リスト処理手法にもとづく文献情報検索について

Rep. Fac. Sci. Kagoshima Univ.. (Math. Phys. Chem.) No. 7, pp. 11–19. 1974

リスト処理手法にもとづく文献情報検索について

富樫昭・藤野精一

ON INFORMATION RETRIEVAL BASED ON LIST PROCESSING TECHNIQUE

By

Akira Togasi and Seiiti Huzino

(1974 年 9 月30日受理)

Abstract

In this paper we have shown that list processing techniques can be applied effectively to the field of information retrieval. We use ternary tree structures as basic elements of list structure by modifiing list processing language LISP which is also modified to be able to use base fields and bugs in L^6 freely in order to increase applicability of list processing techniques in the field.

1 はじめに

最近コンピュータ利用技術の一つとして，リスト処理の手法がしばしば利用されるようになり，非数値計算の分野に応用されてきた（[1], [2], [3]）。

リスト処理（List processing）とは与えられた情報をリスト構造の形でコンピュータの記憶に入れ，それを処理してリスト構造の形で結果を得るコンピュータの応用技術である。（[4]）したがって，このようなリスト処理手法を応用しようとする場合に考えなければならない点は，まず

1° 与えられた情報をどのような形のリスト構造で表現するか？

2° 表現された情報をどのように処理することができるか？

という2点である。この2点の解決は，ともに処理する言語そのものに依存して定まる性質のものである。従来リスト処理言語として有名なものには，代表的なものとして，LISP と L^6 とがある。これらの言語の応用された分野は定理の自動証明，時間割の自動作成，論理回路の自動設計等範囲は非常に広い。

本論文ではこの手法を適用して応用分野の一つである文献情報の収集と検索を実行してみようと思う。処理する言語としては，従来ある言語そのままでは応用分野のもつ性質上不十分である。そこで LISP と L^6 のもつ特長をともにあわせもつようなものを新しく考えることにする。そうすればこのような問題を効率よく実行できるようになる。

ここでは LISP を基調にしてそれを変形して L^6 の機能を若干付与するように改良した言語を

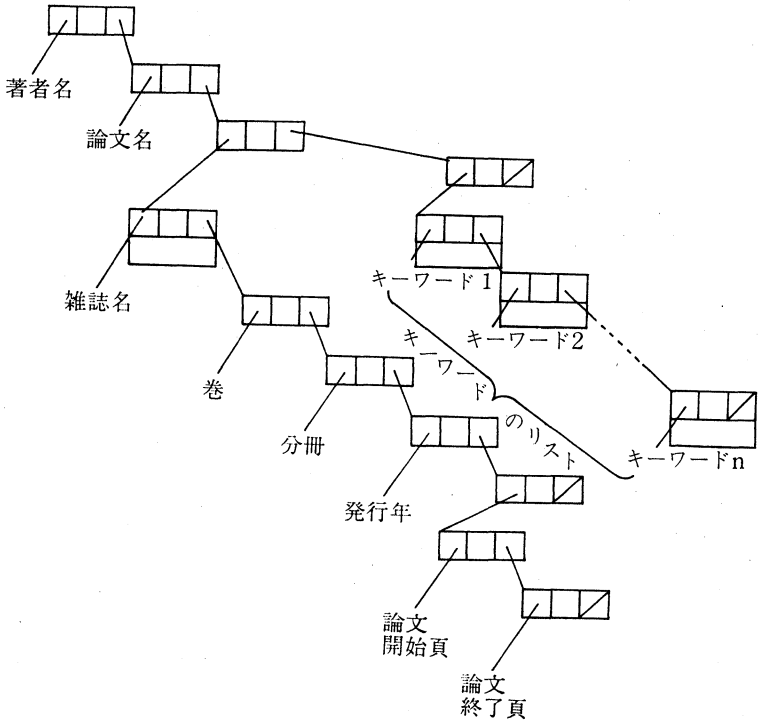
Akira Togasi (Department of Mathematics, Kagoshima University.)
Seiiti Huzino (Department of Mathematics, Kyushu University.)

使用することにした([4], [5])。

改良点の第1は，従来 LISP では2分岐木構造を情報表現の基本においたが，そのままでは記憶をたどることはできても，もとにかえることに非能率的な点があるので，ここでは3分岐木構造を基本におくことにする。この構造は連想記憶を構成するのに都合がよく，したがって検索と整理とを効率よく実行することができる。

2 単位文献リストの表現

まず与えられた文献をコンピュータに入れるには，個々に与えられる各文献をどのような形で(コンピュータの外部および内部で)表現するかを考察する必要がある。そのため，変形する分岐木構造をもとにした次のような構造（第1図）を個々の文献に与えるものとする。



第1図

通常の LISP ではリストを構成する構造部分におかれる語では1語に car 部と cdr 部に対する指示子が入るだけであるが，ここでは1語を3部分にわけ，左の場を car 部，右の場を cdr 部とし，新しく設けた中央の場を ctr 部と称することにする。ctr 部は情報の連想に使用するため新しく設けたもので，使用法についてはこれからのべる。さらに必要に応じて，2語で1個のブロックを構成しその2語ブロックの2番目の語は（この情報がどこから生じたかを示す）情報の原点想起作用に利用することにする。したがって，LISP のリスト構造を作成する場合は，LISP の記法をそのまま使用し，これらに連想想起作用をふくませる場合に，新しく入れた ctr 部が活躍することになる。ctr 部の使用については，その操作を簡略に記すため，次のような2つの関数を導入する。

(i) follow[l; m]
(ii) ctr [e]

follow $[l; m]$ は l が一般に LISP でいうところのリストで，m が F-リストとよばれる新しく ctr 部の使用で作成された構造のとき定義される。F-リストは次のように定義される。

<F-リスト> ::= <リスト>| (<リスト>|<F-リスト>)

すなわち LISP でいうリストは F-リストである。リストと F-リストとの中央部に | を入れてカッコ“( )”でくくってできるものは F-リストを表わす。実際はこの記法は F-リストの外部表現であって，これを内部で構成する操作を関数 follow が表わしている。

follow $[l; m]$ は l がリスト，m が F-リストのとき定義されて，l の根の部分の ctr 部に F-リスト m の指示子を入れて結合する作用をさす。これが記号で $(l|m)$ で表わされる。すなわち，

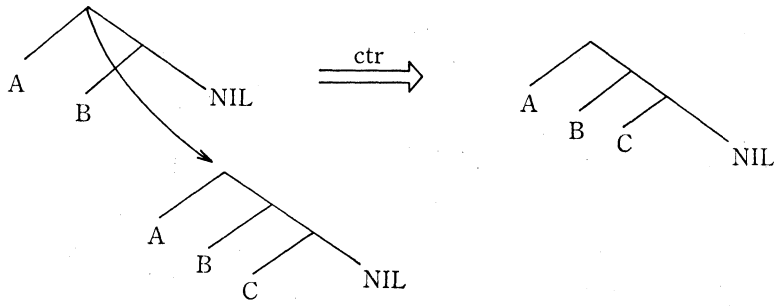
$$\text{follow}\ [l; m] = (l|m)$$

こうすれば l の根の ctr 部に入った m の指示子により m を l から連想することができる。

また ctr $[e]$ は e がリストではない F-リストであるとき，すなわち，e の根の ctr 部に NIL と異なる内容が入っているときに定義され，この根の ctr 部の内容が指示する F-リストをとることを意味する。

たとえば

$\text{ctr}[((A\ B)|(A\ B\ C))] = (A\ B\ C)$ である（第2図参照）



第2図

第2図の根の部分から矢印がでているのは ctr 部に指示子が入っていることを示す。他の表現については [1] 参照。この follow と ctr を導入することによって文献の収集と検索の操作を記述するのが容易になる。第1図の説明にかえろう。第1図は個々の文献をコンピュータが格納した際の内部表現を示している。この内部表現は次の特長をもっている。

（イ）全体は情報構造部分の各語に ctr 部が存在することと，数個所に2語ブロックを使用している点を除いて LISP におけるリスト構造を形成している。

（ロ）各連結単位の ctr 部にはいまのところ何も入っていない。

（ハ）雑誌名とキーワードの連結部分のみが2語ブロックである。

この内部表現を記憶に入れるために使用する外部表現を，次のように規定することにする。

(著者名　論文名　(雑誌名　巻　分冊　発行年　(論文開始頁　論文終了頁))
(キーワード 1　キーワード 2 ··· キーワード n))　(1)

この外部表現は LISP の表現の借用であるが，文献をコンピュータに入れる場合のデータはこの形以外を使用しないことにすれば，LISP の表現をそのまま使用して第1図に示すように内部表現を構成することは，システムをすこし改良することによって容易になすことができる。2番目の‘(’のところで2語ブロックを作成し，それをでた次の内部リストの中の各キーワードの部分の連結に

はすべて2語ブロックを使用して連結すればよいからである。

この際，著者名，雑誌名等はあらかじめ定められた簡略記号で書くようにしておく。また論文名は LISP でいう特性リスト（property list）を使用して書くことにして，それへの指示子を入れるのみにしておけば簡単に記述されるであろう。

一つの文献を入れる際，F-リストの構成部分に要する語数は，キーワードを n 個とすれば第1図からわかるように

$$10+2(n+1)=12+2n$$

である。これに各基本単位である著者名，雑誌名等を各1語と数えると，全部で $8+n$ 個あるから，終止記号 NIL とあわせて

$$12+2n+8+n+1=21+3n$$

語が各単位文献の格納に必要な語の総数である。これは $0\leqq n\leqq 10$ のとき，

$$21\leqq 21+3n\leqq 51$$

であるので，$n=10$ のときでも 51 で，さして多い数ではない。

このような構造の F-リストを単位文献リストといい，第1図をその内部表現とする。外部表現は(1)で表わされる。

3 文献情報の拡充

単位文献リストを一般に e で表現することにする。e をコンピュータに入れる際に，

(i) car[e] はこの文献の著者名，
(ii) cadr[e] は論文名，
(iii) caddr[e] はその論文の掲載されている雑誌についての情報，
(iv) cadddr[e] はこの論文のもつキーワードのリスト

を示していることを注意しておく。このとき単位文献リストをリストにして保存する文献リストの他に，一つの文献ごとに現われる著者名，雑誌名，キーワードをいつもリストにして保存しておいて，新しい情報がつぎつぎに入るたびに，これを新しくしていけば，文献検索を実行する場合に準備ができて非常に都合がよい。それで，単位文献を記憶に入れる際に同時に，次の4個のリストを拡充することにする。

d ····· 文献リスト，
a ····· 著者名リスト，
m ····· 雑誌名リスト，
k ····· キーワードリスト。

以後これらを取り扱うのであるが，他の種類の問題と異り，ここで問題とする文献情報検索の場合には，各 d, a, m, k は新しく情報として，単位文献リスト e が入るたびに更新されて保存されねばならないという点がある。

そのため，LISP の処理手段だけでは保存の面で難点が生じ不十分である。これを補うため，L[6] の虫の機能と基地場の概念とを LISP に補充して拡張する。

いま $u:=v;$ で F-リスト v を記憶にコピーして，それを F-リスト u とよぶという場合の記法とし，

$$u\leftarrow v$$

で F-リスト v に (L^6 でいう) 虫 u をくっつける。すなわち，基地場 u の指示子として，v への指示子を入れることを示すことにする。

これらの機能を LISP に付加すると，各 e を入れる際のコンピュータの作用を簡潔に記すことができる。

文献がコンピュータの中になにも入っていない最初の状態を

$$d:=\mathrm{NIL};\quad a:=\mathrm{NIL};\quad m:=\mathrm{NIL};\quad k:=\mathrm{NIL};$$

として4個の空リストを作成し，しかる後，単位文献リスト e が コンピュータに入ってくる際の各リスト d, a, m, k への処理がなされねばならないが，それらの処理もそれぞれ d-処理，a-処理，m-処理，k-処理と略称することにする。d-処理からはじめて，この順に a, m, k と実行して e を記憶に入れる作業が完了することになる。この全体を in[e] と書くことにすると，in[e] は

begin d-処理; a-処理; m-処理; k-処理 **end**

の形をとっている。

各処理について説明しよう。

(1) ***d*-処理**　単位文献リスト e をそのまま現在の d の前にくっつけて，新しい d を作成する。

$$d:=\mathrm{cons}[e;\ d];$$

これで d-リストを拡充したことになる。

この d-リストが中心になって，処理がつぎつぎに実行される。

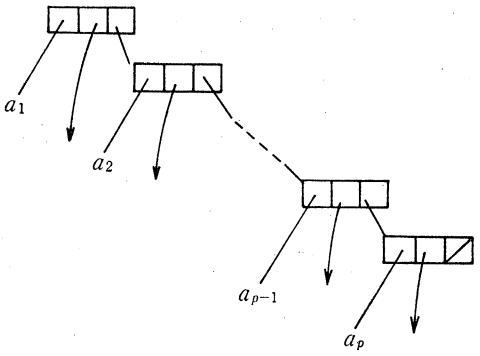
(2) ***a*-処理**　d-リストの car の部分には現在の e が入っている。caar[d] は論文 e の著者名を示すから，現在の a-リストの中に caar[d]，すなわち car[e] と等しいものがあるかどうかをさがす（以後簡単のために，caar[d] のかわりに car[e] と書いて説明をつづける。このようにしても混同の恐れはないであろう。d-リストをもとにして考えることに変りはない。）現在の a-リストを

$$a=(a_1\,a_2\cdots a_p)$$

とする。ここに a_i $(i=1,2,\cdots,p)$ は著者名である。そして上のリスト a は LISP のリスト構造の外部表現をそのまま使用しているが，実際は第3図の示すように，リスト構造の他に，ctr 部がそれぞれある指示子をもっているものである。

この指示子はその著者の論文がすくなくとも1回，過去にコンピュータに入れられたことを示すのに使用されている。

a-処理は次の場合にわけられる。



第3図

(イ) car[e] がどの a_i ($i=1, 2, \cdots, p$) とも異なるとき:
このときはいままでこの著者の文献は記憶に入っていなかったのであるから，car[e] を現在の a-リストの前にくっつけて新しい a-リストとし，その後に ctr[a] に e の指示子を入れる，すなわち，

$$a := \text{cons}\,[\text{car}\,[e];\ a];$$

$$\text{follow}\,[a;\ e];$$

を実行する。(e のところは car[d] であることを注意，以下同様とする。)

(ロ) car[e] がある a_j と一致するとき:
この場合は a_j から後をまとめたリストを A_j としたとき，ctr[A_j] から著者を a_j とする従来の論文がすべて引き出されるようになっているから，その前に新しい同じ著者の論文 e を引き出せるようにするために，現在の ctr[A_j] を ctr[e] に入れ，しかる後に ctr[A_j] に e の指示子を入れることにすれば，このことが達成される。L^6 のリスト要素の増設手法と同様である。

$$\text{follow}[e;\ \text{ctr}[A_j]];\ \text{follow}[A_j{:}e];$$

この A_j を実際に適用するには a の虫 A を作成してこれを動かすことによって達成する（後述)。以上で著者名を中心とする a- 処理は終了する。

(3) ***m*-処理**　a- リストと同じように m-リストは雑誌名のリストが ctr 部を同じ雑誌の論文を連結するために使用されて構成されている。そこで，この論文には，caaddar[e]，すなわち caaddr[e] が e の雑誌名であるから，その内容が現在の m- リストの中にあるかどうかを調べる。

(イ) 現在の m- リストに caddr[e] がないとき
この雑誌名は記憶に入っていない新しい雑誌名であるから，m-リストの前につないで a-処理と同じように論文をひきだせるようにする。

$$m{:}= \text{cons}[\text{caaddr}[e];\ m];$$

$$\text{follow}[m;\ \text{caaddr}[e]]$$

しかし，これだけでは e の示す論文全体の情報が得られない。そこで関数 recall[$l{:}\ m$] を導入する。
この作用はリスト l の根の部分が2語ブロックになっている場合に適用されて，2語ブロックの第2番目の語にリスト m の指示子を入れるというように定める。そうすると recall[$l;\ m$] の実行後，l の指示子がわかれば l から m をよびだすことが可能となる。現在の場合は caaddr[e] より母体の e, すなわち caddr[e] の関係するもとの情報をよびだす必要があるので，上の処理の後にひきつづいて，

$$\text{recall}[\text{caddr}[e];\ e]$$

を実行すればよい。

(ロ) m-リストの中に caaddr[e] と一致するものがあるとき，
このときはこの雑誌の論文が一編以上すでに記憶に入っている。現在の m-リストを

$$m = (m_1\, m_2 \cdots m_q)$$

とする。m_i ($i=1, 2, \cdots q$) は雑誌名で，m の内部表現は第3図の a の内部表現と同じく ctr 部は同じ雑誌の論文を連結するのに使用されているとする。

$m_1, m_2 \cdots$と順に比較して，r 番目の m_r と caaddr[e] とが一致したとする。このときも a-処理の(ロ)と同じような操作を $(m_j\, m_{j+1} \cdots m_r)$ に実行した後，recall を適用すればよい。

follow[caddr[e]; ctr[$(m_j\, m_{j+1} \cdots m_r)$]];

follow [$(m_j\, m_{j+1} \cdots m_r)$; caaddr[$e$]];

recall[caddr[e];e];

(4) **k-処理**　k-リストは

$$k = (k_1\, k_2 \cdots k_s)$$

のようになっていて，各 ctr 部はキーワードを同じくする論文の連結に使用されている。そこで，e の中のキーワードのリストは cadddr[e] でとれるから，このリスト cadddr[e] の中の各キーワードを最初から順に現在の k-リストの要素と比較し，一致するものがあれば follow と recall とを前に使用した m-処理の場合のように使用して連結を完了する。

一致しない場合も m-処理と同様，新しく k-リストの前に一致しなかったキーワードをくっつけて follow, recall 操作を m-処理の(イ)の場合と同様に実行する。

以上で各単位文献リスト e をコンピュータに入れる場合の4個の基本処理である d-処理，a-処理，m-処理，k-処理の機能の説明を終了するが，これを実際に実行するには L^6 のように虫を自由に動かしてやるといちいちリストのコピーをしないですむので，記憶場所がすくなくてすみ，能率的である。たとえば，a-処理を実行した後，a-処理を虫を使用して実際に実行する手順を考えてみよう。ここで a-処理に使用する虫を A とする。この A を使用しての a-処理は次のようにまとめられる。

```
                    A←a;
TREAT:  if equal [A; NIL] then
        begin a := cons [caar[d]; a];
              follow [a; car[d]]
        end;
        if equal [caar[d]; car [A]] then
        begin
          follow [car[d]; ctr[a]];
          follow [a; car [d]]
        end;
        A←car [A]; go to TREAT;
```

ここで注意したいのは $A \leftarrow a$ で a の指示子が A に入るから，car[A] を car[a] と同じに使用できる点である。a をそのまま保持して a の部分構造にある操作をしたときなどは，この L^6 の虫機能は大変重宝である。上の処理手続きは，全体を ALGOL 形式にしてそれに L^6 と LISP 記法を併用してよみやすくしたものである。このような言語で処理できるようシステムを組んでおく。m-処理，k-処理についても，a-処理と同様の虫が活躍することはいうまでもない。本質的な進め方はa-処理の場合と同様であるので，詳細は省略する。

4　文献情報の検索

a-リスト，m-リスト，k-リストを用意した文献リスト d を，現在保有するコンピュータから，

なんらかの文献情報を得る基本的な操作は，つぎの3つである。

(i) キーワードを与えて，そのキーワードをもつすべての文献情報をとり出す，

(ii) 雑誌名を与えて，その雑誌名をもつすべての文献情報をとり出す，

(iii) 著者名を与えて，その著者のすべての文献情報をとりだす。

この3つを組み合わせて，高次の文献情報の検索が種々考えられるが，ここではこの最初の3つの操作を考えてみよう。これらが最も基本的であるからである。この3個の操作を上から順に K-検索，M-検索，A-検索とよぼう。fを検索のために与えられた情報とする。fは次のリストの形で与えることにする。

$$f=(f_1\ f_2)$$

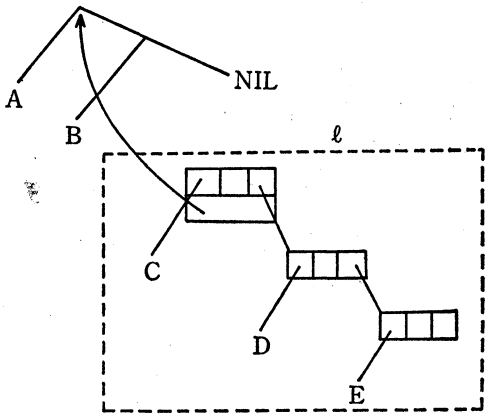
ここに f_1 は K,M,A のいずれかで f_2 は $f_1=K$ のときキーワード名，$f_1=M$ のとき雑誌名，$f_1=A$ のとき著者名とする。このとき情報 f を与えて cadr[f] に一致する文献をすべてとりだす操作を search[f] としよう。いま F-リスト e に対して，print[e] で F-リストの ctr 部を無視してできるリスト e を印刷することを表わすことにする。このとき，search[f] は次のように3つの部分にわかれる。

(1) K-検索：$f_1=K$ のときは，f_2=cadr[f] はキーワードとなっている。このときは，k-リストに cadr[f] と一致するものがあるかどうかを調べる。もし一致するものがあれば（いまこれを k_i としよう）k_i の ctr 部からひき出されるリストをすべて印刷すればよい。

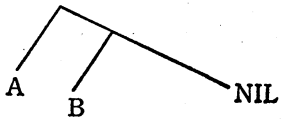
```
                              K←k_i;
RETRIEVAL: if equal [ctr [k]; NIL]
           then go to NEXT;
           print [base [ctr [k]]; k ←ctr [k];
           go to RETRIEVAL;
NEXT:
```

ただし，ここで使用した base[l] は l の根が 2語ブロックのとき定義されて，l の根の2語ブロックの第2番目の語の内容を指示子としてとりだされるリストをさす。たとえば，l が第4図のようになつているとき，base[l] は第5図のようになる。

cadr[f] に一致する k_i をさがす操作は，虫を導入すれば前と同様にできることはいうまでもない。cadr[f] がどのキーワードにも一致しないときは，'SORRY' を印字することにする。



第4図



第5図

(2) M-検索
(3) A-検索

これらの検索については K- 検索と同様な処理をおこなうが，ことなる点は，M-検索の場合は base 処理が必要であるのに対して，A- 検索の場合はその必要がないことである。一致する著者名を見出せば，ctr 部を伝わってつぎつぎとりだし，ctr 部の内容が NIL となるまでくりかえせばよい。

5 結び

以上でリスト処理にもとづく文献情報検索の一試案を簡単に説明した。この方法は検索方法をつぎつぎに高次に改善していくことができる。最初に考えられるのは，

1 キーワードと雑誌名とを与えて，その雑誌の中でこのキーワードをもつ文献をすべてとりだす，
2 キーワードと著者名とを与えて，その著者のこのキーワードをもつ文献をすべてとりだす，
3 雑誌名と著者名とを与えて，その雑誌から，この著者のすべての文献をとりだす

などが考えられる。また，

4 雑誌名と巻数とを与えて，その雑誌のその巻の文献をすべてとりだすとか
5 キーワードと年数とをあたえて，このキーワードをもち，現在よりその年数までさかのぼってあらわれる文献をすべてとりだす

とか，あるいは5の場合を雑誌名，著者名で制限した

6 キーワードと年数と雑誌名とをあたえる，
7 キーワードと年数と著者名とをあたえる

とかその他いろいろ考えられる。また，

8 2次文献リストの構成，すなわち，各単位文献の参考文献をすべてとりだす，

これを一般化して，n 代までさかのぼってさがす，

9 n 次文献リストの構成 ($n \geqq 2$)，

なども考えられる。1～7 までは今まで方法を容易に拡張して実行できるが，8, 9 は単位文献をコンピュータに入れる際に，2次文献リストを入れておく必要があるので，問題が少し難かしくなる。この問題は今後研究すべき問題である。文献情報を利用する者にとっては，必要な情報獲得の手段でもあるので，開拓されるべき問題である。本質的には数個のスタックを併用して解決されよう。

参 考 文 献

[1] S. Huzino, List processing and linguistic analysis (Japanese), Asakura Pub. Co. (1970), pp. 192.
[2] S. Huzino, On an application of list processing techniques to a recognition problem, (to appear).
[3] S. Huzino, On an application of L^6 to the theory of automata, (to appear).
[4] J. McCarthy et al, LISP 1.5 Programmer's Manual, The M.I.T. Press, (1962), pp. 99.
[5] K.C. Knowlton, A programmer's description of L^6, Comm. ACM., 9 (1966), 616–625.