① $C=A+B$
② $A=2$
③ $E=C\times D$
④ $B=1$
⑤ $DB-A$

図-1 単一割り当て言語のプログラム

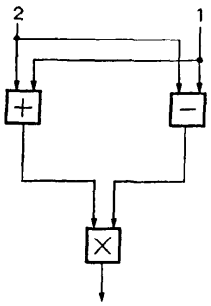

図-2 データ・フローグラフによるプログラム

各代入文はプログラム上の位置とは無関係にオペランドの揃った代入文②と④，①と⑤，③の順（すなわちデータ依存関係）で実行できる．

図-2 のデータ・フローグラフの場合も同様で，各ノードを評価する手順はトークンの入出力依存性（アーク）で表現されている．

従来，関数型言語といえば LISP のような関数の適用を中心とした作用的言語を意味していたが，単一割り当て型言語，データ・フローグラフ言語の出現は関数型言語の新たなプログラミングスタイルを作り出したと考えても良いのではないだろうか．

参 考 文 献

1) 特集 プログラミング言語の最近の動向，情報処理，Vol. 22, No. 6 (1981)
2) Henderson, P.: Functional Programmig Application and Implementation, Prentice-Hall International Series in Computer Science, London (1980).
3) Landin, P. J.: The Next 700 Programming Languages, CACM, Vol. 9, No. 3, pp. 157-166 (Mar. 1966).
4) Landin, P. J.: An Abstract Machine for Designers of Computing Languages, IFIPS 65, pp. 438-439.
5) Arvind, Gostelow, K. P., Plouffe, W.: An Asynchronous Programming Language and Computing Machine, TR 114a, Dept. of Information and Computer Science, University of California, Irvine, California (Dec. 1978).
6) Tesler, L. G. and Enea, H. J.: A Language Design for Concurrent Processes, AFIPS Conf. SJCC, Vol. 32, pp. 403-408 (1968).
7) Chamberlin, D. D.: The "Single-assignment" Approach to Parallel Processing, AFIPS Conf. NCC, Vol. 39, pp. 263-269 (1979).
8) Comte, D., Durrieu, G., Gelly, O., Plas, A. and Syre, J. C.: "Parallelism, Control and Synchronization Expression in a Single Assignment Language," SIGPLAN Notice, Vol. 13, No. 1, pp. 25-33 (Jan. 1978).
9) Dennis, J. B.: First Version of a Data Flow Procedure Language, Lecture Note in Computer Science, Vol. 19, Springer-Verlag, pp. 362-376 (1974).

(昭和 56 年 8 月 24 日受付)

小長谷氏のコメントに対するコメント

横井俊夫（電子技術総合研究所）

小長谷氏のコメントの骨子は，単一代入（割当）規則等による記述が作用的でないことを理由に，関数型言語を大きく，作用的言語とデータ・フロー言語にわけるべきであるというものである．一方，筆者の解説 (81 年 6 月号"関数型言語") は，λ-計算を出発点に，様々の拡張や修飾を施すことにより，有用な言語要素の多くが，関数型言語という枠組の中に包含されるというのが骨子である．単一代入規則も，そのような拡張の 1 つと見なすというもので，陽には述べられてはいないが，データ・フロー言語という言語カテゴリは，あえて設けるほどのものではないというものである．

いずれにしろ，単一代入規則をどう見立てるかが意見の別れめである．λ-計算の拡張の 1 つと見立てた方が妥当だという理由の大筋を以下に述べる．

代入という行為は，(変数) 名のついた記憶領域にある値を書き込むということと，それ以後は，その名前で，値を参照できるようにするという 2 つのことからなる．単一代入規則は，後の機能のみを利用するというものである．この副作用なしに名前付けを行うという仕組は，ラムダ変数への結合ということによって実現できる．

以上は，解説でふれたことである．たとえば，小長谷氏の例は，次のようなラムダ式で表される．

```
((λ(A)
  ((λ(B)
    ((λ(C)
      ((λ(D)
        ((λ(E)
          ……)
         (C×D)))
       (B−A)))
     (A+B)))
   1))
 2)
```

このラムダ式の計算を具体的にどう進めるかは，また別の問題である．外側から順番にリダクションを進めると逐次計算となり，引数の値が求まればリダクションを行うとすれば，データ駆動型の計算となる．ラムダ式による表現では，深い入れ子となるが，これを単一代入文の集合とすることにより，この難はさけられる．

いささか，こじつけめくが，関数への名前付けの仕組（小長谷氏が，作用的言語に入れた機能）も単一代入規則そのものと見ることができる．したがって，この名前付けも，ラムダ結合として表すことができる．ただし，再帰呼び出しに対しては，不十分であることは良く知られていることである．

単一代入規則では，代入文

$C=A+B$;

は，"$A+B$ の計算値に C という名前を付ける" と読まれる．さらに，これを等式と見立てる．あるいは，A, B, C, 3つの変数間に成り立つ関係を表すものと見立てる．

$+(A \quad B \quad C)$;

すると，関係（述語論理）型言語となる．

これは，言語の基本的な分類としては，関数型と関係型を取り上げるべきだとの一例である．この2つの言語は，連続的に関連づけることができ，基本的な性質は同種の言語である．ただし，この言語を具体的に計算する機構は，解説でもふれたように，大きく3種類が考えられる（次表）．

言語＼計算機構	置き換え型	環境評価型	駆 動 型
関 数 型	λ-計算を忠実に行うリダクション・マシン等.	通常計算機上のLisp 等.	データ・フロー計算機上の実行形態.
関係(論理)型	多くの定理正明プログラム.	PROLOG のインプリメンテーション.	研究中で定説をうるにいたっていない. 関数型より，一段複雑になる.

このように，2つの言語と，3つの計算機構という分類をもとにし，色々な事柄を整理すべきである．特に現在はそうすべきで，データ・フロー言語というものを持ち出す理由も利点もあまりない．もちろん，データ・フロー言語という言葉が，世の中で使われていることも事実である．しかし，データ・フロー計算機の研究者達が，その計算機用の高級言語として考案した関数型言語という程度の意味しか持たないと思われる．言語としてのある特徴を持ったカテゴリ名としての意味は，あまりない．関数型にしろ，論理型にしろ，データ・フロー計算機上で実行させるか否かで，言語の本質的な仕様に差が生ずるということはない．皆無と断言するわけではないが．

単一代入規則は，計算結果に名前をつけ，他所で参照するという機構を，即物的に実現する手段であるが，これは，データ・フロー系以外でも取り入れられてしかるべきものである．意味を明確に把握できるようにしたいという発想から生まれた Lucid の繰返しの記述法は，Uインタプリタによって，ループ1つ1つをも並列実行させたいという発想からの Id の記述法に同じになる．tail transfer によって終端再帰を効率良くしたいという SCHEME の発想にも一致する．ストリームの考え方は，関数型言語の枠組の中で経過依存性をどうとらえるかという一般的な議論のうちの1つのアプローチである．I-structure の考え方も，副作用をなくし，複雑なデータ構造を効率良く実現する手法で，Lisp のリスト構造からはじまって，すべての関数型，関係型言語に共通に工夫すべきことである．etc.

以上のように，基本的には，すべて関数型，関係型言語というカテゴリの中で整理されるべき一般的なことがらである．Lisp 等の既存のものに比べ，いわゆるデータ・フロー言語は，より関数型の本来の姿にもどろうとするものとみることができる．もちろん，そうきれい事ばかりではない．対象とするデータ・フロー計算機のアーキテクチャの個性を，その言語仕様に反映することになる．たとえば，VAL は，実行中はフロー・グラフは固定されたままという最も初歩的なデータ・フロー計算機を対象としているので，コンパイル時にグラフを決定できるような仕様となっている．こうなると，言語そのものというより，データ・フロー計算機そのものの是非を議論しなければならなくなる．それは，また別の場所で大いに議論すべきである．データ・フロー計算機の研究が安定期に入り，

アーキテクチャの一般な分類がなされているならまだしも，言語そのものをも含め，すべて再整理しようというのが現状であるから，なおさらである．

また，データ・フロー言語と呼ばれているものを，さらに単一代入系とデータ・フロー・グラフ系に分類するのも，それほどの意義は，認められない．ただし，Davis (Utah 大) 等の GPL のような言語は，別の観点から1つの言語カテゴリを形成すると思われる．図形言語ともいうべきもので，高密度の TV-ディスプレイが一般的ツールとなるこれからのプログラミングを考えると，新しいタイプの言語を目指しているといえる．現在，何箇所かで，この種の言語が研究されているが，母体となっているのは，関数型と関係型の言語である．

いずれにしろ，以上の事柄は，最もホットな話題の1つで，様々の意見があり，今後の研究に期待されるものが大である．筆者の意見は，とりあえず，現状の基本的な骨組みを明確にすることを目的としたため，いささか細部をはしょりすぎたきらいがある．

ご批判，ご検討を乞う．

(昭和56年10月14日受付)