プログラムのページ

担当　鈴木　誠道

7207 LISP による算術式のコンパイル

原田賢一（慶応義塾大学）

はじめに

算術式をコンパイルする技法の1つに，算術式を木構造で表現し，その木構造を走査していって最適なオブジェクト　コードを生成する方法がある[1,2,3]．このような方法は，LISP のもつ再帰的呼出しとリスト処理の機能を利用すると，簡単に記述することができる．以下に，簡単な算術式および算術代入文をコンパイルする LISP プログラムについて説明する．

1. 算術式の表現形式

コンパイルすべき算術式または算術代入文（⟨exp-tree⟩）は，LISP で S-式として直接扱えるように完全かっこ式で表現する．その形式はつぎのとおりである．

⟨exp-tree⟩::=(⟨identifier⟩=⟨exp⟩)|⟨exp⟩
⟨exp⟩::=⟨identifier⟩|(−⟨exp⟩)|(⟨exp⟩⟨op⟩⟨exp⟩)
⟨op⟩::=+|−|*|/
⟨identifier⟩::=⟨op⟩ 以外の原始記号

2. オブジェクト計算機

オブジェクト計算機としては，1個の演算レジスタ（Acc で表わす）をもち，つぎのような1番地方式の命令をもったものを仮定する．

命　令		意　味
LDA	a	Acc ← C(a)
ADD	a	Acc ← C(Acc)+C(a)
SUB	a	Acc ← C(Acc)−C(a)
MLT	a	Acc ← C(Acc)*C(a)
DIV	a	Acc ← C(Acc)/C(a)
NEG		Acc ← −C(Acc)
STO	a	a ← C(Acc)

C(x) は x の内容を意味する．

3. オブジェクト　コード生成のアルゴリズム

演算子+，*に交換律が成り立つことだけを考慮して，中間結果の退避回復の命令が少なくなるようなオブジェクト　コードの生成法を用いる．

一般に，α を根とした木 **T** を考える．α の左側の節を β_l，右側の節を β_r とする．β_l，β_r を根とする

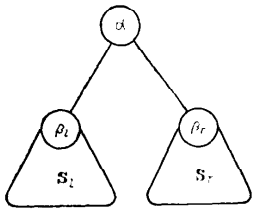

図1　木 **T**

部分木をそれぞれ $\mathbf{S}_l$，$\mathbf{S}_r$ とする（図1）．$\mathbf{S}_l$，$\mathbf{S}_r$ は演算子 α の両オペランドに対応している．α が単項負符号のときは，$\mathbf{S}_r$ はないものとする．**T** に対するオブジェクト　コードを生成し，それを値とする関数 G[**T**] はつぎのように定義される．

T について，以下のステップを 1, 2, 3, …の順に実行してゆき，**T** が i 番目の性質を満たしていれば，i 番目のステップの矢印の右側に示したものが，G[**T**] の値である．

1. **T** が部分木をもたない → (LDA α)
2. α が単項負符号 → G[$\mathbf{S}_l$]⊕(NEG)
   ⊕はリストの結合操作を表わす．
3. α が '=' → G[$\mathbf{S}_r$]⊕(STO β_l)
4. β_r が葉 → G[$\mathbf{S}_l$]⊕(H[α] β_r)
   関数 H[α] は演算子 α に対応する命令コードをとり出すものである．
5. (β_l が葉)∧(α が '+' または '*') →
   G($\mathbf{S}_r$)⊕(H[α] β_l)
6. (β_l が葉)∧(α が '−') →
   G[$\mathbf{S}_r$]⊕(NEG)⊕(ADD β_l)

7. 以上の条件を満足しない →
G[S_r]⊕(STO temp)⊕G[S_l]⊕(H[α] temp)
temp は中間結果の一時退避場所を表わす.

4. プログラムと実行結果

プログラムの作成にはTOSBAC-3400/30のKLISPシステムを使用した[4,5]. 図2にプログラムと評価式を示す. おもな関数の機能はつぎのとおりである.

COMPILE〔s〕: sは1で述べた形式で表現された算術式である. 退避場所の記号を生成するためのカウンタ TSGC をクリアし, CODEGEN〔s〕によって sのオブジェクト コードを生成する.

CODEGEN〔s〕: 2で述べたアルゴリズムによって, 算術式 s をコンパイルする.

CODEGEN1〔s; a; b〕: 算術式 s のオブジェクト コードのうしろに命令 OBJC〔a; b〕をつけ加える.

OBJC〔a; b〕: 1つの命令 (H〔a〕 b) を生成する.

GETTS〔 〕: 計算結果の一時退避場所を表わす記号 '*'n (n は整数) を1つ生成する.

図3に実行結果を示す. 図2に示した評価式を全部処理するのに要した正味の CPU 時間は 660 ms であった. プログラムおよび評価式を主記憶装置に格納するのに要した LISP のセルは 523 個であった.

おわりに, このプログラムを作成するにあたり多大の協力をしてくださった慶応義塾大学 中西正和さんに深く感謝します.

```
*
*         CODE GENERATION OF THE SIMPLE ARITHMETIC EXPRESSION
*
*
COMPILE[S]= PROG[ [TSGC]; TSGC:=0; RETURN[CODEGEN[S]] ];
*
* ------------------------------------------------------------------------
*
*       OBJECT CODE GENERATOR
*
CODEGEN[S]= [
*
    ATOM[S]->OBJC['==';S];
*
    EQUAL[CAR[S];'-']->CODEGEN1[RAND[S];'NEG';NIL];
*
    EQ[OP[S];'=']->CODEGEN1[RAND2[S];OP[S];RAND1[S]];
*
    ATOM[RAND2[S]]->CODEGEN1[RAND1[S];OP[S];RAND2[S]];
*
    AND[ATOM[RAND1[S]];MEMBER[OP[S];'(+ *)']]->
                CODEGEN1[RAND2[S];OP[S];RAND1[S]];
*
    AND[ATOM[RAND1[S]];EQ[OP[S];'-']]->
                CODEGEN1[LIST['=';RAND2[S]];'+';RAND1[S]];
*
    T->PROG[ [TS];
          TS:=GETTS[];
          RETURN[ NCONC[CODEGEN[LIST[TS;'=';RAND2[S]]];
                        CODEGEN[LIST[RAND1[S];OP[S];TS]]] ) ];
*
* ********** ************************************************************
*
*       THE AUXILIARY FUNCTIONS FOR CODEGEN
*
    OP[X]= CADR[X];
    RAND[X]= CADDR[X];
    RAND1[X]= CAR[X];
    RAND2[X]= CADDR[X];
*
    CODEGEN1[S;OP;RAND]= NCONC[CODEGEN[S];OBJC[OP;RAND]];
*
    OBJC[OP;RAND]= LIST[[NULL[RAND]->LIST[OP];T->LIST[MNOPC[OP];RAND]]];
          MNOPC[X]= CDR[SASSOC[X;'((+.ADD)(-.SUB)(*.MLT)(/.DIV)
                                   (=.STO)(==.LDA))';NIL]];
*
    GETTS[]= PROG[ []; TSGC:=ADD1[TSGC]; CLEARBUFF[]; PACK['*'];
                   PACK[TSGC]; RETURN[INTERN[MKNAM[]]] ].
*
*
COMPILE[(A = (B + C))];
COMPILE[(A = (B * (- C)))];
COMPILE[(A = (B - (C - (D * E))))];
COMPILE[(((B / C) + ((- D) * E)) - (F / (G * H)))];
TRACE[(CODEGEN)];
COMPILE[(A = ((B - C) / (D * E)))].
```

図 2 プログラムと評価式

参考文献

1) R. Sethi and J. D. Ullman: The generation of optimal code for arithmetic expression. J. ACM 17, 4 (Oct. 1970), 715-728.
2) I. Nakata: On compling algorithms for arithmetic expressions. Comm. ACM 10, 8 (Aug. 1967), 492-494.
3) Anderson, J. P.: A note on some compiling algorithms. Comm. ACM 7, 3 (March 1964), 149-150.
4) 中西正和: KLISP 説明書 改訂版. 慶応義塾大学情報科学研究所, 1970 年7月.
5) 中西正和: KLISP の拡張機能とその応用. 情報処理. 11, 10 (Oct. 1970), 619-623.

(昭和 47 年 5 月 26 日受付)

```
 FUNCTION EVALQUOTE HAS BEEN ENTERED, ARGUMENTS..
COMPILE
((A = (B + C)))

 END OF EVALQUOTE, VALUE IS..
((LDA B) (ADD C) (STO A))


 FUNCTION EVALQUOTE HAS BEEN ENTERED, ARGUMENTS..
COMPILE
((A = (B * (- C))))

 END OF EVALQUOTE, VALUE IS..
((LDA C) (NEG) (MLT B) (STO A))


 FUNCTION EVALQUOTE HAS BEEN ENTERED, ARGUMENTS..
COMPILE
((A = (B - (C - (D * E)))))

 END OF EVALQUOTE, VALUE IS..
((LDA D) (MLT E) (NEG) (ADD C) (NEG) (ADD B) (STO A))

 FUNCTION EVALQUOTE HAS BEEN ENTERED, ARGUMENTS..
COMPILE
((((B / C) + ((- D) * E)) - (F / (G + H))))

 END OF EVALQUOTE, VALUE IS..
((LDA G) (ADD H) (STO *2) (LDA F) (DIV *2) (STO *1) (LDA D)
     (NEG) (MLT E) (STO *3) (LDA B) (DIV C) (ADD *3) (SUB
    *1))


 FUNCTION EVALQUOTE HAS BEEN ENTERED, ARGUMENTS..
TRACE
((CODEGEN))

 END OF EVALQUOTE, VALUE IS..
NIL


 FUNCTION EVALQUOTE HAS BEEN ENTERED, ARGUMENTS..
COMPILE
((A = ((B - C) / (D + E))))

* 0  ARGUMENTS OF CODEGEN
(A = ((B - C) / (D + E)))

* 1  ARGUMENTS OF CODEGEN
((B - C) / (D + E))

* 2  ARGUMENTS OF CODEGEN
(*1 = (D + E))

* 3  ARGUMENTS OF CODEGEN
(D + E)

* 4  ARGUMENTS OF CODEGEN
D

* 4  VALUE OF CODEGEN
((LDA D))

* 3  VALUE OF CODEGEN
((LDA D) (ADD E))

* 2  VALUE OF CODEGEN
((LDA D) (ADD E) (STO *1))

* 2  ARGUMENTS OF CODEGEN
((B - C) / *1)

* 3  ARGUMENTS OF CODEGEN
(B - C)

* 4  ARGUMENTS OF CODEGEN
B

* 4  VALUE OF CODEGEN
((LDA B))

* 3  VALUE OF CODEGEN
((LDA B) (SUB C))

* 2  VALUE OF CODEGEN
((LDA B) (SUB C) (DIV *1))

* 1  VALUE OF CODEGEN
((LDA D) (ADD E) (STO *1) (LDA B) (SUB C) (DIV *1))

* 0  VALUE OF CODEGEN
((LDA D) (ADD E) (STO *1) (LDA B) (SUB C) (DIV *1) (STO A))

 END OF EVALQUOTE, VALUE IS..
((LDA D) (ADD E) (STO *1) (LDA B) (SUB C) (DIV *1) (STO A))
```

図 3 実行結果