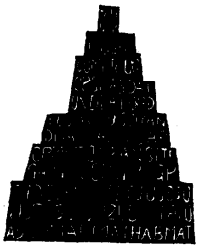

33. 言語設計プロジェクト†

米田信夫††

プログラミングが人間の知的進歩のための絶え間ない戦いであるとすれば，プログラミング言語の設計は正にそのための武器の設計に相当する．設計とは本来人間の欲求と供給可能な技術との間の橋渡し作業であろうが，優れた設計はその両面の可能性を先進的に捉えて，新しい理念を生み，人間の欲求に示唆を与え，先端技術の開発を促す．

プログラミング言語の役割としておよそ考えられる範囲の全体を1つの空間として想定したとき，個々のプログラミング言語が覆う部分は多少とも局部的に限られているし，この空間は技術・知見の進展によって膨張し続ける．プログラミング言語として常に新しい機能が求められ，既存の言語の改定や新しい概念をもりこんだ言語の設計が絶え間なく続くのは，設計者の意欲的・個性的な発想の差や競争による多様もあるとはいえ，本来このような必然性によることである．

現在進行中の言語設計プロジェクトも多い．Ada[1]，CHILL[2] の設計は一段落したようであるが，ESL (European Systems Implementation Language[3])，仕様言語 CLEAR[4] の構想に基づいた CAT[5]，正当性検証を重視した ι-system[6]，強力な集合演算をもって発展中の SETL[7]，Simula 67[8] を継承する Nygaard ほかの Delta/Beta[9] 計画，München グループの CIP (Computer Aided, Intuition Guided Programming[10),11]) 計画，Xerox の Mesa[12] からの発展，等等に際限ない．改定計画として Fortran 82 の声もある．

このような多士済々の中で，ここでは IFIP WG 2.1 による Abstracto 84 計画の概要を紹介しよう．それはこのグループに筆者が関与しているゆえもあるが，とにかくここに世界的な趨勢が反映されていると思われるからである．

● IFIP WG 2.1 と Abstracto 84

国際情報処理連合 IFIP のプログラミング専門委員会 TC 2 の作業グループ WG 2.1 は，IFIP が発足した当初の1962年8月に München で，それまで ACM-GAMM の ALGOL に関する合同委員会に参加していた ALGOL 60 の著者ほかの人達を擁して第1回会合を開いた（日本からは森口繁一現電通大教授が参加）．それ以来 ALGOL 60 の保守がこのグループに引継がれた仕事になっているほか，一般にこれを継承するような汎用の算法言語の設計がこのグループの目標となった．その後 N. Wirth, C. A. R. Hoare, G. Seegmüller らの提案による ALGOL W の検討を経て，Aad van Wijngaarden を中心として設計された ALGOL 68 を1968年12月 München における第11回会合で成立させ，8年後の1976年には第22回会合で改訂 ALGOL 68 を定め，この間に I. D. Hill の起草による修整 ALGOL 60 の審議もしている．また，機関誌 Algol Bulletin を発行している．

現在の主査は N. Y. U. の R. B. K. Dewar；幹事はカナダ Manitoba 大の P. R. King；メンバーとして国際色豊かに名を連ねているのは F. L. Bauer, H. J. Boom, S. R. Bourne, P. Branquart, J. Darlington, A. P. Ershov, G. Goos, P. G. Hibbard, I. D. Hill, P. Jorrand, I. O. Kerner, C. H. A. Koster, C. H. Lindsey, B. J. Mailloux, J. J. Maluszynski, L. G. L. T. Meertens, S. G. van der Meulen, H. Partsch, M. Paul, J. E. L. Peck, W. L. van der Poel, S. A. Schuman, J. T. Schwartz, M. Sharir, M. Sintzoff, R. Uzgalis, E. Wada, A. van Wijngaarden, N. Yoneda の面々である．メンバー予備軍であるオブザーバの層も厚い．

この国際的なグループが現在目差しているのが，プログラムの要求仕様から（実行の対象であるよりは思考の対象であるような）抽象的プログラム，（効率を追求する）具体的プログラムまで，広い範囲の表現媒

† Report on an On-going Language Design Project by Nobuo YONEDA (Faculty of Science, University of Tokyo).

†† 東京大学理学部

体となる目論見の Abstracto 84 である．プログラム開発の中核的推進媒体としては，機械依存性を離れて人間向きの高水準言語を使う方がよいことはもちろんであろうが，さらに抽象データ型の思考方向に添い具体的な算法技術から離れて，要求仕様と抽象的プログラムの間の世界で作業を進めるのがよい；具体的プログラムへの変換は別フェイズの仕事と考える．こういう作業方式を支援するプログラミングシステムの設計が目標である．

Abstracto の名称は，WG 2.1 第24回 Jablona (ポーランド) 会合から使われたらしいが，第25回 Summit (米国) 会合で，Amsterdam からのオブザーバ L. Geurts がその由来を次のように述べている："In a programming class I suggested that we consider a problem in abstracto; later a student asked where he could obtain details of the language Abstracto!" 数字84の方は，ALGOL 60, 68, その改訂を定めた'76年という等差数列の次項である．

立派な構想と名称はできたが，中味の方はまだ模索の段階である．1979年12月第26回 Brussels 会合では，事前に出された宿題8問に対して，多数の参加者それぞれの流儀によるスケッチの展開が提示され，より具体的な Abstracto の枠組を探る討議がなされた．その後，3問に絞った改定問題が配布（この3問については1980年清里での夏のシンポジウム[13]でも議論された）され，1980年8月の Oglebay (米国) 会合で再論された．1981年5月の Nijmegen (オランダ) 会合では，toy problem も結構だが，宿題問 1A "電話帳管理" のようなもっと大規模の問題もしっかり扱おうということになっている．そろそろ具体的な提案も欲しい頃である．第29回は1982年1月東京でという声もあったが英国カンタベリーでという予定になっている．

● 最長上昇列の問題と P+P の解答

前述の宿題中で問7が最長上昇列 (Longest Upsequence) の問題である．これは，もともと NATO 夏期講習会で Dijkstra が出したもので，彼自身の模範解答も示されている[14]．問題の中味は，

> 自然数Nと，整数の長さNの列Aとが与えられるものとして，Aの左から右へ適当に飛び飛びに拾って得られるような部分列 (2^N 通り) のうちで整数の列として単調非減小であるものの最大長を求めるプログラムを作れ

というものである．素朴な攻法ではせいぜい $O(N^2)$ のプログラムが得られるが，模範解答は O(Nlog N) になっているので，天才的飛躍なしでも模範解答以上のものに達するような攻法のレールを敷け．これが改定問題7Aとしての趣旨である．この問題に対して，日本のメンバーとしても文献[13]にあるような趣旨の解答を Oglebay 会合に提出したが，同会合に München グループの Partsch+Pepper がもたらした解答[15]が，Abstracto の考え方を比較的良く反映していると思われるので，以上にその概要を若干修整して紹介しよう．

1. 準備段階

線形順序 "≤" をもった集合 m が（問題の '整数' に相当して）与えられているとし，場合分け上の例外を避ける便宜上，これに仮想元 $-\infty, +\infty$ を追加して集合 $\underline{m}^+$ を作り，すべての $x \in \underline{m}$ に対して $-\infty < x < +\infty$ と約束する．

抽象データ型として m 上の有限列とそれに付随する操作などを次のように規定する．

```
type SEQUENCE≡(sort m) seq, empty,
                append, hdr, last :
sort seq,
seq empty, denote empty by ⟨ ⟩,
funct (seq, m) seq append,
    denote append (a, x) by a & x,
funct (seq) seq hdr,
funct (seq) m+ last,
laws ∀ seq a, m x : hdr (a & x)=a,
  last (a & x)=x, hdr (⟨ ⟩)=⟨ ⟩,
  last (⟨ ⟩)=−∞
endoftype
```

ここで sort というのはデータ型の台集合を示す．'hdr' は 'header' のつもりであるが，hdr や last から undefined が生じないようにしてあることに注意．このデータ型に，問題の表現のため次の定義を補っておく．

```
funct (seq) nat length,
    denote length (a) by |a|,
funct (seq, seq) bool contained,
    denote contained (a, b) by a⊂b,
funct (seq) bool isups,
laws |⟨ ⟩|=0, isups (⟨ ⟩)=true,
∀ seq a, b, m x, y : ⟨ ⟩⊂b=true,
  |a & x|=|a|+1, a & x⊂⟨ ⟩=false,
```

$a \& x \subset b \& y = a \& x \subset b \vee (a \subset b \wedge (x = y))$,

isups $(a \& x) =$ isups $(a) \wedge$ (last $(a) \leq x$),

mode upseq ≡ seq u : isups (u)

ここで mode は sort に条件を付けて制限したものを意味する.

2. 問題の展開

問題は seq A : |A| = N に対して

M = maxlength (A)

を求めることとなる. ただし,

maxlength (seq a) nat

≡max{|u| : $u \in$ upseqset (a)}

=max{nat i : ∃ $u \in$ upseqset (a) : |u| = i},

upseqset (seq a) set upseq

≡{upseq u : $u \subset a$}.

この upseqset の性質を調べると,

upseqset (⟨ ⟩) = {⟨ ⟩},

upseqset $(a \& x)$ = {upseq u : $u \subset a \& x$}

= {upseq u : $u \subset a \vee$ (hdr $(u) \subset a \wedge$ last $(u) = x$)}

= upseqset $(a) \cup$ attach (upseqset (a), x).

ただし, funct attach は

≡(set upseq s, m x) set upseq :

{upseq u : hdr $(u) \in s \wedge$ last $(u) = x$}

と定義されるもので,

attach $(s, x) \neq \phi$

⇔ ∃ upseq $v \in s$: last $(v) \leq x$

⇔ min{last (v) : $v \in s$} $\leq x$

という性質がある. この min は max と共に m の有限部分集合に対して定義されるもので, 空集合 ϕ に対しては min $(\phi) = +\infty$, max $(\phi) = -\infty$ と約束する.

つぎに, プログラム設計上の方針として, 構造のない集合 upseqset (a) に, 問題の要求に合せて長さによる構造をもたせることとする. すなわち,

upset (seq a, nat i) set upseq

≡{upseq u : $u \subset a \wedge |u| = i$}

と定めれば, maxlength (a) は

= max {nat i : upset $(a, i) \neq \phi$}

によって計算されることになる. upset の性質としては

upset $(a, 0) = \{\langle\ \rangle\}$, upset $(\langle\ \rangle, i+1) = \phi$,

upset $(a \& x, i+1)$

= upset $(a, i+1) \cup$ attach (upset (a, i), x)

などが得られるから, 前記 attach の性質から

upset $(a \& x, i+1) \neq \phi$

⇔ upset $(a, i+1) \neq \phi$

$\vee \min\{\text{last}(v) : v \in \text{upset}(a, i)\} \leq x$

となる. この min の値を関数として,

repr(seq a, nat i) m$^+$

≡min {last (v) : $v \in$ upset (a, i)}

と定めてやれば, repr$(a, 0) = -\infty$, repr $(\langle\ \rangle, i+1) = +\infty$ でまた upset $(a, i) \neq \phi$ ⇔ repr $(a, i) < +\infty$ となるので, maxlength $(a) =$ max{nat i : repr $(a, i) < +\infty$}. しかも,

repr $(a \& x, i+1)$

=min ({repr $(a, i+1)$} ∪ {x : repr$(a, i) \leq x$})

=if repr$(a, i) \leq x \wedge x <$ repr$(a, i+1)$

then x else repr$(a, i+1)$ fi

のように, repr はそれ自身で再帰的に定められるので, upset のような集合の管理はしないですむようになる. また, maxlength の方は, ⟨ ⟩ に対して 0 から出発して, 再帰的に

maxlength $(a \& x)$

=if repr$(a,$ maxlength$(a)) \leq x$

then maxlength $(a)+1$ else maxlength (a) fi

と計算される.

3. うまい話から抽象的プログラムへ

前記の関数 repr(a, i) は, i について単調非減小, すなわち

∀ nat i : repr$(a, i) \leq$ repr$(a, i+1)$

という都合のよい性質をもっている. それで seq a と m x に対して, $0 \leqq i_0 \leqq$ maxlength(a) なる nat i_0 で, repr$(a, i_0) \leq x <$ repr (a, i_0+1) となるものが一意的に存在する. この i_0 すなわち

that nat j, $0 \leqq j \leqq$ maxlength (a) :

repr$(a, j) \leq x <$ repr$(a, j+1)$

を用いると, repr$(a \& x, i)$ は

=if $i = i_0+1$ then x else repr(a, i) fi

となり, repr(a, i) から repr$(a \& x, i)$ を更新するべき i は, $i = i_0+1$ の所だけになる.

さて, 所要の M = maxlength (A) を再帰的に計算するのに登場する seq a は A の頭から途中までの部分列, m x は A のある番号 k 番の要素 A_k である. そこで, このような A への参照は,

mode index ≡ nat j : $j \leqq$ N

によるデータを介することとし, 'A_i' のような表現も認めることにすれば, 次のような (抽象的) プログラムができる.

「funct maxlength ≡ (index k) nat :

if $k = 0$ then 0

```
    elif repr(k−1, maxlength(k−1))≤x
    then maxlength (k−1)+1
    else maxlength (k−1)                fi;
  funct repr≡(index k, i) m⁺:
    if k=0 then if i=0 then −∞ else+∞ fi
    else index i₀
          ≡that index j : j≦maxlength (k−1):
            repr(k−1, j)≤A_i<repr(k−1, j+1);
      if i=i₀+1 then A_i else repr (k−1, i) fi
    fi; maxlength (N)                    」
```

4. プログラムの変換

上記のプログラムを得てから後は，プログラムの変換を司る各種のレパートリをライブラリから呼び出し，構文上のパタン合せをコマンドで指示するなどして，プログラムを変形して行く思い入れである．2変数の関数 repr を関数を返す関数 (index k) funct (index) m として再構成する機能，再帰的な計算を下からの積み上げ型に変換する機能，有限変域の関数に対する変更の管理を配列の更新で置換える機能などを駆使して

```
「var nat m:=0; index array m r
        :=(−∞, +∞, ···, +∞);
  for k from 1 to N do
    index i₀
    ≡that index j : j≦m : r[j]≤A_i<r[j+1];
    if m=i₀ then m:=m+1 fi;
    r[i₀+1]:=A_i
  od; m
```

のように，i_0 の決定に2分法を用いれば全体として O (N log N) が達成されるプログラムが得られる．

参 考 文 献

1) Reference Manual for the Ada Programming Language. United States Department of Defense, (July 1980). (複製：プログラム言語 Ada 基準文法書(英文)，bit 別冊 1981. 1 (共立出版))
2) C. C. I. T. T. Study Group XI: CHILL Language Definition, Period 1977-1980 Report, (May 1980).
3) ESL Report on Main Task I (May 1979), Report on Main Task II (Oct. 1979), Siemens/ Cii Honeywell Bull.
4) Burstall, R. M. and Goguen, J. A.: The semantics of CLEAR, a specification Language. Proc. of the 1979 Coppenhagen Winter School on Abstract Software Specification, Lecture Notes in Computer Science, Vol. 86, pp. 292-332. springer-Verlag (1980).
5) Goguen, Joseph A. and Burstall, R. M.: CAT, a system for the Structured Elaboration of Correct Programs from Structured Specifications, Technical Report CSL-118 SRI International (Oct. 1980).
6) Nakajima, R., Nakahara, H. and Honda M.: Hierarchical Program Specification and Verification, a Many-Sorted Logical Approach. Technical Report, Research Institute for Mathematical Sciences, Kyoto, Japan (Jan. 1979).
7) Dewar, R. B. K., Grand, A., Liu, S. C.: Schonberg, E., and Schwarz, J. T.: Programming by Refinement, as Exemplified by the SETL Representation Sublanguage, ACM Trans. on Programming Languages and Systems, Vol. 1, No. 1, pp. 27-49 (July 1979).
8) Dahl, O.-J., Myhrhaug, B. and Nygaard, K.: Common Base Language, Publ. No. S-22, Norwegian Computing Centre (Oct. 1970).
9) Kristensen, B. B., Madsen, O. L., Møller-Pedergen, B. and Nygaard, K.: An Introduction to Beta Programming Language (Language Version as of August 1979). DAIMI IR-20, Computer Science Dept., Aarhus Univ. (Aug. 1979).
10) Bauer, F. L., Broy, M., Ganatz, R., Hesse, W. and Krieg-Brückner, B.: Notes on the Project CIP: Toward a Wide Spectrum Language to Support Program Development by Transformation, TUM-INFO-7722 Tech. Univ. München (July 1977).
11) Bauer, F. L., Partsch, H., Pepper, P. and Wössner, H.: Notes on the Project CIP: Outline of a Transformations System, TUM-INFO-7729 Tech. Univ. München (July 1977).
12) Mitchell, J. G., Maybury, W. and Sweet, R.: Mesa Language Manual, Version 5. 0. CSL-79-3, Xerox Palo Alto Research Center (Apr. 1979).
13) 「作譜工程の評価・改良・自動化」，1980 夏のシンポジウム報告集，情報処理学会—プログラミングシンポジウム委員会.
14) Dijkstra, E. W.: Some Beautiful Arguments Using Mathematical Induction, Acta Informatica Vol. 13, pp. 1-8 (1980).
15) Pepper, P. and Partsch, H.: On the Feedback between Specifications and Implementations: An Example, Working Paper OGY-5, IFIP WG 2. 1 Meeting at Oglebay Park (Aug. 1980).

(昭和 56 年 4 月 10 日受付)