Análisis de Lenguajes de Programación

Trabajo Práctico 4

Nicolás Felipe Del Piano

Ejercicio 1.a

Tenemos la siguiente mónada con su respectiva instancia de la clase Monad.

```
newtype State a = State { runState :: Env → (a, Env) }

instance Monad State where
  return x = State (λs → (s, x))
  m >>= f = State (λs → let (v,s') = runState m s in
    runState (f v) s')
```

Las instancias de Monad deben verificar las siguientes ecuaciones:

- (monad.1)
  $$return\ x \gg\!= f \quad = \quad f\ x$$
- (monad.2)
  $$t \gg\!= return \quad = \quad t$$
- (monad.3)
  $$(t \gg\!= f) \gg\!= g \quad = \quad t \gg\!= (\lambda\ x \to f\ x \gg\!= g)$$

Debemos verificar que efectivamente *State* cumple las tres ecuaciones.

- Prueba de (monad.1)

```
    return x >>= f
= { def. return }
    (State (λs → (x, s))) >>= f
= { def. >>= }
    State (λs → let (v, s') = runState (State (λs → (x, s))) s
                in runState (f v) s')
= { def. runState }
    State (λs → let (v, s') = (λs → (x, s)) s
                in runState (f v) s')
= { aplicación }
    State (λs → let (v, s') = (x, s)
                in runState (f v) s')
= { propiedad de let }
    State (λs → runState (f x) s))
= { extensionalidad }
    State (runState (f x))
= { State inversa de runState }
    f x
```

- Prueba de (monad.2)

```
    t >>= return
= { def. >>= }
    State (λs → let (v,s') = runState t s
                in runState (return v) s')
```

$$
\begin{aligned}
&= \{ \text{ def. return } \} \\
&\quad \mathit{State}\ (\lambda s \to \mathbf{let}\ (v,s') = \mathit{runState}\ t\ s \\
&\qquad\qquad \mathbf{in}\ \mathit{runState}\ (\mathit{State}\ (\lambda s \to (v,s)))\ s') \\
&= \{ \text{ def. runState } \} \\
&\quad \mathit{State}\ (\lambda s \to \mathbf{let}\ (v,s') = \mathit{runState}\ t\ s \\
&\qquad\qquad \mathbf{in}\ (\lambda s \to (v,s))\ s') \\
&= \{ \text{ aplicación } \} \\
&\quad \mathit{State}\ (\lambda s \to \mathbf{let}\ (v,s') = \mathit{runState}\ t\ s \\
&\qquad\qquad \mathbf{in}\ (v,s')) \\
&= \{ \text{ propiedad de let } \} \\
&\quad \mathit{State}\ (\lambda s \to \mathit{runState}\ t\ s) \\
&= \{ \text{ extensionalidad } \} \\
&\quad \mathit{State}\ (\mathit{runState}\ t) \\
&= \{ \text{ State inversa de runState } \} \\
&\quad t
\end{aligned}
$$

▶ Prueba de (monad.3)

Voy a empezar por un lado de la ecuación, llegar a cierto punto y verificar que del otro lado llego a lo mismo.

$$
\begin{aligned}
&\quad (t \gg\!= f) \gg\!= g \\
&= \{ \text{ def. } \gg\!= \} \\
&\quad \mathit{State}\ (\lambda s \to \mathbf{let}\ (v,\ s') = \mathit{runState}\ (t \gg\!= f)\ s \\
&\qquad\qquad \mathbf{in}\ \mathit{runState}\ (g\ v)\ s') \\
&= \{ \text{ def. } \gg\!= \} \\
&\quad \mathit{State}\ (\lambda s \to \mathbf{let}\ (v,\ s') = \mathit{runState}\ (\mathit{State}\ (\lambda s \to \mathbf{let}\ (x,\ y) = \mathit{runState}\ t\ s \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathbf{in}\ \mathit{runState}\ (f\ x)\ y))\ s \\
&\qquad\qquad \mathbf{in}\ \mathit{runState}\ (g\ v)\ s') \\
&= \{ \text{ def. runState } \} \\
&\quad \mathit{State}\ (\lambda s \to \mathbf{let}\ (v,\ s') = (\lambda s \to \mathbf{let}\ (x,\ y) = \mathit{runState}\ t\ s \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathbf{in}\ \mathit{runState}\ (f\ x)\ y)\ s \\
&\qquad\qquad \mathbf{in}\ \mathit{runState}\ (g\ v)\ s') \\
&= \{ \text{ aplicación } \} \\
&\quad \mathit{State}\ (\lambda s \to \mathbf{let}\ (v,\ s') = (\mathbf{let}\ (x,\ y) = \mathit{runState}\ t\ s \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathbf{in}\ \mathit{runState}\ (f\ x)\ y) \\
&\qquad\qquad \mathbf{in}\ \mathit{runState}\ (g\ v)\ s') \\
&= \{ \text{ x = fst (x, y), y = snd (x, y), sustitución } \} \\
&\quad \mathit{State}\ (\lambda s \to \mathbf{let}\ (v,\ s') = (\mathbf{let}\ (x,\ y) = \mathit{runState}\ t\ s \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathbf{in}\ \mathit{runState}\ (f\ (\mathit{fst}\ (x,\ y)))\ (\mathit{snd}\ (x,\ y))) \\
&\qquad\qquad \mathbf{in}\ \mathit{runState}\ (g\ v)\ s') \\
&= \{ \text{ propiedad de let } \} \\
&\quad \mathit{State}\ (\lambda s \to \mathbf{let}\ (v,\ s') = \mathit{runState}\ (f\ (\mathit{fst}\ (\mathit{runState}\ t\ s)))\ (\mathit{snd}\ (\mathit{runState}\ t\ s))) \\
&\qquad\qquad \mathbf{in}\ \mathit{runState}\ (g\ v)\ s') \\
&= \{ \text{ v = fst (v, s'), s' = snd (v, s') } \} \\
&\quad \mathit{State}\ (\lambda s \to \mathbf{let}\ (v,\ s') = \mathit{runState}\ (f\ (\mathit{fst}\ (\mathit{runState}\ t\ s)))\ (\mathit{snd}\ (\mathit{runState}\ t\ s))) \\
&\qquad\qquad \mathbf{in}\ \mathit{runState}\ (g\ (\mathit{fst}\ (v,\ s')))\ (\mathit{snd}\ (v,\ s')))
\end{aligned}
$$

$$
\begin{aligned}
&= \{ \text{ propiedad de let } \} \\
&\quad \mathit{State}\ (\lambda s \to \mathit{runState}\ (g\ (\mathit{fst}\ (\mathit{runState}\ (f\ (\mathit{fst}\ (\mathit{runState}\ t\ s)))\ (\mathit{snd}\ (\mathit{runState}\ t\ s)))))\ (\mathit{snd}\ (\mathit{runState}\ (f\ (\mathit{fst}\ (\mathit{runState}\ t\ s)))\ (\mathit{snd}\ (\mathit{runState}\ t\ s))))) \\
&= \{ \text{ uso de where } \} \\
&\quad \mathit{State}\ (\lambda s \to \mathit{runState}\ (g\ (\mathit{fst}\ r))\ (\mathit{snd}\ r)) \\
&\qquad\qquad \textbf{where}\ \mathrm{r} = \mathit{runState}\ (f\ (\mathit{fst}\ (\mathit{runState}\ t\ s)))\ (\mathit{snd}\ (\mathit{runState}\ t\ s))
\end{aligned}
$$

Veamos que sucede con la otra parte:

$$
\begin{aligned}
&\quad t \gg\!= (\lambda x \to f\ x \gg\!= \mathrm{g}) \\
&= \{ \text{ def. } \gg\!= \} \\
&\quad \mathit{State}\ (\lambda s \to \textbf{let}\ (v,\ s') = \mathit{runState}\ t\ s \\
&\qquad\qquad \textbf{in}\ \mathit{runState}\ ((\lambda x \to f\ x \gg\!= \mathrm{g})\ \mathrm{v})\ \mathrm{s'}) \\
&= \{ \text{ aplicación } \} \\
&\quad \mathit{State}\ (\lambda s \to \textbf{let}\ (v,\ s') = \mathit{runState}\ t\ s \\
&\qquad\qquad \textbf{in}\ \mathit{runState}\ (f\ v \gg\!= \mathrm{g})\ \mathrm{s'}) \\
&= \{ \text{ def. } \gg\!= \} \\
&\quad \mathit{State}\ (\lambda s \to \textbf{let}\ (v,\ s') = \mathit{runState}\ t\ s \\
&\qquad\qquad \textbf{in}\ \mathit{runState}\ (\mathit{State}\ (\lambda s \to \textbf{let}\ (x,\ y) = \mathit{runState}\ (f\ v)\ s \\
&\qquad\qquad\qquad\qquad\qquad\qquad \textbf{in}\ \mathit{runState}\ (g\ x)\ \mathrm{y}))\ \mathrm{s'}) \\
&= \{ \text{ def. runState } \} \\
&\quad \mathit{State}\ (\lambda s \to \textbf{let}\ (v,\ s') = \mathit{runState}\ t\ s \\
&\qquad\qquad \textbf{in}\ (\lambda s \to \textbf{let}\ (x,\ y) = \mathit{runState}\ (f\ v)\ s \\
&\qquad\qquad\qquad\qquad \textbf{in}\ \mathit{runState}\ (g\ x)\ \mathrm{y})\ \mathrm{s'}) \\
&= \{ \text{ aplicación } \} \\
&\quad \mathit{State}\ (\lambda s \to \textbf{let}\ (v,\ s') = \mathit{runState}\ t\ s \\
&\qquad\qquad \textbf{in}\ (\textbf{let}\ (x,\ y) = \mathit{runState}\ (f\ v)\ s' \\
&\qquad\qquad\qquad\qquad \textbf{in}\ \mathit{runState}\ (g\ x)\ \mathrm{y})) \\
&= \{ \mathrm{x} = \mathrm{fst}\ (\mathrm{x},\ \mathrm{y}),\ \mathrm{y} = \mathrm{snd}\ (\mathrm{x},\ \mathrm{y}),\ \text{sustición } \} \\
&\quad \mathit{State}\ (\lambda s \to \textbf{let}\ (v,\ s') = \mathit{runState}\ t\ s \\
&\qquad\qquad \textbf{in}\ (\textbf{let}\ (x,\ y) = \mathit{runState}\ (f\ v)\ s' \\
&\qquad\qquad\qquad\qquad \textbf{in}\ \mathit{runState}\ (g\ (\mathit{fst}\ (x,\ y)))\ (\mathit{snd}\ (x,\ y)))) \\
&= \{ \text{ propiedad de let } \} \\
&\quad \mathit{State}\ (\lambda s \to \textbf{let}\ (v,\ s') = \mathit{runState}\ t\ s \\
&\qquad\qquad \textbf{in}\ \mathit{runState}\ (g\ (\mathit{fst}\ (\mathit{runState}\ (f\ v)\ s')))\ (\mathit{snd}\ (\mathit{runState}\ (f\ v)\ s'))) \\
&= \{ \mathrm{v} = \mathrm{fst}\ (\mathrm{v},\ \mathrm{s'}),\ \mathrm{s'} = \mathrm{snd}\ (\mathrm{v},\ \mathrm{s'}),\ \text{sustitución } \} \\
&\quad \mathit{State}\ (\lambda s \to \textbf{let}\ (v,\ s') = \mathit{runState}\ t\ s \\
&\qquad\qquad \textbf{in}\ \mathit{runState}\ (g\ (\mathit{fst}\ (\mathit{runState}\ (f\ (\mathit{fst}\ (v,\ s')))\ (\mathit{snd}\ (v,\ s')))))\ (\mathit{snd}\ (\mathit{runState}\ (f\ (\mathit{fst}\ (v,\ s')))\ (\mathit{snd}\ (v,\ s'))))) \\
&= \{ \text{ propiedad de let } \} \\
&\quad \mathit{State}\ (\lambda s \to \mathit{runState}\ (g\ (\mathit{fst}\ (\mathit{runState}\ (f\ (\mathit{fst}\ (\mathit{runState}\ t\ s)))\ (\mathit{snd}\ (\mathit{runState}\ t\ s)))))\ (\mathit{snd}\ (\mathit{runState}\ (f\ (\mathit{fst}\ (\mathit{runState}\ t\ s)))\ (\mathit{snd}\ (\mathit{runState}\ t\ s))))) \\
&= \{ \text{ uso de where } \} \\
&\quad \mathit{State}\ (\lambda s \to \mathit{runState}\ (g\ (\mathit{fst}\ r))\ (\mathit{snd}\ r)) \\
&\qquad\qquad \textbf{where}\ \mathrm{r} = \mathit{runState}\ (f\ (\mathit{fst}\ (\mathit{runState}\ t\ s)))\ (\mathit{snd}\ (\mathit{runState}\ t\ s))
\end{aligned}
$$

Como llegamos a lo mismo en ambos desarrollos, demostramos que la ecuación (monad.3) se cumple.
Efectivamente, de la prueba de las tres ecuaciones, *State* es una mónada.

Aclaración: usé where para simplificar la lectura.

Ejercicio 1.b

Implementado en el código.

Ejercicio 2.a

```
instance Monad StateError where
    return x = StateError (λs → Just (x s))
    m >>= f = StateError (λs → case runStateError m s of
                    Nothing → Nothing
                    Just (v, s') → runStateError (f v) s')
```

Ejercicio 2.b

```
instance MonadError StateError where
    throw = StateError (λs → Nothing)
```

Ejercicio 2.c

```
instance MonadState StateError where
    lookfor v = StateError (λs → Just (lookfor' v s, s))
            where lookfor' v ((u, j):ss) | v == u = j
                                         | v /= u = lookfor' v ss
    update v i = StateError (λs → Just ((), update' v i s))
            where update' v i [] = [(v, i)]
                  update' v i ((u, _):ss) | v == u = (v, i):ss
                    update' v i ((u, j):ss) | v /= u = (u, j):(update' v i ss)
```

Ejercicio 2.d

Implementado en el código.

Ejercicio 3.a

```
newtype StateErrorTick a = StateErrorTick {runStateErrorTick :: Env → (Maybe (a,Int),Env)}
```

Ejercicio 3.b

```
class Monad m ⇒ MonadTick m where
    tick :: m ()
```

Ejercicio 3.c

```
instance MonadTick StateErrorTick where
    tick = StateErrorTick (λs → (Just ((),1),s))
```

Ejercicio 3.d

```
instance MonadError StateErrorTick where
    throw = StateErrorTick (λs → (Nothing, s))
```

Ejercicio 3.e

```
instance MonadState StateErrorTick where
    lookfor v = StateErrorTick (λs → (Just (lookfor' v s,0), s))
            where lookfor' v ((u, j):ss) | v == u = j
                                         | v /= u = lookfor' v ss
    update v i = StateErrorTick (λs → (Just ((),0), update' v i s))
            where update' v i [] = [(v, i)]
                  update' v i ((u, _ ):ss) | v == u = (v, i):ss
                    update' v i ((u, j):ss) | v /= u = (u, j):(update' v i ss)
```

Ejercicio 3.f

Implementado en el código.