の形式および data allocation に関して興味ある system を作っておりぜひ一読をすすめる.

なお上記文献中（3）（4）（8）は情報処理学会に申し込めば入手できるものと思われる.

現在 ALGOL language に関する情報交換のための centre としてつぎの2箇所がある.

Editor, Communications ACM
Carnegie Inst. of Tech.,
Pittsburgh, 13, Pa., U.S.A.
Editor of the ALGOL-Bulletin
Regnecentralen
Gl. Carlsbergvej 2,
Copenhagen, Valby
Denmark

前者は Communications of ACM という monthly の journal を発行しており，後者は ALGOL-Bulletin を不定期に発行している. ALGOL-Bulletin の方はときどき supplement を発行しており（8）の文献もその一つである. 現在ドイツを中心として

Handbook for Automatic Computation

が企画されている（全体で数巻）. この Handbook はあらゆる種類の数学計算の test ずみの algorithm を集成するものであって，この algorithm は ALGOL language で書かれることになっている. volume 1a は ALGOL language の使用法の解説，volume 1b はその translator の解説にあてられる予定である.

また一般的な automatic programming に関する本につぎのものがある.

Annual Review in AUTOMATIC PROGRAMMING:

Edited by Richard Goodman M.A., B. Sc. Pergamon Press. 現在 Volume 1（1960）と Volume と 2（1961）が出ている.

"ALGOL 60 Language について" に対する remark*

高 橋 秀 俊**

1) 〔例 1〕について

この例で $A(x)+B(x)$ と $B(x)+A(x)$ とが違う結果を与えるのは，$A(x)$ という function designator が $Z:=Z+1$ という"副作用"をもつ procedure を呼ぶからである. このように，同じ式が計算の順序によって違う結果を生ずるのは artithmetic expression というものに対する我々の常識に反するものであって，不都合というべきである.

この不都合の原因を除くには，function designator を定義する procedure が副作用を生むことを禁止するのが妥当であろう. たとえば

"function designator を定義する procedure declaration の中には，その function designator 以外の non-local variable に値を assign するような statement があらわれてはならない". という条項を加えれば一応問題は解決されると思われる. この制限は多少窮屈かもしれないが，読んでわかりやすいという ALGOL の特徴を活かすためにはやむを得ないのではあるまいか.

なお，これに類する case の一つとして，function designator が input に関する procedure である場合がある. たとえば，READ が一つの input procedure で，input device から読み出された数値をあらわすとすると，これが2度以上あらわれる statement，たとえば

$A:=\mathrm{READ}+B\times\mathrm{READ};$

は，一方の READ が第1に読み出された数値，他方は第2に読み出された数値をあらわし，どちらが先に演算されるかで異ってしまう. これを有効に防ぐには，input procedure を function designator の形にすることを禁止するか，または，そのような function designator が一つの expression に2回以上使われることを禁止するかしなければならない.

2) 〔例 3〕についての注釈

ここで6行目に P という label が存在しないなら

* Remarks to "On ALGOL 60 Language." by Hidetosi Takahasi (University of Tokyo)
** 東京大学理学部

ば，つまりこの中側の block で P が使われなかったならば，7行目の **go to** P; は2行目の P への jump を意味するわけである（block の外の label を中から呼ぶことは許されている）．

3) Parameter called by value に関して

value で呼ばれる formal parameterが procedure の中で assignment statement の左辺にあらわれることは許される．その場合，その formal parameter であらわされる local variable は変ってしまうが，procedure call の中にあらわれる actual parameter である variable には何の影響もない．

これに反して，もしこの formal parameter が name で呼ばれたときは，actual parameter 自身が値を assign されることになる．したがって value で呼ばれるか name で呼ばれるかの差違は明白である．

4) 〔例 6〕〔例 8〕〔例 9〕について

これらはみな注（1）でのべた条件に合わないためにおこる面倒である．なお，この〔例8〕，〔例9〕のようなことに recursive な procedure を使うことは，実際のプログラミングの常識からいえば論外である．また，一般に，recursive に自分自身を呼び出す procedure で，変数が name で呼ばれると，はなはだ面倒な手続きを要する．

5) 3.1 節，3.3.1 節

この節でいわれていることは，すべて注(1) にのべた“副作用”の結果である．

6) 3.3.2 節，〔例 11〕

ここでのべられている ambiguity は，ALGOL 60 を制定したときのミスというべきで，これは，**do** の後の statementを <unconditional statement> とすべきであった．これ以外の proposal は，どれも曖昧な表現を許しておいて何とかそれの解釈を一定しようというもので，compile に難題を提供し，また，使用者が意図したのと違った解釈がなされて，trouble を生ずる以外に益はなかろう．**begin end** を入れるだけではっきりするものを無理に曖昧な文を認めようと努力するのは理解に苦しむ．

7) 3.5 節〔例 12〕

これが sneaky なのは **real procedure** が副作用をもつからであって，variable W が formal parameter としてあらわれないゆえではない．一般の（function designator をあらわさない）procedure に対してここにのべられたような制限を付するのは考えものである．そのため，compiler は余分の手数を負わされる．しかし，不注意な誤りを防ぐ効果はあろう．

8) 3.6 節〔例 13〕

好ましくないのは数字を label に使うことではなくて，こんな奇怪な procedure を考え出した人の頭だろう！