mao処理系

佐藤晴郎

2003.2.17

今回、mao の開発にあたり、W.Kernighan・R.Pike[?] に記載されている hoc (high-order calculator の略) を参考に作りはじめることとなった。hoc とは名前が表す通り、電卓の機能をかなり拡張したものでプログラミング言語インタプリタである。

2003 年 2 月現在、W.Kernighan・R.Pike[?] に掲載されている hoc よりも新しいバージョンのものが以下のウェブサイトからダウンロードできる。

`http://cm.bell-labs.com/cm/cs/upe/index.html`

1 mao記述言語、必要とするライブラリ

mao は C 言語、Yacc を用いて書かれている。丸めの制御に Bias/Profil ライブラリ (詳細は第 2 節) 、入力作業の効率化のために GNU Readline ライブラリ (詳細は付録**??**) を使用している。

以上まとめると mao のコンパイルに必要なものは以下のものである。

- C 言語コンパイラ
- Bias/Profil ライブラリ
- GNU Readline ライブラリ
- Yacc もしくは Bison (付録**??** 参照)
- make (これは必ずしも必要ではない)

Bias/Profil ライブラリは様々なシステムの丸めの制御を可能であるが IEEE754 の丸め制御も可能である。このことから mao (mao の本質的な部分) は IEEE754 に準拠したコンピュータを前提に作ることにした。そのようなコンピュータであれば、mao を移植することは容易であろう。2003 年 2 月現在までで動作実績のある OS 環境は Windows2000 Professional,Solaris,Linux である。

2 Bias/Profil

Bias/Profil は Olaf Knüppel により開発された。Profil とは通常必要とされる区間演算および実数演算を提供する C++のライブラリである。Profil は Bias の関数を参照しており、Bias は区間演算および実数演算を提供する C 言語のライブラリである。

参考までに 2003 年 2 月現在の公式ウェブページを記述しておく。

`http://www.ti3.tu-harburg.de/Software/PROFILEnglisch.html`

3 Readline ライブラリ

mao では入力作業を効率化するため、GNU Readline ライブラリをもちいており、同ツールの機能を利用することが出来る。GNU Readline ライブラリの機能については、簡単ではあるが付録**??**に記述しておくので、参照してほしい。

4 値

mao では値を主に機械区間として扱う。機械区間は二つの倍精度浮動小数点数をもつ。ここからしばらく、区間の間を “,” で区切るところを “@” で表す。これは mao の言語の仕様である。先ほど、機械区間として扱うと書いたが、$[a, b]$ が機械区間であるとは

$$[a@b] \quad (a \leq b, a \in \mathrm{F}, b \in \mathrm{F})$$

が成り立つことである。ただし、点区間は次のように略して書く。

$$[a@a] = a \quad (a \in \mathrm{F})$$

$a, b \in \mathrm{R}$ でかつ $a, b \notin \mathrm{F}$ を定数として扱おうとした時、それは

$$[\vee a@ \wedge b] \quad (\vee \text{は下向きの丸め}, \wedge \text{は上向きの丸め})$$

と自動的に変換され機械区間として扱われる。

5 区間のチェック

区間を直接入力する際はその入力が正しいかのチェックを行う。このチェックに使用される条件は区間の定義を拡張したものであり、以下に示していく。

まず $[a,b]$ が機械区間であるとは

$$[a@b] \quad (a \leq b, a \in \mathrm{F}, b \in \mathrm{F})$$

である。$a,b \in \mathrm{F}$ という条件は実はかなり限定された条件になっていて応用しがたいものである[1]。そこで mao では $a,b \in \mathrm{IF}$ とし、条件を緩めることにした。また $a \leq b$ の条件の代わりに以下のような条件をつけ、入力の際にその条件との適合確認を行うことにした。

$$a \leq b \quad (a,b \in \mathrm{IF})$$

(区間演算の $\leq$ については第**??**節に記載。)

もし、条件を満たさない場合にはその旨を表示され、ユーザーがどのようにするかを選択することが出来る。しかし、この確認では時として正しい入力も条件を満たさないとされてしまう。

理解が容易になるよう、ここからは具体的に例をあげて考えてみる。整数しか表現できず、小数点以下の入力があった場合その数を含む機械区間 (要素が整数) としてその数を表すシステムが存在したとする。このシステム上で 1.5 は [1@2] とシステム内部で表現される。また 1.6 も [1@2] とシステム内部で表現される。それではこのシステム上で区間の入力の際、その要素を小数で入力してみよう。例えば [1.5@1.6] の入力はまず 1.5 が [1@2](= a と表すことにする) 、1.6 が a とシステム内部で表現され、その二つの値を用い、区間として表現することとなる。[1.6@1.5] の入力は本来、区間の定義に合わない形である。この入力は入力エラーとすべきであろう。しかしこの入力も 1.6 が a 、1.5 が a とシステム内部で表現され、それから二つの値を用い、区間として表現することとなる。すると上の正しい入力の場合と判別をする事ができないのである。

そこで mao ではそのような入力があった際はユーザーに確認を求めることとしたのである。この mao 内のチェック作業は設定を変更することにより、ユーザが制御することが出来る。しかし基本設定では常にチェック作業を行い、条件が満たされなければユーザーに確認を求めるという設定にしてある。

入力エラーが起こらない場合を一つ目に二つ目に入力エラーになる極めて特別な例を以下に示す。これは実際の mao の入出力例である。

```
> [ 1 @ 2 ]
```

[1] もし、この条件であるのならば a,b には定数の入力しかできないこととなり、数式または変数などを扱うことが出来ず (数式、変数の値が点区間になり、数と同値として扱うことが出来る場合もあるが、ほとんどの場合がそうはならない。) やはり非常に不便なのである。

```
[ 1.00000000000000 × 10^{0} @ 2.00000000000000 × 10^{0} ]

> [ 1/3 @ 1/3]
WARNING----------
 (前) [3.3333333333333314830e-01 @ 3.3333333333333370341e-01]
 ? ( <= ) ?
 (後) [3.3333333333333314830e-01 @ 3.3333333333333370341e-01]
 区間の内部の大小関係（区間計算上における）が
 正しいと言い切れません。
-----------------
[3.3333333333333314830e-01 @ 3.3333333333333370341e-01]
として計算を続行しますが、よろしいですか？Yes=1 No=0 1
[ 3.33333333333333 × 10^{-1} @ 3.33333333333334 × 10^{-1} ]
```

本来は同じ式であるので、条件の確認をパスしなければならないだろう。しかし、現段階の mao では次のように処理され、チェック内の条件が満たされないとなる。この入力はまず 1/3 という数ではなく、1/3 という数式に分別される。数式として処理されると 1/3 は二進数で正確に表現できないことから機械区間として表現される。つまり言い方をかえれば、1/3 という入力は 1/3 に極めて近いぼやけた値が入力されたということになる。機械区間として表現されてしまっている以上、本当の値がその機械区間の中の何処にあるかは限定できない。点区間ではない同じ区間を比べるのであるから条件をみたすことができない。

6 mao の構成ファイル

mao は以下のファイルで構成されており、以下にそのファイルがどのようなファイルであり、またどのようなことが書かれているかを記す。

- mao.y: mao の本体となる部分。字句解析、構文解析などを担当する。
- mao.h: mao のヘッダーファイル
- code.h: code 関係のヘッダーファイル。 (code とは mao インタプリタが呼び出す命令である。)
- code0.c: mao のプログラム構築など本質的な部分にかかわる code
- code1.c: 画面への出力 code
- code2.c: 丸めに関する code
- code3.c: 基本的な演算子 (四則演算等) の計算部分。code4 で利用する関数
- code4.c: 基本的な演算子 (四則演算等) の code
- code5.c: 不等式、包含関係に関する code
- code6.c: if、while、for の制御 code
- code7.c: 行列の初期化、基本的な演算 特別な行列の作成の code

- code8.c: 行列の要素に対する操作 code
- code9.c: LU 分解、連立方程式等の code
- code10.c: 既存ファイルからの読み込み、設定に関する code
- ftos.c: printf 文で丸め誤差を出さないために値を文字に変換している。
- math.c: 標準の数値演算関数のエラーからの復帰を担当する。
- init.c: 組み込み関数、組み込み定数、予約語等を定義する。
- myget.c: getc 関数、ungetc 関数に追ずる関数
- symbol.c: 記号表の作成、記号表からの引用を担当する。
- makefile: コンパイル作業を軽減するためのファイル

7 丸めの制御

mao は区間演算をサポートするように開発されている。そのため、丸めの制御は必要不可欠である。具体的な mao での丸めの制御方法は Bias の以下の 3 つの関数を呼び出している。

- `_BiasRoundUp()`: 丸めを上方向に設定する。
- `_BiasRoundDown()`: 丸めを下方向に設定する。
- `_BiasRoundNear()`: 丸めを近似に設定する。

上の関数は呼び出すと、その後の丸めの方法が常にそれに設定される。

8 数の入力

mao では数の入力時に丸め誤差が発生しないように特別な処理をしてある。処理というのは、まず数を整数部と小数部に分け、各々を整数として扱う。小数部の場合は最上桁より一つ上の桁に 1 を加える。その後、C 言語の scanf 関数を呼び出し、数に変換する。

その後、丸めを制御しながら整数部と小数部を足す (小数部の最上桁の 1 はこの時、除く)。

これらのプログラムは mao.y の字句解析部、code0.c に書かれている。

9　数の出力

Bias/Profil を用いても C 言語の printf 文内の丸めの制御はできない。そこでその部分についてはこちらで記述することにした。実際、mao 内でどのように処理されているかというと、数の丸めを考慮しながら、数を文字列に変換する。その文字列を C 言語の printf 文で出力するという形である。

このプログラムは ftos.c に書かれている。

10　仮想コンピュータ

mao の処理系は一種の仮想コンピュータ (スタックマシン) になっている。mao は入力があるとそれに対するプログラムを作る。そしてその命令を配列 (配列名 prog) として記憶し入力の区切りになった時点でプログラムを実行する。命令の配列のほかに具体的なデータをためておくスタックを用意してあり、その値を操作することで、結果を得る。

11　mao のプログラムのポイント

ここからは mao 処理系のプログラムでポイントとなる部分を幾つか、あげておく。プログラムのソースを解読するための参考として読み進めていただきたい。

11.1　スタックの操作

スタックの操作は以下の三つのコードで行われる。

- push：スタックにデータをプッシュする。
- pop：スタックの一番上の値をポップする。返し値として Datum 型を返す。
- popreno：スタックの一番上の値をポップする。返し値はなし。

11.2　スタックマシンとしての実行

スタックマシンとしての実行は execute という関数の中で行われる。execute は STOP との命令があると execute のループを抜ける。またその際に呼び出す命令が配列 (prog) に書かれた code 命令である

11.3 記号

変数名などの記号は Symbol リストの中の next フィールドによってリスト状につながっている。それをアクセスする手段のなかに lookup 関数と install 関数というものがある。lookup 関数はそのリストを頭から順に検索をかけ、一致すれば Symbol のポインを返す。一方 install は Symbol リストの先頭に変数名等をおく働きがある。最初の段階で与えられる記号表は init.c に書かれている。

11.4 bltin

bltin は組み込み関数であるが Bias の関数を呼び出している。Bias の関数の呼び出しのため、すこし凝った型変換をしている。

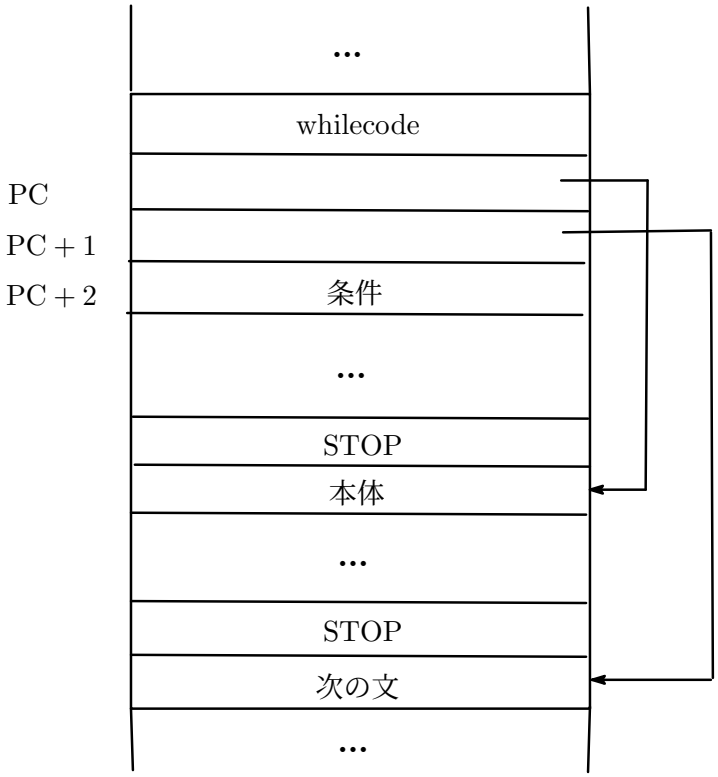
11.5 制御フロー

mao では if、if-else、while、for の制御フローを用いることが出来る。本節では具体的に while を例にとって話を進めていく。

キーワード while にであうと whilecode が prog 上に積まれスタックマシン上にその whilecode 命令の位置を積むこととなる。その時、同時にそれに続く二つの場所も確保される。そこには後で値が入ることとなる。次に生成されるコードは条件部にあたる部分で cond がそれに値する。mao での while 文の構文解析は次のようになっている。

```
| while '(' cond ')' stmt end {
        ($1)[1] = (Inst)$5; /* body of loop */
        ($1)[2] = (Inst)$6; } /* end, if cond fails */
```

この具体的な説明は省略するが、prog は次のような状態になる。



この様な状態になっている prog について、execute を複数回、呼び出すことでループ文は成り立っている。