リストの概念*

藜沼　良一**

1. はしがき

これまでの電子計算機の使い方は，計算機という名前が示すように，計算ということが大部分であった．したがって，扱われる情報も操作も，数値に関するものが主体をなしていた．

ところが，計算機を数値計算でない問題に利用することが多くなると，計算機の設計，プログラムの方法が，今までどおりのものでよいかどうかが問題になってくる．たとえば，数値計算でなければ，浮動小数点のような演算は不用になるだろう．

それでは，算術的でない問題にはどんな操作が必要か，また，情報の扱い方はどんな方法がいちばんよいのかということになると，現在のところ，ほとんどわかっていないというのが実情である．けれど，リストというのは，この問題に対してかなり役立つもので，特に LISP のような考え方は，この方向の機械計算やプログラム法の第一歩をもたらすものと思われる．

2. 人間と計算機の違い

電子計算機を算術的でない問題に利用することを考えるには，人間と計算機との違いを見ておくことが重要である．これにより，最もよい方法が見つかるかも知れないからである．

そこで，まず，計算能力について考えてみよう．人間は足し算とか掛け算とかをするには，何か補助器具を必要とする（たとえば，紙と鉛筆，ソロバン，電動計算機など）．これらができると称する人でも，その能力はきわめて低く，電動計算機や電子計算機に比べれば，ばかみたいなものである．

それから，人間は非常によく間違える．計算違いをするのはもちろん，読み違い，聞き違い，いい違い，書き違い等々，間違えるのが当然のように思われている．

このほかに，人間は疲れたり，あきたりするなど，機械としてはかなり性能の低いものであることがわかる．にもかかわらず，機械よりすぐれた働きをすることも事実である．たとえば，薬屋の前を通りかかったとき，歯みがきを買うのを思い出したりする．

この思い出すという働きは，計算機にはちょっとできない事柄である．歯みがきを買うというのを，計算機に行なわせるとしたら次のようになろう．

（1） 歯みがきが必要かどうかを質問し，

（2） 薬屋がこれに適した店かどうかを判断する．

ところが，人間は絶えず，歯みがきについて考え続け，店が変わるたびに検討しているわけではなかろう．

それから，人間はかなり論理的でない働きをする．たとえば

甲：昨日，学会で会った背の高い人の名前は，何といったかな．

乙：知らないけど，たしか，タで始まっていたように思う．

甲：ああ，そうだ．山田さんだ．

このような働きも計算機では，どうしたらよいのか，わからないであろう．

ここで，人間と計算機との違いを見つけるのが目的ではないが，算術的でない問題の扱いは，これまでの計算機に対する考え方では解決できないことに気がつかれるであろう．

リストが万能とは思えないが，ほんのわずかではあるが，人間と計算機の違いを，のりこえる方向に進めるのではないかと思う．

3. リスト

計算機のメモリは普通，0番から順々に番号がついていて，情報の扱いやプログラムの実行も，この順に行なうのが便利なようにできている．情報の記憶を例にとれば，次のように並べるのが普通である．

メモリの番号	情　報
K+1	1番目の情報
K+2	2番目の情報
…	…
K+N	N番目の情報

したがって，命令や情報が直線的に配列できる場合にはきわめて便利である．しかし，高次元の行列や樹枝状の情報配置のように，問題によっては，直線的に配列するのが非常にめんどうになる．

さらに，このような配置法には，少なくとも次のよ

* An Introduction to Lists. by Ryoiti Tadenuma (Electrtechnical Laboratry)

** 電気試験所電子計算機部

うな二つの欠点がある．

（1） この配列の中間に，新しい情報を加えようとすると，それ以下の情報をすべて動かす必要がある．

（2） 情報を加えたとき，次の表に割り込む可能性がある．したがって，他の表まで移動させなければならなくなるかも知れない．

リストあるいはリスト構造というのは，順々に配列していく方法をとらず，この欠点をさけようとするものである．次に，リストのいろいろな種類を示そう．

3.1. リスト-1

次の例に示すように，情報と一しょに，次にくる情報の位置をも併記する形式のものである．各欄の意味は，次のとおりである．

リスト－1

A	B	L
1	2番目の情報	7
…	……	…
3	1番目の情報	1
4	最後の情報	0
…		
7	3番目の情報	4

A：情報の位置
B：情報の内容
L：次の情報の位置

このリスト構造で，4番のL欄に0とあるのは，これがリスト構造の最後であることを示す．

新しい情報を加えるには，L欄を1カ所直すだけでよい．たとえば，2番目と3番目の情報の間に，新しい情報を加えると，次のようになる．すなわち，1番のL欄を8に直し，8番に新しい情報をおけばよい．

新しい情報の追加

A	B	L
1	2番目	8
…		
3	1番目	1
4	最後	0
…		
7	3番目	4
8	新しい情報	7

逆にある情報を，この表から除くのも同じようにL欄を直すだけで簡単にできる．

3.2. リスト-2

リスト－1では，最初の情報から始めて，次々に情報を求めていくのはL欄だけでできるから，きわめて簡単である．しかし，逆方向，すなわち，最後の情報からさか上ってゆくのは，このままでは困難である．そのためにはけっきょく，前後両方向の位置を示せる形式にする必要がある．

次のは，このリストの例である．この構造のものならば，前後のつながりが容易に求められるが，欄が一つふえる欠点がある．

人間は一連の情報のつながりがあると，前からは比較的容易にさがすが，その逆はあまり得意ではなさそうに思う．たとえば，イロハ歌で，タの次はと聞かれると，はじめからたどっていって，レであることを答える．逆方向に，ンからはじめるということは，ほとんどできない．そうすると，この形のリストはあまり重要でないかも知れないという気もする．

リスト－2

A	B	F	L
1	これ	0	2
2	は	1	3
3	本	2	4
4	です	3	0

F：前の情報の位置

3.3 リスト-3

次の二つの文をリスト-1で表わしてみよう．

（1） あれは本でない．

（2） これも本である．

この文には，「本」と「で」が共通している．このように，一つの情報がいくつかの表に現われるとき，そのつど，これを配置するのでは経済的でないし，人間も本をいくつも記憶しているとは思えない．そこで，本を唯一つで済まそうとすると，1の欄にいくつも記入する必要がでてくる．そこで，いくつもあるときは，何コかにわけて入れる必要がある．

リスト－3

A	B	L	Y
1	本	2	0
2	で	3	5
5		4	0
3	ある	0	0
4	ない	0	0

上のリスト-3はY欄を新しく作り，この欄によって，何回も行なわれているかどうかを判断するようにしたものである．

この形式のリストでは非常にむだが多く，使用回数だけ入れたのと同じ程度になる．このむだをなくするために考えられたのが，次のリストである．

3.4. リスト-4

この構造のものは，情報の種類を集めた字引と，配列を示す接続表とに分け，接続表のみをリスト構造で表わしたものである．先の例文を表わすと，次のようになる．

字引

情報	位置
これ	1
あれ	2
は	3
も	4
本	5
で	6
ある	7
ない	8

接続表

A	B	C
1	4	0
2	3	0
3	5	0
4	5	0
5	6	0
6	7	9
7	0	0
8	0	0
9	7	0

この接続表A，C欄が0であれば，Bが情報の位置を示すが，0でないときは，その位置を調べると，別の接続が与えられるようになっている．

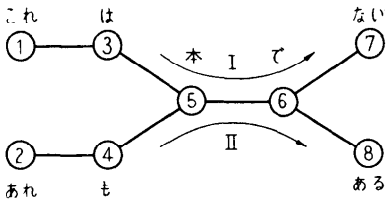
3.5 リスト-5

前のリスト構造をよく調べると，「で」の次が「ある」か「ない」かのどちらかになるかを決めることができないことが知られよう．したがって，原文（1），（2）のほかに

（3） これも本でない．

（4） あれは本である．

という文が作られてしまう．この原因は，次の図より明らかであろう．すなわち，⑤，⑥を通るとき，①からはじまる流れIと，②からはじまる流れIIが混同してしまうことによるのである．



原因がわかれば，助けるのは容易である．すなわち，リスト-2 型の接続表に，も一つの欄を作り，混乱しそうなところには流れを区別する印をつけておけばよい．これがリスト-5 におけるS欄の意味である．

この型のリストによると，単語を組み合わせた句というものを，機械的に作れる利点がある．たとえば，単語「で」と「ある」とのつながりが非常に多く使われれば，A-Tの組合せにおいて，6-8となるものが多く現われよう．このようなものは，A, T, L あるいはA, F, L の組合わせを調べるだけで見つけることができる．

リスト-5

A	F	T	L	S
1	0	3		
2	0	4		
3	1	5		
4	4	5		
5	3	6	10	
10	4	6		1
6	5	8	11	1
11	5	7		
7	6	0		
8	6	0		

A：情報の番号
F：前の情報の番号
T：後の情報の番号
L：別の情報の流れ
S：流れを区別する印

「で」と「ある」の組合わせが非常に多ければ，これをつないだ「である」を字引に入れたほうが経済的になる．このような過程で，新しい句を作らせることができる．この過程は人間が新しい型を構成する場合とも非常に類似しているのではなかろうか．

字引と接続表とを用いれば，情報を構成するいっさいの資料がそろえられるから，どんな処理操作も接続表だけで行なえるわけである．しかし，処理結果をとりだすには，はじめに用意した字引だけではきわめてめんどうになる．このため，いままでの入力用の字引のほかに，ちょうど，逆の構成になる出力用の字引を用意する必要がある．もちろん，字引ひきのような操作ができれば，一方だけで済ますことができる．

4. リストの処理

前節においては，いろいろの型のリストを紹介したから本節は，これらのリストを処理することについて述べよう．

リストでは，情報資料だけでなく，処理法，すなわち，プログラムも情報資料の一種と考え，リスト構造で表わす．このため，字引に相当する基本操作（基本単語）と，接続表に相当するプログラムとに分けて説明するのが都合がよい．

4.1. 基本単語

どのような操作を基本単語に選ぶかは，処理言語によって異なり，少なくないものは数個，多いものは数百個になっている．しかし，基本となるような操作は，次のようなものであろう．

〔除去〕：リスト構造の一部を除く．

〔挿入〕：リスト構造の一要素を，別のリスト構造でおきかえる．

〔分割〕：ある要素で，リスト構造を分割する．

〔置換〕：ある要素をすべて別のリストあるいはリスト構造で，おきかえる．

〔調査〕：ある条件を満たすリストをさがす．

4.2 接続表

これは，それぞれのリストに応じた形式のものが用いられ，基本単語を用いて作られたプログラムをリスト構造にして表わす．

サブルーチンなどは，前節で説明した句の一種と考えることができるが，これを字引に追加するような操作は，どのリスト処理言語にも用意されている．そうして，これは，基本単語と接続表との組合せを処理するようになっている．

5. IPL-V と LISP

いろいろの型のリストを紹介したが，これらのすべてが開発されているわけではない．本節では，代表的なリストと，その処理言語について紹介する．詳細に

ついては，それぞれの文献を参照されたい．

5.1. IPL-V

RAND において開発された一連の IPL (Information Processing Lauguage) の5番目のもので，おそらく，最も多くの機種に使えるようになっているものであろう．

IPL に使われるリストは，**3節**のリスト-1 型のものと，ほとんど同じである．単純な構造と，複雑な構造とに分かれ，次に示すようになっている．

単純なリスト構造

NAME	PQ Symbol	Link
L 0	0	36
36	S 1	50S
50S	S 2	I 3
I 3	S 3	0

NAME の欄は，メモリの番号に相当するものである．PQ Symbol の欄は，実際の情報を示し，LINK は情報のつながりを示している．

この例で，L0 の PQ Symbol の欄が0になっているのは，これが単純なリスト構造であることを示している．

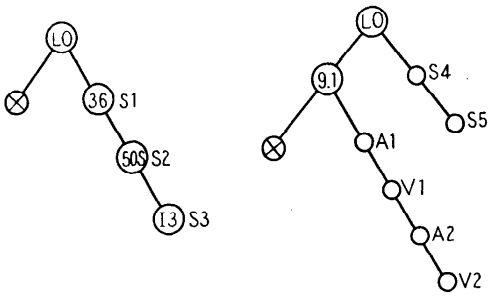
複雑なリスト構造のものは，次の例に示すような型式に作られる．

複雑なリスト構造

NAME	PQ Symbol	LINK
L 0	9.1	
	S 4	
	S 5	0
9.1	0	
	A 1	
	V 1	
	A 2	
	V 2	0

この例では，L0 において，9.1 なるリストに分枝していることが示される．リスト9.1は単純なリスト構造になっている．

この二例を木の形で表わすと，それぞれ次の図のようになる．



リスト構造の例

この言語には，リストの処理，演算，入出力，その他で約150種の単語（命令）が用意されている．プログラムは，これらのリスト構造に作られるから，情報と単語との混同をさけるため，単語は特定の文字で始まるようにしている．このため，この文字で始まる NAME や SYMBOL の使用は許されない．

新しいサブルーチンや回帰的ルーチンの作成のために，三種の独得な単語が用意されている．

5.2. LISP

MIT において，McCarthy によって開拓されたもので，基本単語が非常に少なくないのが特徴である．

LISP で扱う情報の最小単位は，ATOM と称する文字や記号の列である．ATOM をさらに小さい単位に分解する操作は，LISP には用意されていない．

リストの形式は，リスト-1 とほとんど同じであるが，B 欄，L 欄の両方に，情報でも，つながりでも書けるようになっている．

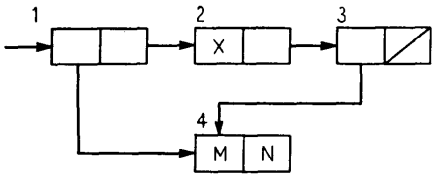
LISP のリスト

N	A	D
1	4	2
2	×	3
3	4	0
4	M	N

上の構造は

$$MN \times MN$$

を LISP のリストで表わしたものである．



LISP の基本単語は，次の6種だけである．

car [x] : x を与えて，そのA部を求める．

cdr [x] : x を与えて，そのD部を求める．

cons [x; y] : x, y を，それぞれ，A部，D部とするトリスを作る．

atom [x] : x が ATOM かどうかを調べる．

eq [x; y] : x と y が，同じ ATOM かどうかを調べる．

cond [$e_1 \rightarrow f_1$; ……; $e_n \rightarrow f_n$] : e_1 より，順々に，その真偽値を調べる．そして，最初に真になった e に対応する f を選ぶ．

この6種を組み合わせて，新しい操作を作って，リストの処理を行なう．この構成規則を M-言語と名づ

けている．これに対して，ATOM と，その構成規則を S-言語という．M-言語と S-言語との混同をさけるため，M-言語は小文字，S-言語は大文字で正別している．

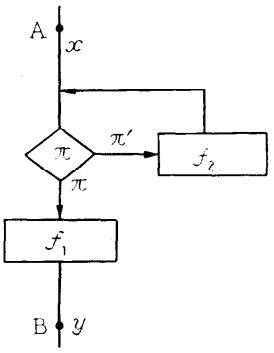
このように分けると M-言語は，S-言語を値域および定義域とする関数と考えることができる．基本的な LISP は，このようなものであるが，M-単語をも定義域に入れるように，拡張されたものが使用されている．

実際に，LISP を計算機で扱うには，M-言語を S-言語に翻訳する必要がある．そのために，いくつかの翻訳の規則が与えられている．

LISP は，6種の基本単語の組合せだけで，いっさいの操作を行なうのであるから，どの程度の能力のものかを確かめておく必要がある．そこで，言葉を一つ定義しておこう．

計算機の状態は，状態変数というベクトル x を考えれば，一意に記述できるとしよう．これに対して，別の状態変数 y があり，y が x より，基本単語の組合せで得られるとき，y は LISP で記述可能ということにする．

いま，次の図のような流れ図で表わされる処理を考えよう．記号を，次のように決めておく．



x：入口A点における状態変数
y：出口B点における状態変数
f_1, f_2：LISP で記述可能な函数．
π：LISP で記述可能な論理函数
f：この流れ図により表わされる函数

そうすると，f は次の式で記述できる．

$$f(x)=\mathrm{cond}\,[\pi(x)\rightarrow f_1(x);\ \pi'(x)\rightarrow f\{f_2(x)\}]$$

この式で，f 以外は，すべて LISP で記述可能であるから，f の範囲は，帰納的函数論によって与えることができる．

上の例では，特別な流れ図について考えたが，一般の場合は，多変数の帰納的函数論によって基礎づけられる．

6. あとがき

リストの応用まで述べることはできなかったけれど，以上でリストの概念は明らかになったことと思う．

リスト処理とプッシュ・ダウン・メモリとの関係についても説明できなかったが，簡単に述べておこう．

IPL, LISP, いずれも，リスト-1 型のものであるから，逆順に処理するのは，きわめて困難である．これを解決する一便法として，情報の分岐する位置だけを記憶しておくという方法が考えられている．これに都合のよいのがプッシュ・ダウン・メモリというものである．

このメモリはちょうど，棚のようなもので，上へ上へと積み重ねていく．したがって，この棚のものを使おうとすると，いちばん上のものしか取り出せない．このような構造のものが，このメモリである．情報の分岐点をこの棚にどんどんのせていくと，逆行するとき，ちょうど，都合よく，いちばん新しい分岐点が求められるというわけである．

参考文献

1) John McCarthy, Recursive Functions of Symbolic Expressions and Their Computation by Machine, Comm. ACM March, 1960.
2) Lisp 1.5 Programmer's Manual MIT Computation Center and Research Lab. Rand corp. 1960.
3) 高須　達，記号操作と定理の機械的証明，第3回プログラミング，シンポジウム，1962.
4) 五十嵐滋，List Processing について，情報処理，4，5，(1963)
5) 蓼沼良一，込山敏子，リスト処理言語 情報処理学会機械翻訳研究委員会 1963.
6) A. Newell, F.M. Tonge, An Introduction to Information Processing Language V. Comm. ACM, March, 1960.
7) A.J. Perlis, Charles Thornton, Symbol Manipulation by Threaded List, Comm. ACM. March. 1960.
8) Peter Kugel, Programming Techniques for Processing Clues and Hints, Computer and Automation, Aug, 1963.
9) P.M. Woodward and D.P. Jenkins, Atoms and Lists, Comm. ACM, Mach. 1960.
10) Daniel G. Borrow, Bertram Raphael, A Comparison of List-Processing Computer Languages, Comm. ACM, April, 1964.
11) R.W. Hsu, Characteristics of four List-Processing Languages, NBS Report No. 8163, Sept, 1963.

（昭和39年9月19日受付）