Chapter 26: Unfold

Zhenjiang Hu, Wei Zhang

School of Computer Science
Peking University

December 29, 2023

Outline

foldr

The computation pattern by h is captured by a higher-order function *foldr*.

$$
\begin{array}{lcl}
foldr & :: & (\alpha \to \beta \to \beta) \to \beta \to [\alpha] \to \beta \\
foldr\ f\ e\ [\,] & = & e \\
foldr\ f\ e\ (x : xs) & = & f\ x\ (foldr\ f\ e\ xs)
\end{array}
$$

foldr

The computation pattern by h is captured by a higher-order function *foldr*.

$$\begin{array}{lcl} foldr & :: & (\alpha \to \beta \to \beta) \to \beta \to [\alpha] \to \beta \\ foldr\ f\ e\ [\,] & = & e \\ foldr\ f\ e\ (x : xs) & = & f\ x\ (foldr\ f\ e\ xs) \end{array}$$

In BMF, we write it as right-to-left reduction.

$$\begin{array}{lcl} \oplus \not\leftarrow_e [\,] & = & e \\ \oplus \not\leftarrow_e (a : x) & = & a \oplus (\oplus \not\leftarrow_e x) \end{array}$$

Outline

unfold

unfold is the essential and simplest computation pattern for producing (possibly infinite) lists.

unfold

unfold is the essential and simplest computation pattern for producing (possibly infinite) lists.

$$unfold :: (b \to Bool) \to (b \to a) \to (b \to b) \to b \to [a]$$
$$unfold\ p\ f\ g\ x = \textbf{if}\ p\ x\ \textbf{then}\ [\,]\ \textbf{else}\ f\ x : unfold\ p\ f\ g\ (g\ x)$$

unfold

unfold is the essential and simplest computation pattern for producing (possibly infinite) lists.

$$
\begin{array}{l}
\mathit{unfold} :: (b \rightarrow \mathit{Bool}) \rightarrow (b \rightarrow a) \rightarrow (b \rightarrow b) \rightarrow b \rightarrow [a] \\
\mathit{unfold}\ p\ f\ g\ x = \textbf{if}\ p\ x\ \textbf{then}\ [\,]\ \textbf{else}\ f\ x : \mathit{unfold}\ p\ f\ g\ (g\ x)
\end{array}
$$

This reads that

- it generates an empty list if the input satisfies p;
- otherwise, it generates a list whose head is produced by f on the input and whose tail is recursively produced from the new input created by g on the input.

Specification with *unfold*

Example: *upto* in *unfold*

Considering the function *upto*:

$$upto\ (3, 6) = [3, 4, 5, 6]$$

we can define it as an *unfold*.

Specification with *unfold*

Example: *upto* in *unfold*

Considering the function *upto*:

$$upto\ (3,6) = [3,4,5,6]$$

we can define it as an *unfold*.

$$\begin{aligned} upto\ (m,n) &= unfold\ fstGreater\ fst\ succFst\ (m,n) \\ \textbf{where}\ fstGreater\ (x,y) &= x > y \\ succFst\ (x,y) &= (x+1,y) \end{aligned}$$

Specification with *unfold*

Example: *upto* in *unfold*

Considering the function *upto*:

$$upto\ (3,6) = [3,4,5,6]$$

we can define it as an *unfold*.

$$\begin{aligned} upto &= unfold\ fstGreater\ fst\ succFst \\ \textbf{where}\ fstGreater\ (x,y) &= x > y \\ succFst\ (x,y) &= (x+1,y) \end{aligned}$$

Specification with *unfold*

Example: *map* in *unfold*

$$map\ f = unfold\ (== [\,])\ (f \cdot head)\ tail$$

Specification with *unfold*

Example: *map* in *unfold*

$$map\ f = unfold\ (==[\,])\ (f \cdot head)\ tail$$

Homework BMF 4-1

Given two *sorted* lists (xs, ys), the function *merge* (xs, ys) merges them into one sorted list. Define *merge* as an unfold.

Specification with *unfold*

Example: *fib* in *unfold*

Considering the function *fib* for generating infinite sequence of all Fibonacci numbers:

$$fib\ (0,1) = [0,1,1,2,3,5,8,13,21,\ldots]$$

we can define it as an *unfold*.

Specification with *unfold*

Example: *fib* in *unfold*

Considering the function *fib* for generating infinite sequence of all Fibonacci numbers:

$$fib\ (0,1) = [0,1,1,2,3,5,8,13,21,\ldots]$$

we can define it as an *unfold*.

$$\begin{aligned} fib &= unfold\ (const\ False)\ fst\ (\lambda(x,y).\,(y,x+y)) \\ &\quad \textbf{where}\ const\ x\ y = x \end{aligned}$$

Specification with *unfold*

Example: *fib* in *unfold*

Considering the function *fib* for generating infinite sequence of all Fibonacci numbers:

$$fib\ (0, 1) = [0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots]$$

we can define it as an *unfold*.

$$\begin{array}{l} fib = unfold\ (const\ False)\ fst\ (\lambda(x, y).\,(y, x + y)) \\ \qquad \mathbf{where}\ const\ x\ y = x \end{array}$$

Homework BMF 4-2

Change the above definition of *fib* to generate all Fibonacci numbers that are less than 1000,000.

Specification with *unfold*: $unfold^{\infty}$

$unfold^{\infty}$, a special case of *unfold*, is used to generate streams (infinite lists, denoted by $[a]^{\infty}$).

$$unfold^{\infty}\ f\ g = unfold\ (const\ False)\ f\ g$$

Specification with *unfold*: $unfold^{\infty}$

$unfold^{\infty}$, a special case of *unfold*, is used to generate streams (infinite lists, denoted by $[a]^{\infty}$).

$$unfold^{\infty}\ f\ g = unfold\ (const\ False)\ f\ g$$

It is characterized by the following two equations.

$$\begin{array}{lcl} head \cdot unfold^{\infty}\ f\ g & = & f \\ tail \cdot unfold^{\infty}\ f\ g & = & unfold^{\infty}\ f\ g \cdot g \end{array}$$

Specification with *unfold*: $unfold^{\infty}$

Examples

$$
\begin{array}{lcl}
fib & = & unfold^{\infty}\ fst\ (\lambda(x,y).\,(y, x+y)) \\
from & = & unfold^{\infty}\ id\ succ \quad \textbf{where}\ succ\ n = n+1 \\
iterate\ f & = & unfold^{\infty}\ id\ f \\
ones & = & unfold^{\infty}\ (const\ 1)\ id
\end{array}
$$

When can a function be described by an *unfold*?

Not all functions can be described by a single *unfold*.

Example

The function

$$\begin{array}{lcl} mults & :: & Nat \to [a]^{\infty} \\ mults\ n & = & [n \times 0, n \times 1, n \times 2, \ldots] \end{array}$$

cannot be described by a single *unfold*.

When can a function be described by an *unfold*?

Not all functions can be described by a single *unfold*.

Example

The function

$$\begin{array}{lcl} mults & :: & Nat \to [a]^{\infty} \\ mults\ n & = & [n \times 0, n \times 1, n \times 2, \ldots] \end{array}$$

cannot be described by a single *unfold*.
Proof Sketch. The existence of f and g such that $mults = unfold^{\infty}\ f\ g$ will destroy the equation $tail \cdot unfold^{\infty}\ f\ g = unfold^{\infty}\ f\ g \cdot g$.

When can a function be described by an *unfold*?

Not all functions can be described by a single *unfold*.

Example

The function

$$\begin{array}{lcl} mults & :: & Nat \to [a]^\infty \\ mults\ n & = & [n \times 0, n \times 1, n \times 2, \ldots] \end{array}$$

cannot be described by a single *unfold*.
Proof Sketch. The existence of f and g such that $mults = unfold^\infty\ f\ g$ will destroy the equation $tail \cdot unfold^\infty\ f\ g = unfold^\infty\ f\ g \cdot g$.

Can you construct a theory for unfold (like what we have for homomorphism)?