数理解析研究所講究録
第612巻 1987年 90-93

数式処理と数学記号のあいまい性

電子技術総合研究所　元吉文男　(Fumio Motoyoshi)

数式処理システムのプログラムを書いていると、厳密であるはずの数学記号のなかには、習慣や文脈によって異なって解釈されるものが出現する。ここではそのうちからいくつかを紹介するとともに、我々の所で開発中の数学コンサルタントシステムＧＡＬＯＩＳにおいて、どのように解決する予定であるかを述べる。

記法のあいまい性

一般に数学記号を書く場合に、あいまい性が残らない範囲で余分な括弧などを省略するが、このなかには習慣あるいは文脈によって、さらには読者の常識と推論を期待して書かれるものがあり、ある決まった手続きによるものではない場合がある。数式処理システムが結果の出力を行う場合になるべく自然な形で行おうとすると、この問題が生じてくる。その例をいくつか上げてみる。

三角関数を書く場合には、一般の関数とは異なりその引数は普通は括弧でくくらない。このためにあいまい性が生じることがある。

```
sin xy
```

と書けばこれは

```
sin(xy)
```

のことである。しかし

```
sin x sin y
```

と書けばこれは

```
sin(x) sin(y)
```

のことだと思う人が多い。それでは

```
sin xf(y)
```

と書いたらどうであろうか。これは

```
sin(xf(y))
```

とも

```
sin(x)f(y)
```

とも解釈できる。このようなあいまい性を生じる場合に人間は、括弧を入れるよりも

f(y) s i n x

と書くのではないだろうか。計算機にこのような処理をさせるには、いままでのような単純な手続きではうまく実現できない。

また添字付きの変数を書く場合にもあいまい性がある。たとえば行列Aの要素を

$a_{i,j}$

などと書くが、この記法では

a(1, 1)

と

a(1 1)

が共に

a_{11}

となってしまう。

現在の数式処理システムにおいては、このようなあいまい性の生じることのないように、括弧やカンマを挿入してこの問題を避けているが、人間とのインターフェースをよくするには、人間が書くのと同じ程度に括弧やカンマを省略するのが望ましい。この問題はつきつめて考えると、文脈や人間の常識・推論までも考慮しなければならず、人工知能の問題になってくる。しかもこの機能は必須のものではないので、ここに力を入れる余裕がないのが現状である。

以上で述べた問題は記法の問題であり、それほど本質的ではないが、以下で述べることは実際の処理にかかわってくる問題である。

関数と関数値

関数と関数値は混乱して使用されており、また逆にそのことをうまく利用しているものもある。

たとえば、次の式があるとする。

f(x)＝x＋a

これは、fという関数を定義しているのか、fのxにおける値が x＋a であるのかがわからない。この2つは同じことのようであるが、異なる意味を持つ場合もある。いまの例では

f(a)＝2a

となるが、これはfの定義と解釈するわけにはいかない。このように同じ形の式がある場合には関数の定義を表わし、別の場合には特定の引数値に対する関数値を表わす、というように2通りに使いわけられている。

ＧＡＬＯＩＳでは、この混乱を避けるために、関数定義の場合には次のようにラムダ式で行うことにする予定である：

f＝λ(x)(x＋a)

このようにすると、副次効果として関数f自体をデータとして扱え

H(f)

のように汎関数の扱いが容易になる。

ここで簡単にラムダ式を使用した場合の計算方法を紹介する。最初に束縛変数と自由変数の説明をする。変数ｘは式λ〔ｘ〕ｆ中で束縛されているという。またｘがｇ中で束縛されていればｇを含む式中でも束縛されている。それ以外の変数は自由変数という。ラムダ計算の原則は

(λ〔ｘ〕ｆ)(ａ)

という式はｆの中のｘという自由変数にａを代入したものと等しいとするものである。ただし、このときにｘとａは次の条件を満たしていなければならない：

・ａの中の自由変数が代入によって束縛されてはならない。

以上の話は引数が１個の場合であったが、これはほとんどそのままｎ個の場合に拡張できる。

数式処理においてラムダ式をもとに関数を処理しようとすると、式を与えてその中の変数を引数とする関数を作る機能が必要になる。その変数をｘとすると、これは一見

λ〔ｅ〕(λ〔ｘ〕ｅ)

とすればよさそうであるが、これは上に述べた条件に反している（ｅの中にｘがある場合には代入ができない）。そこで本体の式の構造を考えずに（ラムダ式を認識せずに）代入を行う機能が必要となる。これをλ'で表わすことにする。すると前の式は

λ'〔ｅ〕(λ〔ｘ〕ｅ)

と書ける。さらに式と変数を引数として、その変数を引数とする関数を値とする関数は

λ'〔ｅ,ｘ〕(λ〔ｘ〕ｅ)

と書けることになる。

逆に関数から式を作ることは簡単で、その関数をｆとすれば

ｆ(ｘ)

がｘにおけるｆの表わす式となる。

<u>微分</u>

微分についてもあいまいな点がある。次の例をみてみよう。

$f(x)=x^2+2a^2$

でｆが定義されるものとする（これは前節で述べたように便宜的な記法である）。このとき次の式を考える：

$f'(a)$, $\frac{df(a)}{da}$, $\frac{df}{da}(a)$

この場合多くの人はこれらの値はそれぞれ

２ａ , ６ａ , ４ａ

とするのではなかろうか。

そこで微分にも関数を関数にマップするものと、式を式にマップするものとの２種類を考えることにする。そうして

$$\frac{dx^2}{dx}=2x$$

という式の意味は、x^2という式を２ｘという式にマップすることであり、関数から関数へのマップではないとする。

我々は、関数から関数へのマップを基本演算として採用し、これをもとに他の演算を導くことにした。具体的には、与えられた関数の第ｉ引数で偏微分した関数を値とする汎関数を基本にする。これを∂_iと書くことにすると、上のｄ／ｄｘという演算は前節のλを使用することにとって、

$$\frac{d}{dx} = \lambda'(e)((\partial_1(\lambda(x)e))(x))$$

となる。なお∂_1はよく使用されるので∂と書くことにする。この式にａを引数として与えると

$$\frac{da}{dx} = (\partial(\lambda(x)a))(x)$$

となる。

またｆとｇを関数としたときに

$(f+g)(x) = f(x)+g(x)$

$(fg)(x) = f(x)g(x)$

$(f \cdot g)(x) = f(g(x))$,

とすると、微分の公式は

$\partial(f+g) = (\partial f)+(\partial g)$

$\partial(fg) = (\partial f)g+f(\partial g)$

$\partial(f \cdot g) = ((\partial f) \cdot g)(\partial g)$

と書くことができる。

この∂を使用すると式が長くなるが、あいまい性がなくなるので、数式処理のように完全に機械的な処理ではかえってすっきりとするのではないだろうか。