Short Note: Verification of In-Place List-Reversal with Effect Sets

Martin Berger∗ Kohei Honda† Nobuko Yoshida*

This short note presents a modular correctness proof of a simple in-place list reversal algorithm that is a stable of the relevant literature, cf. (Reynolds, 2002). To keep this note short, we assume familiarity with (Berger *et al.*, 2006). The algorithm in question is:

$$\begin{array}{lcl} M & \stackrel{\text{def}}{=} & r := \mathsf{nil}; (\mu f.\lambda().(\mathtt{case}\ !p\ \mathtt{of}\ \{\mathsf{nil} \triangleright () \mid a :: l \triangleright N; f()\}))() \\ N & \stackrel{\text{def}}{=} & \mathtt{let}\ tmp = !l\ \mathtt{in}\ N' \\ N' & \stackrel{\text{def}}{=} & l := !r; r := !p; p := tmp \end{array}$$

This algorithm satisfies the following natural specification.

$$\{\underbrace{\mathsf{List}(p,n) \wedge \mathsf{path}(p,i,q) \wedge \mathsf{Dist}}_{B(pniq)}\}\ M\ \{\mathsf{path}(!r,i,q)\} \mathbin{@} S(n) \tag{1}$$

Intuitively, $\mathsf{List}(p,n)$ says that p stores a non-circular list of length n, and $\mathsf{path}(p,i,q)$ expresses the existence of an i-long path starting at p that ends at q. Dist gives all the relevant inequalities that need to hold for the algorithm to work and for the proof to go through, like $r \neq p$ and so on. Finally, $S(n)$ lists all relevant effects: r and each $\pi_2(l)$ such that l is reachable from p. These predicates are straightforward and defined below. We note that (1) does not give the strongest possible specification, for example, as pointed out in (Bornat, 2006), the algorithm also works with circular lists. It would be easy to adapt our approach to reversing circular lists. We continue with the definition of a tailor-made reachability predicate $x \hookrightarrow y$.

$$x \hookrightarrow y \quad \stackrel{\text{def}}{=} \quad x = y \vee (!x \neq \mathsf{nil} \wedge \pi_2(!x) \hookrightarrow y)$$

This predicate ignores the "left" components of the pairs making up lists, as they will not be modified by the list reversal algorithm. The next predicate gives the distinction that underlies our correctness proof.

$$\mathsf{Dist} \quad \stackrel{\text{def}}{=} \quad (p \hookrightarrow x \wedge r \hookrightarrow y) \supset x \neq y.$$

∗ Deptartment of Computing, Imperial College London.
† Department of Computer Science, Queen Mary, University of London.

$$\frac{\{C\}\ M\ \{C'\} @ S}{\{C \wedge [!S]C_0\}\ M\ \{C' \wedge C_0\} @ S}\ [\mathit{Invariance}] \quad \frac{\{C\}\ M\ \{C'\} @ S}{\{C\}\ M\ \{C'\} @ S \cup S'}\ [\mathit{Weak}]$$

$$\frac{\{C\}\ M\ \{C_0\} @ S \quad \{C_0\}\ N\ \{C'\} @ S' \quad S' \text{-}^{!S}}{\{C\}\ M;N\ \{C'\} @ S \cup S'}\ [\mathit{Seq}]$$

$$\frac{\{C_1\}\ M\ \{C_1' \wedge C\} @ S_M \quad \{C \wedge C_2\}\ N\ \{C_2'\} @ S_N \quad S_N \text{-}^{!S_M}}{\{C_1 \wedge [!S_M]C_2\}\ M;N\ \{C_1' \wedge C_2'\} @ S_M \cup S_N}\ [\mathit{Seq\text{-}I}]$$

$$\frac{\begin{array}{l}\{C\}\ M :_m \{C_0\} @ S_M \\ \{C_0[\mathsf{nil}/m]\}\ N_1 :_u \{C'\} @ S_N \\ \{\exists al.(m = \langle a,l\rangle \wedge C_0)\}\ N_2 :_u \{C'\} @ S_N \quad S_N \text{-}^{!S_M}\end{array}}{\{C\}\ \mathtt{case}\ M\ \mathtt{of}\ \{\mathsf{nil} \triangleright N_1 \mid a::l \triangleright N_2\} :_u \{C'\} @ S_M \cup S_N}\ [\mathit{ListCase}]$$

Fig. 1. Auxiliary rules of inference used in the derivation, all of which are easy to justify.

Now we can give the remaining predicates.

$$\begin{aligned}\mathsf{path}(p,0,q) &\stackrel{\text{def}}{=} !p = \mathsf{nil} \\ \mathsf{path}(p,n+1,q) &\stackrel{\text{def}}{=} \exists ar.(!p = a::r \wedge \mathsf{path}(r,n,q)) \\ \mathsf{List}(p,n) &\stackrel{\text{def}}{=} \exists q.\mathsf{path}(p,n,q) \wedge \neg\exists q.\mathsf{path}(p,n+1,q). \\ S(n) &\stackrel{\text{def}}{=} \{a \mid a = r \vee (n > 1 \wedge p \hookrightarrow a)\}\end{aligned}$$

For convenience, we sometimes confuse effect sets with their constituting effect comprehensions, i.e. $S' \equiv [!S]S'$ is short for $C_1 \equiv [!S]C_1$, assuming that $S' \stackrel{\text{def}}{=} \{a|C_1\}$. We write $S^{\text{-}!S'}$ to mean the formula $S' \equiv [!S]S'$. The overall structure of the proof follows the syntax of M. Auxiliary rules streamling our reasoning are to be found in Figure 1.

1	$\{\mathsf{T}\}\ r := \mathtt{nil}\ \{!r = \mathsf{nil}\} @ r$	omitted
2	$\{[!r]B(pniq)\}\ r := \mathtt{nil}\ \{!r = \mathsf{nil} \wedge B(pniq)\} @ r$	Invar., 1
3	$B(pniq) \supset [!r]B(pniq)$	
4	$\{B(pniq)\}\ r := \mathtt{nil}\ \{!r = \mathsf{nil} \wedge B(pniq)\} @ r$	Cons, 2, 3
5	$\{!r = \mathsf{nil} \wedge B(pniq)\}\ (\mu f.\lambda().M')()\ \{\mathsf{path}(p,i,q)\} @ S(n)$	see below
6	$S(n) \equiv [!r]S(n)$	
7	$\{B(pniq)\}\ M\ \{\mathsf{path}(p,i,q)\} @ S(n) \cup \{r\}$	Seq, 4, 5, 6
8	$S(n) \cup \{r\} = S(n)$	
9	$\{B(pniq)\}\ M\ \{\mathsf{path}(p,i,q)\} @ S(n)$	Cons, 7, 8

We turn to Line 5, but omit the straightforward derivation for $(\mu f.\lambda().M')()$, focussing instead on the body of the recursion to establish:

$$\{P \wedge \forall k < n.E(k)\}\ \mathtt{case}\ !p\ \mathtt{of}\ \{\mathsf{nil} \triangleright () \mid a::l \triangleright N; f()\}\ \{Q\} @ S(n) \qquad (2)$$

and for this purpose, we define some more auxiliary predicates.

$$P \stackrel{\text{def}}{=} \mathsf{List}(p,n) \wedge \mathsf{List}(!r,n_r) \wedge \mathsf{Dist} \wedge \mathsf{path}(p,i,s) \wedge \mathsf{path}(!r,j,t)$$
$$Q \stackrel{\text{def}}{=} (i > 0 \supset \mathsf{path}(!r,i,s)) \wedge \mathsf{path}(!r,j+n,t)$$

Now we can specify the key invariant.

$$E(n) \quad \stackrel{\text{def}}{=} \quad \forall n_r ijst.\{P\} f \bullet ()\{Q\} \mathbin{@} S(n)$$

We shall now turn to deriving the body of the `case`-distinction using [*ListCase*]. We can clearly derive

$$\{P \wedge \forall k < n.E(k)\}\ !p :_m \{P \wedge \forall k < n.E(k) \wedge !p = m\} \mathbin{@} \emptyset \tag{3}$$

and also

$$\frac{1\ \ \{!p = \mathsf{nil} \wedge P \wedge \forall k < n.E(k)\}\ ()\ \{Q\} \mathbin{@} \emptyset}{2\ \ \{!p = \mathsf{nil} \wedge P \wedge \forall k < n.E(k)\}\ ()\ \{Q\} \mathbin{@} S(n) \quad \mathsf{Weak},\ 1}$$

This covers the base case. For the inductive step in the `case`-statement we need to establish that (omitting an application of [*Cons*]):

$$\{\exists al.(!p = \langle a,l\rangle \wedge P \wedge \forall k < n.E(k))\}\ N; f()\ \{Q\} \mathbin{@} S(n).$$

We do this in two steps. First we note that N preforms a straightforward rotation of content, provided r,l,p are mutually distinct pointers. We express this last assumptions by Dist. Then we easily derive, details omitted:

$$\{\mathsf{Dist} \wedge !r = z \wedge !p = x \wedge !l = y\}\ N\ \{!p = y \wedge !r = x \wedge !l = z\} \mathbin{@} \{r,l,p\}$$

Defining

- $P' \stackrel{\text{def}}{=} !r = z \wedge !p = x \wedge !l = y \wedge P \wedge \forall k < k.E(k)$ and
- $Q' \stackrel{\text{def}}{=} P \wedge \forall k < k.E(k) \wedge !p = y \wedge !r = x \wedge !l = z$,

we use [*Invariance*] and [*Cons*] to add:

$$\{P'\}\ N\ \{Q'\} \mathbin{@} \{r,l,p\}$$

But clearly

$$P \supset P' \qquad \text{and} \qquad Q' \supset \underbrace{\left(\begin{array}{c} \mathsf{List}(!t,n_r+1) \wedge \mathsf{List}(p,n-1) \wedge \mathsf{path}(!r,j+1,s) \\ \wedge \\ (i > 0 \supset \mathsf{path}(p,i-1,t)) \wedge (i = 0 \supset \mathsf{path}(!r,0,s)) \end{array}\right)}_{Q''}.$$

It is straightforward to see that

$$\{Q''\}\ f()\ \{Q\} \mathbin{@} S(n-1) \tag{4}$$

(this is simply the recursive call). Unfortunatly we cannot directly apply [*Seq*] now, because the set $S(n-1)$ in (4) does not satisfy $S(n-1)^{\text{-}!\{r,l,p\}}$. The culprit is $!p$ which occurs in $S(n-1)$. But, instead of $S(n-1)$ we can use

$$S'(n-1) \stackrel{\text{def}}{=} \{a \mid a = r \vee (n > 1 \wedge (a = p \vee \pi_2(y) \hookrightarrow a))\}$$

Then [*Cons*] allows us to infer

$$\{Q''\}\ f()\ \{Q\} \mathbin{@} S'(n-1)$$

with $S'(n-1)^{\text{-}!\{r,l,p\}}$, hence we can apply [*Seq*] and obtain

$$\{P\}\ f()\ \{Q\} \mathbin{@} \{l,r,p\} \cup S(n-1)$$

But, under P, clearly $S(n) = \{l,r,p\} \cup S(n-1)$, so in fact

$$\{P\}\ f()\ \{Q\} \mathbin{@} S'(n)$$

as required, completing the proof of (2).

References

Berger, Martin, Honda, Kohei, & Yoshida, Nobuko. (2006). *A Logical Analysis of Aliasing in Imperative Higher-Order Functions*. Submitted.

Bornat, Richard. (2006). *List reversal: back into the frying pan*. Available from `http://homepages.phonecoop.coop/randj/richard`.

Reynolds, John C. (2002). Separation logic: a logic for shared mutable data structures. *Proc. LICS'02*.