Trabajo Práctico 3
Análisis de Lenguajes de Programación

Agustín Fernández Bergé y Ramiro Gatto
Legajos: F-3726/5 y G-5803/3

03/12/2024

Ejercicio 1.a

Dada la siguienete monada State:
$instance\ Monad\ State\ where$
$\quad return\ x = State(\lambda s \to (x :!:\ s))$
$\quad m >>= f = State(\lambda s \to let\ (v :!:\ s') = runState\ m\ s$
$\qquad\qquad\qquad\qquad\qquad in\ runState(f\ v)\ s')$
Hay qur probamos que en efecto es una monada, para esto debemos probar las tres leyes de monada.

Monad.1

Debemos probamos que: $return\ s >>= k = k\ a$
Demostración:
$return\ a >>= k$
$=< def.\ return >$
$State(\lambda s \to (a :!:\ s)) >>= k$
$=< def.\ >>=>$
$State(\lambda s \to let\ (v :!:\ s') = runState(State(\lambda s'' \to (a :!:\ s'')))\ s$
$\qquad\qquad in\ runState\ (k\ v)\ s')$
$=< def.\ runState.State = id >$
$State(\lambda s \to let\ (v :!:\ s') = (\lambda s'' \to (a :!:\ s''))s$
$\qquad\qquad in\ runState(k\ v)\ s')$
$=< def.\ App. >$
$State(\lambda s \to let\ (v :!:\ s') = (a :! : s)$
$\qquad\qquad in\ runState(k\ v)\ s')$
$=< a = v$ y $a = s' >$
$State(\lambda s \to let\ (v :!:\ s') = (a :! : s)$
$\qquad\qquad in\ runState(k\ a)\ s)$
$=< def.let >$
$State(\lambda s \to runState(k\ a)\ s)$
$=< def.\ App. >$
$State(runState(k\ a))$
$=< def.\ runState.State = id >$
$k\ a$

Monad.2

Debemos probamos que: $m >>= return = m$
Demostración:
$m >>= return$
$=< def.\ >>=>$
$State(\lambda s \to let\ (v :!:\ s') = runState\ m\ s$
$\qquad\qquad in\ runState(return\ v)\ s')$
$=< def.\ return >$
$State(\lambda s \to let\ (v :!:\ s') = runState\ m\ s$
$\qquad\qquad in\ runState(State(\lambda s'' \to (v :!:\ s'')))\ s')$
$=< def.\ runState.State = id >$
$State(\lambda s \to let\ (v :!:\ s') = runState\ m\ s$
$\qquad\qquad in\ (\lambda s'' \to (v :!:\ s''))\ s')$
$=< def.\ App. >$
$State(\lambda s \to let(v :!:\ s') = runState\ m\ s$
$\qquad\qquad in\ (v :!:\ s'))$
$=< Prop.\ let >$
$State(\lambda s \to runState\ m\ s)$
$=< def.\ App. >$
$State(runState\ m)$
$=< def.\ runState.State = id >$
m

Monad.3

Debemos probar que: $(m >>= k) >>= h = m >>= (\lambda x \to kx >>= h)$
Demostración:

I) Primera mitad:
$(m >>= k) >>= h$
$=< def. >>=>$
$State(\lambda s1 \to let\ (v1 :!: s1') = runState\ m\ s1$
$\quad in\ runState(k\ v1)\ s1') >>= h$
$=< def. >>=>$
$State(\lambda s \to let\ (v :!: s') = runState(State(\lambda s1 \to let\ (v1 :!: s1') = runState\ m\ s1$
$\quad in\ runState(k\ v1)\ s1'))\ s$
$\quad in\ runState(h\ v)\ s')$
$=< def.\ runState.State = id >$
$State(\lambda s \to let\ (v :!: s') = (\lambda s1 \to let\ (v1 :!: s1') = runState\ m\ s1$
$\quad in\ runState(k\ v1)\ s1')\ s$
$\quad in\ runState(h\ v)\ s')$
$=< def.\ App. >$
$State(\lambda s \to let\ (v :! : s') = (let(v1 :! : s1') = runState\ m\ s$
$\quad in\ runState(k\ v1)\ s1')$
$\quad in\ runState(h\ v)\ s')$
$=< Prop.\ Let >$
$State(\lambda s \to let\ (v1 :!: s1') = runState\ m\ s$
$\quad (v :! : s') = runState(k\ v1)\ s1'$
$\quad in\ runState(h\ v)\ s')$

II) Segunda mitad:
$m >>= (\lambda x \to k\ x >>= h)$
$=< def. >>=>$
$State(\lambda s \to let\ (v1 :!: s1') = runState\ m\ s$
$\quad in\ runState((\lambda x \to kx >>= h)v)s')$
$=< def.\ App >$
$State(\lambda s \to let(v1 :!: s1') = runState\ m\ s$
$\quad in\ runState(k\ v >>= h)\ s')$
$=< def. >>=>$

$State(\lambda s \rightarrow let(v1 :!: s1') = runState\ m\ s$
$\quad in\ runState(State(\lambda s'' \rightarrow let\ (v :!: s') = runState\ (k\ v)\ s''$
$\quad\quad in\ runState(h\ v)\ s')$
$\quad)\ s1')$
$=< def.\ runState.State = id >$
$State(\lambda s \rightarrow let\ (v1 :!: s1') = runState\ m\ s$
$\quad in(\lambda s'' \rightarrow let\ (v :!: s') = runState\ (k\ v)\ s''$
$\quad\quad in\ runState\ (h\ v)\ s')$
$\quad s1')$
$=< def.\ App. >$
$State(\lambda s \rightarrow let\ (v1 :!: s1') = runState\ m\ s$
$\quad in\ let\ (v :! : s') = runState\ (k\ v)\ s1'$
$\quad\quad in\ runState\ (h\ v)\ s')$

$=< def.\ Let >$
$State(\lambda s \rightarrow let\ (v1 :!: s1') = runState\ m\ s$
$\quad (v :!: s') = runState\ (k\ v)\ s1$
$\quad in\ runState\ (h\ v)$ s')

De I y II se tiene que vale: $(m >>= k) >>= h = m >>= (\lambda x \rightarrow kx >>= h)$

Conclusion

Como se cumple, monad.1, monad.2 y monad.3, tenemos que en efecto es una monada