

# Bird Meertens Formalism

– An Application –

Zhenjiang Hu, Wei Zhang

Department of Computer Science and Technology  
Peking University

December 24, 2021

# Outline

- 1 Automatic Parallelization
  - Parallelizing List Functions
  - Parallelizing Tree Functions

## Maximum Prefix Sum Problem

Design a D&C parallel program that computes the maximum of all the prefix sums of a list.

$$mps [1, -2, 3, -9, 5, 7, -10, 8, -9, 10] = 5$$

# Review: List Homomorphism

Function  $h$  on lists is a list homomorphism, if

$$\begin{aligned}h [] &= e \\h [a] &= f a \\h (x ++ y) &= h x \odot h y\end{aligned}$$

for some  $\odot$ .

## Properties

- Suitable for parallel computation in the D&C style
- Basic concept for skeletal parallel programming
- Enjoy many nice algebraic properties (1st, 2nd, 3rd Homomorphism theorems)

# Review: Existence of Homomorphism

## Existence Lemma

The list function  $h$  is a homomorphism iff the implication

$$h\ v = h\ x \wedge h\ w = h\ y \Rightarrow h\ (v ++ w) = h\ (x ++ y)$$

holds for all lists  $v, w, x, y$ .

# The Third Homomorphism Theorem (Gibbons:JFP95)

A function  $f$  can be described as a foldl and a foldr

$$\begin{aligned} h &= \oplus \swarrow e \\ h &= \otimes \searrow e \end{aligned}$$

# The Third Homomorphism Theorem (Gibbons:JFP95)

A function  $f$  can be described as a foldl and a foldr

$$\begin{aligned} h &= \oplus \swarrow_e \\ h &= \otimes \nearrow_e \end{aligned}$$

that is,

$$\begin{aligned} h ([a] ++ x) &= a \oplus h x \\ h (x ++ [a]) &= h x \otimes a \end{aligned}$$

iff there **exists** an associative operator  $\odot$  such that

$$h(x ++ y) = h x \odot h y.$$

# The Third Homomorphism Theorem (Gibbons:JFP95)

A function  $f$  can be described as a foldl and a foldr

$$\begin{aligned} h &= \oplus \swarrow_e \\ h &= \otimes \nearrow_e \end{aligned}$$

that is,

$$\begin{aligned} h ([a] ++ x) &= a \oplus h x \\ h (x ++ [a]) &= h x \otimes a \end{aligned}$$

iff there **exists** an associative operator  $\odot$  such that

$$h(x ++ y) = h x \odot h y.$$

Two sequential programs guarantee existence of a parallel program!



# Proof of the Third Homomorphism Theorem

**Proof.** Let  $h \ v = h \ x$  and  $h \ w = h \ y$ . Then:

$$\begin{aligned}
 & h \ (v ++ w) \\
 = & \quad \{ \ h = \oplus \not\rightarrow_e \ \} \\
 & \oplus \not\rightarrow_e (v ++ w) \\
 = & \quad \{ \text{property of right-to-left reduction} \ \} \\
 & \oplus \not\rightarrow_e \oplus \not\rightarrow_e w \ v \\
 = & \quad \{ \ h \ w = h \ y \ \} \\
 & \oplus \not\rightarrow_e \oplus \not\rightarrow_e y \ v \\
 = & \quad \{ \text{property of right-to-left reduction} \ \} \\
 & \oplus \not\rightarrow_e (v ++ y) \\
 = & \quad \{ \ h = \oplus \not\rightarrow_e \ \} \\
 & h \ (v ++ y) \\
 = & \quad \{ \text{symmetrically, since } h = \otimes \not\rightarrow_e \ \} \\
 & h \ (x ++ y)
 \end{aligned}$$

# Proof of the Third Homomorphism Theorem

**Proof.** Let  $h \ v = h \ x$  and  $h \ w = h \ y$ . Then:

$$\begin{aligned}
 & h \ (v ++ w) \\
 = & \quad \{ \ h = \oplus \not\rightarrow_e \ \} \\
 & \oplus \not\rightarrow_e (v ++ w) \\
 = & \quad \{ \text{property of right-to-left reduction} \ \} \\
 & \oplus \not\rightarrow_e \oplus \not\rightarrow_e w \ v \\
 = & \quad \{ \ h \ w = h \ y \ \} \\
 & \oplus \not\rightarrow_e \oplus \not\rightarrow_e y \ v \\
 = & \quad \{ \text{property of right-to-left reduction} \ \} \\
 & \oplus \not\rightarrow_e (v ++ y) \\
 = & \quad \{ \ h = \oplus \not\rightarrow_e \ \} \\
 & h \ (v ++ y) \\
 = & \quad \{ \text{symmetrically, since } h = \otimes \not\rightarrow_e \ \} \\
 & h \ (x ++ y)
 \end{aligned}$$

By the Existence Lemma,  $h$  is a homomorphism.

# Examples

- $\text{sum } [1, 2, 3] = 6$

# Examples

- $\text{sum } [1, 2, 3] = 6$

$$\text{sum } (a : x) = a + \text{sum } x$$

$$\text{sum } (x ++ [a]) = \text{sum } x + a$$

# Examples

- $\text{sum } [1, 2, 3] = 6$

$$\begin{aligned}\text{sum } (a : x) &= a + \text{sum } x \\ \text{sum } (x ++ [a]) &= \text{sum } x + a\end{aligned}$$

- $\text{sort } [1, 3, 2] = [1, 2, 3]$

# Examples

- $\text{sum } [1, 2, 3] = 6$

$$\text{sum } (a : x) = a + \text{sum } x$$

$$\text{sum } (x ++ [a]) = \text{sum } x + a$$

- $\text{sort } [1, 3, 2] = [1, 2, 3]$

$$\text{sort } (a : x) = \text{insert } a (\text{sort } x)$$

$$\text{sort } (x ++ [b]) = \text{insert } b (\text{sort } x)$$

# Examples

- $\text{sum } [1, 2, 3] = 6$

$$\text{sum } (a : x) = a + \text{sum } x$$

$$\text{sum } (x ++ [a]) = \text{sum } x + a$$

- $\text{sort } [1, 3, 2] = [1, 2, 3]$

$$\text{sort } (a : x) = \text{insert } a (\text{sort } x)$$

$$\text{sort } (x ++ [b]) = \text{insert } b (\text{sort } x)$$

- $\text{psums } [1, 2, 3] = [1, 1 + 2, 1 + 2 + 3]$

# Examples

- $\text{sum } [1, 2, 3] = 6$

$$\begin{aligned}\text{sum } (a : x) &= a + \text{sum } x \\ \text{sum } (x ++ [a]) &= \text{sum } x + a\end{aligned}$$

- $\text{sort } [1, 3, 2] = [1, 2, 3]$

$$\begin{aligned}\text{sort } (a : x) &= \text{insert } a (\text{sort } x) \\ \text{sort } (x ++ [b]) &= \text{insert } b (\text{sort } x)\end{aligned}$$

- $\text{psums } [1, 2, 3] = [1, 1 + 2, 1 + 2 + 3]$

$$\begin{aligned}\text{psums } (a : x) &= a : (a+) * (\text{psums } x) \\ \text{psums } (x ++ [b]) &= \text{psums } x ++ [\text{last } (\text{psums } x) + b]\end{aligned}$$



# A Challenge Problem

It remains as a challenge to automatically derive *efficient* an associative operator  $\odot$  from  $\oplus$  and  $\otimes$ .

# Parallelization Theorem

Let  $f^\circ$  denote a weak right inverse of  $f$ .

$$\begin{array}{rcl}
 f(a : x) & = & a \oplus f x \\
 f(x ++ [b]) & = & f x \otimes b \\
 \hline
 f(x ++ y) & = & f x \odot f y \\
 \text{where } a \odot b & = & f(f^\circ a ++ f^\circ b)
 \end{array}$$

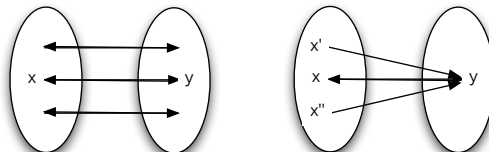
# Weak (Right) Inverse

- $g$  is an **inverse** of  $f$ , if

$$g y = x \Leftrightarrow f x = y$$

- $g$  is a **weak (right) inverse** of  $f$ , if for  $y \in \text{image}(f)$

$$g y = x \Rightarrow f x = y$$



# Properties of Weak Inverse

- Weak inverse always **exists** but may **not be unique**.

**Example:** Function *sum*

$$\begin{aligned} \text{sum } [] &= 0 \\ \text{sum } (a : x) &= a + \text{sum } x \end{aligned}$$

can have infinite number of weak inverse:

# Properties of Weak Inverse

- Weak inverse always **exists** but may **not be unique**.

**Example:** Function *sum*

$$\begin{aligned} \text{sum } [] &= 0 \\ \text{sum } (a : x) &= a + \text{sum } x \end{aligned}$$

can have infinite number of weak inverse:

$$\begin{aligned} g_1 y &= [y] \\ g_2 y &= [0, y] \\ &\dots \end{aligned}$$

# Parallelizing *sum*

From

$$\textcircled{1} \quad \textit{sum} (a : x) = a + \textit{sum} x$$

$$\textcircled{2} \quad \textit{sum} (x ++ [b]) = \textit{sum} x + b$$

$$\textcircled{3} \quad \textit{sum}^\circ y = [y]$$

we soon obtain

$$\textit{sum} (x ++ y) = \textit{sum} x \odot \textit{sum} y$$

where

$$\begin{aligned} a \odot b &= \textit{sum} (\textit{sum}^\circ a ++ \textit{sum}^\circ b) \\ &= \textit{sum} ([a] ++ [b]) \\ &= a + b \end{aligned}$$

# Parallelizing *sum*

From

$$\textcircled{1} \quad \textit{sum} (a : x) = a + \textit{sum} x$$

$$\textcircled{2} \quad \textit{sum} (x ++ [b]) = \textit{sum} x + b$$

$$\textcircled{3} \quad \textit{sum}^\circ y = [y]$$

we soon obtain

$$\textit{sum} (x ++ y) = \textit{sum} x \odot \textit{sum} y$$

where

$$\begin{aligned} a \odot b &= \textit{sum} (\textit{sum}^\circ a ++ \textit{sum}^\circ b) \\ &= \textit{sum} ([a] ++ [b]) \\ &= a + b \end{aligned}$$

That is,

$$\textit{sum} (x ++ y) = \textit{sum} x + \textit{sum} y.$$

# Weak inversion is not easy!

- What is a weak inverse for *sum*?

$$\text{sum } [] = 0$$

$$\text{sum } (a : x) = a + \text{sum } x$$



# Weak inversion is not easy!

- What is a weak inverse for *sum*?  $sum^\circ y = [y]$

$$\begin{aligned} sum [] &= 0 \\ sum (a : x) &= a + sum x \end{aligned}$$

# Weak inversion is not easy!

- What is a weak inverse for *sum*?  $sum^\circ y = [y]$

$$\begin{aligned} sum [] &= 0 \\ sum (a : x) &= a + sum x \end{aligned}$$

- What is it for *mps*?

$$\begin{aligned} mps [] &= 0 \\ mps (a : x) &= 0 \uparrow a \uparrow (a + mps x) \end{aligned}$$

# Weak inversion is not easy!

- What is a weak inverse for *sum*?  $sum^\circ y = [y]$

$$\begin{aligned} sum [] &= 0 \\ sum (a : x) &= a + sum x \end{aligned}$$

- What is it for *mps*?  $mps^\circ y = [y]$

$$\begin{aligned} mps [] &= 0 \\ mps (a : x) &= 0 \uparrow a \uparrow (a + mps x) \end{aligned}$$

# Weak inversion is not easy!

- What is a weak inverse for *sum*?  $\underline{\text{sum}^\circ y = [y]}$

$$\begin{aligned}\text{sum } [] &= 0 \\ \text{sum } (a : x) &= a + \text{sum } x\end{aligned}$$

- What is it for *mps*?  $\underline{\text{mps}^\circ y = [y]}$

$$\begin{aligned}\text{mps } [] &= 0 \\ \text{mps } (a : x) &= 0 \uparrow a \uparrow (a + \text{mps } x)\end{aligned}$$

- What is it for  $f = \text{mps} \triangle \text{sum}$ ?

$$f\ x = (\text{mps } x, \text{sum } x)$$

# Weak inversion is not easy!

- What is a weak inverse for *sum*?  $sum^\circ y = [y]$

$$\begin{aligned} sum [] &= 0 \\ sum (a : x) &= a + sum x \end{aligned}$$

- What is it for *mps*?  $mps^\circ y = [y]$

$$\begin{aligned} mps [] &= 0 \\ mps (a : x) &= 0 \uparrow a \uparrow (a + mps x) \end{aligned}$$

- What is it for  $f = mps \triangle sum$ ?  $f^\circ (p, s) = [p, s - p]$

$$f x = (mps x, sum x)$$

# Weak inversion is challenging

Can you find a weak inverse for  $f$ ?

$$f\ x = (mss\ x, mps\ x, mts\ x, sum\ x)$$

where

$$mss\ [] = 0$$

$$mss\ (a : x) = (a + mps\ x) \uparrow mss\ x \uparrow 0$$

$$mts\ [] = 0$$

$$mts\ (a : x) = (a + sum\ x) \uparrow mts\ x \uparrow 0$$

# Weak inversion is challenging

Can you find a weak inverse for  $f$ ?

$$f\ x = (mss\ x, mps\ x, mts\ x, sum\ x)$$

where

$$mss\ [] = 0$$

$$mss\ (a : x) = (a + mps\ x) \uparrow mss\ x \uparrow 0$$

$$mts\ [] = 0$$

$$mts\ (a : x) = (a + sum\ x) \uparrow mts\ x \uparrow 0$$

$$\underline{f^\circ\ (m, p, t, s) = [p, s - p - t, m, t - m]}$$

# Derivation of Weak Right Inverse

- Idea:

deriving a weak right inverse



solving conditional linear equations



# Derivation of Weak Right Inverse

- Idea:

deriving a weak right inverse



solving conditional linear equations

- Consider to find a weak right inverse for  $f$  defined by

$$f\ x = (mps\ x, sum\ x)$$

Let  $x_1, x_2$  be a solution to the following equations:

$$\begin{aligned}mps\ [x_1, x_2] &= p \\sum\ [x_1, x_2] &= s\end{aligned}$$

then

$$f^\circ (p, s) = [x_1, x_2]$$

# Derivation of Weak Right Inverse

- Idea:

deriving a weak right inverse



solving conditional linear equations

- Consider to find a weak right inverse for  $f$  defined by

$$f\ x = (mps\ x, sum\ x)$$

Let  $x_1, x_2$  be a solution to the following equations:

$$\begin{array}{rcl} 0 \uparrow x_1 \uparrow (x_1 + x_2) & = & p \\ x_1 + x_2 & = & s \end{array}$$

then

$$f^\circ (p, s) = [x_1, x_2]$$

# Derivation of Weak Right Inverse

- Idea:

deriving a weak right inverse



solving conditional linear equations

- Consider to find a weak right inverse for  $f$  defined by

$$f\ x = (mps\ x, sum\ x)$$

Let  $x_1, x_2$  be a solution to the following equations:

$$x_1 = p$$

$$x_2 = s - p$$

then

$$f^\circ (p, s) = [x_1, x_2]$$

# Derivation of Weak Right Inverse

- Idea:

deriving a weak right inverse



solving conditional linear equations

- Consider to find a weak right inverse for  $f$  defined by

$$f\ x = (mps\ x, sum\ x)$$

Let  $x_1, x_2$  be a solution to the following equations:

$$x_1 = p$$

$$x_2 = s - p$$

then

$$f^\circ (p, s) = [p, s - p]$$

# Conditional Linear Equations

$$\begin{aligned}t_1(x_1, x_2, \dots, x_m) &= c_1 \\t_2(x_1, x_2, \dots, x_m) &= c_2 \\&\vdots \\t_m(x_1, x_2, \dots, x_m) &= c_m\end{aligned}$$

# Conditional Linear Equations

$$\begin{aligned}t_1(x_1, x_2, \dots, x_m) &= c_1 \\t_2(x_1, x_2, \dots, x_m) &= c_2 \\&\vdots \\t_m(x_1, x_2, \dots, x_m) &= c_m\end{aligned}$$

$$t ::= n \mid x \mid n \times \mid t_1 + t_2 \mid p \rightarrow t_1; t_2$$

$$p ::= t_1 < t_2 \mid t_1 = t_2 \mid \neg p \mid p_1 \wedge p_2 \mid p_1 \vee p_2$$

# Conditional Linear Equations

$$\begin{aligned}t_1(x_1, x_2, \dots, x_m) &= c_1 \\t_2(x_1, x_2, \dots, x_m) &= c_2 \\&\vdots \\t_m(x_1, x_2, \dots, x_m) &= c_m\end{aligned}$$

Conditional linear equations can be efficiently solved by using  
Mathematica. [PLDI'07]

# Can we generalize the idea from lists to trees?

$$\begin{array}{lcl}
 f(a : x) & = & a \oplus f x \\
 f(x ++ [b]) & = & f x \otimes b \\
 \hline
 f(x ++ y) & = & f x \odot f y \\
 \text{where } a \odot b & = & f(f^\circ a ++ f^\circ b)
 \end{array}$$



$f$  is a bottom-up tree reduction

$f$  is a top-down tree reduction

$$\begin{array}{lcl}
 f(t_1 \triangleleft t_2) & = & f t_1 \odot f t_2 \\
 \text{where } a \odot b & = & f(f^\circ a \triangleleft f^\circ b)
 \end{array}$$



# Can we generalize the idea from lists to trees?

$$\begin{array}{lcl}
 f(a : x) & = & a \oplus f x \\
 f(x ++ [b]) & = & f x \otimes b \\
 \hline
 f(x ++ y) & = & f x \odot f y \\
 \text{where } a \odot b & = & f(f^\circ a ++ f^\circ b)
 \end{array}$$



$f$  is a bottom-up tree reduction

$f$  is a top-down tree reduction

$$\begin{array}{lcl}
 f(t_1 \triangleleft t_2) & = & f t_1 \odot f t_2 \\
 \text{where } a \odot b & = & f(f^\circ a \triangleleft f^\circ b)
 \end{array}$$

Yes, see POPL'09.

# Can we generalize the idea from lists to trees?

$$\begin{array}{lcl}
 f(a : x) & = & a \oplus f x \\
 f(x ++ [b]) & = & f x \otimes b \\
 \hline
 f(x ++ y) & = & f x \odot f y \\
 \text{where } a \odot b & = & f(f^\circ a ++ f^\circ b)
 \end{array}$$



$f$  is a bottom-up tree reduction

$f$  is a top-down tree reduction

$$\begin{array}{lcl}
 f(t_1 \triangleleft t_2) & = & f t_1 \odot f t_2 \\
 \text{where } a \odot b & = & f(f^\circ a \triangleleft f^\circ b)
 \end{array}$$

Yes, see POPL'09. In fact, all the ideas in this course can be naturally generated to trees.