ECMA における COBOL の活動*

西　村　恕　彦**

1. はじめに

COBOL の文法は発表された当時からあいまいな点があって，翻訳ルーチンの作成者も，利用者もいろいろ悩まされた．それにもかかわらず COBOL はたいへんよく普及し，利用されるようになってきた．翻訳ルーチンを手ばやく，あるいは小型計算機用に作るために，外部言語仕様を簡約，必須，選択の三つのレベルの部分集合にわける構想が DOD（アメリカ国防総省）の CODASYL（データ組織言語会議）にあった．

いっぽう COBOL を ISO（国際標準機構）を通じて標準化しようとする動きがあって，これも言語の記述を厳密にし，かつ部分集合をとりきめる作業にかかわっている．そして ASA（アメリカ規格協会）は引受け団体として，ECMA（ヨーロッパ計算機製造業者協会）はヨーロッパの立場から標準案を出しあっている．これは 1965 年現在ではまだ予備作業の段階であって，かたまった案にまでは到達していない．そこで部分集合に関しては CODASYL，ASA，ECMA の三つの案が出たことになるが，それを通読すると次のような印象をうける．

（1） ASA は CODASYL と何の関係もない．

（2） ECMA は ASA と何の関係もない．

ところで最近発行された COBOL 1965 年版をみると，これまでの仕様書にくらべて格段にわかりやすく，また厳密になっていて，これまでにあいまいだった点が多く解決された．その他に次の二点が明記されている．

（3） CODASYL は部分集合を作る作業を放棄する．

（4） 仕様書を作るにあたって，ECMA は全面的に CODASYL と協力した．

このように ECMA が COBOL に関して実質的な作業をしていることがわかったので，1965 年 9 月にヨーロッパに行ったさいに ECMA に立寄った．もちろん会員や委員が常駐しているわけではないが，事務局の Dara HEKIMI 氏と Jean BESSE 氏とに会い，資料を交換して若干の討論をすることができた．それを以下に報告したい．

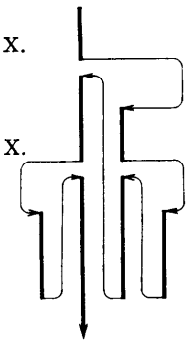
2. COBOL のあいまいな点

情報処理学会の COBOL 研究会で，仕様のあいまいな点とまちがえやすい書き方とをまとめて COBOL Confidentials と称し，その解明にあたってきた．そのうち前者に関するメモ[1]を ECMA に渡した．その各項目を説明し，読者諸氏の参考に供し，できればご意見をうかがいたい．

a. 二重の PERFORM の共通の出口

次のプログラムは文法的に正しいか，正しいとすれば機械語プログラムはどう作るか．ただし PERFORM 命令は FORTRAN でいうと DO と CALL とをいっしょにしたような命令であって，図にみるとおり

```
A.
B.      PERFORM D THRU X.
C.
  D.
  E.    PERFORM G THRU X.
  F.
    G.
    H.
    X. EXIT.
Y.
```



AB-(DE-(GHX)-FGHX)-CDE-(GHX)-FGHXY

という経路をとおると期待される．Xにある復帰機構は FORTRAN の CONTINUE や RETURN とちがって，作り付けにはできない．つまり PERFORM で呼ばれたときには復帰機構として働き，そうでない（串ざし実行の）ときにはまっすぐとおりぬける．

復帰と串ざしとを切りかえること自体はむつかしくない．文法的にいって，二重の PERFORM が完全に入れ子になって（たとえば B. PERFORM D THRU

* COBOL Activities at ECMA, by Hirohiko Nisimura, Electrotechnical Laboratory of MITI.

** 通産省電気試験所

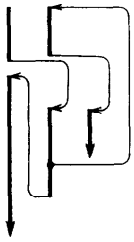
Y.）いればまったく正しい．二重の PERFORM が履行する（たとえば B. PERFORM D THRU H.）ことは明白に禁止されている．

そして上記のような共通の出口についてはあいまいである．これが図のように実行できず，堂々めぐりに入ってしまう翻訳ルーチンもある．図のように実行させるには棚上げ方式や（島内，IBM[4] など），戻り番地を交換してリストにつないでいく方法（島内，MELCOM など）がしられている．

b. PERFORM の範囲の外への脱出

次のプログラムは文法的に正しいか，正しいとすれば機械語プログラムはどう作るか．

```
R.
S.    PERFORM U THRU W.
T.
  U.
  V.  IF …THEN GO TO R.
  W.
X.
```



FORTRAN でテーブルサーチなどしているときに，DO ループの内からその範囲の外へ飛びだすのはごくありふれたやりかたである．つまりループをいろいろな条件によって打ちきったり，または正常に終了させたりして，それぞれに行き先がちがうのである．

そのやりかたで PERFORM を利用すると，a 項を実行**できない**翻訳ルーチンでは何も問題なく b 項が実行**できる**にもかかわらず，a 項を実行できるようにした翻訳ルーチンでは棚があふれたり，復帰機構が無効になったりするおそれがある．

文法書には「入り口（U）から引きつづいて実行していくと出口（W）に到達するようにしておかねばならない」という趣旨が述べられているので，多くの人はこの b 項の書き方は，禁止されているとみなしている．

しかしループからの脱出が条件によって何処へでも行けるようにしたいことがあるので，たとえば IBM では PERFORM の範囲から脱出するときに復帰機構を解除するような算法を考慮している[4]という．しかしこれはたとえば V から R に行って（S に行かないで）すぐに W に戻ってくる（つまり PERFORM の範囲から脱出したわけではなかった）場合が考えられるので，一般には脱出したのかどうか判定できないと思われる．

なおテーブルサーチだけに関しては SEARCH 命令を利用すれば多数の行き先を設定できるので，あえてこのようなあいまいな書き方を使わないでよい．

c. 手続き名のない手続きを書けるか

次のようなプログラムを好んで書く人がいる．つまり，プログラムの始まりへの飛びこしはけっしてあらわれないから，そこへ名前をつけておく必要がないというわけである．

```
PROCEDURE DIVISION.
        OPEN
K.      READ
        COMPUTE
```

しかしこの書き方は 1965 年版では明白に禁止されているらしい．すなわち文の集まりに名前をつけるとパラグラフとなり，パラグラフの集まりに名前をつけるとセクションになるので，パラグラフに属さない文があってはならない．

逆に文がなくて名前だけのパラグラフ（ALGOL でいえば空文）もあってはならない．

d. COBOL で書いたプログラムの終り

ソースプログラムの書き終りの示し方には，プログラム自身に属する命令で内側から終らせる（FORTRAN の END 命令の）やりかたと，外側からモニタカードなどで仕切られるやりかたとがある．COBOL は後者で翻訳ルーチンの手間はかわらないが，ドキュメンテーションの立場からは何かとりきめてあるほうがよいともいう．ECMA でも相当に議論があったが，まとまらなかったと聞いた．

e. 1 桁の英数字情報の字類別判定法

これは文法の問題ではないが，たとえば IBM 1401 のような計算機では +1 と “A” とがまったく同じ形で格納されるので，数字か英字かを判定できない．

f. COBOL 部分集合における予約語の定義

MOVE とか READ とかいう語は固有の意味をもっているので，プログラマがデータや手続きの名前としてかってにつけてはいけない．そこで，ある小さな COBOL が予約語の小さな表をもっていることにすると，その表にない語はデータ名として用いてよいことになり，そのプログラムは部分集合の翻訳ルーチンにはかかるが，大きい翻訳ルーチンにはかけられなくなる．すなわち，上向きの共通性（upward compatibility）が保証されなくなる．

逆にすべての予約語を共有することにすると，記憶容量やチェックの手間がかかる．部分集合の問題は，CODASYL ではタッチしないので，ISO で標準案をきめるときに気をつけるべき事項である．

g. 数式の計算精度と共通性

FORTRAN や ALGOL ではソースプログラムに共通性があっても，計算機によって演算の桁数や四捨五入の仕方がちがうので，最終結果の下位の桁の値が食いちがうことはときどき経験される．

COBOL ではソースプログラムにも，結果のデータにも完全な共通性がたもてるように配慮されている．すなわち名前のついているデータはすべて桁数と位取りとが指定されている．だから演算の中間結果をいちいち格納するように書いたプログラムの結果は厳密にきまっている．ところが長い数式を評価してゆく過程での中間結果の桁数のとりかたは作成者のとりきめによって任意でよく，したがって結果に共通性がなくてもよいときめてある（仕様書，II-1-3，III-7-26）．これでは不十分なのではないかという印象も，一部でもたれている．

h. WRITE 命令と自動的な行送り

あるレコードを印刷すると行が送られるが，一行送ってから印刷するのか，印刷してから一行送るのかがはっきり定義されていない．

i. 報告書における自動的な行送り

報告書機能（report writer）ではいちいち行送りを指定せねばならず，自動的な行送りはできないような印象をうけるが，はたしてそうなのか．

LINE NUMBER IS PLUS 1; と書くと，べたづめに印刷されるのか，一行おきに印刷されるのか．各行ごとにこれを指定せねばならないのか．

j. 小切手改変防止記号（星印）とあたい零

出力データについて有効数字の左側の零を星印でおきかえる指定ができる．たとえば A; PICTURE IS ******; と指定しておいて，MOVE 123 TO A; とやると A のなかには“*** 123”が入る．MOVE 0 TO A; とやると A の内容は“******”となる．銀行の通帳などみていると，払出欄か預入欄のどちらか一方には星印のついた数値が印字されており，他方は空白になっていることが多い．すなわちこの場合に無関係な欄まで全桁が星印で埋まっていると，ずいぶん重苦しいものになってしまうだろう．だから零のときに空白にする書き方が併用できるとよい．

3. COBOL の翻訳

COBOL-61 の書き方の骨組を，英仏独伊の四ヵ国語の対訳の形で示した文書[5]は，かなりまえから日本に来ていた．これら各国語の COBOL は，しかし教育，説明のためだけのもので，公式語および機械にかける言語は**もとの英語のものに限定する**ことを強く勧告している．これを作った目的，動機を次のように説明していた．COBOL を使う層が広くなって，教育，訓練の重要さが増したこと，生半可（?）な英語の知識では理解できない，また誤解しやすい表現があること，放っておくと一国内でもばらばらな訳語が使われるようになるだろうこと，などである．

たとえば MULTIPLY A BY B; は B=B×A の意味である．日本でも講習にあたって，ことに学卒者でない受講者の場合には，筆者の経験では予約語と超変数との双方に少なくとも日本語の注釈が必要で，できれば完全に日本語でやったほうが能率がよい．また解説書を与えても横文字の部分だけは読まないで飛ばしてしまう読者があるという．

この文書では予約語は訳してあるが超変数は英語のままで，それは簡単な単語だからという説明だった．しかし literal（書いたとおりの定数），integer（整定数または整数基本項目）などの語はかならずしも自明とは思われない．

以前の「和訳 COBOL」では予約語，超変数とも英語でおしとおし，そのスタイルが各社の説明書にも踏襲された．しかし ALGOL，FORTRAN の JIS や筆者の教科書では超変数は，日本語か片仮名になっていて，ECMA とは対照的である．

わが国ではローマ字で日本語を表記するのは抵抗が大きく，すでに片仮名（富士通，日電など），日本語（日電）の COBOL が稼動している．計算機用言語を浅くとも広く利用させるねらいと，言語に国際標準を設けるのとではどこかに妥協点をみつけねばならず，日本ではことにそれは困難な作業である．

フランス語の SOAP や FORTRAN，ロシヤ語の ALGOL はその後どうなっただろうか．

なお ECMA のこの仕事と関係があるかどうか聞きもらしたが，私的な仕事で 1965 年版のドイツ訳が出ている由である．

“COBOL Ausgabe 1965”, 29.50 マルク
Mr. H. Hoseit
8, München 23, Osterwaldstrasse 57,

Germany

4. ECMA の提案

ECMA から CODASYL へはかなりきちんとした形でさまざまの提案がいっているらしい[6,7]. そのなかにはわれわれも気付いた 1965 年版の小さな誤り[2]に関する訂正もあるが，実質的な変更提案もある.

割り算の剰余を求める書き方を 1965 年版以後に誰かが提案しているらしい.

DIVIDE A BY B GIVING M, N △ROUNDED,
P REMAINDER R;

これによって A÷B の商が M, N, P に入るとともに剰余が R に入る. ところがその剰余を求める算法が実は相当に面倒なことを指摘して，次の形にするように反対提案している.

DIVIDE A BY B GIVING Q
REMAINDER R, S, T;

すなわち剰余を求める割り算は，切り捨てによるただひとつの商（Q）しか許さないことに制限する.

名詞がいくつかならんだときに，そのあいだは単に間隔で区切られているだけでもよく，コンマをはさんでもよい.

MOVE A TO R S T;

ところがこの規則によると x_{ijk} というような添字の書き方も，コンマがあってもなくてもよいことになる.

X(I J K) または X(I, J, K).

コンマのない書き方はたいへん抵抗が強いから，添字についてだけはコンマを強制してはどうか.

手続き部門においては，手続き名（ラベル）のあとに文を書いたものが手続きとなる. 手続き名のつかない文，文のない手続き名はともに禁止されている. ところが環境部門，データ部門においては本文のない見出しだけを書いてもよいように受けとられる記述があるが，これは禁止してはどうか. ただし筆者はこれに賛成ではない. あらかじめ印刷してあるコーディング用紙を用いるようなときには，本文が空な，きまった見出しがあって差支えないと考える.

正書法の記述中の “a space” は原則として “a space or spaces” と読みかえてよいのではないか. そうでないと， PROCEDURE△DIVISION. は正しいが PROCEDURE△△DIVISION. は正しくないことになる.

ソースプログラムで行をかえると，原則としてさきの行の終りに間隔が規定される. ただしあとの行のはじめに continuation のハイフンがあると，行末，行初の間隔は空とされる. この部分は以前の仕様書ではあいまいで，解釈は各社まちまちだった.

以上のほかに，いわゆるバッカス記法に近い形で，COBOL の文法を定義記述する試みもはじめられている[8,9]. 超言語の記法に，ある要素の長さの制限，桁位置の指定，任意の順序のならべかえなどがつけくわわっている. 作業そのものはごく一部に手をつけたにとどまっているが，いずれは DOD の正式の文書のなかに入るようになると期待される.

5. そのほか

ECMA が COBOL の作業に本腰を入れられるのは，ヨーロッパ全体としての関心の支持があるからで，それはアイビーエム社が COBOL 翻訳ルーチンを納入し稼動しはじめたこの一，二年のうちに急激に高まったものだと聞いた. 国際標準を作る作業も ASA で進んでおり，66 年中に（第 3 回目の？）提案が出て，67 年はジュネーブで会議を開きたい予定だという.

終りに ECMA を訪問する機会を与えて下さった森口繁一先生はじめ関係各位に感謝いたします.

参考文献

1) COBOL 研究会: COBOL Ambiguities, メモ, 1966-9.
2) 西村: Some Questions on COBOL Edition 1965, メモ, 1966-9.
3) COBOL 研究会: COBOL Compilers made in Japan, メモ, 1966-9.
4) 竹下 享: COBOL Ambiguities について，私信，1966-11.
5) ECMA: COBOL Translations, 1965.
6) ECMA: Proposals submitted by ECMA TC 6, ECMA/TC 6/66/24.
7) ECMA: Proposals referring to COBOL Edition 1965, ECMA/TC 6/66/27.
8) ECMA: Formalism for Syntactic Description
9) ECMA: Syntactic Definions of General Nature, メモ

（昭和 41 年 12 月 6 日受付）