Remarks on the failed proof of $take\ k\ xs +\!\!+\ drop\ k\ xs = xs$

James Caldwell

November 30, 2007

In Hudak's text *The Haskell School of Expression: Learning Functional Programming through Multimedia* [2] the following property is claimed to hold for all infinite lists.

$$take\,k\,xs +\!\!+\, drop\,k\,xs \;= xs$$

The definitions of append (++)[2, pp.330], *take* [2, pp. 324] and *drop* [2, pp.324] are given as follows:

$$\begin{array}{lcl}
+\!\!+ & :: & [a] \rightarrow [a] \rightarrow [a] \\
\texttt{[]} +\!\!+ ys & = & ys \\
(x : xs) +\!\!+ ys & = & x : (xs +\!\!+ ys) \\
 & & \\
take & :: & Int \rightarrow [a] \rightarrow [a] \\
take\ 0\ \_ & = & \texttt{[]} \\
take\ \_\ \texttt{[]} & = & \texttt{[]} \\
take\,k\,(x : xs)\mid k > 0 & = & x : take\,(k-1)\,xs \\
take\ \_\ \_ & = & error \\
 & & \\
drop & :: & Int \rightarrow [a] \rightarrow [a] \\
drop\ 0\ \_ & = & \texttt{[]} \\
drop\ \_\ \texttt{[]} & = & \texttt{[]} \\
drop\,k\,(\_ : xs)\mid k > 0 & = & drop\,(k-1)\,xs \\
drop\ \_\ \_ & = & error
\end{array}$$

On the errata sheet, the definition of *drop* is modified as follows to make the base case of the proof work when $k = 0$.

$$\begin{array}{lcl} \mathit{drop} & :: & \mathit{Int} \to [a] \to [a] \\ \mathit{drop}\; 0\; \mathit{xs} & = & \mathit{xs} \\ \mathit{drop}\; \_\; \texttt{[]} & = & \texttt{[]} \\ \mathit{drop}\, k\, (\_ : \mathit{xs}) \mid k > 0 & = & \mathit{drop}\, (k-1)\, \mathit{xs} \end{array}$$

1 In the book

The method of proof by induction for infinite lists as presented in chapter 14 says

"... to prove a property P, expressed as an equation in Haskell, is true of all infinite lists, we proceed in two steps.

1. *Prove $P(\perp)$ (base case)*
2. *Assume that $P(xs)$ is true (the induction hypothesis), and prove that $P(x : xs)$ is true (the induction step)."*

Without elaboration with quantifiers, the base case of the proof of the following equality is presented in the book (pp. 204) and is corrected on the errata sheet [1]. The claim is made in the text that the proof of the induction step is straightforward (and thus omitted).

Theorem 1.1. *take k xs* ++ *drop k xs* = *xs*

Proof: By induction on the infinite list xs. There are two cases.
(base case:) Show $\mathit{take}\ k \perp \mathbin{+\!\!+} \mathit{drop}\ k \perp = \perp$ Now, either $k = 0$ or not.
If $k = 0$ then:

$$\begin{array}{l} \mathit{take}\ 0 \perp \mathbin{+\!\!+} \mathit{drop}\ 0 \perp \\ \Longrightarrow \texttt{[]} \mathbin{+\!\!+} \mathit{drop}\ 0 \perp \\ \Longrightarrow \mathit{drop}\ 0 \perp \\ \Longrightarrow \perp \end{array}$$

If $k \neq 0$ then we have the following computation.

$$\begin{array}{l} \mathit{take}\ k \perp \mathbin{+\!\!+} \mathit{drop}\ k \perp \\ \Longrightarrow \perp \mathbin{+\!\!+} \mathit{drop}\ k \perp \\ \Longrightarrow \perp \end{array}$$

(Induction step:) Assume the induction hypothesis:

$$(\mathbf{I.H.})\quad \mathit{take}\ k\ \mathit{xs} \mathbin{+\!\!+} \mathit{drop}\ k\ \mathit{xs} = \mathit{xs}$$

and show

$$take\ k\ (x : xs) \mathbin{+\!\!+} drop\ k\ (x : xs) = (x : xs)$$

Now, since the type of *take* and *drop* are $Int \rightarrow [a] \rightarrow [a]$ we know $k :: Int$ We proceed by cases on k: either $k < 0$ or $k = 0$ or $k > 0$.
case 1.($k = 0$)

$$\begin{aligned} & take\ 0\ (x : xs) \mathbin{+\!\!+} drop\ 0\ (x : xs) \\ & \ \Rightarrow [\,] \mathbin{+\!\!+} drop\ 0\ (x : xs) \\ & \ \Rightarrow drop\ 0\ (x : xs) \\ & \ \Rightarrow (x : xs) \end{aligned}$$

case 2.($k > 0$)

$$\begin{aligned} & take\ k\ (x : xs) \mathbin{+\!\!+} drop\ k\ (x : xs) \\ & \ \Rightarrow x : (take\ (k-1)\ (xs)) \mathbin{+\!\!+} drop\ k\ (x : xs) \\ & \ \Rightarrow x : (take\ (k-1)\ (xs)) \mathbin{+\!\!+} drop\ (k-1)\ xs \\ & \ \Rightarrow x : (take\ (k-1)\ (xs) \mathbin{+\!\!+} drop\ (k-1)\ xs) \end{aligned}$$

At this point in the proof we would like to invoke the induction hypothesis **(I.H.)** bu we can not match k with $k - 1$ and so the substitution fails. We are stuck. We need to state the theorem so that we are claiming this is true for all k.
case 3.($k < 0$)

$$\begin{aligned} & take\ k\ (x : xs) \mathbin{+\!\!+} drop\ k\ (x : xs) \\ & \ \Rightarrow \perp \mathbin{+\!\!+} drop\ k\ (x : xs) \\ & \ \Rightarrow \perp \end{aligned}$$

So this branch of the proof has failed as well. Interestingly, in Hugs, negative values of k return the empty list.

The inductive step of the proof has failed for two reasons. First: the unquantified induction hypothesis can not be instantiated with $(k - 1)$ and; second: when $k < 0$ the proof fails with *take* (and *drop*) as defined in the book – but not as defined in the Haskell-98 prelude.

2 Restatement of the theorem

We restate the theorem by universally quantifying over the free variables in the equation. Instead of restricting the properties provable by induction on infinite lists to simply be an equation in Haskell, we restrict the properties provable in this way to be properties of the form $\forall \bar{x} : \bar{T}.E$ where E is an equation in Haskell.

Then, to prove $\forall xs : [a].P(xs)$, where xs is an infinite list and P is a property of the appropriate form, prove two things.

Base case: $P(\bot)$

Induction step: Assume $P(xs)$ for arbitrary list xs and show $P(x : xs)$.

We redefine *take* and *drop* so that they return the empty list if the argument is negative.

$$\begin{array}{lcl}
\mathit{take} & :: & \mathit{Int} \to [a] \to [a] \\
\mathit{take}\ k\ \_ \mid k \leq 0 & = & \texttt{[]} \\
\mathit{take}\ \_\ \texttt{[]} & = & \texttt{[]} \\
\mathit{take}\, k\, (x : xs) \mid k > 0 & = & x : \mathit{take}\,(k-1)\, xs \\
\\
\mathit{drop} & :: & \mathit{Int} \to [a] \to [a] \\
\mathit{drop}\ k\ xs \mid k \leq 0 & = & xs \\
\mathit{drop}\ \_\ \texttt{[]} & = & \texttt{[]} \\
\mathit{drop}\, k\, (\_ : xs) \mid k > 0 & = & \mathit{drop}\,(k-1)\, xs
\end{array}$$

Note that these are the defintions in the Haskell prelude.

Here is the restated theorem using the universal quantifiers.

Theorem 2.1. $\forall xs : [a].\forall k : \mathit{Int}.\ \ \mathit{take}\ k\ xs \mathbin{+\!\!+} \mathit{drop}\ k\ xs = xs$

Exercise .1. Prove this theorem by induction on infinite lists.

References

[1] `http://www.haskell.org/soe/Bug/erata.htm`.

[2] Paul Hudak. *The Haskell School of Expression: Learning Functional Programming through Multimedia.* Cambridge University Press, 2000.