6J-1

コンビネータマシン用 関数型プログラミング言語JCの開発

杉野 順清　　飯田 三郎
(豊橋技術科学大学 情報工学系)

1 はじめに

計算可能なアルゴリズムは，λ式で表現することができる．また，そのλ式はコンビネータという一種の組合せ演算子に変換できる．我々は，このコンビネータ式を直接実行するコンビネータマシンを開発中である．本稿では，このコンビネータマシンのためのプログラミング言語JCの設計方針およびそのコンパイラについて報告する．

2 λ式とコンビネータ

λ式とは関数抽象と関数適用による計算を抽象化したものであり，計算は束縛変数への代入により行なわれる．例えば，自然数 n に2を加算する関数は $(\lambda n.(+2n))$ と表現され，その関数へ3を適用すると $(\lambda n.(+2n))3 \Rightarrow (+23) \Rightarrow 5$ と計算される．

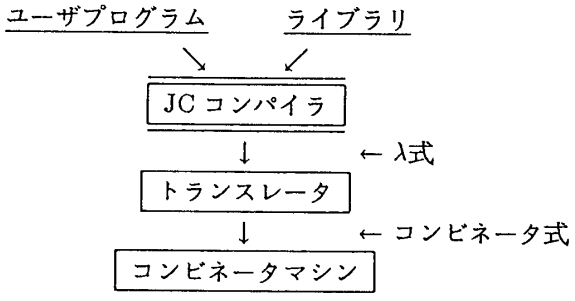
一方，コンビネータとは一種の組合せ演算子であり，各々の簡約規則にしたがって簡約を行なうことにより計算が進む．コンビネータには，一般に以下のような5種類のものが用いられる．

$$\begin{aligned} \mathbf{S}xyz &\Rightarrow xz(yz) \\ \mathbf{K}xy &\Rightarrow x \\ \mathbf{I}x &\Rightarrow x \\ \mathbf{B}xyz &\Rightarrow x(yz) \\ \mathbf{C}xyz &\Rightarrow xzy \end{aligned}$$

λ式は比較的簡単なアルゴリズムでこれらの列に変換することができ，このコンビネータ式を簡約することと元のλ式を実行することとは計算上等しい．

3 システム概要

以下に本システムの概要を示す．



ここで，ユーザプログラムを高級言語とするとλ式は中間言語に相当し，コンビネータ式はマシン語ということもできる．

4 言語の設定

プログラミング言語JCは以下のような特徴を持つ．

- 関数型言語である
  コンビネータ式はλ式と密接な関係を持つため必然的に関数型言語と親和性が高い．
- 強い型付けを持つ
  強い型付けを持つことによりコンパイル時にプログラムの誤りを発見できる可能性が高い．またコンビネータマシン上で動的なデータ型を取扱う必要がなくなるため実行が高速化される．
- グローバル変数を持たない
  最終的な目標は並列コンビネータマシンを開発することにあるので，グローバル変数を持つ言語仕様ではそれに対応できなくなる可能性がある．

5 ストリクト関数とprimitive宣言子

コンビネータ式のみで整数などを表現し，その演算もコンビネータ式で表現することは可能である．しかしそのような表現法では実際の計算量が非常に大きくなるため，数値などに対してはそれを直接取り扱うことができるように拡張するのが一般的である．コンビネータマシン上ではそのような演算は，引数がすべて評価されないと実行できないストリクト関数として表現されている．このような関数はコンビネータマシン上ではマイクロプログラムにより実行されるので，将来拡張される可能性がある．JC上でもこれに柔軟に対応するために，primitive宣言子を用意した．以下にその例を示す．

```
\primitive operator+ "ADDI"
        int->int->int;
\primitive operator- "SUBI"
        int->int->int;
```

int の加算および減算を
定義する primitive 宣言子の例

これにより新しいストリクト関数にもコンパイラ自身を変更することなく対応できる．

Development of a Functional Programming Language JC for a Combinator Machine.
Junsei Sugino (sugino@kiku.tutics.tut.ac.jp), Saburou Iida (iida@kiku.tutics.tut.ac.jp)
Toyohashi University of Technology

6 関数名の overload

JC で用いることができる関数名や演算子名に対してはすべて overload ができる．overload とは異なった型で異なった演算を行なうものに同じ名前を与えることである．例えば上の primitive 宣言に string 型同士の連結を求めるものを加えると下のようになる．

```
\primitive operator+ "CONCAT"
                string->string->string;
```

string の連結の定義

a+b という式は，a b が int 型なら (\ADDI a b) が生成され string 型なら (\CONCAT a b) が生成される．

7 強い型付けと型推論

JC は強い型付けを持つ言語であるが，Pascal のように変数や関数の宣言は必要でない場合が多い．それは，コンパイラ自体に型推論機能があるからである．型推論とはある式の型を決定するためにその式中ですでに分かっている型から式自体の型を推論するもので，主に型同士の単一化により決定される．下の Program 1 では，10 と 20 が整数型であるため a と b も整数型と自動的に推論され，sq は整数を受け取り整数を返す関数と推論される．

8 プログラミング

JC のプログラミングは関数の宣言により行なわれる．例えば以下のような Program 1 においては

```
a = 10;
b = 20;
sq x = x * x;
sq a + sq b.
```

Program 1

a および b は 10 または 20 を返す定数関数で，sq は引数を 1 つとりそれを 2 乗する関数である．答として 10 の 2 乗と 20 の 2 乗の和を計算する．繰り返し計算は再帰関数で行なう．例えばフィボナッチ数の 10 を求めるプログラムは以下のようになる．

```
fib n = take n 10 (fiblist 0 1);
      where
   take n [x:r] = x, n = 0;
                = take (n - 1) r;
   fiblist m n  = [m:fiblist n (m + n)]
      end

fib 10.
```

Program 2

9 翻訳例

以下に Program 1 と Program 2 をλ式およびコンビネータ式に翻訳した結果を示す．

```
(\lambda sq a b . (\ADDI (sq a) (sq b))
  (\lambda x . \MULI x x) 10 20)
(S (B C (B (B B) (B \ADDI))) I
  (S \MULI I) 10 20)
```

Program 1 の翻訳結果

```
((\lambda (n)
  (\fix (\lambda (f n) \unpair
      (\lambda (x r) (\ZEROPINT n) x
       (f (\SUBINT n 1) r)))) n
  ((\fix (\lambda (f m n) \pair m
       (f n (\ADDINT m n)))) 0 1))
 10)
((C (Y (B (B U) (B
      (S (B C (B (B B) \ZEROPINT)))
            (C B (C \SUBINT 1)))))
    (Y (B (S (B B P))
       (C (B B S) \ADDINT))
       0 1)) 10)
```

Program 2 の翻訳結果

ただし Y = S (C B (S I I))(C B (S I I)) である．翻訳結果中に現れる\ADDI や\SUBI は，コンビネータマシン上で用いることができるストリクト関数名を示す．

10 まとめ

- プログラミング言語 JC を用いることにより簡単に関数型プログラミングが可能であり，またコンパイル時にバグを発見できる可能性も高くなった．
- JC が出力するコンビネータ式には評価順序による結果の違いが生じないので並列コンピュータへの応用が期待できる．
- 一方現在の JC には多相型や抽象型が実現されていないのでプログラミングの生産性の向上のためにも将来それらを導入する予定である．

参考文献

[1] D.A.Turner, A New Implementation Technique for Applicative Languages, *SOFTWARE-PRACTICE AND EXPERIENCE, VOL. 9*, 31-49(1979)

[2] D.A.Turner, Miranda: A non-strict functional language with polymorphic types, *LNCS #201*, Springer-Verlag, 1-16(1985)

[3] 中島玲二, 数理科学ライブラリー 3 数理情報学入門, 朝倉書店 (1982)