第18章：序列理论概述(2)

胡振江，张伟

信息学院计算机科学技术系

2021年12月15日

序列理论 (Theory of Lists)

- BMF (Bird Meertens Formalism)

A set of combinators
(higher order functions on Lists)

+

A set of rules/laws
(properties)

Calculational Functional Programming
(Constructive Functional Programming)

参考资料： Richard Bird, Lecture Notes on Constructive Functional Programming, Technical Monograph PRG-69, Oxford University, 1988.

MSS的问题描述：利用特定的组合子

```
[-]_ : ∀ {A : Set} (x : A) → List A
[-] x = x ∷ []

inits : ∀ {A : Set} → List A → List (List A)
inits = scanl _++_ [] ∘ map ([-]_)

tails : ∀ {A : Set} → List A → List (List A)
tails = scanr _++_ [] ∘ map ([-]_)

segs : ∀ {A : Set} → List A → List (List A)
segs = foldr _++_ [] ∘ map tails ∘ inits

sum : List ℕ → ℕ
sum = foldr _+_ zero

max : List ℕ → ℕ
max = foldr _⊔_ zero

mss : List ℕ → ℕ
mss = max ∘ map sum ∘ segs
```

Hom: Map/Reduce

```
reduce : ∀{A : Set} → (_⊕_ : A → A → A) → (e : A)
         → IsMonoid _⊕_ e → List A -> A
reduce _⊕_ e _ []        = e
reduce _⊕_ e p (x ∷ xs) = x ⊕ reduce _⊕_ e p xs
```

```
hom : ∀{A B : Set} (_⊕_ : B → B → B) (e : B)
      (p : IsMonoid _⊕_ e) → (A → B)
      → List A -> B
hom _⊕_ e p f = reduce _⊕_ e p ∘ map f
```

Promotion Laws

```
map-promotion :
   ∀{A B : Set} (f : A → B)
   → map f ∘ flatten ≡ flatten ∘ map (map f)

reduce-promotion :
   ∀{A : Set} (_⊕_ : A → A → A) (e : A)
       (p : IsMonoid _⊕_ e)
   → reduce _⊕_ e p ∘ flatten
       ≡ reduce _⊕_ e p ∘ map (reduce _⊕_ e p)
```

```
flatten : ∀{A : Set} → List (List A) → List A
flatten = foldr _++_ []
```

Hom-Hom Fusion

```
hom-hom : ∀{A B C : Set} (_⊕_ : C → C → C) (e : C)
    (p : IsMonoid _⊕_ e)(g : A → List B) (f : B → C)
    → hom _⊕_ e p f ∘ hom _++_ [] _ g
       ≡ hom _⊕_ e p (hom _⊕_ e p f ∘ g)
```

$$
\begin{aligned}
& \oplus/ \cdot f* \cdot +\!\!+/ \cdot g* \\
= & \quad \{ \text{ map promotion } \} \\
& \oplus/ \cdot +\!\!+/ \cdot f** \cdot g* \\
= & \quad \{ \text{ reduce promotion } \} \\
& \oplus/ \cdot (\oplus/)* \cdot f** \cdot g* \\
= & \quad \{ \text{ map distribution } \} \\
& \oplus/ \cdot (\oplus/ \cdot f* \cdot g)*
\end{aligned}
$$

记号：

```
reduce _⊕_ e _ = ⊕/
map f = f∗
```

Accumulation Lemma

```
acc-lemma : ∀{A : Set} (_⊕_ : A → A → A) (e : A)
   → scanl _⊕_ e ≡ map (foldl _⊕_ e) ∘ inits
```

O(n) O(n²)

$$(\oplus \not\!\!\!\twoheadrightarrow_e) = (\oplus \not\rightarrow_e) * \cdot inits$$

Honor's Rule

```
R-Dist : ∀{A : Set} (_⊕_ : A → A → A)(_⊗_ : A → A → A) → Set
R-Dist {A} _⊕_ _⊗_ = ∀ (a b c : A)
    → (a ⊕ b) ⊗ c ≡ (a ⊗ c) ⊕ (b ⊕ c)
```

```
horner-rule : ∀{A : Set} (_⊕_ : A → A → A) (e-⊕ : A)
               (_⊗_ : A → A → A) (e-⊗ : A)
    → (p : IsMonoid _⊕_ e-⊕)
    → (q : IsMonoid _⊗_ e-⊗)
    → (rdist : R-Dist _⊕_ _⊗_)
     -----------------------------
    → reduce _⊕_ e-⊕ p ∘ map (reduce _⊗_ e-⊗ q) ∘ tails
       ≡ foldl (λ a b → (a ⊗ b) ⊕ e-⊗ ) e-⊗
```

$$\oplus/ \cdot \otimes/ * \cdot \mathit{tails} = \odot \not\rightarrow_e$$

where

$$e = id_{\otimes}$$

$$a \odot b = (a \otimes b) \oplus e$$

北京大學 PEKING UNIVERSITY

高效mss的程序推导

$$
\begin{array}{cl}
 & mss \\
= & \quad \{ \text{ definition of } mss \ \} \\
 & \uparrow / \cdot + / * \cdot segs \\
= & \quad \{ \text{ definition of } segs \ \} \\
 & \uparrow / \cdot + / * \cdot +\!\!+ / \cdot tails * \cdot inits \\
= & \quad \{ \text{ map and reduce promotion } \} \\
 & \uparrow / \cdot (\uparrow / \cdot + / * \cdot tails) * \cdot inits \\
= & \quad \{ \text{ Horner's rule with } a \odot b = (a + b) \uparrow 0 \ \} \\
 & \uparrow / \cdot \odot \not\rightarrow_0 * \cdot inits \\
= & \quad \{ \text{ accumulation lemma } \} \\
 & \uparrow / \cdot \odot /\!\!\!/\!\!\!\rightarrow_0
\end{array}
$$

作业

18-4. 利用agda实现mss的程序推导。