Monads

Samuel Mimram

1 Monads in Haskell

Here is an excerpt of `http://www.haskell.org/haskellwiki/Monad`:

```
Monads can be viewed as a standard programming interface to various data or control structures,
which is captured by the Monad class.  All common monads are members of it:
```

```
class Monad m where
  (>>=) :: m a -> (a -> m b) -> m b
  return :: a -> m a
```

```
In addition to implementing the class functions, all instances of Monad should obey the
following equations:
```

```
return a >>= k  =  k a
m >>= return  =  m
m >>= (\x -> k x >>= h)  =  (m >>= k) >>= h
```

1. What does the `Maybe` monad defined below do?

   ```
   data Maybe a = Nothing | Just a

   instance Monad Maybe where
       return         = Just
       Nothing  >>= f = Nothing
       (Just x) >>= f = f x
   ```

2. What does the `List` monad defined below do?

   ```
   instance Monad [] where
       m >>= f  = concatMap f m
       return x = [x]
   ```

3. A *Kleisli triple* $(T, \eta, (-)^*)$ on a category $\mathcal{C}$ consists of

   – a function $T : \mathrm{Ob}(\mathcal{C}) \to \mathrm{Ob}(\mathcal{C})$,
   – a function $\eta_A : A \to TA$ for every object A of $\mathcal{C}$,
   – a morphism $f^* : TA \to TB$ for every morphism $f : A \to TB$,

   such that for every objects A, B, C and morphisms $f : A \to TB$ and $g : B \to TC$,

   $$\eta_A^* = \mathrm{id}_{TA} \qquad f^* \circ \eta_A = f \qquad g^* \circ f^* = (g^* \circ f)^*$$

   Show that Kleisli triples are in bijection with monads on $\mathcal{C}$.

4. Construct the Kleisli category associated to a Kleisli triple.

2 Usual monads

Recall that in a cartesian closed category $\mathcal{C}$ we have a bijection $\mathcal{C}(S \times A, B) \simeq \mathcal{C}(A, S \Rightarrow B)$. You can use the fact that λ-calculus is an internal language for cartesian closed categories.

1. Describe the adjoint functors and the bijection.
2. Describe the unit and the counit of the associated adjunction, the resulting *state monad*, and the Kleisli triple.
3. Describe the Kleisli category.
4. Implement the *read* and *write* operations.

Let us study some other monads. For each of those, provide the Kleisli triple, as well as the implementation of the expected operations.

1. Define the *reader monad* such that TA is a value of type A depending on a state in S.
2. The *stream monad* is such that $TA = A^{\mathbb{N}}$. Complete the description.
3. Define the *log monad* which models a situation where each command might write some lines in a log file.
4. Define the *flag monad* where a program might set a flag during its execution (and can never unset).
5. How can you unify the two previous monads. What is an algebra for the resulting monad?
6. The *continuation monad* is such that $TA = (A \Rightarrow S) \Rightarrow S$. Complete the description.

3 Monads in Rel

We define **Rel** as the 2-category whose 0-cells are sets, 1-cells $R : A \to B$ are relations $R \subseteq A \times B$, there is a unique 2-cell $\alpha : R \Rightarrow R' : A \to B$ whenever $R \subseteq R'$.

1. Recall both horizontal and vertical compositions in **Rel**.
2. Show that a left adjoint in **Rel** is a function.
3. What is a monad in **Rel**?

4 Monads in Span

The 2-category of **Span** is the category where

- a 0-cell is a set
- a 1-cell from A to B is a *span*: $A \xleftarrow{s} I \xrightarrow{t} B$
- a 2-cell $f : (s, t) \to (s', t')$ is a function making the following diagram commute

$$\begin{tikzcd}
& I \arrow[dl, "s"'] \arrow[dr, "t"] \arrow[d, "f"] & \\
A & I' \arrow[l, "s'"] \arrow[r, "t'"'] & B
\end{tikzcd}$$

Horizontal composition of 1-cells is given by pullback.

1. What is an endomorphism $A \to A$? A 2-cell between such endomorphisms?
2. Detail the structure of 2-category.
3. Is it really a 2-category?
4. What is a monad in this "2-category"?