ALGOL—部分集合について*

清　水　留　三　郎**

ALGOL 60 は算法言語の体系化の結晶といえるが，対応する機械語への翻訳とその結果でき上がった機械語のプログラムの実行を能率よく行ないたいという希望と相容れない部分をその言語体系に含んでいる．そこで速く簡単に実行できるようにすることと ALGOL 60の報告の中に述べられている五つの問題点を回避ないしは解決することを目的として，ALGOL 60 の構文の自由度に制限を加えて ALGOL 60 の部分集合を作ることになった．部分集合というのは，それで書いたプログラムはすべて full ALGOL のコンパイラーの下でも正しく働くという保証を与えることができるものということである．ここに紹介する部分集合は，IFIP の van der Poel を議長とする ALGOL に関する Working Group 2.1（情報処理学会からの委員は森口繁一教授）によって作成され，1964 年 IFIP の総会で承認された．部分集合にはほかに ECMA (European Computer Manufacturers Association) の部分集合があるが，IFIP の部分集合は ECMA の部分集合のさらに部分集合になっている．

IFIP の部分集合は ALGOL 60 の報告に対する添削および変更の形で記述されているが，ここではできるかぎり ALGOL 60 の報告の順序に従って該当する項の番号とともにそれらを紹介し，その目的および影響を説明する．

2.1. 文　字　の

<letter>::=a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|
q|r|s|t|u|v|w|x|y|z|A|B|C|D|
E|F|G|H|I|J|K|L|M|N|O|P|Q|
R|S|T|U|V|W|X|Y|Z

の |A|B|C|……|Y|Z を削除し，'注：実用上小文字でなく大文字である必要がある場合は，大文字を小文字の金物表現と見なす．' を付け加える．

2.4.3. の

'identifier は自由に選んでよい' を 'identifier は自由に選んでよいが，最初の 6 字が同じ二つの相異なる identifier の値は undefined である' に置き換える．

これは翻訳のときに identifier の表から対応する identifier を検索する作業を著しく簡単にする．6 字としたのは一語に 6 字はいる機械が多いからである．

3.3.1. の

<multiplying operator>::=×|/|÷

の |÷ を削除する．

3.3.4.2. の

'The operations <term>/<factor> and <term>÷<factor> both denote' を 'The operation <term>/<factor>denotes' で置き換え，最後の一文 'The operator÷is defined……(cf. sections 3.2.4. and 3.2.5).' を削除する．

3.3.5.1. の

second: ×/÷

の ÷ を削除する．

いわゆる整数割算は部分集合には含まれない．

3.3.4. 演算子と型 に

'しかし，もし算術式の型がその式の値を計算するかあるいは actual parameter の型か値を識別するかしなければ決定できない場合，型は real であるものとする．' を付け加える．

procedure body の中に name で呼ばれる formal parameter が現われると，対応する actual parameter を調べてみなければ型がわからないし，その procedure が 2 箇所以上で呼ばれていれば actual parameter の型は一方に定まるとは限らない．したがって翻訳の仕事の一部をその program の実行の過程まで残さねばならない．部分集合では，後に出てくる制限と合わせて，翻訳を実行と切り離すことができるように考慮されている．

3.3.4.3. に

'例外として，底 a および指数 i がともに整数型であるならば，指数は unsigned integer でなければならない，さもなければ答は undefined である．' を付け加える．

a↑i は底 a が整数型であっても i<0 のときだけは実数型に変わる．このような型の変化を避けるため部分集合では整数の整数乗に制限が課せられている．

---

* On Subset ALGOL 60, by Tomesaburo Simizu (University of Tokyo)

** 東京大学計算センター

3.5.1. の

＜label＞::＝＜identifier＞|＜unsigned integer＞

の |＜unsigned integer＞ を削除し，

3.5.5. Unsigned integers as labels. を削除する．

3.5.1. の

‘＜simple designational expression＞::＝……

＜designational expression＞::＝……’

を ‘＜designational expression＞::＝＜label＞|

＜switch designator＞’

で置き換える．

3.5.3. の

‘In the general case……result is already found.’

を削除する．

部分集合では unconditional designational expression のみが使える．条件によって行先を変えたければ conditional statement の形で書くことができるからである．これによって conditional designational expression に対して，あたかも同等の conditional statement でていねいに書かれたかのごとく，compiler が気をきかせなければならない労を軽減することができる．

4.3.5. Go to an undefined switch designator. の ‘go to statement の designational expression が値の undefined な switch designator ならば，その go to statement は dummy statement と同等である．’ の ‘dummy statement と同等’ を ‘undefined’ で置き換える．

実行段階で switch designator の subscript の値を調べる手間を省くことができる．

4.6.1. の

‘＜for clause＞::＝for＜variable＞:＝＜for list＞

do’を‘＜for clause＞::＝for＜variable identifier＞

:＝＜for list＞ do’

で置き換える．

for clause の中の controlled variable は，部分集合では simple variable に限られている．プログラムの実行の速さは for statement の実行の速さに最も依存するので，その部分をできるかぎり能率のよい機械語に翻訳できるようにすることが目的である．

4.7.3.2. の

‘value list にでてこない formal parameter は，procedure body の全体を通じて，対応する actual parameter で置き換えられる．この置き換えは文法的に可能なところでは，actual parameter をカッコでくくってから行なわれる．’ の後の部分を ‘この actual parameter は identifier でなければならない．さもなければ名前の置き換えは undefined である．’ で置き換える．

たとえば ALGOL 60 の報告にある．

```
procedure innerproduct (a, b, k, p, y);
value k; integer k, p; real y, a, b;
begin real s; s:=0;
for p:=1 step 1 until k do s:=s+a×b;
y:=s
end innerproduct
```

は，procedure statement

innerproduct (a [t, p, u], b[p], 10, p, y)

で呼ばれれば三次元の配列 a とベクトル b の積を計算し，procedure statement

innerproduct (a[p], b[p], n, p, y)

で呼ばれればベクトル a と b との内積を計算せねばならない．このような procedure declaration は誠に巧妙ではあるが，compiler にとってははなはだやっかいなしろ物であり，部分集合では許されない．

4.7.5. 制限 に

‘procedure body の中の statement の実行の際にその procedure 自身を呼ぶことは許されないし，名前で呼ばれる formal parameter に対応する actual parameter の評価の際にも，またその procedure の中の declaration にでてくる式の評価の際にもそれは許されない．’ をつけ加える．

5.4.4. Values of function designators. の ‘procedure body の中で，assignment statement の left part 以外にその procedure identifier が現われることは，その procedure 自身を呼ぶことを示す．’ を削除する．

いわゆる recursive procedure および recursive call は部分集合には含まれない．これで formal parameter が procedure でも push down store を用意しなくて済む．

4.7.5.5. の本文を

‘名前で呼ばれる場合，actual parameter の種類と型は対応する formal parameter の種類と型とに同一でなければならない．’ で置き換える．

procedure declaration と procedure statement との間に相違がある場合，compiler が型の変換など

を行なう責任があるかどうかの問題を書く者の責任にゆだねた.

5. 宣言 の

'宣言にさらに宣言子 **own** をつけてもよい. これは次の効果をもつ：**block** に再びはいった場合, own な量の値は出たときの値と変わらない. 一方 own でない変数の値は undefined である. ' を削除する.

5.1.1. の

<local or own type>::=<type>|**own**<type>

の |**own** <type> を削除する.

5.1.3. の最後の文章を削除する.

初期値の与え方などに問題のある own の概念は, 部分集合には含まれない.

5. に

'**assignment statement** の **left part list** に現われ, label あるいは **procedure identifier** でない identifier は宣言の中でも自分の宣言の前に現われてはならない. ' を追加する.

ALGOL 60 では declaration が statement に先行することを規定しているのみで, 宣言どうしの前後関係には触れていない. これは sequential translation には困ることなので, 他の宣言に使われる identifier の宣言を先行させることになっている.

5.3.1. の

'<switch list>::=<designational expression>|'

<switch list>, <designational expression>

を '<switch list>::=<label>|<switch list>, <label>' で置き換える.

3.5.3. の最後の文章を削除する.

部分集合では switch list の中は label だけに制限された. これで switch declaration にいわゆる jump table を対応させることが簡単になる.

5.4. PROCEDURE DECLARATIONS. に

'同じ identifier は formal parameter list に2度以上現われてはならない. ' を追加する.

対応する actual parameter が異なっている場合に起こる問題を回避した.

5.4.4. Values of function designators. に

'function designator は procedure statement の形で使われるとしたら dummy statement と同等になるようなものでなければならない. function designator の値は宣言されている変数と actual parameter で決定される. ' を追加する.

いわゆる side effect のある function を宣言することは部分集合では許されない.

5.4.5. Specifications. の

'値で呼ばれる (4.7.3.1 項参照) formal parameter の specification はなされねばならないが名前で呼ばれる (4.7.3.2. 項参照) formal parameter の specification は省略してもよい. ' を 'formal parameter がもしあれば, その specification はすべてなければならない. ' に置き換える.

procedure declaration を簡単に実行し, 速く実行できるよう, かならず細かく指定することを部分集合では要求している.

以上で部分集合の紹介と解説を終えるが, もうおわかりいただけたとおり, 部分集合とはいえ, その言葉から何か小さなものと思ったならば見当違いで, 実用の算法言語としては最も大きいものである. ALGOL を机上の言語としてではなく, 実用上でも経済的な言語としたいという部分集合作成の意図が各計算機に ALGOL 部分集合の compiler が作られることで生かされることを期待したい.

参考文献

1) IFIP, Report on SUBSET ALGOL 60, CACM, **7**, 10 (1964), pp. 626-628.
2) Naur, P. (ed.), Revised Report on the Algorithmic Language ALGOL 60, CACM, **6**, 1 (1963), pp. 1-17.

(昭和39年9月15日受付)