構成的アルゴリズム論
Left Reductions

胡 振江

東京大学 計数工学科

2008 年 2 月 2 日

A Problem

Min-Max Problem

Given is a list of lists of numbers. Required is an efficient algorithm for computing the minimum of the maximum numbers in each list. More succinctly, we want to compute

$$\mathit{minimax} \;=\; \downarrow/ \cdot \uparrow/*$$

as efficiently as possible.

Left Reduction

$$\oplus\!\not\rightarrow_e [x_1, x_2, \ldots, x_n] = (((e \oplus x_1) \oplus x_2) \oplus \cdots) \oplus x_n$$

In the monoid view of lists, the formal definition of $\oplus\!\not\rightarrow_e$ is as follows.

$$\begin{aligned}
\oplus\!\not\rightarrow_e [\,] &= e \\
\oplus\!\not\rightarrow_e [a] &= e \oplus a \\
\oplus\!\not\rightarrow_e (x \mathbin{+\!\!+} y) &= \oplus\!\not\rightarrow_{e'} y \ \textbf{where}\ e' = \oplus\!\not\rightarrow_e x
\end{aligned}$$

$\oplus\not\rightarrow_e$: A Well-Defined Function

There is an instructive alternative way of seeing that $\oplus\not\rightarrow_e$ is well-defined. Define h by

$$\begin{array}{lcl} h\ [] & = & id \\ h\ [a] & = & (\oplus a) \\ h\ (x +\!\!+\, y) & = & h\ y \cdot h\ x \end{array}$$

Obviously, h is a homomorphism from $([a], +\!\!+, [])$ to $(\beta \rightarrow \beta, \cdot, id_\beta)$. Now we have

$$\oplus\not\rightarrow_e x = h\ x\ e$$

and so $\oplus\not\rightarrow_e$ is well-defined.

Left Reduction is Important

Every set of equations of the following form

$$\begin{array}{lcl} f\ [] & = & e \\ f\ (x +\!\!\!+ [a]) & = & F(a, x, f\ x) \end{array}$$

can be defined in terms of a left reduction:

$$f = \pi_2 \cdot \oplus \not\rightarrow_{e'}$$

where

$$\begin{array}{lcl} e' & = & ([], e) \\ (x, u) \oplus a & = & (x +\!\!\!+ [a], F(a, x, u)) \end{array}$$

Specialization Lemma

Every homomorphism on lists can be expressed as a left (or also a right) reduction. More precisely,

$$
\begin{aligned}
\oplus/ \cdot f* \;&=\; \odot \not\rightarrow_e \\
\text{where}\quad & \\
e \;&=\; id_{\oplus} \\
a \odot b \;&=\; a \oplus f\ b
\end{aligned}
$$

Exercise: Prove the specialization lemma.

Left Reductions and Loops

A left reduction $\oplus \not\rightarrow_e x$ can be translated into the following program in a conventional *imperative* language.

```
|[ var r;
   r := e;
   for b in x
      do r := r oplus b;
   return r
]|
```

Left Zeros

Left reductions require that the argument list be traversed in its entirety. Such a traversal can be cut short if we recognize the possibility that an operator may have *left-zeros*.

Definition: Left Zero

ω is a left-zero of $\oplus$ if for all a the following holds.

$$\omega \oplus a = \omega$$

Exercise: Prove that if ω is a left-zero of $\oplus$ then

$$\oplus \not\rightarrow_{\omega} x = \omega$$

for all x. (by induction on snoc list x.)

Implementation of Left Reduction with Left-zero Check

From the fact that $\oplus\!\not\rightarrow_{e}(x +\!\!+\, y) = \oplus\!\not\rightarrow_{(\oplus\not\rightarrow_{e} x)}\; y$, we have the following program for left-reduction.

```
|[ var r;
   r := e;
   for b in x while not left-zero(r)
      do r := r oplus b;
   return r
]|
```

Minimax

Let us return to the problem of computing

$$minimax \;=\; \downarrow/ \cdot \uparrow/ *$$

efficiently. Using the specialization lemma, we can write

$$minimax \;=\; \odot \not\to_{\infty}$$

where ∞ is the identity element of $\downarrow$, and

$$a \odot x \;=\; a \downarrow (\uparrow/x).$$

Exercise: Prove that $-\infty$ is the left-zero of $\odot$.

Since $\downarrow$ distributes through $\uparrow$ we have

$$a \odot x \;=\; a \downarrow (\uparrow / x) \;=\; \uparrow / (a \downarrow) * x$$

Using the specialization lemma a second time, we have

$$\begin{aligned} a \odot x \;&=\; \oplus_a \not\rightarrow_{-\infty} x \\ \text{where } b \oplus_a c \;&=\; b \uparrow (a \downarrow c) \end{aligned}$$

Exercise: Prove that a is the left-zero of $\oplus_a$.

An Efficient Implementation of `minimax xs`

```
|[ var a; a := infinity;
   for x in xs while a <> -infinity
      do a := a odot x;
   return a
]|
```

where the assignment `a := a odot x` can be implemented by the loop:

```
|[ var b; b := -infinity;
   for c in x while c <> a
      do b := b max (a min c);
   a := b
]|
```

About the Final Examination

- 時間：2月9日 8:30-10:00
- 場所：6号館63号室
- 教科書、講義資料など持ち込み可
- 成績：全体の70%

ご協力ありがとうございました。