Monad-Based Programming WS 2021

Assignment 5

Deadline for solutions: 21.01.2022

Exercise 1 Triples* Need Not Be Kleisli (6 Points)

Complete the proof from the lecture that Kleisli triples bijectively correspond to the triples (T, η, μ). To that end

(1) define a Kleisli triple from a monad, given as a triple (T, η, μ) and verify the axioms of Kleisli tripples;

(2) define a monad in the form (T, η, μ) from a Kleisli triple and verify the axioms of monads;

(3) show that the passage $(T, \eta, \_^*) \to (T, \eta, \mu) \to (T, \eta, \_^*)$ yields an identity;

(4) show that the passage $(T, \eta, \mu) \to (T, \eta, \_^*) \to (T, \eta, \mu)$ yields an identity.

Exercise 2 Success and Failure of Monad Laws (9 Points)

List monad is defined as follows in Haskell

```
instance Monad [] where
  return x = [x]
  xs >>= f = concat (map f xs)
```

(1) Give a category-theretic defintion of this monad (i.e. in terms of η and μ) over the category of sets using the connection betwenn category-theoretic and Klesili presentations, using the previous excercise.

Note that the elements of $[A]$ can be understood as expressions of the form $a_1 \vee \ldots \vee a_n$, where $a_i \in A$. For convenience, let us denote such an expression as false if $n = 0$ and $a_1^\vee$ if $n = 1$. We could write e.g. $(a \vee b) \vee c$, which is the same as $a \vee (b \vee c)$, since both expressions correspond to the same list $[a, b, c]$.

(2) Describe η and μ in terms of this presentation and use it to show that the list monad is really a monad.

(3) Modify the presentaion from the previous clause and the argument, so as to show that the *finite powerset monad* (i.e. the one obtained by replacing finite lists by finite sets) is also a monad.

Consider the following code next

```
import Prelude hiding (and,or)

newtype DNF a = DNF { unDNF :: [[a]] }
```

---

* "*Triple*" is an old fashioned term for "*monad*".

```
  deriving (Eq, Ord, Show, Read)

newtype DNF a = DNF { unDNF :: [[a]] }
  deriving (Eq, Ord, Show, Read)

true :: DNF a
true = DNF [[]]

false :: DNF a
false = DNF []

or :: DNF a -> DNF a -> DNF a
or (DNF a) (DNF b) = DNF $ a ++ b

and :: DNF a -> DNF a -> DNF a
and (DNF []) (DNF bs) = false
and (DNF (a : as)) (DNF bs) = DNF $ (map (a++) bs) ++ (unDNF $ DNF as `and` DNF bs)

instance Monad DNF where
  return a = DNF $ [[a]]

  DNF [] >>= k = false
  DNF (a : as) >>= k = (foldl and true (map k a)) `or` (DNF as >>= k)
```

In a nutshell, we switched from lists ($[A]$) to iterated lists ($[[A]]$), but the notation is selected to be more suggestive – intuitively, instead for finite disjunctions of "atoms" from A, we are dealing with negation-free disjunctive normal forms (DNF) over A.

(4) Again, describe η and μ, derived from the above definition.

(5) Show that the specified iterated list monad is actually **not** a monad. Program an example showing that, i.e. provide two instances of the left and the right hand side of the monad law that fails.

Hint: It is advisable to view μ as a certain normalization procedure and exploit the discrepancy between $a^{\wedge} \vee (a \wedge b)$ and $(a^{\wedge})^{\vee}$, which are distinct but logically equivalent DNF's.

Exercise 3 Monads on Posets (6 Points)

A closure operator T over a poset (=partially ordered set), say $\mathcal{C}$, satisfies properties:

$$X \leq TX \qquad \text{(extensiveness)}$$
$$X \leq Y \quad \text{implies} \quad TX \leq TY \qquad \text{(monotonicity)}$$
$$TTX = TX \qquad \text{(idempotence)}$$

For example, if $\mathcal{C}$ is the standard partial order on real numbers, then the operator rounding up a number to the closest integer is a closure operator.

Recall from the lecture that we can view $\mathcal{C}$ as a category: $Ob(\mathcal{C})$ is the set of elements, $\mathsf{Hom}_{\mathcal{C}}(X, Y) = \{*\}$ if $X \leq Y$ and $\mathsf{Hom}_{\mathcal{C}}(X, Y) = \{\,\}$ otherwise.

Prove that T is a monad on $\mathcal{C}$ iff T is a closure operator.

Exercise 4 Monads from Monoids (9 Points)

A category $\mathcal{C}$ is called *monoidal* if it is equipped with the following data

- a bifunctor $\otimes\colon \mathbf{C}\times\mathbf{C}\to\mathbf{C}$ (*tensor product*);
- an object I (*unit object*);
- three natural isomorphisms: $\alpha_{A,B,C}\colon A\otimes(B\otimes C)\cong(A\otimes B)\otimes C$ (*associator*), $\lambda_A\colon I\otimes A\cong A$ (*left unitor*) and $\rho_A\colon A\otimes I\cong A$ (*right unitor*);
- the following laws (coherence conditions):

$$\begin{tikzcd}
A\otimes(I\otimes B) \arrow[rr, "\alpha_{A,I,B}"] \arrow[dr, "\mathsf{id}_A\otimes\lambda_B"'] & & (A\otimes I)\otimes B \arrow[dl, "\rho_A\otimes\mathsf{id}_B"] \\
& A\otimes B &
\end{tikzcd}$$

$$\begin{tikzcd}
A\otimes(B\otimes(C\otimes D)) \arrow[r, "\mathsf{id}_A\otimes\alpha_{B,C,D}"] \arrow[d, "\alpha_{A,B,C\otimes D}"'] & A\otimes((B\otimes C)\otimes D) \arrow[r, "\alpha_{A,B\otimes C,D}"] & (A\otimes(B\otimes C))\otimes D \arrow[d, "\alpha_{A,B,C}\otimes\mathsf{id}_D"] \\
(A\otimes B)\otimes(C\otimes D) \arrow[rr, "\alpha_{A\otimes B,C,D}"'] & & ((A\otimes B)\otimes C)\otimes D
\end{tikzcd}$$

An instructive example is a category with (selected) finitary products (Cartesian category), where we can take

- $\otimes$ to be $\times$,
- I to be the initial object 1,
- $\alpha_{A,B,C} = \langle\mathsf{id}\times\mathsf{fst},\mathsf{snd}\circ\mathsf{snd}\rangle\colon A\times(B\times C)\cong(A\times B)\times C$, $\lambda_A=\mathsf{snd}\colon 1\times A\cong A$, $\rho_A=\mathsf{fst}\colon A\otimes I\cong A$ (where $f\times g=\langle f\circ\mathsf{fst},g\circ\mathsf{snd}\rangle$ for $f\colon A\to B$, $g\colon C\to D$).

The coherence conditions can easily be obtained using equational reasoning from the following complete axiomatization of binary products:

$$\mathsf{fst}\circ\langle f,g\rangle = f \qquad \mathsf{snd}\circ\langle f,g\rangle = g \qquad \langle\mathsf{fst},\mathsf{snd}\rangle=\mathsf{id} \qquad h\circ\langle f,g\rangle=\langle h\circ f,h\circ g\rangle$$

A *monoid* in a monoidal category $\mathcal{C}$ is a triple $(M,\epsilon,\odot)$ where M is an object in $\mathcal{C}$; $\odot$ (*multiplication*) is a morphism $M\otimes M\to M$ and ϵ (*unit*) is a morphism $I\to M$ such that the following diagrams commute:

$$\begin{tikzcd}
M\otimes I \arrow[r, "\mathsf{id}_M\otimes\epsilon"] \arrow[dr, "\rho_M"'] & M\otimes M \arrow[d, "\odot"] & I\otimes M \arrow[l, "\epsilon\otimes\mathsf{id}_M"'] \arrow[dl, "\lambda_M"] \\
& M &
\end{tikzcd}
\qquad
\begin{tikzcd}
M\otimes(M\otimes M) \arrow[rr, "\alpha_{M,M,M}"] \arrow[d, "\mathsf{id}_M\otimes\odot"'] & & (M\otimes M)\otimes M \arrow[d, "\odot\otimes\mathsf{id}_M"] \\
M\otimes M \arrow[r, "\odot"] & M & M\otimes M \arrow[l, "\odot"']
\end{tikzcd}$$

It is easy to check that in a Cartesian category, these diagrams precisely capture the property that ϵ is a monoid unit, i.e. $\odot\circ\langle\mathsf{id},!\rangle=\odot\circ\langle !,\mathsf{id}\rangle=\mathsf{id}$ (first diagram) and that monoid multiplication is associative, i.e. $\odot\circ(\odot\times\mathsf{id})=\odot\circ(\odot\times\mathsf{id})\circ\alpha$ (second diagram).

Every monoid $(M,\epsilon,\odot)$ gives rise to a monad T_M, with

1. $T_M X = M\otimes X$;
2. $\eta_X=(\epsilon\times\mathsf{id})\circ\lambda^{-1}\colon X\to M\otimes X$;
3. $\mu_X=(\odot\otimes\mathsf{id})\circ\alpha\colon M\otimes(M\otimes X)\to M\otimes X$.

(1) Prove by diagram chasing that T_M, thus defined, is indeed a monad.

You can make free use of the following (famous) *Mac Lane's coherence theorem*:

Theorem: every well-formed diagram, with morphisms made of $\alpha, \lambda, \rho, \alpha^{-1}, \lambda^{-1}, \rho^{-1}, \mathsf{id}$ and $\otimes$ commutes.

For example, the monad law $\mu \circ \eta = \mathsf{id}$ is shown with the following diagram:

$$
\begin{tikzcd}
M \otimes X \arrow[r, "\lambda^{-1}"] \arrow[dr, "\lambda^{-1}\otimes\mathsf{id}"'] \arrow[ddrr, bend right, "\mathsf{id}"'] \arrow[r, phantom, "\text{(Mac Lane)}"', near end, shift right=4ex] & I \otimes (M \otimes X) \arrow[r, "\epsilon\otimes\mathsf{id}"] \arrow[d, "\alpha"] \arrow[dr, phantom, "\text{(naturality } \alpha)"] & M \otimes (M \otimes X) \arrow[d, "\alpha"] \\
 & (I \otimes M) \otimes X \arrow[r, "(\epsilon\otimes\mathsf{id})\otimes\mathsf{id}"] \arrow[dr, "\lambda\otimes\mathsf{id}"'] \arrow[d, phantom, "\text{(}\lambda\text{ isomorphsim)}"] & (M \otimes M) \otimes X \arrow[d, "\odot\otimes\mathsf{id}"] \arrow[dl, phantom, "\text{(axiom)}"] \\
 & & M \otimes X
\end{tikzcd}
$$

It is relatively easy to see that under $\otimes = \times$, $I = 1$, T_M is the familiar writer monad `Write m` from Haskell.

(2) Using the axiomatization of binary coproducts dual to the above axiomatization of finite products, prove that in a monoidal category with $\otimes = +$ (what is I?) *any* object E can be made into a monoid. What is the induced monad?