Trabajo Práctico 4
Análisis de Lenguajes de Programación

Alumnas:

Cipullo, Inés

Sullivan, Katherine

Universidad Nacional de Rosario

2021

Semántica de While

Enunciamos la semántica del comando `while`:

$$\frac{}{\langle \textbf{while } b \ c, \sigma\rangle \rightsquigarrow \langle \textbf{if } b \textbf{ then } c; \textbf{while } b \ c \textbf{ else skip}, \sigma\rangle} \text{ WHILE}$$

Ejercicio 1.a

Debemos probar que `State` es una mónada. Para eso, verificaremos que se cumplan las tres propiedades monádicas:

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

monad.1

Debemos probar que

$$return\ x \gg= f = f\ x$$

Veamos que

$return\ x \gg= f$
$= \{def\ \ \gg=\}$
$State\ (\lambda s \to let\ (v :!: s') = runState\ (return\ x)\ s\ in\ runState\ (f\ v)\ s')$
$= \{def\ return\}$
$State\ (\lambda s \to let\ (v :!: s') = runState\ (State\ (\lambda s_1 \to let\ (x :!: s_1)))\ s\ in\ runState\ (f\ v)\ s')$
$= \{def\ runState\}$
$State\ (\lambda s \to let\ (v :!: s') = (x :!: s)\ in\ runState\ (f\ v)\ s')$
$= \{def\ let\}$
$State\ (\lambda s \to runState\ (f\ x)\ s))$
$= \{def\ runState\ y\ def\ (f\ x)\ (*)\}$
$State\ (\lambda s \to g(s))$
$= \{def\ (f\ x)\}$
$f\ x$

$(*)$ Como $f\ x$ es de tipo $State\ b$, lo podemos definir como $f\ x = State\ (\lambda t \to g(t))$, donde g es una función genérica de tipo $a \to b$, donde a es el tipo de x.

Por lo que vale (monad.1).

monad.2

Debemos probar que

$$t \gg= return = t$$

Veamos que

$$\begin{aligned}
&t \gg= return\\
&= \{def \ \gg=\}\\
&State\ (\lambda s \rightarrow let\ (v :!: s') = runState\ t\ s\ in\ runState\ (return\ v)\ s')\\
&= \{def\ return\}\\
&State\ (\lambda s \rightarrow let\ (v :!: s') = runState\ t\ s\ in\ runState\ (State\ (\lambda s_1 \rightarrow (v :!: s_1)))\ s')\\
&= \{def\ runState\}\\
&State\ (\lambda s \rightarrow let\ (v :!: s') = runState\ t\ s\ in\ (v :!: s'))\\
&= \{def\ let\}\\
&State\ (\lambda s \rightarrow runState\ t\ s)\\
&= \{def\ t\ (**)\}\\
&State\ (\lambda s \rightarrow runState\ (State\ (\lambda y \rightarrow h(y)))\ s)\\
&= \{def\ runState\}\\
&State\ (\lambda s \rightarrow h(s))\\
&= \{def\ t\}\\
&t
\end{aligned}$$

$(**)$ Como t es de tipo $State\ a$, lo podemos definir como $t\ = State\ (\lambda y \rightarrow h(y))$

Por lo que vale (monad.2).

monad.3

Debemos probar que

$$(t \gg\!= f) \gg\!= g = t \gg\!= (\lambda x \to f\ x \gg\!= g))$$

Veamos que

$$
\begin{aligned}
& t \gg\!= (\lambda x \to f\ x \gg\!= g)) \\
& = \{def\ \gg\!=\} \\
& State\ (\lambda s \to let\ (v :!: s') = runState\ t\ s \\
& \qquad\qquad in\ runState\ ((\lambda x \to State\ (\lambda p \to let\ (r :!: p') = runState\ (f\ x)\ p \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad in\ runState\ (g\ r)\ p'))\ v)\ s') \\
& = \{def\ aplicacion\ (\lambda x)\} \\
& State\ (\lambda s \to let\ (v :!: s') = runState\ t\ s \\
& \qquad\qquad in\ runState\ (State\ (\lambda p \to let\ (r :!: p') = runState\ (f\ v)\ p \\
& \qquad\qquad\qquad\qquad\qquad\qquad in\ runState\ (g\ r)\ p'))\ s') \\
& = \{def\ runState\} \\
& State\ (\lambda s \to let\ (v :!: s') = runState\ t\ s \\
& \qquad\qquad in\ (\lambda p \to let\ (r :!: p') = runState\ (f\ v)\ p\ in\ runState\ (g\ r)\ p')\ s') \\
& = \{def\ aplicacion\ (\lambda p)\} \\
& State\ (\lambda s \to let\ (v :!: s') = runState\ t\ s \\
& \qquad\qquad in\ let\ (r :!: p') = runState\ (f\ v)\ s'\ in\ runState\ (g\ r)\ p') \\
& = \{def\ let\} \\
& State\ (\lambda s \to let\ (v :!: s') = runState\ t\ s \\
& \qquad\qquad\quad (r :!: p') = runState\ (f\ v)\ s' \\
& \qquad\qquad in\ runState\ (g\ r)\ p') \\
& = \{def\ let\} \\
& State\ (\lambda s \to let\ (r :!: p') = runState\ (State\ (\lambda s'' \to let\ (v :!: s') = runState\ t\ s'' \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad in\ runState\ (f\ v)\ s'))\ s \\
& \qquad\qquad in\ runState\ (g\ r)\ p') \\
& = \{def\ \gg\!=\} \\
& State\ (\lambda s \to let\ (r :!: p') = runState\ (t \gg\!= f)\ s\ in\ runState\ (g\ r)\ p') \\
& = \{def\ \gg\!=\} \\
& (t \gg\!= f) \gg\!= g
\end{aligned}
$$

Por lo que vale (monad.3).