TẠP CHÍ KHOA HỌC TÍNH TOÁN VÀ ĐIỀU KHIỂN TẬP VI (1990), Số 3 (27-33)

MỘT PHƯƠNG PHÁP XÂY DỰNG HỆ SOẠN THẢO VÀ PHÂN TÍCH CÔNG THỨC TOÁN HỌC

TRẦN ĐÌNH KHANG, HOÀNG KIẾM

MỞ ĐẦU

Sự phát triển trong lĩnh vực tin học đem lại cho máy tính những khả năng mới mà một trong những hướng nghiên cứu trong thời gian gần đây là xây dựng các hệ trợ giúp giáo dục và đào tạo, bao gồm việc soạn thảo các bài toán một cách tự nhiên cũng như lập luận, tính toán thông minh.

Đối tượng sử dụng các hệ này rất rộng rãi, từ học sinh phổ thông tới sinh viên đại học, từ các kỹ sư, nhân viên kỹ thuật tới các nhà nghiên cứu. Khi xử lý một biểu thức toán họic phức tạp, người sử dụng thay vì viết một chương trình tính toán, có thể soạn thảo công thức đó vào máy, và ngay lập tức nhận được kết quả. Người ta còn có thể nhận được các chỉ dẫn để giải các bài toán, cũng như vẽ đồ thị, giải phương trình, thống kê, xấp xỉ hàm,... Có thể kể ra đây tên một số sản phẩm làm theo hướng này như DERIVE, MATHCAD, với những khả năng tính toán, giải phương trình, vẽ đồ thị 2 - 3 chiều, tính đạo hàm, tích phân,...

Ở các hệ trên, việc soạn thảo công thức phải tuân theo một khuôn mẫu nhất định. Bài báo này sẽ trình bày một phương pháp soạn thảo công thức tự nhiên hơn làm công cụ cho nhiều ứng dụng tiếp theo.

Phần đầu tiên của bài giới thiệu phương pháp soạn thảo công thức toán học, tiếp theo trình bày thuật toán chuyển đổi công thức soạn thảo về biểu thức dạng thủ tục và phần cuối cùng là một vài suy nghĩ và hướng phát triển của hệ.

1. PHƯƠNG PHÁP SOẠN THẢO CÔNG THỨC TRÊN MÁY TÍNH

Với 256 ký tự mã ASCII cho chế độ màn hình văn bản hạn chế rất nhiều cho việc soạn thảo các công thức toán học, trong khi đó kỹ thuật đồ họa trên máy tính ngày càng phát triển. Cách soạn thảo dưới đây sẽ sử dụng chế độ đồ họa để thể hiện công thức trên màn hình.

Một công thức có thể chia thành các ký hiệu và các xâu ký tự. Có các ký hiệu như dấu sigma, căn thức, tích phân, đạo hàm, đường thẳng, mở đóng ngoặc,... Các ký hiệu này có thể được định nghĩa bởi một cửa sổ hình chữ nhật cho phép người sử dụng vẽ nó theo các kích cỡ khác nhau.

Các ký hiệu lưu dưới dạng xâu "$\backslash n, x_1, y_1, x_2, y_2$", trong đó

$n = 2$ cho đường thẳng

$n = 3$ cho căn thức ($\surd$)

$n = 4$ cho Sigma ($\sum$)

$n = 5$ cho tích phân ($\int$), ...

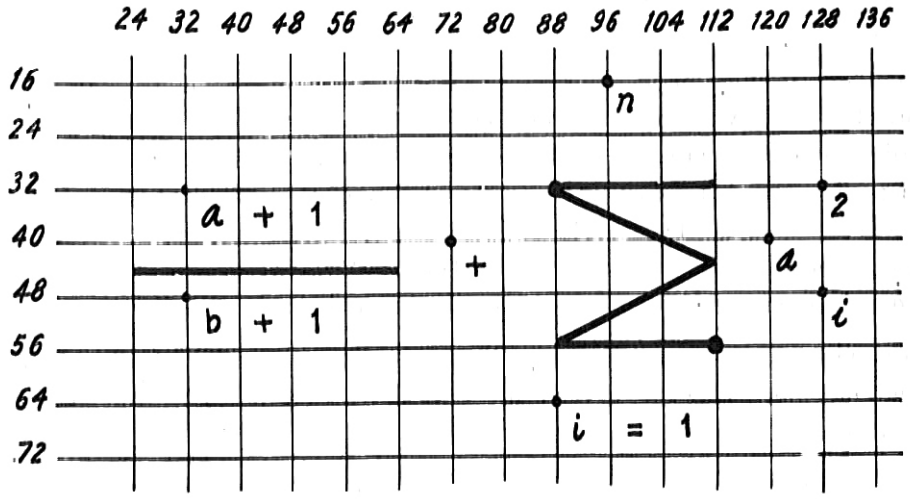
với (x_1, y_1) là tọa độ điểm trên trái và (x_2, y_2) là tọa độ điểm dưới phải của cửa sổ chứa ký hiệu.

Các xâu ký tự được lưu trong " \ 1, x_1, y_1, text" trong đó (x_1, y_1) là tọa độ điểm trên trái của xâu text.

Ví dụ. Biểu thức

$$\frac{a+1}{b+1} + \sum_{i=1}^{n} a_i^2 \tag{1.1}$$

có thể được biểu diễn trên màn hình



và lưu trong xâu

" \ 1, 32, 32, a + 1 \ 2, 24, 44, 64, 44, \1, 32, 48, b + 1 \ 1, 72, 40, +
\4, 88, 32, 112, 56 \ 1, 88, 64, i = 1 \ 1, 96, 16, n \ 1, 120, 40, a (1.2)
\1, 128, 32, 2 \ 1, 128, 48, i"

Lưu ý rằng ở ví dụ này, biểu thức được soạn thảo khá ngay ngắn trong khi các cửa sổ có thể xê dịch ở các vị trí và kích cỡ khác nhau.

Cách biểu diễn này giúp cho việc soạn thảo công thức đẹp, hiển thị nhanh trên màn hình và có thể xóa sửa dễ dàng. Để vẽ một ký hiệu, trước hết phải định nghĩa cửa sổ chứa nó, rồi vẽ ký hiệu đó lên màn hình và nối xâu biểu diễn ký hiệu vào xâu biểu thức. Để xóa, sửa, dịch chuyển các ký hiệu và xâu ký tự, cần phải đưa con trỏ về vị trí nhận biết chúng rồi thực hiện các chức năng tương ứng trên màn hình và trong xâu biểu thức. Tương tự như vậy, có thể thực hiện các phép toán sao chép, phóng to, thu nhỏ,...

Sau đây là thủ tục minh họa bằng ngôn ngữ C vẽ một dấu sigma trong cửa sổ (x_1, y_1, x_2, y_2) đã được định nghĩa:

```
draw – sigma (int x1, int y1, int x2, int y2)
{
        move to (x2, y1);
        line to (x1, y1);
        line to (x2, (y1 + y2)/2);
        line to (x1, y2);
        line to (x2, y2);
}
```

Hạn chế của phương pháp soạn thảo này là những khó khăn trong việc nhận dạng công thức, đưa về biểu thức dạng thủ tục mà ta sẽ bàn đến ở phần 2.

2. TỰ ĐỘNG PHÂN TÍCH VỀ MỘT BIỂU THỨC DẠNG THỦ TỤC

Ta hãy quay lại với ví dụ ở phần 1, vấn đề đặt ra là làm cách nào để biến đổi xâu (1.2) thành một biểu thức kiểu như

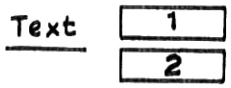
"(a + 1)/(b + 1) + sigma (i = 1_n, a_i^2)" (2.1)

để thực hiện các phép tính toán, rút gọn tiếp theo.

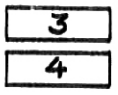
Những thông tin chủ yếu có thể dựa vào là:

a. *Tọa độ của các ký hiệu và xâu ký tự*

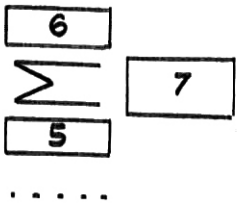
Ví dụ một xâu sẽ nằm trong dấu căn nếu như cửa sổ giới hạn của nó nằm trong cửa sổ của căn thức. Ta tạm qui ước sự phụ thuộc của các ký hiệu bằng các giá trị sau



Các xâu nằm trong vùng 1 có thể là chỉ số trên hoặc mũ của text, các xâu nằm trong vùng 2 là chỉ số dưới.



Các xâu nằm trong vùng 3 sẽ thuộc về số bị chia và vùng 4 thuộc về số chia.



Các xâu trong vùng 5 và 6 là biến chạy, trong vùng 7 là biểu thức của sigma

Tuy nhiên, việc xác định các cửa sổ phụ thuộc sẽ gặp nhiều khó khăn nếu không xét đến các yếu tố tiếp. Ví dụ rất khó xác định cận phải của các vùng 1, 2, 7, cận trên của 1, 3, 6 và cận dưới của 2, 4, 5,... hoặc trong xâu

$$a_j b_j \tag{2.2}$$

"i" thuộc về chỉ số dưới của "a" trong khi sự phụ thuộc của "b" là chỉ số trên của "i" lại không đúng.

b. Trục ngang của biểu thức

Mỗi xâu biểu thức đều nằm trên một trục ngang, qua đó có thể xác định cận phải của các vùng phụ thuộc. Ví dụ ở xâu (2.2), cửa sổ chỉ số dưới của xâu "a" không thể vượt quá cận trái của "b" được.

Hoặc trong xâu

$$\frac{\frac{a+1}{b+1}+c}{d} \tag{2.3}$$

vùng "số chia" của phép chia "$a + 1$" cho "$b + 1$" không thể vượt quá cận trên của đường thẳng dài hơn thể hiện phép chia của toàn bộ biểu thức nằm trên cho "d".

c. Cú pháp của biểu thức dạng thủ tục

Đó là các tính chất đặc biệt như tổng số ngoặc mở phải bằng tổng số ngoặc đóng, hoặc một tính phân sẽ kết thúc bằng "d" như "dx", "dy", "dt", "$du1$", ...

d. Tính liên thông của các ký hiệu và xâu ký tự

Gọi P là ma trận phụ thuộc của các phần tử trong một biểu thức, thì P_{ij} sẽ là giá trị của sự phụ thuộc xâu thứ j vào xâu thứ i. $P_{ij} = 0$ có nghĩa là không có sự phụ thuộc nào.

Có thể biểu diễn tính liên thông như sau:

$$\text{Nếu } P_{ij} \neq 0 \text{ và } P_{jk} \neq 0 \text{ thì } P_{ik} = P_{ij} \tag{2.4}$$

Ở trong xâu (2.2) có

$P(\text{"}a\text{"}, \text{"}i\text{"}) = 2$
$P(\text{"}i\text{"}, \text{"}b\text{"}) = 1$
$P(\text{"}b\text{"}, \text{"}j\text{"}) = 2$
$P(\text{"}a\text{"}, \text{"}b\text{"}) = 0$
$P(\text{"}i\text{"}, \text{"}j\text{"}) = 0$ ...

như vậy $P(\text{"}a\text{"}, \text{"}i\text{"}) \neq 0$, $P(\text{"}i\text{"}, \text{"}b\text{"}) \neq 0$, trong khi $P(\text{"}a\text{"}, \text{"}b\text{"}) = 0 \neq 2 = P(\text{"}a\text{"}, \text{"}i\text{"})$ không thỏa mãn tính liên thông. Nguyên nhân do $P(\text{"}i\text{"}, \text{"}b\text{"}) = 1$ là không đúng, cần phải sửa lại = 0.

e. Các yếu tố khác

Có những cách viết thông dụng như $\log_a b$ là một hàm số hoặc $\sin^2 x$ phải chuyển về dạng "$\sin(x)\text{^}2$",...

Từ những phân tích trên, đã có thể xây dựng một ma trận phụ thuộc cho biểu thức (1.1):

		a+1	-	b+1	+	Σ	i=1	n	a	i	2
1	a+1										
2	___	3		4							
3	b+1										
4	+										
5	Σ						5	6	7	7	7
6	i=1										
7	n										
8	a									2	1
9	i										
10	2										

(2.5)

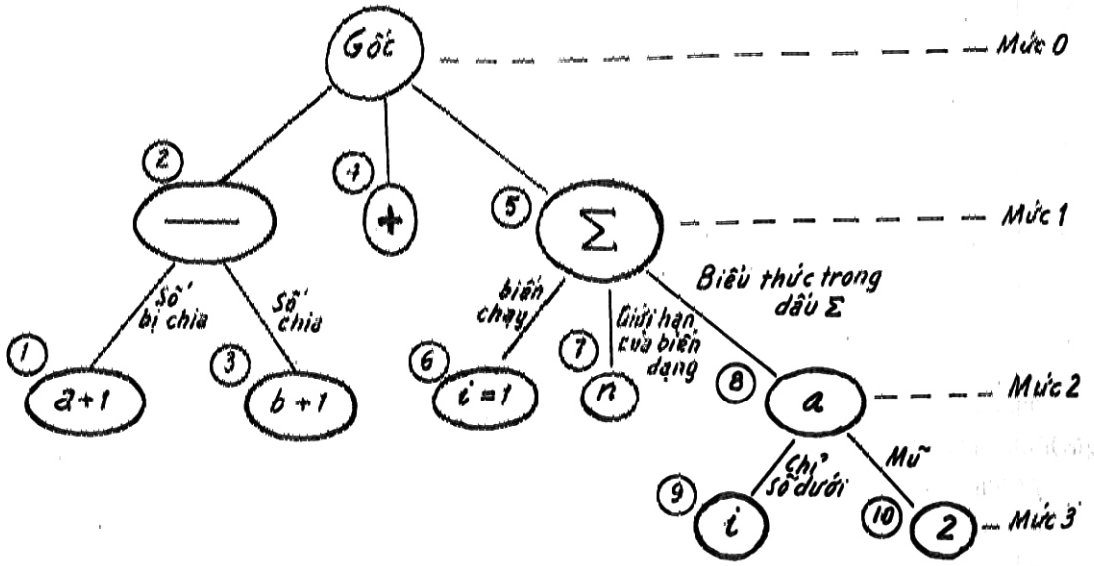
Thuật toán tính xâu biểu thức kết quả từ ma trận phụ thuộc có thể được mô tả theo ngôn ngữ hình thức như sau:

/* Thủ tục (2.6) */
$k = 1$ /* Đặt các giá trị ban đầu */
$A^1 = \{1, 2, ..., n\}$ /* với n là số các phần tử */
RES = "A^1"
While ($k > 0$) {
For ($i = 1$; $i \leq k$; $i++$) {
Từ tập A^i tính ra xâu kết quả tạm thời RESULT^i
/* Sẽ được trình bày rõ hơn ở dưới */
Thay thế xâu RESULT^i vào A^i trong RES
} /* kết thúc vòng lặp For. */
$k = 0$
Tính xem trong RES còn có tập con nào không để nâng giá trị của k và gán vào A^k
} /* Kết thúc vòng lặp While */
RES là xâu kết quả.
/* Kết thúc thủ tục (2.6) */

Tiếp theo là thủ tục mô tả quá trình tính toán một xâu kết quả tạm thời RESULT^i từ tập con A^i:

/* Thủ tục (2.7) */
Chia A^i thành các tập con $B_1, ..., B_m$ không phụ thuộc nhau ($\forall i_1 \in B_i$, $j_1 \in B_j \Rightarrow P_{i_1 j_1} = 0$)
For ($j = 1$; $j \leq m$; $j++$) {
Trong B_j tìm phần tử PIVOT có nhiều phần tử khác phụ thuộc vào nhất
Khôi phục xâu kết quả tương ứng Result$_j$ dựa vào kiểu của phần tử PIVOT
} /* Kết thúc vòng lặp For */
Sắp xếp các xâu Result$_1$,..., Result$_m$ theo tọa độ từ trái sang phải để kết nối lại thành xâu RESULT^i
/* Kết thúc thủ tục (2.7) */

Về bản chất, thủ tục (2.6) và (2.7) gộp lại thành một bài toán đệ qui tính toán trên một cây biểu thức. Ví dụ từ ma trận phụ thuộc (2.5) có cây biểu thức sau:



Sau mỗi lần thực hiện các lệnh trong vòng lặp While của thủ tục (2.6), cây biểu thức sẽ được phân tích sâu hơn một mức:

Giá trị ban đầu: RES = "A^{1n}", $k = 1$, $A^1 = \{1,2,3,4,5,6,7,8,9,10\}$

Mức 1: RES = "$A^1 + A^{2n}$", $k = 2$, $A^1 = \{1,2,3\}$, $A^2 = \{5,6,7,8,9,10\}$

Mức 2: RES = "$(a + 1)/(b + 1)$ + sigma (i = 1_n, A^1)", $k = 1$, $A^1 = \{8,9,10\}$

Mức 3: RES = "$(a + 1)/(b + 1)$ + sigma (i = 1_n, a_i-2)", $k = 0$

3. KẾT LUẬN

Phương pháp được trình bày ở phần 2 có thể chia làm hai công đoạn chính là xây dựng ma trận phụ thuộc và khôi phục xâu biểu thức, trong đó việc tính xâu kết quả đã được mô phỏng bằng một thuật toán chính xác. Vì vậy độ tin cậy của lời giải sẽ được xác định dựa trên ma trận phụ thuộc, tức là phụ thuộc chủ yếu vào tọa độ của các xâu ký tự và ký hiệu, hay nói đúng hơn là phụ thuộc vào quá trình soạn thảo.

Có thể coi đây là một phần của bài toán nhận dạng công thức toán học, trong đó phần còn lại của bài toán này là biến đổi một trang ảnh được quét vào từ thiết bị đọc thành xâu dạng soạn thảo được trình bày ở phần 1.

Từ đó có thể mở ra những hướng ứng dụng mới trong quá trình tự động phân tích giải bài toán hoặc trợ giúp tính toán thông minh, giải bài toán theo vết và khám phá ra những công thức mới.

Nhận ngày 24-4-1990