FORTRAN 語のための変換プログラムの問題*

浅　井　　清**

Abstract

The purpose of this note is to find suitable features of FORTRAN conversion programs. For this, the author gives brief descriptions of some existent conversion programs, some examples of FORTRAN program conversions and desirable features of conversion programs

1. はじめに

ここで問題とする変換プログムとは，ある言語で書かれている計算機プログラムを，その言語と類似の他の言語のプログラムへ変換する作業を補助する計算機プログラムをいう．以下の議論では，FORTRAN 語が対象となる言語の例として取り上げられる．

近年になって計算機の発達に伴って，原子力に関する研究機関や企業において使用される計算機が多様化し，また，これらの研究機関・企業間の研究協力や技術情報の交流が，計算機プログラムの交換という形式で行なわれるようになってきた．原子力関係のプログラム(原子力コードと呼ぶ)は，その対象となる問題の性格から長いプログラムが多く，なかには 20,000 個以上の FORTRAN 文から成っているものもある．しかも，計算機の大型化・高性能化に伴って，一つの仕事の単位としてのプログラムも長くなってきている．そして，これらの原子力コードは，世界中の研究機関・企業において作成されており，使用されているプログラミング言語にもさまざまの相違がある．たとえば，英国の KDF-9, IBM 7030, CDC 3600 の FORTRAN 語などは，標準的な FORTRAN 語との文法的な差が大きすぎて，これらの言語を使用して書かれたプログラムは，標準的な FORTRAN 語用のコンパイラでは翻訳できない文が多くなる．このような事情から，近年になって，プログラムを変換する変換プログラムの重要性が議論されるようになったが，この議論は，全く相対立する二つの意見によって二分されている．その一つは，変換プログラムは必要であると主張する意見であり，これは英国やヨーロッパで作成されたプログラムを書き換えた経験を持つ研究者・技術者に多い意見である．もう一つは，変換プログラムは不必要とする意見であり，これは主として標準的な FORTRAN II プログラムから FORTRAN IV への書き換えに従事した人たちに多い意見である．変換プログラムに課せられた第一の課題は，これら相対立する二つの主張をどのようにして満足させるかということにある．変換プログラム不用論の一番大きな根拠は，変換プログラムの経済性にあるが，変換プログラムの効率が悪い理由は，次の四つが考えられる．

(i) 変換プログラムは原則としてどの機種においても使用できる必要があり，それ自身の変換に手間どってはならないということから，問題向き言語（たとえば，FORTRAN 語）で書かれているのが普通である．このために変換プログラムの実行効率が悪くなっている（IBM 7090 を使用して実現された変換プログラムの処理速度は，後述する FLIC, SIFT 90 などでも 200～250 枚/分）．

(ii) 長い（たとえば，FORTRAN 語・10,000 個以上の文から成る）プログラムを変換すれば，変換だけで計算機の時間を消費し，それだけで計算機使用料はばく大なものになる．ところが，実際にはプログラム全体の 1 割以下の文のみが変換の対象となっているに過ぎない場合が多い．この点で，変換プログラムはコンパイラやインタプリタ，あるいはプログラミング言語のシンタックス・チェッカーなどとは本質的に異なっている．しかし，従来の変換プログラムは，使用者の意図に反して，その持てる機能のすべてを変換すべきプログラムの一つ一つの文に作用させるように作られている．

(iii) 変換すべき事象が多様なために，ある機能の

* Problems on Conversion Programs for FORTRAN Languages, by Kiyoshi Asai (Computing Center, Japan Atomic Energy Res. Inst.)

** 日本原子力研究所計算センター

みを特別に効率を上げた変換プログラムを作成できない．

（iv） ある要素(プログラム中の変数やストリング)が一つの集合に含まれているかどうかを知る機能（後述）の効率が変換プログラムの効率に大きな影響を持っているにもかかわらず，従来の変換プログラムはこの点に注意を払っていない．

現在までに発表されている変換プログラムの持っているこれらの欠点は，おそらくわれわれが変換プログラムの特性を十分に捨象していないことにあると思われる．本小編の目的は，この特性について議論することにあるが，そのまえに現存する変換プログラムでFORTRAN 語で書かれたプログラムの変換を行なうものを二，三紹介したい．ここで挙げる例のうち，初めの二つはプログラムの変換をストリングのパターンの認識とその置き換えとしてとらえている．3番目のものは，いくつかのプログラム単位を首尾一貫した変換の対象としてみなし，それらのプログラム単位をコンパイラが行なうと同程度まで分解し，その後あらかじめ与えられている自己の言語の文法のわくのなかで分解したプログラムを再構成する．この三つの例は，変換プログラムとして単純なものから非常に大がかりなもの，およびその中間的なものの例として取り上げた．もちろん，このほかに多くの変換プログラムが存在するが，それらの機能は以下に述べるものと大同小異である．

2. 変換プログラムの例

2.1 GENSUB

GENSUB は英国で作成されたストリング（文字のつながりをストリングと呼ぶ）変換のための変換プログラム〔1〕である．このプログラムは IBM 1401 のアセンブリ言語である SPS (Symbolic Programming Systems)で書かれている．その変換のための基本的な考え方を図で示すと **Fig. 1** のようになる．プログラムは，その内部に刺激と応答の対応表を持っており，その対応表に含まれているストリングと同一のストリングが入力されたときに対応する応答を出力する．刺激と応答の対応表の例をあげると，次のようなものがある．

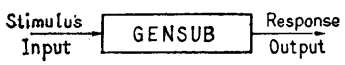

Fig. 1 The stimulus and response scheme of GENSUB.

刺激	応答
READ INPUT TAPE n, list	READ (5, n) list
COSF	COS

GENSUB の特徴をあげると

（a） 刺激と応答の対応表のなかのストリングは，アルファベット順の木構造 (tree structure) を持っている．このためにストリング検索の時間とメモリを節約することができる．木構造の例を **Fig. 2** で示す．

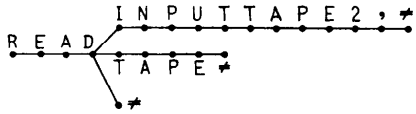

Fig. 2 The tree representation of strings.

（b） 対応表は FORTRAN Ⅱの入力に対して FORTRAN Ⅳの出力を出すように作成されている．入力は FORTRAN Ⅱの一つの文を単位として行なう．

（c） 刺激と応答の対応表は GENSUB のプログラムのなかで定められており，プログラムを書き換えなければ，刺激と応答の組合せを変えることはできない．

（d） 書き換えることができない文にフラグ（しるし）をつける．たとえば，入力 IF ACCUMULATOR OVERFLOW に対して

IF ACCUMULATOR OVERFLOW **** TYPE 5

を出力する．

（e） 文法の構造に関する問題（たとえば，一つの文の前後関係，一つの文中に現われるストリングの前後関係をしらべ，そのストリングの文法上の意味を知ることなど）にはタッチしない．

（f） プログラムは IBM 1401 で

GENSUB 本体：700 文字
入出力領域　：2900 文字
処理速度　　：100～150 文/分．

2.2 FLIC

FLIC は，GENSUB をもとにして英国で作成されたストリング変換のプログラムであるが，主として FORTRAN プログラムの変換に便利なように作成されている〔2〕．FLIC の基本的な考え方は GENSUB と同一であるが，GENSUB にはない種々の機能を持っている．FLIC の特徴を列挙すると

（a） 刺激と応答の対応表は簡単な表で与えることができ，その対応表は FLIC 自身のなかには存在しない．対応表は使用者が作成することができて，使用者によってストリングの対応関係の形式で与えられた

表を解読するために補助プログラム DUSTER がある．DUSTER は対応表に含まれているストリングを本構造表現に変形し，それをテープに書く．FLIC はテープ上の情報（刺激と応答の表）に従って入力を処理する．これを図で示すと **Fig. 3** のようになる．

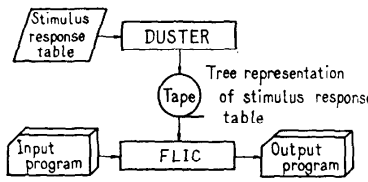

Fig. 3　The conversion scheme FLIC program

（b） FLIC は JOKER SEARCH と呼ばれる key character 検索の機能を持っている．JOKER 数字は 16 個あり，必要なときに使用者が対応表を使用して与える．これらの数字は，それぞれ固有の機能を持っていて，あるストリングの前後の key character の種類をしらべ，逆にそのストリングの文法的な意味を判定すること，およびそのストリングの処理方法を指示することなどに利用される．

例　JOKER 数字 10 は，それがストリング α に対して与えられると，入力されたプログラムのなかで刺激となるストリング α が出現したとき，そのストリングの前に＋，－，*，／，(，＝のいずれかの文字が現われたかどうかを捜す．いま対応表に

刺激	応答
10 RETURN	RETURN

と定められていれば，入力ストリング

A=B+RETURN (1)

は変形されないで，そのまま出力される．これは RETURN がこの場合は変数名とみなされたことを意味する．

（c） 文単位で書き換えられないものにはフラグをつけることができる．そのフラグに対応する注釈も使用者が対応表のなかで与えることができる．

（d） 対応表を作成するときに，ストリングをアルファベット順に並べなければならず，まちがえやすい．

（e） プログラムは IBM 7090 FORTRAN II で

DUSTER：100 文
FLIC　：100 文
処理速度：200～250 文/分

2.3 FAUST

FAUST は FORTRAN 語を FORTRAN 語へ変換するために日本で作成されたプログラムである〔3〕．FAUST の特徴は，報告〔3〕のまえがきを引用させていただくと，次のようにまとめられる．

（a） 変換操作を小さな作業段階に分解し，その各作業段階をサブルーチン化する．サブルーチン間の情報の受け渡しのための表は，各種の FORTRAN 語に適用できる形にする．したがって，内部に組み込んである表と，いくつかのサブルーチンを取り換えることにより，種々の EORTRAN 語間の変換プログラムを作成できる．

（b） 画一的な処理が可能なように，変換される原始プログラムをひとたび符号化しておき，最後にそれを FORTRAN 語の形に再構成する．

（c） プログラム全体をひとまとめにして処理し，副プログラム間の相互関係もしらべる．

（d） 変換のみならず既製のプログラムの修正のためにも利用できるよう，原始プログラムの内部仕様に関する各種のデータを印刷する．以上の特徴から FAUST の機能を簡単に書けば **Fig. 4** のようになる．

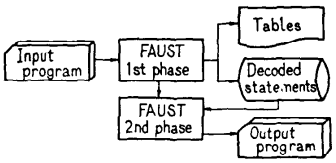

Fig. 4　The conversion schem of FAUST program.

（e） プログラムは HITAC 5020F FORTRAN 語とアセンブリ言語で書かれており

8000 文 (FORTRAN)
1000 文 (アセンブリ語)
処理速度：400～500 文/分.

このほかによく知られている変換プログラムとして，IBM 7090 用に作成された FORTRAN II から FORTRAN IV への変換プログラム SIFT 90 がある．これはプログラムが大がかりな上に処理速度があまりよくない(200～250文／分)．その上，FORTRAN II と FORTRAN IV とで文法が極端に異なるのは入出力文の部分で，COMMON 文中の変数の並べ換えが問題となるプログラムは，筆者の知る限りではあまり存在しない．ところが，SIFT 90 は COMMON 文の変数の並べ換えが大きな機能となっている．このために SIFT 90 は GENSUB と FLIC の中間程度の働きしかなかったといってよいであろう．

3. プログラム変換の例

変換プログラムの持つ機能として基本的なものは（1）文字の早い認識，（2）ストリング処理のさまざまな機能，（3）ストリングを要素とする集合についての集合論的な演算のさまざまな機能の三つに集約されるように思われる．特に，（3）の機能の効率が変換プログラムの効率そのものを決定するというのが筆者の主張である．この主張の説明のために，プログラム変換の簡単な具体例を二，三あげてみる．

3.1 ストリング処理の機能

READ INPUT TAPE 2, m, list

なる文を

READ (I, m) list

のように書き換える．手順を分解してみると

（a） ストリングを読む．

（b） READ 文を認識する．

（c） 新しいパターン READ (I, m) を作り出す．

（d） list を複写する．

このとき変換プログラムとして最小限必要となる機能を挙げると

（i） ストリングの読み込み，書き出し．

（ii） ストリング中に文字のポインタを設定し，ストリング中の文字処理の進行状況を早く知ること．

（iii） パターン・マッチングの機能．

（iv） ストリングの置き換え．

3.2 記憶の割当て

E2は English Electric KDF 9 計算機のための FORTRAN 語の方言である EGTRAN の略称，また，S2は IBM 7030 STRETCH FŌRTRAN 語の略称である．E2, S2 は共に FŌRTRAN II によく似ているが，配列に関する記憶場所の割当て法は通常の FORTRAN 語とは全く異なっている．E2, S2 では，主プログラム・副プログラムの別なく DIMENSION 文中に可変（variable）DIMENSIONを使用することができるが，その DIMENSION 文中に現われる配列名とその宣言子添字となる変数の名前は COMMON 文中に定義されていなければならない．しかし，FORTRAN IV と異なり，この可変配列そのものが共通ブロック中に割り当てられることを意味するわけではない．その配列の割当て場所(番地)は，そのプログラムの実行される前に（すなわち，プログラム・ローディングのすぐ後で）計算されて共通ブロック以外の場所に割り当てられる．それからその配列のためにとられた一連の番地の最初の番地の値を保持する制御変数が，その配列名で共通ブロック中のただ一語に割り当てられる．いいかえれば，COMMON 文中に現われた配列名は，その配列のために他の場所に確保されている番地の最初の番地を値として持つ一語の共通ブロック要素である．このような言語で書かれたプログラムを FORTRAN IV へ変換することを考えてみる．いま，次のような宣言を持つ E2 のプログラムを，固定された宣言文を持つ FORTRAN IV のプログラムへ書き換える．

```
DIMENSION     A(N, N), B(M, N)……
COMMON        A, B, N, M, ……
EQUIVALENCE   (A(N, 1), B(1, N)), ……
```

上の文を次のように最き換える．

```
DIMENSION     A(10, 10), B(20, 10), ……
COMMON        N, M, Q01, Q02, ……
EQUIVALENCE   (A(10, 1), B(1, 10)), ……
```

書き換えの手順を分解すると

（a） 原始プログラム中から DIMENSION, COMMON, EQUIVALENCE 文のみをひろい出す．

（b） 可変（variable）DIMENSION 文中の配列名 A, B, ……とその配列に対する宣言子添字の変数名 N, M……を要素とする集合を作り，それを DIM と名づける．

（c） COMMON 文中の配列名・配列に対する宣言子添字変数名，その他の変数名を要素とする集合を作り，それを COM と名づける．

（d） EQUIVALENCE 文中の配列名・配列に対する宣言子添字変数名，およびその他の変数名を要素とする集合を作り，それを EQUIV と名づける．

（e） あらかじめ設定された集合 FIX（宣言子添字の変数名とその変数名にとって代わるべき整定数を要素とする集合）の要素 a で，a∈ DIM, a∈ EQUIV となる a については，DIM, EQUIV のなかの a を集合 FIX で与えられた整定数で置き換える．

（f） 必要ならば COM の要素をダミー変数名 Qon で置き換える．

必要となる機能

（i） ストリング a が集合に含まれているかどうかを速く知る機能．

（ii） 可変な容量を持つ集合を作る機能（ストリングの数はプログラム単位によって可変だから）．

（iii） ストリング a をストリング b で置き換える操作．

（iv） あらかじめ用意されているストリング Qon

の n を n＝n＋1 と設定するための操作.

FORTRAN Ⅳ の整合配列 (adjustable dimension) の機能を利用して E 2, S 2 の可変配列の特質を生かそうとするなら, 上の(ⅰ)〜(ⅳ)に加えて

(ⅴ) 集合の包含関係, 論理積, 論理和, 補集合を求める演算が必要になる.

3.3 配列の取扱い

固定配列から整合配列への書き換えを例としてとると, その手順は

(a) 整合配列に変更する配列名を要素とする集合 SET 1 と, その宣言子添字となる変数名の集合 SET 2 (たとえば, DN (NG, NG) なる配列について) を与える.

(b) 各プログラム単位で定義されている DIMENSION 文をしらべ, そのなかの配列名 DN について DN∈SET 1, かつ DN NG∈SET 2 ならば, 副プログラムの配列名のリストに DN, NG を加える.

必要となる機能

(ⅰ) 集合の論理和, 論理積, 包含関係.

4. 変換プログラムとは何か

いままで, 主として FORTRAN を対象にして考えてきたが, どのような言語であれ, 原始プログラム間の変換に要求される機能であって, 通常の問題向き言語のコンパイラと共通するのは, せいぜい文法のシンタックス・チェックの段階までである. コンパイラを例にとれば, その第1段階である文法検査のルーチンと, 第2段階であるプログラム中に現われる変数, およびストリングの割り当てまでが変換プログラムの持つべき機能であって, それ以上にコンパイラの持っている以後の各段階であるインデックス・レジスタの割り当てルーチン, 算術式の翻訳ルーチン, シンボリックな機械語を数値的機械語へ置き換えるルーチンなどの機能は, 変換プログラムにとって必要なものではない. このような立場から変換プログラムの位置づけを行なうならば, 変換プログラムは人間＋変換プログラム＋コンパイラの三者協同作業のひとつを受け持つべきであって, 変換プログラムに全能のものを期待してはならない.

以上のことから変換プログラムについてひとつの行き方が考えられる. それは, 変換プログラムをある言語のために特別に作成することをやめ, その代わりに記号処理言語を使用して, 個々の問題単位で必要に応じて変換プログラムを作成するという方向である.

この観点から代表的な記号処理言語で, その取り扱うデータに特別の構造を要求されない COBOL〔4〕, EOL〔5〕, SNOBOL〔6〕について簡単な比較表を **Table 1** で示す.

Table 1　The comparison Table of Some Symbol Manipulation Languages

機能 ＼ 言語	算術演算	集合演算	動的レイベル	可変長のストリング	ストリングへの早いアクセス	文字の早い認識	文字を利用する早いブランチ	間接番地	可変長ストリングの入出力	プログラミングの難易	互換性
COBOL*	○	△	×	△	△	△	△	×	△	△	△
EOL	○	×	×	△	△	△	×	×	△	×	×
SNOBOL	△	△	△	○	△	△	△	○	×	○	△

(注) ○: よい
△: 可能, ただし, 効率はよくない.
×: 文法的に不可能, または効率がひどく落ちる.
*: COBOL は IBM 7044 COBOL.

リスト処理言語のうち, LISP や SLIP は文字の取扱いの不便さ, 効率の悪さなどから比較にならない. 最近, 英国において **Table 1** の集合演算と互換性を重視した SYNCON (Syntax Conversion Program) と呼ばれる変換プログラムが開発されたが, まだ正式には発表されておらず, 詳細は不明である. 筆者を含むグループにおいて, Table 1 のほとんどの項目を満たすと思われる SPM (String Processing Machine) と呼ぶストリング処理のための仮想機械を FORTRAN Ⅳ を用いて作成し, 現在試験的に使用しているが, その詳細は別の機会にゆずりたい.

参 考 文 献

1) Bindon, D. C.: GENSUB-A General Substitution Comment Programme. Nov., 1963, Computer Group, Winfrith, Atomic Energy Establishment, U. K.
2) Basher, J. C.: The FLIC Conversion Code, AEE-R439 May, 1965, AEE. Winfrith, U. K.
3) 渡辺, 漆原: FORTRAN 言語間の自動変換プログラム, 第9回プログラミングシンポジウム報告集, 1968 年1月, 情報処理学会.
4) IBM 7044 OS COBOL, Form C 28-6336-3
5) Lukaszewicz, L.; EOL-A Symbol Manipulation Language, The Computer Journal, May, 1967.
6) Farber, et al.: The SNOBOL 3 Programming Language, Bell System Tech. Jour. vol. 45, No. 6

(昭和43年12月10日受付)