ttp/papers/working/squigol-exercises.tex

Tom Pressburger
March 27, 1989

1 Exercises

Exercise 1. Define $P \triangleleft S$ (read "S filtered by P"), for a predicate P, to be the subsequence of elements of S that satisfy P; i.e. $P\triangleleft$ filters out from a given sequence S those elements that do not satisfy P. For example, $(<3) \triangleleft [1,2,3,2,1,3] = [1,2,2,1]$. So $P\colon \alpha \rightarrow \mathbf{B} \Rightarrow P \triangleleft \_ \colon [\alpha] \rightarrow [\alpha]$.

1. Convince yourself that $P\triangleleft$ is a $+\!\!+$-homomorphism. Find $\oplus$ and f so that $P \triangleleft s = \oplus / f \star s$.
2. For arbitrary P, Q, f, figure out an equivalent expression for each of the following. Make sure your answer type checks. Derive your answer using the laws and your answer to the preceding exercise.
   (a) $P \triangleleft P \triangleleft s$
   (b) $P \triangleleft Q \triangleleft s$
   (c) $P \triangleleft f \star s$
3. Define

$$\begin{array}{rcl} \mathbf{N} & \hat{=} & [0,1,2,\ldots] \\ \mathbf{E}\,x & \hat{=} & 2 \mid x \\ \mathbf{P}\,x & \hat{=} & \exists\,(i:\mathbf{N})\; i*i = x \\ \mathbf{sqr}\,x & \hat{=} & x*x \\ \mathbf{dbl}\,x & \hat{=} & 2*x \end{array}$$

   The expression $m \mid n$ means that m divides evenly into n; i.e. $n \equiv 0 \bmod m$.

   Find an equivalent for each expression below.
   (a) $\mathbf{E} \triangleleft \mathbf{N}$
   (b) $\mathbf{P} \triangleleft \mathbf{N}$
   (c) $\mathbf{E} \circ \mathbf{sqr}$
4. Using the results of previous exercises, derive an equivalent for $\mathbf{P} \triangleleft \mathbf{E} \triangleleft \mathbf{N}$.

Exercise 2. Let $\mathbf{MSSQ}\; s = \uparrow / \sum \star \mathbf{seqs}\; s$. Derive the "obvious" algorithm.

2 Solutions

Exercise 1.

1. Yes, $P\triangleleft$ is a $+\!\!+$-homorphism because

$$P \triangleleft (s_1 +\!\!+ s_2) = (P \triangleleft s_1) +\!\!+ (P \triangleleft s_2). \tag{1}$$

   The proof of such a thing depends on how sequences are defined formally (as a mapping from $[1 \mathinner{..} \#s]$ to values, or as the datatype generated by the free constructor CONS), and how $\triangleleft$ was defined.

   By our homorphism lemma, because $P\triangleleft$ is a homomorphism, it can be written as $(\oplus/) \circ (f\star)$, where $\oplus$ is the right-hand side joining operation. Equation 1 above shows that $\oplus = +\!\!+$. From a previous result, $f = (P\triangleleft) \circ [\,]$. Using our intuition about $\triangleleft$ results in

$$\begin{array}{lcll} f & = & (\lambda x.\textbf{if } P\ x \textbf{ then } [x] \textbf{ else } [\,]) & \text{or, expressed functionally,} \\ f & = & (P \longrightarrow [\,]; K_{[\,]}) & \end{array}$$

2. (a) $P \triangleleft P \triangleleft s = P \triangleleft s$. Although this is obvious, we can try to prove it using the definition of $\triangleleft$ above.

$$\begin{array}{lcll}
(P\triangleleft) \circ (P\triangleleft) & = & (+\!\!+/) \circ f\star \circ (+\!\!+/) \circ f\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (f\star)\star \circ f\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (f\star \circ f)\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ ((f\star) \circ (P \longrightarrow [\,]; K_{[\,]}))\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (P \longrightarrow (f\star) \circ [\,]; (f\star) \circ K_{[\,]})\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (P \longrightarrow [\,] \circ f; K_{[\,]})\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (P \longrightarrow [\,] \circ (P \longrightarrow [\,]; K_{[\,]}); K_{[\,]})\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (P \longrightarrow (P \longrightarrow [\,] \circ [\,]; [\,] \circ K_{[\,]}); K_{[\,]})\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (P \longrightarrow [\,] \circ [\,]; K_{[\,]})\star & \\
& = & (+\!\!+/) \circ (+\!\!+/)\star \circ (P \longrightarrow [\,] \circ [\,]; K_{[\,]})\star & +\!\!+/ \text{ is a morphism over } +\!\!+ \\
& = & (+\!\!+/) \circ ((+\!\!+/) \circ (P \longrightarrow [\,] \circ [\,]; K_{[\,]}))\star & \\
& = & (+\!\!+/) \circ ((P \longrightarrow (+\!\!+/) \circ [\,] \circ [\,]; (+\!\!+/) \circ K_{[\,]}))\star & \\
& = & (+\!\!+/) \circ ((P \longrightarrow [\,]; K_{[\,]}))\star & \\
& = & P\triangleleft &
\end{array}$$

   (b) For predicates P and Q, defined on all of the domain of elements of s, we have

$$P \triangleleft Q \triangleleft s = Q \triangleleft P \triangleleft s = (\lambda x.P\ x \wedge Q\ x) \triangleleft s = (P \wedge Q) \triangleleft s.$$

   However, if the domain of P is smaller than that of the domain of elements of s, then $P \triangleleft s$ may be meaningless, or a type error.
   Proof:

$$\begin{array}{lcll}
(P\triangleleft) \circ (Q\triangleleft) & = & (+\!\!+/) \circ f_P\star \circ (+\!\!+/) \circ f_Q\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (f_P\star)\star \circ f_Q\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (f_P\star \circ f_Q)\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ ((f_P\star) \circ (Q \longrightarrow [\,]; K_{[\,]}))\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (Q \longrightarrow (f_P\star) \circ [\,]; (f_P\star) \circ K_{[\,]})\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (Q \longrightarrow [\,] \circ f_P; K_{[\,]})\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (Q \longrightarrow [\,] \circ (P \longrightarrow [\,]; K_{[\,]}); K_{[\,]})\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (Q \longrightarrow (P \longrightarrow [\,] \circ [\,]; [\,] \circ K_{[\,]}); K_{[\,]})\star & \\
& = & (+\!\!+/) \circ (+\!\!+/) \circ (Q \wedge P \longrightarrow [\,] \circ [\,]; K_{[\,]})\star & \text{and as before} \\
& = & (Q \wedge P) \triangleleft s &
\end{array}$$

(c) $P \triangleleft f \star s = f \star (P \circ f) \triangleleft s$

This transformation transposes the order of $\star$ and $\triangleleft$. This may be a good idea if f is expensive and/or P is "sparse" (is rarely true), and $P \circ f$ simplifies. See below.

Proof:

$$\begin{aligned}
(P\triangleleft) \circ (f\star) &= (+\!\!+/) \circ (P \longrightarrow []; K_{[\,]})\star \circ (f\star) \\
&= (+\!\!+/) \circ ((P \longrightarrow []; K_{[\,]}) \circ f)\star \\
&= (+\!\!+/) \circ ((P \circ f) \longrightarrow [] \circ f; K_{[\,]} \circ f)\star \\
&= (+\!\!+/) \circ ((P \circ f) \longrightarrow f\star \circ []; f\star \circ K_{[\,]})\star \\
&= (+\!\!+/) \circ (f\star \circ ((P \circ f) \longrightarrow []; K_{[\,]}))\star \\
&= (+\!\!+/) \circ (f\star)\star \circ ((P \circ f) \longrightarrow []; K_{[\,]})\star \\
&= f\star \circ (+\!\!+/) \circ ((P \circ f) \longrightarrow []; K_{[\,]})\star \\
&= f\star \circ (P \circ f)\triangleleft
\end{aligned}$$

3. Rewrite

(a) $\mathbf{E} \triangleleft \mathbf{N} = \mathbf{dbl} \star \mathbf{N}$

(b) $\mathbf{P} \triangleleft \mathbf{N} = \mathbf{sqr} \star \mathbf{N}$

(c) $\mathbf{E} \circ \mathbf{sqr} = \mathbf{E}$. This one requires the elementary number theory fact that if p is a prime, then $p \mid a * b \equiv (p \mid a) \vee (p \mid b)$. Then

$$(\mathbf{E} \circ \mathbf{sqr})\, x \equiv 2 \mid x * x \equiv (2 \mid x) \vee (2 \mid x) \equiv 2 \mid x \equiv \mathbf{E}\, x.$$

4.

$$\begin{aligned}
\mathbf{P} \triangleleft \mathbf{E} \triangleleft \mathbf{N} &= \mathbf{E} \triangleleft \mathbf{P} \triangleleft \mathbf{N} \\
&= \mathbf{E} \triangleleft \mathbf{sqr} \star \mathbf{N} \\
&= \mathbf{sqr} \star (\mathbf{E} \circ \mathbf{sqr}) \triangleleft \mathbf{N} \\
&= \mathbf{sqr} \star \mathbf{E} \triangleleft \mathbf{N} \\
&= \mathbf{sqr} \star \mathbf{dbl} \star \mathbf{N} \\
&= (\mathbf{sqr} \circ \mathbf{dbl}) \star \mathbf{N}
\end{aligned}$$

Exercise 2. Let $\mathbf{MSSQ}\ s = \uparrow/\sum \star\, \mathbf{seqs}\ s$. Derive the "obvious" algorithm.

The obvious algorithm is to form the sum of the positive numbers. If there are none, then the answer is 0, because the empty sequence is considered a subsequence of any sequence.

We can obtain a closed-form expression for the result by employing the homomorphism form of **seqs**, where $\mathbf{seqs} = (\times_{+\!\!+}/) \circ [K_{[\,]}, []]\star$.

We'll use the following lemmas.

$$\begin{aligned}
(\textstyle\sum\star) \circ (\times_{+\!\!+}/) &= (\times_{+}/) \circ (\textstyle\sum\star)\star \\
\textstyle\sum \circ [] &= \mathbf{id} \\
(\uparrow/) \circ (\times_{+}/) &= (+/) \circ (\uparrow/)\star \\
(\uparrow/) \circ [f, g] &= \uparrow \circ f \times g \\
\uparrow \circ K_c \times \mathbf{id} &= (c\,\uparrow)
\end{aligned}$$

A functional style derivation is as follows.

$$\begin{aligned}
\mathbf{MSSQ} &= (\uparrow/) \circ (\textstyle\sum\star) \circ \mathbf{seqs} \\
&= (\uparrow/) \circ (\textstyle\sum\star) \circ (\times_{+\!\!+}/) \circ [K_{[\,]}, []]\star \\
&= (\uparrow/) \circ (\times_{+}/) \circ (\textstyle\sum\star)\star \circ [K_{[\,]}, []]\star
\end{aligned}$$

$$
\begin{aligned}
&= (\uparrow/) \circ (\times_{+}/) \circ ((\textstyle\sum\star) \circ [K_{[\,]}, []])\star \\
&= (\uparrow/) \circ (\times_{+}/) \circ [\textstyle\sum \circ K_{[\,]}, \sum \circ []]\star \\
&= (\uparrow/) \circ (\times_{+}/) \circ [K_0, \mathbf{id}]\star \\
&= (+/) \circ (\uparrow/)\star \circ [K_0, \mathbf{id}]\star \\
&= (+/) \circ ((\uparrow/) \circ [K_0, \mathbf{id}])\star \\
&= (+/) \circ (\uparrow \circ K_0 \times \mathbf{id})\star \\
&= (+/) \circ (0 \uparrow)\star
\end{aligned}
$$

Less functionally, we have

$$
\begin{aligned}
\mathbf{MSSQ}\, s &= \uparrow/\textstyle\sum \star \mathbf{seqs}\, s \\
&= \uparrow/\textstyle\sum \star \times_{++}/(\lambda x.[[\,],[x]]) \star s \\
&= \uparrow/\times_{+}/(\textstyle\sum\star) \star \lambda x.[[\,],[x]] \star s \\
&= \uparrow/\times_{+}/(\lambda x.\textstyle\sum \star [[\,],[x]]) \star s \\
&= \uparrow/\times_{+}/\lambda x.[\textstyle\sum[\,], \sum[x]] \star s \\
&= \uparrow/\times_{+}/\lambda x.[0,x] \star s \\
&= +/(\uparrow/) \star \lambda x.[0,x] \star s \\
&= +/(\lambda x.0 \uparrow x) \star s \\
&= +/(0 \uparrow) \star s
\end{aligned}
$$

We could derive a version that doesn't create intermediate lists as follows, using a singleton split.

$$
\begin{aligned}
\mathbf{MSSQ}\,[\,] &= +/(0\uparrow) \star [\,] = 0 \\
\mathbf{MSSQ}\,[x] \mathbin{+\!\!+} s &= +/(0\uparrow) \star ([x] \mathbin{+\!\!+} s) \\
&= (+/(0\uparrow) \star [x]) + (+/(0\uparrow) \star s) \\
&= (0 \uparrow x) + \mathbf{MSSQ}\, s \\
&= (\mathbf{if}\ 0 < x\ \mathbf{then}\ x\ \mathbf{else}\ 0) + \mathbf{MSSQ}\, s \\
&= \mathbf{if}\ 0 < x\ \mathbf{then}\ x + \mathbf{MSSQ}\, s\ \mathbf{else}\ \mathbf{MSSQ}\, s
\end{aligned}
$$

Or we could reason about the various cases without ever deriving the closed form expression.

$$
\begin{aligned}
\mathbf{MSSQ}\,[\,] &= \uparrow/\textstyle\sum \star \mathbf{seqs}\,[\,] = \uparrow/\sum \star [[\,]] = \uparrow/[\sum\,[\,]] = \uparrow/[0] = 0 \\
\mathbf{MSSQ}\,[x] &= \uparrow/\textstyle\sum \star \mathbf{seqs}\,[x] = \uparrow/\sum \star [[\,],[x]] = \uparrow/[\sum\,[\,], \sum\,[x]] = \uparrow/[0,x] = 0 \uparrow x \\
\mathbf{MSSQ}\, s_1 \mathbin{+\!\!+} s_2 &= \uparrow/\textstyle\sum \star \mathbf{seqs}\,(s_1 \mathbin{+\!\!+} s_2) \\
&= \uparrow/\textstyle\sum \star ((\mathbf{seqs}\, s_1) \times_{++} (\mathbf{seqs}\, s_2)) \\
&= \uparrow/(\textstyle\sum \star \mathbf{seqs}\, s_1) \times_{+} (\sum \star \mathbf{seqs}\, s_2) \\
&= (\uparrow/\textstyle\sum \star \mathbf{seqs}\, s_1) + (\uparrow/\sum \star \mathbf{seqs}\, s_2) \\
&= \mathbf{MSSQ}\, s_1 + \mathbf{MSSQ}\, s_2
\end{aligned}
$$

Of course, from the above equations, we see that **MSSQ** is a homomorphism, with $\oplus = +$ and $f = (0 \uparrow)$, hence **MSSQ** $= (+/) \circ (0 \uparrow)\star$.

The advantage of the functional derivation is that it produces a single expression, so that further reasoning does not necessarily require case splits. A disadvantage is that the notation and rules are unfamiliar, and data is not named.