Eine Behauptung ...

Schon bekannte Funktionen:

$$
\begin{aligned}
&\texttt{filter} :: (a \to \text{Bool}) \to [a] \to [a] \\
&\texttt{filter}\ p\ [\,] \qquad\quad = [\,] \\
&\texttt{filter}\ p\ (a : as) \mid p\ a \qquad\quad = a : (\texttt{filter}\ p\ as) \\
&\qquad\qquad\qquad\quad \mid \texttt{otherwise} = \texttt{filter}\ p\ as
\end{aligned}
$$

$$
\begin{aligned}
&\texttt{map} :: (a \to b) \to [a] \to [b] \\
&\texttt{map}\ h\ [\,] \qquad = [\,] \\
&\texttt{map}\ h\ (a : as) = (h\ a) : (\texttt{map}\ h\ as)
\end{aligned}
$$

Für jede Wahl von p, h und l gilt:

$$\texttt{filter}\ p\ (\texttt{map}\ h\ l) \;=\; \texttt{map}\ h\ (\texttt{filter}\ (p \circ h)\ l)$$

. . . und ihr Beweis

Per Induktion über die Länge der Liste:

1. $\texttt{filter}\ p\ (\texttt{map}\ h\ [\,]) = \texttt{map}\ h\ (\texttt{filter}\ (p \circ h)\ [\,])$? OK
2. $\texttt{filter}\ p\ (\texttt{map}\ h\ (a : as)) = \texttt{map}\ h\ (\texttt{filter}\ (p \circ h)\ (a : as))$?

$$
\begin{aligned}
& \texttt{filter}\ p\ (\texttt{map}\ h\ (a : as)) \\
= \;& \texttt{filter}\ p\ ((h\ a) : (\texttt{map}\ h\ as)) \\
= \;& \textbf{if}\ p\ (h\ a)\ \textbf{then}\ (h\ a) : (\texttt{filter}\ p\ (\texttt{map}\ h\ as)) \\
& \qquad\qquad \textbf{else}\ \ \texttt{filter}\ p\ (\texttt{map}\ h\ as) \\
= \;& \textbf{if}\ (p \circ h)\ a\ \textbf{then}\ (h\ a) : (\texttt{map}\ h\ (\texttt{filter}\ (p \circ h)\ as)) \\
& \qquad\qquad \textbf{else}\ \ \texttt{map}\ h\ (\texttt{filter}\ (p \circ h)\ as) \\
= \;& \textbf{if}\ (p \circ h)\ a\ \textbf{then}\ \texttt{map}\ h\ (a : (\texttt{filter}\ (p \circ h)\ as)) \\
& \qquad\qquad \textbf{else}\ \ \texttt{map}\ h\ (\texttt{filter}\ (p \circ h)\ as) \\
= \;& \texttt{map}\ h\ (\textbf{if}\ (p \circ h)\ a\ \textbf{then}\ a : (\texttt{filter}\ (p \circ h)\ as) \\
& \qquad\qquad\qquad\qquad \textbf{else}\ \ \texttt{filter}\ (p \circ h)\ as) \\
= \;& \texttt{map}\ h\ (\texttt{filter}\ (p \circ h)\ (a : as))
\end{aligned}
$$

OK, sofern

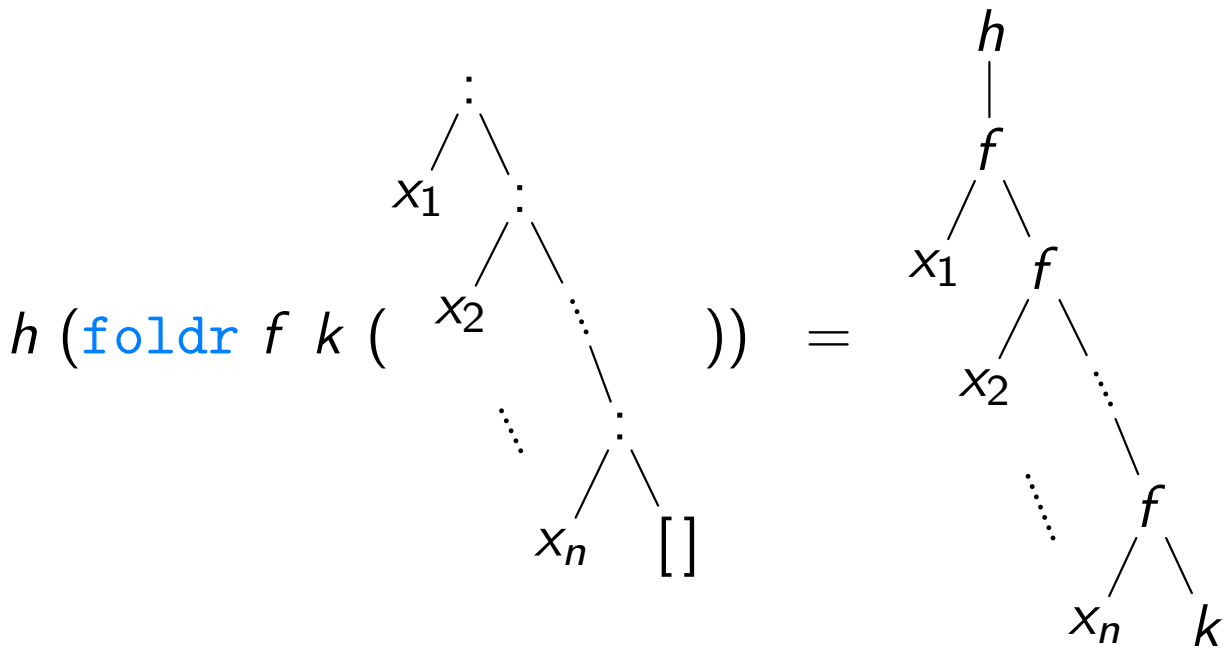
$\texttt{filter}\ p\ (\texttt{map}\ h\ as) = \texttt{map}\ h\ (\texttt{filter}\ (p \circ h)\ as)$

Ein weiteres Beispiel

Angenommen, wir wollen (für gewisse h, f, k und l)

$$h\ (\texttt{foldr}\ f\ k\ l)$$

durch einen einzelnen Aufruf von `foldr` ersetzen.



Angenommen, wir wüssten ein g, so dass für alle x und y,

$$h\ (f\ x\ y)\ =\ g\ x\ (h\ y)$$

. . . , dann:

$$h\ (\texttt{foldr}\ f\ k\ l)\ =\ \texttt{foldr}\ g\ (h\ k)\ l$$

Beweis per Induktion über die Länge der Liste:

1. $h\ (\texttt{foldr}\ f\ k\ [])\ =\ \texttt{foldr}\ g\ (h\ k)\ []$? OK
2. $h\ (\texttt{foldr}\ f\ k\ (a : as))\ =\ \texttt{foldr}\ g\ (h\ k)\ (a : as)$? OK

$$\begin{aligned} & h\ (\texttt{foldr}\ f\ k\ (a : as)) \\ = \quad & h\ (f\ a\ (\texttt{foldr}\ f\ k\ as)) \\ = \quad & g\ a\ (h\ (\texttt{foldr}\ f\ k\ as)) \\ = \quad & g\ a\ (\texttt{foldr}\ g\ (h\ k)\ as) \\ = \quad & \texttt{foldr}\ g\ (h\ k)\ (a : as) \end{aligned}$$

Auf anderen Datentypen

Zum Beispiel:

$$\textbf{data}\ \text{Tree}\ a\ =\ \text{Leaf}\ a \mid \text{Node}\ (\text{Tree}\ a)\ (\text{Tree}\ a)$$

$$\begin{aligned}
&\texttt{flat} :: \text{Tree}\ a \to [a] \\
&\texttt{flat}\ (\text{Leaf}\ a) &&= [a] \\
&\texttt{flat}\ (\text{Node}\ t_1\ t_2) &&= (\texttt{flat}\ t_1) +\!\!+ (\texttt{flat}\ t_2)
\end{aligned}$$

$$\begin{aligned}
&\texttt{incr} :: \text{Tree Int} \to \text{Tree Int} \\
&\texttt{incr}\ (\text{Leaf}\ a) &&= \text{Leaf}\ (a+1) \\
&\texttt{incr}\ (\text{Node}\ t_1\ t_2) &&= \text{Node}\ (\texttt{incr}\ t_1)\ (\texttt{incr}\ t_2)
\end{aligned}$$

Behauptung:

$$\texttt{flat}\ (\texttt{incr}\ t)\ =\ \texttt{map}\ (+1)\ (\texttt{flat}\ t)$$

Auf anderen Datentypen

Beweis per struktureller Induktion:

1. $\texttt{flat}\ (\texttt{incr}\ (\text{Leaf}\ a)) = \texttt{map}\ (+1)\ (\texttt{flat}\ (\text{Leaf}\ a))$?
   OK
2. $\texttt{flat}\ (\texttt{incr}\ (\text{Node}\ t_1\ t_2)) = \texttt{map}\ (+1)\ (\texttt{flat}\ (\text{Node}\ t_1\ t_2))$?
   OK (mit Hilfe: $(\texttt{map}\ h\ l_1) +\!\!+ (\texttt{map}\ h\ l_2) = \texttt{map}\ h\ (l_1 +\!\!+ l_2)$)

$$
\begin{aligned}
& \texttt{flat}\ (\texttt{incr}\ (\text{Node}\ t_1\ t_2)) \\
= \;& \texttt{flat}\ (\text{Node}\ (\texttt{incr}\ t_1)\ (\texttt{incr}\ t_2)) \\
= \;& (\texttt{flat}\ (\texttt{incr}\ t_1)) +\!\!+ (\texttt{flat}\ (\texttt{incr}\ t_2)) \\
= \;& (\texttt{map}\ (+1)\ (\texttt{flat}\ t_1)) +\!\!+ (\texttt{map}\ (+1)\ (\texttt{flat}\ t_2)) \\
= \;& \texttt{map}\ (+1)\ ((\texttt{flat}\ t_1) +\!\!+ (\texttt{flat}\ t_2)) \\
= \;& \texttt{map}\ (+1)\ (\texttt{flat}\ (\text{Node}\ t_1\ t_2))
\end{aligned}
$$

Manchmal nicht so einfach

Naives Umdrehen einer Liste:

$$\begin{aligned}
&\texttt{reverse} :: [a] \to [a] \\
&\texttt{reverse}\ [\,] \quad\ = [\,] \\
&\texttt{reverse}\ (a : as) = (\texttt{reverse}\ as) +\!\!+ [a]
\end{aligned}$$

Zu beweisen:

$$\texttt{map}\ h\ (\texttt{reverse}\ l) \ = \ \texttt{reverse}\ (\texttt{map}\ h\ l)$$

Induktionsschritt:

2. $\texttt{map}\ h\ (\texttt{reverse}\ (a : as)) \ = \ \texttt{reverse}\ (\texttt{map}\ h\ (a : as))$?

   OK

$$\begin{aligned}
& \texttt{map}\ h\ (\texttt{reverse}\ (a : as)) \\
= \ & \texttt{map}\ h\ ((\texttt{reverse}\ as) +\!\!+ [a]) \\
= \ & (\texttt{map}\ h\ (\texttt{reverse}\ as)) +\!\!+ (\texttt{map}\ h\ [a]) \\
= \ & (\texttt{reverse}\ (\texttt{map}\ h\ as)) +\!\!+ [h\ a] \\
= \ & \texttt{reverse}\ ((h\ a) : (\texttt{map}\ h\ as)) \\
= \ & \texttt{reverse}\ (\texttt{map}\ h\ (a : as))
\end{aligned}$$

Manchmal nicht so einfach

Effizientes Umdrehen einer Liste:

$$\begin{aligned}
&\texttt{reverse}' :: [a] \to [a] \to [a] \\
&\texttt{reverse}'\ [\,]\ \ \ \ \ \ \ bs = bs \\
&\texttt{reverse}'\ (a : as)\ bs = \texttt{reverse}'\ as\ (a : bs)
\end{aligned}$$

Zu beweisen:

$$\texttt{map}\ h\ (\texttt{reverse}'\ l\ [\,])\ =\ \texttt{reverse}'\ (\texttt{map}\ h\ l)\ [\,]$$

Induktionsschritt:

2. $\texttt{map}\ h\ (\texttt{reverse}'\ (a : as)\ [\,])\ =\ \texttt{reverse}'\ (\texttt{map}\ h\ (a : as))\ [\,]$?

$$\begin{aligned}
& \texttt{map}\ h\ (\texttt{reverse}'\ (a : as)\ [\,]) \\
= \quad & \texttt{map}\ h\ (\texttt{reverse}'\ as\ [a]) \\
? \quad & \\
= \quad & \texttt{reverse}'\ (\texttt{map}\ h\ as)\ [h\ a] \\
= \quad & \texttt{reverse}'\ ((h\ a) : (\texttt{map}\ h\ as))\ [\,] \\
= \quad & \texttt{reverse}'\ (\texttt{map}\ h\ (a : as))\ [\,]
\end{aligned}$$

Freie Theoreme [Wadler, 1989]

Ursprüngliches Beispiel:

```
filter :: (a → Bool) → [a] → [a]
filter p []                = []
filter p (a : as) | p a           = a : (filter p as)
                  | otherwise     = filter p as
```

Für jede Wahl von p, h und l gilt:

$$\texttt{filter}\ p\ (\texttt{map}\ h\ l) \quad = \quad \texttt{map}\ h\ (\texttt{filter}\ (p \circ h)\ l)$$

Bewiesen per Induktion.

Aber auch möglich ohne Induktion!

Freie Theoreme [Wadler, 1989]

Ursprüngliches Beispiel:

$$\texttt{filter} :: (a \to \text{Bool}) \to [a] \to [a]$$

$$\texttt{takeWhile} :: (a \to \text{Bool}) \to [a] \to [a]$$

$$\texttt{g} :: (a \to \text{Bool}) \to [a] \to [a]$$

Für jede Wahl von p, h und l gilt:

$$\begin{aligned}
\texttt{filter}\ p\ (\texttt{map}\ h\ l) &= \texttt{map}\ h\ (\texttt{filter}\ (p \circ h)\ l) \\
\texttt{takeWhile}\ p\ (\texttt{map}\ h\ l) &= \texttt{map}\ h\ (\texttt{takeWhile}\ (p \circ h)\ l) \\
\texttt{g}\ p\ (\texttt{map}\ h\ l) &= \texttt{map}\ h\ (\texttt{g}\ (p \circ h)\ l)
\end{aligned}$$

Aber wie soll man so etwas beweisen können?

Versuch einer Argumentation

- `g` :: $(a \to \text{Bool}) \to [a] \to [a]$ muss für jede mögliche Instanziierung von a einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l und dem Eingabeprädikat p abhängen.
- Die einzig möglichen Grundlagen zur Entscheidung sind die Länge von l und die Ergebnisse von p auf Elementen von l.
- Aber, die Listen (`map` h l) und l haben stets die selbe Länge.
- Und Anwendung von p auf ein Element von (`map` h l) hat stets das selbe Ergebnis wie Anwendung von $(p \circ h)$ auf das entsprechende Element von l.
- Also wählt `g` mit p stets „die selben“ Elemente aus (`map` h l) wie es `g` mit $(p \circ h)$ aus l tut, außer dass im ersten Fall die entsprechenden Abbilder unter h ausgegeben werden.
- Also ist (`g` p (`map` h l)) gleich (`map` h (`g` $(p \circ h)$ l)).
- Genau das wollten wir beweisen!

Geht es vielleicht etwas formaler?

Gegeben:

$$\mathtt{g} :: (a \to \mathrm{Bool}) \to [a] \to [a]$$

Behauptung, für jede Wahl von p, h und l:

$$\mathtt{g}\ p\ (\mathtt{map}\ h\ l) \;=\; \mathtt{map}\ h\ (\mathtt{g}\ (p \circ h)\ l)$$

Wie wäre es mit Induktion über den Syntaxaufbau von $\mathtt{g}$?
Problem: lässt Typinformation außer Acht

Wie wäre es dann mit Induktion nur über alle Funktionen des Typs $(a \to \mathrm{Bool}) \to [a] \to [a]$?
Problem: dieser Bereich nicht kompositionell

Wir müssen also allgemein Aussagen der obigen Art für alle Typen „gleichzeitig“ beweisen!

Geht es vielleicht etwas formaler?

Strategie:

- zu jedem Typ angeben, welche möglichen Varianten (semantisch, nicht syntaktisch) von Elementen diesen Typs es gibt
- das Ganze möglichst kompositionell über den Aufbau von Typen
- später per Induktion über die Syntax beweisen, dass jede Funktion die semantischen Einschränkungen ihres Typs erfüllt

Erster Punkt an Beispielen

Frage: Welche Funktionen haben den Typ „$(a, a) \to a$“ ?

Antwort: nur `fst` und `snd` (und semantisch äquivalente)

Frage: Welche Funktionen haben den Typ „$a \to a$“ ?

Antwort: nur `id`

Frage: Welche Funktionen haben den Typ „$a \to [a]$“ ?

Antwort: $\texttt{f}_0$, $\texttt{f}_1$, $\texttt{f}_2$, . . . , wobei:

$$\begin{aligned} \texttt{f}_0\ \_ &= [\,] \\ \texttt{f}_1\ a &= [a] \\ \texttt{f}_2\ a &= [a, a] \\ &\ldots \end{aligned}$$

Frage: Welche Funktionen haben den Typ „$[a] \to [a]$“ ?

Welche Möglichkeiten gibt es für g :: $[a] \to [a]$?

Zunächst wieder intuitiv:

- Die Ausgabe- kann nur Elemente der Eingabeliste enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von dieser Liste abhängen, und zwar von ihrer Länge.
- Folglich sind für eine feste Eingabelänge die Länge der Ausgabeliste sowie die Anordnung der Elemente darin stets fix.

Formal beschrieben:

Für jedes g :: $[a] \to [a]$ gibt es:

- eine Funktion $f : \mathbb{N} \to \mathbb{N}$, so dass für jedes $n \in \mathbb{N}$ aus einer Liste der Länge n mit g stets eine Liste der Länge $f(n)$ wird;
- eine Funktion $h : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, so dass für jede Liste einer Länge $n \in \mathbb{N}$ und für jedes $0 \leq m < f(n)$, das m-te Element der Ausgabeliste das $h(n, m)$-te Element der Eingabeliste ist.

Geht es vielleicht etwas formaler?

Strategie:

- zu jedem Typ angeben, welche möglichen Varianten (semantisch, nicht syntaktisch) von Elementen diesen Typs es gibt
- das Ganze möglichst kompositionell über den Aufbau von Typen
- später per Induktion über die Syntax beweisen, dass jede Funktion die semantischen Einschränkungen ihres Typs erfüllt