計算機プログラムの数理
-- 数式運算によるプログラミング --

胡 振江
東京大学情報理工学系研究科

2002年7月1日

Programming

Given a problem

Write a correct and efficient program
to solve the problem

Top-down Programming

A big problem

Simpler problem1, …, Simpler problem n

Solve simpler problems

Combine to form a final program

A Running Example

- Maximum Segment Sum
  Find the maximum of the sums of all segments

  mss [31, -41, 59, 26, -53, 58, 97, -93, -23, 84] =187

- Top-down programming
  - `segs` : find all the segments
  - `sums` : compute the sums
  - `max` : find the maximum

Program Outline

```
/* Program-1 */
main(){
    int x[]= ...
    int n= ...
      ...
    /* segs */
    /* sums */
    /* max */
      ...
}
```

A Limitation

How to design an O(n) program for mss

– Topdown programming gives a concise but usually inefficient program
– It is complicated for programmers to make it efficient. (usually ask help from a compiler)

Calculational Programming

Calculational Programming

A concise algorithm

using operators suitable for program transformation

Apply calclationnal rules to improve the algorithm

Map the efficient algorithm to concrete program

Theory of Lists: BMF

- Lists (Arrays)
  - [ ] empty list
  - x:xs is a list by inserting element x to a list xs

Example: 1 : (2 : (3 : []))

Abbreviation

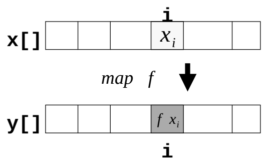
$$x_1 : (x_2 : (..., (x_n : []))) = [x_1, x_2, ..., x_n]$$

Loops and Laws

- Basic loop patterns over lists
  - *map*
  - *fold*s
  - *scan*
- Calculational laws: General
  - Promotion
  - Fusion
  - Honer's rule

Map (1)

$$map\ \ f\ [x_0, x_1, \cdots, x_{n-1}] = [f\ x_0,\ f\ x_1, \cdots,\ f\ x_{n-1}]$$




```
for(i=0;i<n;i++)
    y[i]= f(x[i]);
```

Map (2)

- Definition

  map f [] = []

  map f (x:xs) = f x : map f xs

- Exampes

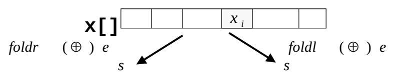
  map double [1,2,3,4] = [1,4,6,8]

  map (+1) [1,2,3,4] = [2,3,4,5]

  map (1:) [[1,2],[3,4],[5,6,7]] = [[1,1,2],[1,3,4],[1,5,6,7]]

Folds

$$foldr\,(\oplus)\,e\,[x_0, x_1, \cdots, x_{n-1}] = x_0 \oplus (x_1 \oplus (\cdots (x_{n-1} \oplus e)))$$
$$foldl\,(\oplus)\,e\,[x_0, x_1, \cdots, x_{n-1}] = (((e \oplus x_0) \oplus x_1) \cdots) \oplus x_{n-1}$$




```
s=e;
for(i=n-1;i>=0;i--)
    s = x[i] oplus s;
```

```
s=e;
for(i=0;i<n;i++)
    s = s oplus x[i];
```

Folds: Examples

- Definition

  foldr (⊕) e [] = e

  foldr (⊕) e (x:xs) = x ⊕ foldr (⊕) e xs

- Examples

  sum = foldr (+) 0

  product = foldr (*) 1

  maximum = foldr (max) (-∞)

Duality of *folds*

For associative operator ⊕ with unit e

$$foldr\ (\oplus)\ e = foldl\ (\oplus)\ e$$

concat

$$concat = foldr\,(+\!\!+)\,[\,] = foldl\,(+\!\!+)\,[\,]$$

```
Concat [[1,2],[3],[],[4,5]] = [1,2,3,4,5]
```

Scan

- Accumulate folding results

$$scanl(\oplus)\ e\,[x_0, x_1, \cdots, x_{n-1}]$$
$$= [e, e\oplus x_0, (e\oplus x_0)\oplus x_1, \cdots, (((e\oplus x_0)\oplus x_1)\cdots)\oplus x_{n-1}]$$

```
O(n)
```

```
Scanl (*) 1 [1,2,3,4] = [1,1!,2!,3!,4!]
```

inits

$$inits\,[x_0, x_1, \cdots, x_{n-1}] =$$
$$[[\,], [x_0], [x_0, x_1], \cdots, [x_0, x_1, \cdots, x_{n-1}]]$$

```
inits = scanl (⊕) [[]]
  where xs ⊕ x = xs ++ [last xs ++ [x]]
```

Segments

tails

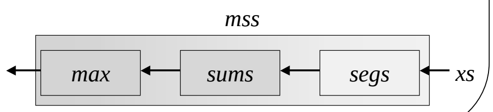
$$tails\,[x_0, x_1, \cdots, x_{n-1}] =$$
$$[[x_0, x_1, \cdots, x_{n-1}], \cdots, [x_{n-2}, x_{n-1}], [x_{n-1}], [\,]]$$

segs

$$segs = concat \bullet map\ tails \bullet inits$$

Function Composition

$$mss\ xs = max(sums(segs\ xs))$$
$$\Leftrightarrow$$
$$mss = max \bullet sums \bullet segs$$



Calculation Laws

- General Laws
  - Promotion
  - Fusion
- Specific Laws
  - Horner's law
  - ...

A Simple Law

$$map\ f \bullet map\ g = map\ (f \bullet g)$$

$$\begin{aligned}
&(map\ f \bullet map\ g)\,[x_0, x_1, \cdots, x_{n-1}] \\
&= map\ f\ (map\ g\,[x_0, x_1, \cdots, x_{n-1}]) \\
&= map\ f\,[g\ x_0, g\ x_1, \cdots, g\ x_{n-1}] \\
&= [\,f\,(g\ x_0),\ f\,(g\ x_1), \cdots,\ f\,(g\ x_{n-1})] \\
&= [(\,f \bullet g\,)x_0, (\,f \bullet g)x_1, \cdots, (\,f \bullet g)\ x_{n-1}] \\
&= map\ (\,f \bullet g\,)\,[x_0, x_1, \cdots, x_{n-1}]
\end{aligned}$$

Promotion Laws

Operations on a compound structure can be ***promoted*** into its components:

• *concat* gives a list of lists

• operation on *concat*ed lists into components

```
map f . Concat => ?
foldr (⊕) e . Concat => ?
```

Two Promotion Laws

map promotion law

$$map\ f \bullet concat = concat \bullet map\ (map\ f\,)$$

fold promotion

$$foldl\ \ (\oplus)\ e \bullet concat\ \ = \ \ foldl\ \ (\oplus)\ e \bullet map\ \ (\ foldl\ \ (\oplus)\ e\,)$$

Fusion Laws

Successive operations on a list can be ***fused*** into a single operation.

fold-map fusion

$$\frac{a \otimes x = a \oplus f\ x}{foldl\ (\oplus)\ e \bullet map\ f\ =\ foldl\ (\otimes)\ e}$$

fold -scan **fusion**

$$\begin{aligned}
&foldl\ (\oplus)\ e \bullet scanl\ (\otimes)\ d = \\
&\qquad fst \bullet\ foldl\ (*)\ (e \oplus d, d) \\
&\text{where } (u, v) * x = (u \oplus w, w),\ w = v \otimes x
\end{aligned}$$

$$fst\ (x, y) = x$$
$$snd\ (x, y) = y$$

Horner's Rule

Calculation of polynomials of order n by

n additions and n multiplication

$$a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = ((a_n x + a_{n-1})x + \cdots + a_1)x + a_0$$

More General Form

$$x_0 \times x_1 \times x_2 + x_1 \times x_2 + x_2 + 1$$
$$= ((1 \times x_0 + 1) \times x_1 + 1) \times x_2 + 1$$

$$\mathit{foldl}\ (+)\ 0\ (\mathit{map}\ (\ \mathit{foldl}\ (\times)\ 1)\ (\mathit{tails}\ [x_0, x_1, x_2]))$$
$$= \mathit{foldl}\ (\oplus)\ 1\ [x_0, x_1, x_2]$$
$$\mathit{where} \quad a \oplus b = a \times b + 1$$

Generalized Horner's Rule

For operators $\oplus$ and $\otimes$ satisfying

$$(x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)$$

with left - unit e of $\oplus$, i.e., $e \oplus x = x$,

$$\mathit{foldl}\ (\oplus)\ e \bullet \mathit{map}\ (\ \mathit{foldl}\ (\otimes)\ d\ ) \bullet \mathit{tails} = \mathit{foldl}\ (*)\ d$$

where

$$x * y = (x \otimes y) \oplus d$$

Maximum Segment Sum

$$\mathit{mss} = \{\text{Def.}\mathit{mss}\}$$
$$\mathit{max} \bullet \mathit{map}\ \mathit{sum} \bullet \underline{\mathit{segs}}$$
$$= \{\text{Def.}\mathit{segs}\}$$
$$\mathit{max} \bullet \underline{\mathit{map}\ \mathit{sum} \bullet \mathit{concat}} \bullet \mathit{map}\ \mathit{tails} \bullet \mathit{inits}$$
$$= \{\mathit{map}\ \text{ promotion}\}$$
$$\underline{\mathit{max}} \bullet \mathit{concat} \bullet \mathit{map}\ (\mathit{map}\ \mathit{sum}) \bullet \mathit{map}\ \mathit{tails} \bullet \mathit{inits}$$
$$=$$

Maximum Segment Sum (Cont.)

$$\underline{\mathit{max}} \bullet \mathit{concat} \bullet \mathit{map}\ (\mathit{map}\ \mathit{sum}) \bullet \mathit{map}\ \mathit{tails} \bullet \mathit{inits}$$
$$= \{\text{Def.}\mathit{max}\}$$
$$\underline{\mathit{foldl}\ (\cup)\ (-\infty) \bullet \mathit{concat}} \bullet \cdots$$
$$= \{\ \mathit{fold}\ \text{ promotion}\}$$
$$\mathit{max} \bullet \underline{\mathit{map}\ \mathit{max} \bullet \mathit{map}\ (\mathit{map}\ \mathit{sum}) \bullet \mathit{map}\ \mathit{tails}} \bullet \mathit{inits}$$
$$= \{\mathit{map}\ \text{ distribution}\}$$
$$\mathit{max} \bullet \mathit{map}\ (\underline{\mathit{max}} \bullet \mathit{map}\ \underline{\mathit{sum}} \bullet \mathit{tails}) \bullet \mathit{inits}$$
$$=$$

Maximum Segment Sum (Cont.)

$$\mathit{max} \bullet \mathit{map}\ (\underline{\mathit{max}} \bullet \mathit{map}\ \underline{\mathit{sum}} \bullet \mathit{tails}) \bullet \mathit{inits}$$
$$= \{\text{Def.}\ \mathit{max} \text{ and } \mathit{sum}\}$$
$$\mathit{max} \bullet \mathit{map}\ (\ \underline{\mathit{foldl}\ (\cup)\ (-\infty) \bullet \mathit{map}\ (\ \mathit{foldl}\ (+)\ 0) \bullet \mathit{tails}})$$
$$\bullet\ \mathit{inits}$$
$$= \{\text{Horner's rule}: x * y = (x + y) \cup 0\}$$
$$\mathit{max} \bullet \underline{\mathit{map}\ (\ \mathit{foldl}\ (*)\ 0) \bullet \mathit{inits}}$$
$$= \{\mathit{scan}\ \text{ lemma}\}$$
$$\underline{\mathit{max}} \bullet \mathit{scanl}\ (*)\ 0$$
$$=$$

Maximum Segment Sum (Cont.)

$$
\begin{aligned}
& \underline{max} \bullet scanl\,(*)\,0 \\
= \; & \{\text{Def.} \mathit{max}\} \\
& \underline{foldl\,(\cup)\,(-\infty) \bullet scanl\,(*)\,0} \\
= \; & \{\, \textit{fold-scan} \text{ fusion}: \\
& \qquad (u,v) \div x = (u \cup w, w),\; w = (v+x) \cup 0\} \\
& fst \bullet foldl\,(\div)\,(\underline{-\infty \cup 0},0) \\
= \; & \{-\infty \cup 0 = 0\} \\
& fst \bullet foldl\,(\div)\,(0,0)
\end{aligned}
$$

Final Program *mss*

$$
\begin{aligned}
& mss\,[x_0, x_1, \cdots, x_{n-1}] \\
= \; & (fst \bullet foldl\,(\div)\,(0,0))\,[x_0, x_1, \cdots, x_{n-1}] \\
= \; & fst((((0,0) \div x_0) \div x_1) \div \cdots) \div x_{n-1})
\end{aligned}
$$

Pseudo-C code

```
(mss,s)=(0,0);
for(i=0;i<n;i++)
    (mss,s) oslash= x[i];
```

A Linear-time Program in C

```
/* Program-5 */
main(){
    int x[]= ...
    int n= ...
    int mss=0;
    int s=0;
    for(i=0;i<n;i++){
        s += x[i];
        if(s<0)
            s=0;
        if(mss<s)
            mss= s;
    }
}
```

$\underline{O(n)}$ operations

Report 2

Please solve Problem 5 to Problem 8, and send your report by email to

hu@ipl.t.u-tokyo.ac.jp

no later than

July 31st, 2002.

The subject of your email should be

MSP #2.