11. 実習用FORTRANコンパイラCAFT作成上の問題点と効率

慶応義塾大学情報科学研究所
土居範久・大野義夫・山本喜一
慶応義塾大学工学部
大駒誠一・中西正和

1. はじめに

慶応大学では，UNIVAC 1106 システムを使って毎年1,000名をこえる学生を対象とした計算機の入門教育を行なってきたが，既存の RFOR (Reentrant FORTRAN)，FORTRAN V の２つの FORTRAN コンパイラはいずれも次のような点で入門教育には不適当であった。

- 連結編集に時間がかかる
- コンパイル時，連結編集時にディスク，ドラムへのアクセスが頻繁で，ターンアラウンド時間が長い
- コンパイラがリエントラントになっていないために，コアの使用効率が悪い (FORTRAN V)。あるいは，リエントラントになっていても，数フェーズにわかれているために，その効果が減殺される (RFOR)
- 診断メッセージの形式が統一されていない
- 実行時のチェックがほとんど行なわれない
- 必要のない余分の情報（割付番地など）が大量に印刷される

これらの点を解決すべく，新たに FORTRAN コンパイラとして CAFT (the keio CAfeteria oriented reentrant Fortran Translator) を作成した。

2. 方　針

CAFT の設計・作成にあたっては次の点に留意した。

イ．コンパイラの構成に関して

- コンパイルは１パスで行ない，中間結果を外部記憶に書き出すことをしない
- 目的プログラムを保存せず，コンパイル即実行方式とする
- 連結編集を行なわない
- コンパイラをリエントラントにする

ロ．コンパイラの作成に関して

- 作成労力を節約するために，コンパイラを FORTRAN で書く。ただし，コンパイルの速度に重大な影響をもつ部分はアセンブリ言語を使う

・RFOR, FORTRAN V とのファイルの互換性を保ち，作成労力を節約するために，入出力ルーチンは FORTRAN V のもの（現在はリエントラントではない）を用いる

・基本外部関数の計算ルーチンは FORTRAN V のものを用いる

ハ．言語仕様に関して

・JIS FORTRAN 水準 7000 を完全に含む

・RFOR, FORTRAN V との互換性をできるだけ保つ

・入門教育用として適当・必要と思われる拡張を行なう

例：標準書式による入出力　　WRITE (6,*) X,Y

整数・実数の混合した式を許す

添字に任意の整数式を許す

ニ．機能に関して

・トレース機能を用意する

・文の実行回数などのヒストグラム（プロファイル）が作成できるようにする

・実行速度よりもコンパイル速度を重視し，目的プログラムの最適化は最小限にとどめる

・文法チェックをできるだけていねいに行なう

・JIS 7000 に違反した箇所を指摘する

・実行時チェックもできるだけ行なう

・連結編集をしないので，ライブラリの利用はできない。また，他言語との連絡もとれない

ホ．その他

・できるだけ簡単な制御カードで使えるようにする

・余分な情報の出力を抑え，実行結果の印刷もコンパクトにする

3. 構　成

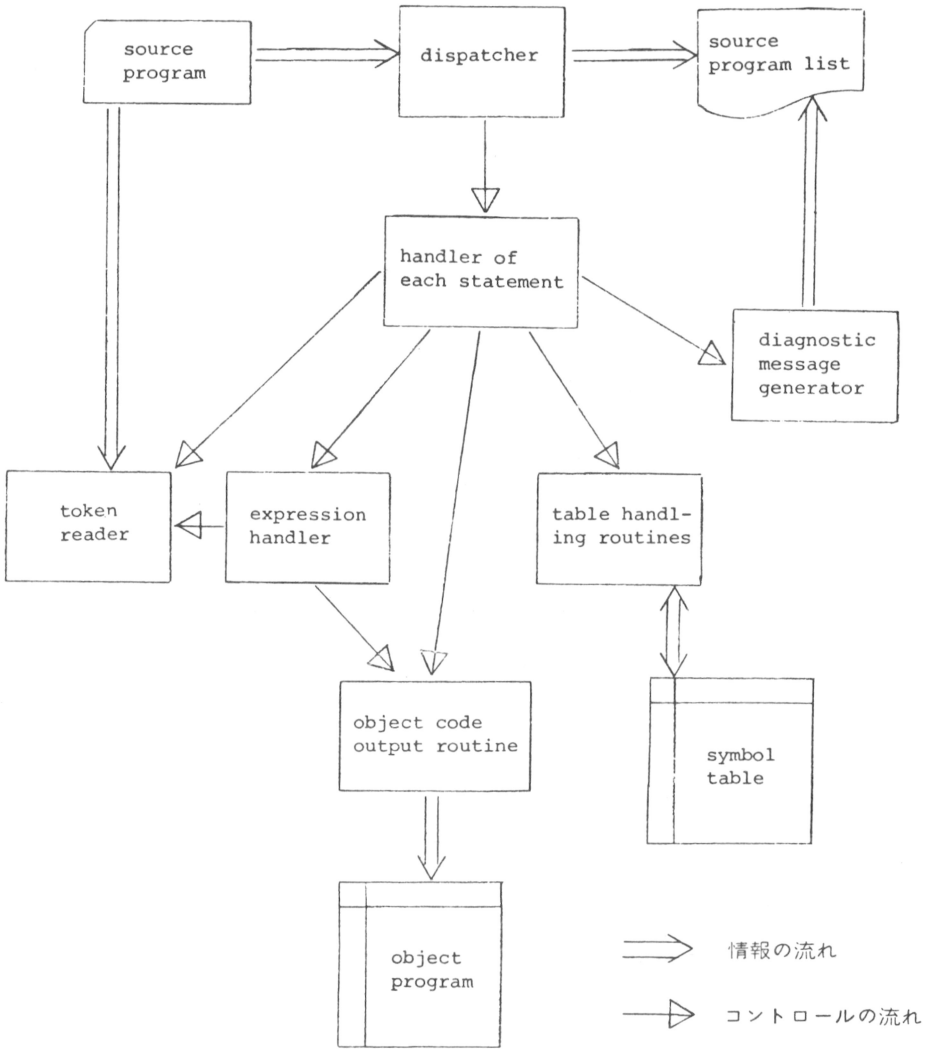
CAFT コンパイラの構成を第1図に示す。

4. オプション

CAFT を呼び出す制御カードには，次のようにしてオプションを指定することができる。

@ＣＡＦＴ，オプション文字の列

オプション文字は 12 種類あり，次のようなことが指定できる。下線を施した部分が省略時解釈である。



第1図　ＣＡＦＴコンパイラの構成

・実行時に添字のチェックを行なう／行なわない

・実引数と仮引数の型の一致のチェックを行なう／行なわない

・診断メッセージのうち程度の軽いものを印刷しない／全部印刷する

・原始プログラムのリストを印刷する／印刷しない

・コンパイルや実行に関する統計値を印刷する／印刷しない

・トレースができるような目的プログラムを生成する／トレースのことは一切考慮しない

・文の種類ごと，文ごとのプロファイルを作成する／作成しない

5. 処理上の問題点

コンパイラ作成にあたって，次のような点が問題となった。

イ．言語仕様の拡張に伴なうもの

・多重代入文

```
I = 5
A(I) = I = X = 3.14
```

X に 3.14, I に 3, A(5) に 3.0 が入る。

・文字代入文

```
DO 5 I = H4,L,M
DO 5 I = 4H,L,M
```

・DO 文　パラメタに整数型の式を許したことと，FORTRAN V で増分パラメタの省略時解釈として－1をとることがあるため，次のように定めた。

```
DO 5 I = 3, -4        増分パラメタとして－1を仮定する
DO 5 I = (3), -4      増分パラメタとして＋1を仮定する
```

・WRITE 文

```
WRITE (6,10) FUNC(X)
```

関数 FUNC の中で入出力を行なわないことのチェックが必要となる。

・標準書式　FORTRAN V では，標準書式による入力のときデータの区切りにコンマを用い，標準書式による出力のときデータの区切りは空白となる。このため，入力と出力の対称性が保たれないが，そのままとした。

・出力並びの中の括弧

```
WRITE (6,10) (A(I,J),J.EQ.1,K)        冗長な括弧
WRITE (6,10) (A(I,J),J=1,K)           DO 形並びの括弧
WRITE (6,10) (A(I,J)+J)/K             式の括弧
```

ロ．コンパイルを1パスで行なったことに伴なうもの

・外部名　組込み関数，基本外部関数の多くについて自動型づけ（実引数の型に応じて関

数を自動的に選択する）を行なうため，これらの関数名と同じ名前を関数副プログラム名として使用することを禁止せざるをえなかった。

・DO の範囲ととびこしの関係などについて十分なチェックができなかった。

6. エラー処理

CAFT がコンパイル時，実行時に出す診断メッセージには次のような情報が含まれている。このため，コンパイル時に作成した記号表は実行時まで保存する。

・エラーの程度（ REMARK, WARNING, ERROR, FATAL ERROR ）

・種類を示すコード （3文字）

・補助情報（変数名など）

・該当箇所を示す行番号

・実行時エラーについては，主プログラムまでのバックトラック

次のチェックは実行時に行なう。このため，目的プログラムの効率がかなり低下した。

・添字の値（チェックをはずすことも可）

・計算形 GO TO 文の整数式の値

・割当形 GO TO 文の変数の値

・DO 文，DO 形並びのパラメタの値

・DO の範囲ととびこしとの関係

・文関数参照・副プログラム呼出しで引数の型の対応（チェックをはずすことも可）

・RETURN k で k の値

次のものはチェックできなかった。

・未定義変数の参照

・入出力並びと欄記述子の型の対応

7. トレース

第2図のプログラムをトレースした例を第3図に示す。

代入文のトレース，関数呼出しのトレースを含む目的プログラムは，格納番地の決定と値の確定の時点が異なるために，実行時に入れ子の構造になる。たとえば，次のような代入文

```
X(K) = B + F(K)
```

で，X は1次元の配列，F は関数名とする。トレースすべきものは X(K) と K であるが，

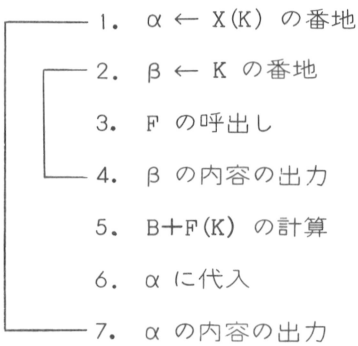
```
      1              DOUBLE PRECISION B
      2              DIMENSION X(2)
      3              TRACE ON
... REMARK ...      < TR-Ø >
      4              CALL SUB(3.14,A)
      5              READ(5,1ØØ) B
      6          1ØØ FORMAT(D2Ø.1Ø)
      7              X(1) = X(2) = F(A,B)
... REMARK ...      < EQ-1 >
      8              STOP
      9              END


     1Ø              SUBROUTINE SUB(X,Y)
     11              TRACE ON Y,*
... REMARK ...      < TR-Ø >
     12              IF (X .GT. 1) GO TO 1Ø
... REMARK ...      < MM-Ø >
     13              Y = X * X
     14              RETURN
     15           1Ø Y = X + 1
... REMARK ...      < MM-Ø >
     16              RETURN
     17              END


     18              FUNCTION F(X,Y)
     19              DOUBLE PRECISION Y
     2Ø              F = SIN(X) + SQRT(Y)
... REMARK ...      < GN-1 >  DSQRT
     21              RETURN
     22              END
```

第2図　トレースのための例題

```
<    4>    TRACE      CALL  SUB
<   12>    TRACE      IF (.TRUE.)
<   12>    TRACE      GO TO    1Ø
<   15>    TRACE      Y      = 4.14ØØØØ+ Ø
<   16>    TRACE      RETURN
<    4>    TRACE      A      = 4.14ØØØØ+ Ø
<    5>    TRACE      B      = 3.25ØØØØØØØØØØØØØØ+   1
<    7>    TRACE      FUNCTION CALL
<    7>    TRACE      B      = 3.25ØØØØØØØØØØØØØØ+   1
<    7>    TRACE      A      = 4.14ØØØØ+ Ø
<    7>    TRACE      X      (     2)= 4.86Ø268+ Ø
<    7>    TRACE      X      (     1)= 4.86Ø268+ Ø
```

第3図　トレースの例

これは次のような順序で実行される。



このような入れ子の目的プログラムを生成することにより，副プログラム F の中に多くの代入文が存在しても X(K) の番地は保存される。

CAFT では，上記の α，β に対応する格納場所はプッシュダウン スタックとし，記憶場所の利用効率を高めるように配慮した。

トレース オプションの指定があると，トレースすべき変数の番地の退避やトレース情報の出力のための目的プログラムが追加されるので，TRACE 文によって指定されたトレースすべき変数がほんのわずかであっても，目的プログラムの実行速度はきわめて遅くなる。また，トレース情報の出力の量は初心者には見当がつかず，いたずらに多量の紙を消費する可能性がある。そのうえ，きわめて簡単なプログラムに対しても安易にトレースしがちであり，アルゴリズムの検討という重要な教育項目に対する効果が裏目に出る場合も考えられる。したがって，トレース機能は初心者用というよりはむしろ熟練者用あるいは教育者用というべきものと考えられる。

8. プロファイル

CAFT ではオプションの指定により原始プログラムにどんな種類の文があり，どの文が何回実行されたかをヒストグラムとして印刷することができるようになっている。

イ．文の種類 (STATEMENT HISTOGRAM / STATIC)

どんな種類の文がいくつあったかを示す (第5図)。第4図の連立1次方程式を解くプログラムに 10 元のデータを与えた場合で，以下も同じ。

ロ．文の種類ごとの実行回数 (STATEMENT HISTOGRAM / DYNAMIC)

どんな種類の文を何回実行したかを示す (第6図)。

ハ．文の実行回数 (EXECUTION PROFILE)

どの文を何回実行したかを示す (第7図)。

```
      C
      C     LINEAR  EQUATIONS
      C
 1          REAL  A(50,51)
 2          READ (5,100) N
 3    100   FORMAT (I5)
 4          N1 = N + 1
 5             DO 200 I = 1, N
 6             READ  (5,300) (A(I,J), J=1, N1)
 7    200      WRITE (6,350) (A(I,J), J=1, N1)
 8    300      FORMAT (10F8.4)
 9    350      FORMAT (1H0, 10F9.4)
10             DO 700 K = 1, N
11             K1 = K + 1
12                DO 400 J = K1, N1
13    400         A(K,J) = A(K,J) / A(K,K)
14                DO 600 I = 1, N
15                IF (I .EQ. K) GO TO 600
16                   DO 500 J = K1, N1
17                   A(I,J) = A(I,J) - A(I,K) * A(K,J)
18    500            CONTINUE
19    600         CONTINUE
20    700      CONTINUE
21          WRITE(6,800) (I,A(I,N1), I=1,N)
22    800   FORMAT (1H0, 6H    X(,I2,2H)=, E13.6)
23          STOP
24          END
```

第4図　プロファイルのための例題

<<<<<<<<<<　STATEMENT HISTOGRAM / STATIC　>>>>>>>>>>

STATEMENT	COUNT	%	1--------10--------20--------30--------40--------50
ASSIGNMENT	4	14.29	****
COMMENT	3	10.71	***
CONTINUE	3	10.71	***
DO	5	17.86	*****
END	1	3.57	*
FORMAT	4	14.29	****
GO TO	1	3.57	*
IF LOGICAL	1	3.57	*
READ	2	7.14	**
REAL	1	3.57	*
STOP	1	3.57	*
WRITE	2	7.14	**
<TOTAL>	28		EACH * = 1

第5図　文の種類

<<<<<<<<<<　STATEMENT HISTOGRAM / DYNAMIC　>>>>>>>>>>

STATEMENT	COUNT	%	1--------10--------20--------30--------40--------50
ASSIGNMENT	561	39.76	**************************
CONTINUE	605	42.88	*****************************
DO	112	7.94	******
GO TO	10	.71	*
IF LOGICAL	100	7.09	*****
READ	11	.78	*
STOP	1	.07	
WRITE	11	.78	*
<TOTAL>	1411		EACH * = 20

第6図　文の種類ごとの実行回数

```
<<<<<<<<<<   EXECUTION  PROFILE   >>>>>>>>>>

  LINE     COUNT       %        1--------10--------20--------30--------40--------50

     2         1       .07
     4         1       .07
     5         1       .07
     6        10       .71      *
     7        10       .71      *
    10         1       .07
    11        10       .71      *
    12        10       .71      *
    13        55      3.93      ******
    14        10       .71      *
    15       100      7.14      **********
    16        90      6.42      *********
    17       495     35.33      **************************************************
    18       495     35.33      **************************************************
    19       100      7.14      **********
    20        10       .71      *
    21         1       .07
    23         1       .07

<TOTAL>      1401                EACH * =     10
```

第7図　文の実行回数

```
<<<<<<<<<<   EXECUTION TIME PROFILE   >>>>>>>>>>

  LINE  TIME(MS)       %        1--------10--------20--------30--------40--------50

     1        .6      .03
     2       3.4      .19
     4        .8      .05
     5        .4      .02
     6     177.0    10.09      **********
     7     174.0     9.92      **********
    10        .6      .03
    11       7.8      .44
    12       5.8      .33
    13      73.4     4.19      *****
    14       6.2      .35
    15      73.8     4.21      *****
    16      53.6     3.06      ***
    17     738.4    42.10      ******************************************
    18     319.2    18.20      ******************
    19      64.4     3.67      ****
    20       6.2      .35
    21      48.2     2.75      ***

<TOTAL>    1753.8                EACH * =   16.0 MS
```

第8図　文の実行時間

ニ．文の実行時間 (EXECUTION TIME PROFILE)

どの文を何ミリ秒実行したかを示す（第8図)。UNIVAC 1106 の時計ルーチンを1回呼ぶたびに 400 ～ 500 マイクロ秒かかるので，このヒストグラムは意味がない。実際より数十倍も実行時間がかかったかのようにカウントされてしまう。時計は 200 マイクロ秒きざみで，プロファイル ルーチンを作成したときには，時計ルーチンの時間が利用者に加算されるとは思っていなかったので，むしろ逆にあまり通らない部分は全然カウントされないであろうと予想していた。

9. 効　率

CAFT の主目的は入門教育における実習プログラムを効率よく処理することである。文献［1］に掲載されているプログラムと例題として使用したプログラムの計121個を使って効率を比較した結果を 表1 に示す。これらのプログラムはカード枚数計 4757 枚（1プログラムあたり 39 枚)，うちプログラム カードは 3579 枚（1プログラムあたり 29 枚)で，典型的な実習プログラムといえよう。コンパイル速度は FORTRAN V がやや遅いが，これは目的プログラムの最適化を行なっているためである。CAFT の連結編集時間は，変

表1 効率の比較

	CPU 時間（ミリ秒）				入出力語数	処理料金
	コンパイル	連結編集	実　行	計	(K 語)	(円)
CAFT	58,857	822	47,870	107,549	7,053	6,864
	486	6	395	896	59	57
	(16.4)					
FORTRAN V	68,077	293,500	23,195	384,869	76,441	9,213
	562	2,425	191	3,180	632	76
	(19.0)					
RFOR	59,691	335,383	25,144	420,313	70,345	10,301
	493	2,771	207	3,473	581	85
	(16.6)					

上段は全プログラムの合計。下段は1プログラムあたりの平均。（ ）内はプログラム カード1枚あたりのコンパイル時間。

数・配列の番地割付けに要した時間である。実行時間は実習プログラムの特長としてきわめて短いが，CAFT は他にくらべて2倍ほどかかっている。これは実行時にチェックをいろいろと行なうためと，変数の値の参照を間接番地指定で行なっているためである。実習プログラムの場合には，連結編集時間の割合が大きいため，合計時間や処理料金の点では CAFT がもっともすぐれている。

チャネルの負荷の軽減に関しては，表1の入出力語数から明らかなように，CAFT は FORTRAN V の 9.23 %，RFOR の 10.03 % にすぎず，大きな効果があがっている。

10. 教育上の効果

診断メッセージの種類が多い（FORTRAN V のライブラリが出すものを除いて 381 種類）ので，デバッグに役立っている。またコンパイル エラーがなくてしかも答が正しくないときには，トレース機能を利用して誤りを発見することができる。

初心者に対して入出力の書式を説明するのはかなり面倒であるが，標準書式を使うことにより，最初から言語の細部に立ち入らなくてもすむようになった。

プログラム相談員からの反応として最大のものは，診断メッセージを文章にしてほしいということで，メッセージの一覧表を見ながらデバッグすることにかなり強い抵抗がみられた。

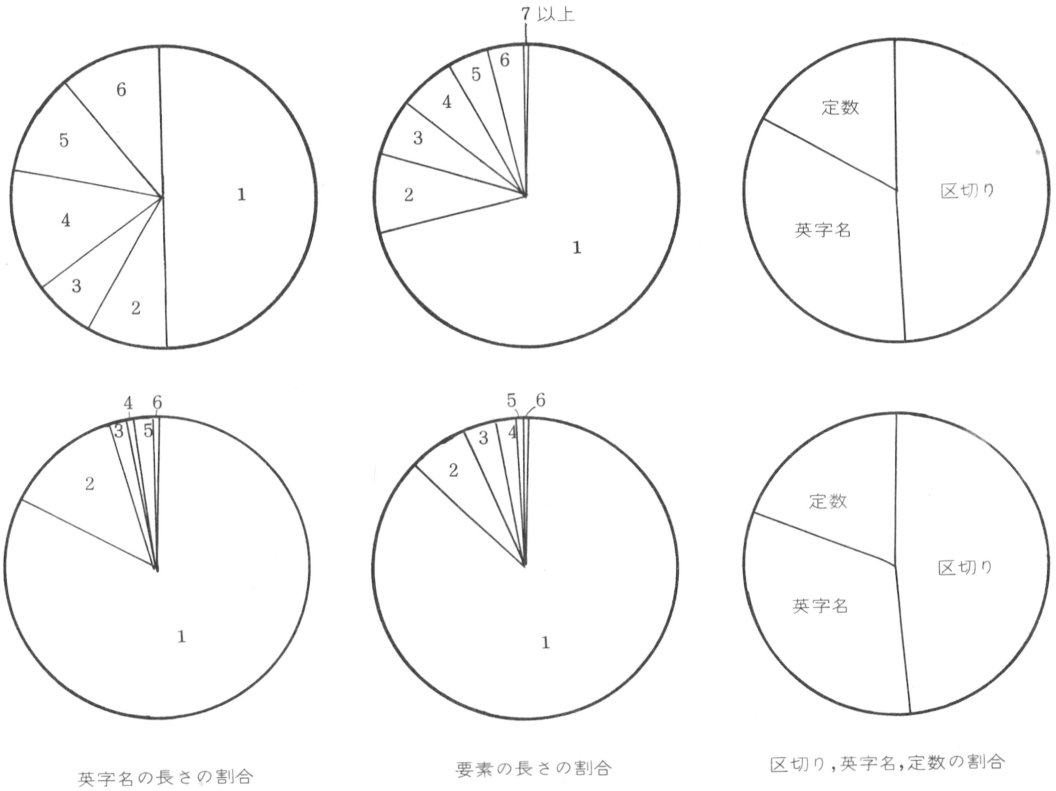
付．プログラム要素の構成比

CAFT コンパイラの token reader は，原始プログラムの中の区切り，英字名，定数の数や長さの統計をとる機能ももっている。第9図はその構成比の例を示したものである。上段は文献［1］の 25 個の例題プログラムの平均で，下段は文献［2］の 19 個のプログラムの平均である。この結果 FORTRAN プログラムの構成要素は1桁のものが圧倒的に多い（7～8割）ことがわかったので，1桁の要素だけ特別あつかいするようにしたところ，token reader の平均処理時間は約1割短縮された。コンパイラ全体からみればわずかではあるが。NHK 講座の英字名は1桁のものがとくに多くなっている。これは BASIC と対比したプログラムになっているためである。

参考文献

［1］ 浦　昭二「FORTRAN 入門」1972．

［2］ 森口繁一他「NHK コンピュータ講座テキスト」1975．



第9図　FORTRAN プログラムの構成要素比
上段　文献［1］，下段　文献［2］