]
  - Effect Handlers in Haskell, Evidently[2]
- Some sections today's discussion isn't going to cover
  - Efficiency/Performance of the library or effect system itself
  - How to use effect systems
  - Comparison of effect system libraries or how to choose one

[1] Generalized Evidence Passing for Effect Handlers
[2] Effect Handlers in Haskell, evidently

Recap, what's it all about

- **Separate syntax from semantics**
- **Interpret your abstract syntax tree in various ways**
- **Not losing performance while having both**

Recap, why effect systems

- Monads to model effects but monads don't compose[3]
- transformers/mtl has limitations
  - Monad transformer stacks are rigid
  - Doesn't allow handling something like `Reader Int (Reader String)` due to functional dependencies

```
class Monad m => MonadReader r m | m -> r
```

  - More than a few effects in stack become unwieldy
  - n-square instances problem

[3] Composing Monads by Mark Jones and Luc Duponcheel

Evidence passing based effect system libraries

- `EvEff`
- `MpEff`
- `speff`
- `cleff` based on `ReaderT IO`
- `effectful` based on `ReaderT IO`
- others?

Some myths[4]

- Many extensible effects libraries are implemented with free(r) monads (True)
- Therefore extensible effects = free(r) monads (False)
- Free(r) monads require certain mathematical concepts to grasp (True)
- Free(r) monads don't have very good performance (True, to some extent)
- Therefore extensible effects are slow, ivory-towerish toys (False)

[4] ReaderT pattern is just extensible effects

Effects can be implemented in various way

- `free` monads
- `ReaderT IO`
- `CPS`
- `delimited continuations`

What's the gist

- How do you pass the `handler` for the `effect`?

Generalized control flow

- Languages that expose a yield primitive actually have a way to access delimited continuations! Central result of the paper by James-Sabry [5][6].

[5] Yield: Mainstream Delimited Continuations
[6] Delimited Continuations are all you need

WTH are delimited continuations

- Delimited Continuations for Everyone[7]

[7] Delimited Continuations for Everyone

How does one define an effect

```
data Reader a e ans = Reader { ask :: !(Op () a e ans)
                             }

data State a e ans = State { get :: !(Op () a e ans)
                           , put :: !(Op a () e ans)
                           }
```

Multi-prompt delimited control

```
data Ctl e a = Pure { result :: !a }
             | forall h b e' ans.
               Control {
       -- prompt marker to yield to (in type context `::ans`)
                 marker :: Marker h e' ans,
       -- the final action, just needs the resumption (:: b -> Eff e' ans)
                 op     :: !((b -> Eff e' ans) -> Eff e' ans),
       -- the (partially) build up resumption;
       -- (b -> Eff e a) :~: (b -> Eff e' ans)` by the time
       -- we reach the prompt
                 cont   :: !(b -> Eff e a) }

data Context e where
  CCons :: !(Marker h e' ans) -> !(h e' ans) -> !(ContextT e e')
        -> !(Context e) -> Context (h :* e)
  CNil  :: Context ()
```

Key ideas

- Multi Prompt
- Evidence Passing
- Tail Resumptive Operations
- Bubbling Yields
- Short cut resumptions
- Monadic Translation
- Bind-inlining and Join-Point Sharing

Dig in

- Dig in to the paper!

Resources

- Alexis King on “Delimited Continuations, Demystified” @ZuriHac2023
- GHC Proposal: Delimited continuation primops
- Delimited Continuations
- Efficient Compilation of Algebraic Effect Handlers - Ningning Xie
- From Folklore to Fact: Comparing Implementations of Stacks and Continuations
- Compiler and Runtime Support for Continuation Marks
- Capturing the Future by Replaying the Past Functional Pearl
- From Delimited Continuations to Algebraic Effects in Haskell
- Concurrent System Programming with Effect Handlers
- Eff Directly in OCaml
- Retrofitting Effect Handlers onto OCaml

Questions?

- Reach out on
  - Email: sanchayan@sanchayanmaity.net
  - Mastodon: https://sanchayanmaity.com/@sanchayan
  - Blog: https://sanchayanmaity.net
  - Telegram:
    - t.me/fpncr
    - t.me/SanchayanMaity