

Chapter 28: Unfold

Zhenjiang Hu, Wei Zhang

School of Computer Science
Peking University

December 26, 2025

Outline

- 1 Review: Foldr
- 2 Unfold

foldr

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

$$\begin{aligned} \text{foldr} &:: (\alpha \rightarrow \beta \rightarrow \beta) \rightarrow \beta \rightarrow [\alpha] \rightarrow \beta \\ \text{foldr } f \ e \ [] &= e \\ \text{foldr } f \ e \ (x : xs) &= f \ x \ (\text{foldr } f \ e \ xs) \end{aligned}$$

foldr

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

$$\begin{aligned} \text{foldr} &:: (\alpha \rightarrow \beta \rightarrow \beta) \rightarrow \beta \rightarrow [\alpha] \rightarrow \beta \\ \text{foldr } f \ e \ [] &= e \\ \text{foldr } f \ e \ (x : xs) &= f \ x \ (\text{foldr } f \ e \ xs) \end{aligned}$$

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

$$\begin{aligned} \bigoplus \leftarrow_e [] &= e \\ \bigoplus \leftarrow_e (a : x) &= a \oplus (\bigoplus \leftarrow_e x) \end{aligned}$$

Outline

- 1 Review: Foldr
- 2 Unfold

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.

$$\begin{aligned} \text{unfold} &:: (b \rightarrow \text{Bool}) \rightarrow (b \rightarrow a) \rightarrow (b \rightarrow b) \rightarrow b \rightarrow [a] \\ \text{unfold } p \ f \ g \ x &= \mathbf{if } p \ x \ \mathbf{then } [] \ \mathbf{else } f \ x : \text{unfold } p \ f \ g \ (g \ x) \end{aligned}$$

unfold

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

$$\begin{aligned} \text{unfold} &:: (b \rightarrow \text{Bool}) \rightarrow (b \rightarrow a) \rightarrow (b \rightarrow b) \rightarrow b \rightarrow [a] \\ \text{unfold } p \ f \ g \ x &= \mathbf{if } p \ x \ \mathbf{then } [] \ \mathbf{else } f \ x : \text{unfold } p \ f \ g \ (g \ x) \end{aligned}$$

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*:

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

we can define it as an *unfold*.

Specification with *unfold*

Example: *upto* in *unfold*

Considering the function *upto*:

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

we can define it as an *unfold*.

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

Specification with *unfold*Example: *upto* in *unfold*

Considering the function *upto*:

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

we can define it as an *unfold*.

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

Specification with *unfold*Example: *map* in *unfold*
$$\text{map } f = \text{unfold } (== []) (f \cdot \text{head}) \text{ tail}$$

Specification with *unfold*

Example: *map* in *unfold*

$$\text{map } f = \text{unfold } (== []) (f \cdot \text{head}) \text{ tail}$$

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:

$$\text{fib } (0, 1) = [0, 1, 1, 2, 3, 5, 8, 13, 21, \dots]$$

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:

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

we can define it as an *unfold*.

$$fib = unfold\ (const\ False)\ fst\ (\lambda(x, y). (y, x + y))$$

where *const* $x\ y = x$

Specification with *unfold*

Example: *fib* in *unfold*

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

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

we can define it as an *unfold*.

$$fib = unfold\ (const\ False)\ fst\ (\lambda(x, y). (y, x + y))$$

where *const* $x\ y = x$

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.

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

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

Examples

fib = $unfold^\infty \text{fst } (\lambda(x, y). (y, x + y))$
 $from$ = $unfold^\infty \text{id succ}$ **where** $\text{succ } n = n + 1$
 $iterate\ f$ = $unfold^\infty \text{id } f$
 $ones$ = $unfold^\infty (\text{const } 1) \text{id}$

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

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

Example

The function

$$\begin{aligned} \text{mults} &:: \text{Nat} \rightarrow [a]^\infty \\ \text{mults } n &= [n \times 0, n \times 1, n \times 2, \dots] \end{aligned}$$

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

$$\begin{aligned} \text{mults} &:: \text{Nat} \rightarrow [a]^\infty \\ \text{mults } n &= [n \times 0, n \times 1, n \times 2, \dots] \end{aligned}$$

cannot be described by a single *unfold*.

Proof Sketch. The existence of f and g such that $\text{mults} = \text{unfold}^\infty f g$ will destroy the equation $\text{tail} \cdot \text{unfold}^\infty f g = \text{unfold}^\infty f g \cdot g$.

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

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

Example

The function

$$\begin{aligned} \text{mults} &:: \text{Nat} \rightarrow [a]^\infty \\ \text{mults } n &= [n \times 0, n \times 1, n \times 2, \dots] \end{aligned}$$

cannot be described by a single *unfold*.

Proof Sketch. The existence of f and g such that $\text{mults} = \text{unfold}^\infty f g$ will destroy the equation $\text{tail} \cdot \text{unfold}^\infty f g = \text{unfold}^\infty f g \cdot g$.

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