TD3 – Adjunctions and monads

Samuel Mimram

15 octobre 2009

1 Kleisli categories

1. Given a monad (T, μ, η) on a category $\mathcal{C}$, we write $\mathcal{C}_T$ for the *Kleisli category* associated to the monad: its objects are the objects of $\mathcal{C}$ and morphisms $f : A \to B$ of $\mathcal{C}_T$ are the morphisms $f : A \to TB$ of $\mathcal{C}$, the composition of two morphisms $f : A \to TB$ and $g : B \to TC$ being given by $g \circ f = \mu_C \circ Tg \circ f$ and identities by $\mathrm{id}_A = \eta_A$. Show that the axioms of categories are satisfied.
2. Give a direct description of the Kleisli category associated to the exception monad.

2 Non-determinism monad

1. We write **Mon** for the category of monoids. Describe the functor $U : \mathbf{Mon} \to \mathbf{Set}$ which sends a monoid to its underlying set. The functor U is often called a *forgetful functor* because it "forgets" about the structure of monoid on a set.
2. Give an explicit description of the monoid freely generated by a set.
3. Construct a functor $F : \mathbf{Set} \to \mathbf{Mon}$ which sends a set on the monoid it freely generates.
4. Show that F is left adjoint to U.
5. Define a structure of monad on the functor $U \circ F : \mathbf{Set} \to \mathbf{Set}$.
6. Similarly define a monad $T : \mathbf{Set} \to \mathbf{Set}$ from an adjunction between **Set** and the category **CMon** of commutative monoids.
7. Describe the Kleisli category $\mathbf{Set}_T$ and explain why we can see its morphisms as non-deterministic programs.
8. Other variant: construct similarly the powerset monad on **Set** which to every set associates the set of its subsets, and give a direct description of the associated Kleisli category.

3 Free category on a graph

A *graph* is defined as a diagram $V \overset{s}{\underset{t}{\leftleftarrows}} E$ in **Set**.

1. Define the notion of morphism of graph. We write **Graph** for the category thus constructed.
2. Define the forgetful functor $U : \mathbf{Cat} \to \mathbf{Graph}$.
3. Show that this functor $F : \mathbf{Graph} \to \mathbf{Cat}$ admits a left adjoint.

4 Terminal objects and products by adjunctions

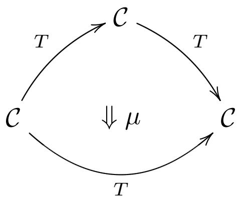
1. Show that the category **Cat** has a terminal object **1**.
2. Given a category $\mathcal{C}$, describe the *terminal functor* $T : \mathbf{Cat} \to \mathbf{1}$.
3. Given a category $\mathcal{C}$, show that the terminal functor $T : \mathcal{C} \to \mathbf{1}$ has a right (resp. left) adjoint iff the category $\mathcal{C}$ admits a terminal (resp. initial) object.
4. Given a category $\mathcal{C}$, describe the *diagonal functor* $D : \mathcal{C} \to \mathcal{C} \times \mathcal{C}$ and show that the category $\mathcal{C}$ admins cartesian products (resp. coproducts) iff the diagonal functor admits a right (resp. left) adjoint.

5 Monads generated by an adjunction

1. Recall that a functor $F : \mathcal{C} \to \mathcal{D}$ is left adjoint to a functor $G : \mathcal{D} \to \mathcal{C}$ iff there exists two natural transformations
$$\eta : \mathrm{id}_\mathcal{C} \to G \circ F \qquad \text{and} \qquad \varepsilon : F \circ G \to \mathrm{id}_\mathcal{D}$$
respectively called the *unit* and *counit* of the adjunction, such that
$$\varepsilon_F \cdot F\eta = \mathrm{id}_F \qquad \text{and} \qquad G\varepsilon \cdot \eta_G = \mathrm{id}_G \tag{1}$$
Describe the unit and counit corresponding the adjunctions studied in previous questions.

2. Recall that a 2-category of categories, functors and natural transformations can be defined. What are the vertical and horizontal compositions in this category? What is the "exchange law" in a 2-category?

3. For every monad $T : \mathcal{C} \to \mathcal{C}$, the multiplication μ can be thus seen as a 2-cell



in this 2-category. By constructing the Poincaré dual of this diagram, we thus get a representation of the natural transformation μ using *string diagrams*. Similarly, give the string diagrammatic representation of the laws defining a monad as well as the laws (1).

4. Given an adjunction $(F, G, \eta, \varepsilon)$, show that the functor $G \circ F$ can be equipped with a structure of monad.

5. [Optional] Show the property mentioned in question 1.

6. [Optional] Show that if T is a monad on a category $\mathcal{C}$ then the category $\mathcal{C}$ is in adjunction with the category $\mathcal{C}_T$.

6 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. Show that this notion of monad is equivalent to the categorical definition of monads.

2. 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
```

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

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