記号処理 4 － 4
（1978. 5. 19）

生成順序と圧縮型くず集め

寺島 元章 （電気通信大学計算機科学科）
後藤 英一 （東京大学理学部情報科学科）

1. 序

生成順序（genetic order）と呼ばれる線型順序を、リスト構造の記憶領域上のアドレスを用いて確立できること，及びこの生成順序を保存するリスト処理系、特にくず集め（garbage collection）について報告する。 生成順序は、そのリスト処理系がこの生成順序を保存する場合に限ってそのリスト構造が生成された時点で規定される。 その場合、リスト処理系は生成順序を保存するくず集めを行う必要がある。 生成順序を保存するくず集めで、現役（使用中）のリストセルを連続した領域に再配置する機能をもつものを圧縮型生成順序保存（compactifying genetic order preserving）くず集めと呼ぶ。 圧縮型生成順序保存くず集めで、既存の技法の処理時間、使用領域量に問題のあるポインター補正に関して、二つのポインター補正技法を考案した。 生成順序木を使用したポインター補正法と、補正ビット累算器を使用した高速ポインター補正法である。

次に、退役のシステム作成記号を要素にもつ属性リストを回収する逆行（backward）くず集めについて述べる。 この様な属性リストは、もはやそれを参照する事ができないため、使用済のものであり、回収可能である。

2. 生成順序と圧縮型生成順序保存くず集め

生成順序と呼ぶ線型順序を、そのリスト処理系がこの生成順序を保存する場合に限って、リスト構造にその記憶領域上のアドレスを用いて確立することができる。 この場合、処理系のくず集めは、生成順序保存型でなければならない。 LISP 1.5 [12] の古典的くず集めは、生成順序保存型であることに注目すべきである。 また Wegbreit [17] の圧縮型（compactifying）くず集めも生成順序保存型であるが，Bobrow [3]，Cheney [4]，Hansen [9] の圧縮型，詰込み型（compacting）くず集めは生成順序保存型ではない。（詰込み型とはリストの cdr セルを除去し現役のリストセルを連続した領域に再配置することを意味し [4, 9]，圧縮型とはリストの cdr セルを除去せずに再配置を行なうことを意味する。[3, 17]）

検索，分類と併合の多くの技法は，データーのある種の線型順序を利用しており、種々のデーター構造に対する演算の高速化に適用、応用されてきた。 これらの技法すべてが、生成順序保存の処理系で生成順序を利用することによりリスト構造に容易に適用できる事を指摘しておく。 生成順序は、未だほとんどのリスト処理技法で使用されていないが，その生成順序の使用の有効性を指摘することは十分に価値のあることであろう。 そこでまず、リスト処理技法、特にくず集めに関して生成順序保存型と非保存型に分類した。

生成順序保存くず集めは三段階から成る。 現役セルのマーキング，ポインター補正，セルの再配置である。 マーキング と再配置のアルゴリズムは比較的平易であり、高速アルゴリズム（記憶領域に比例した時間量で行える。例えば [10] 参照）も容易に設計できる。 ポインター補正では、そのよく知られた技法は、前二者と比較して、時間、記憶領域、あるいは時間と記憶領域をより多く要する。従って、ここではポインター補正の技法だけを論じる。 一つのポインター補正

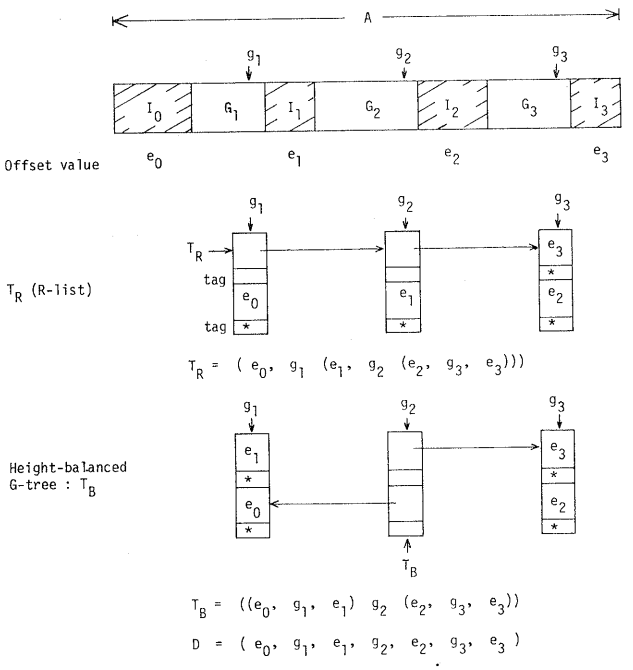
技法は、生成順序二分木（genetic ordered binary tree，略して生成順序木と言う）－二分木で木の頂点のアドレスを頂点のデーターとして扱うもの－を使用する。この技法は外部記憶を必要としない。　また、この技法は生成順序を利用したアルゴリズムの例でもある。　すなわち、生成順序の生成順序係存処理系のインプリメンテーションへの自己適用例である。　他の一つは、高速補正技法（1個のポインター補正を O(1) 時間で行う）で、ハードウェアインプリメンテーションに適すると考えられるが、補正ビット累算器（offset register）のための外部記憶を必要とする。

2.1　生成順序木を使用したポインター補正技法

Aをアドレス空間とする。　ポインター補正段階の開始時点で、Aは、現役セルから成る島々（islands）と、それらの島々の間の間隙（gap(s)）に分割されているものとする。（図1参照）

$$A = I_0 \cup G_1 \cup I_1 \quad \cdots \quad G_n \cup I_n$$

ここに I_i は i番目の島で、G_i は島 I_{i-1} と I_i の間の間隙である。　Aが一島だけである様な場合は考えないものとする。　一般性を失う事なく、すべての島々とその間隙とは空でないと仮定してよい。　現役のポインターが I_i に属して



A : the address space
I_i: i-th island
G_i: gap between I_{i-1} and I_i
e_i: the offset value of I_i
g_i: an address chosen from G_i

図1．島、間隙と生成順序木表現

いれば、そのポインターは補正値 e_i, すなわち I_i が再配置時に移動する量により補正される。g_i でそれぞれの間隙 G_i から選ばれたアドレスを表わすものとすると、ポインター p を補正するための補正値は次の条件式で定義される。

$$\text{補正値} = \begin{cases} e_0 & p < g_1 \text{ ならば} \\ e_i & g_i < p < g_{i+1} \quad (1 \leq i < n) \text{ ならば} \\ e_n & g_n < p \text{ ならば} \end{cases} \tag{2.1}$$

明らかに、この条件式は、補正及び間隙データー e_i と g_i の $2n+1$ 組 $D = (e_0, g_1, e_1, \cdots g_n, e_n)$ で十分に規定することができる。

定義 1. D の生成順序木とは、入れ子となった順序付組の各二番目の要素が g_i ($1 \leq i \leq n$) である様にカッコ対を挿入する事で得られた順序付組として定義される。(図1参照) 各組 $(T' \; g_i \; T'')$ は生成順序木の頂点と呼ばれる。ここに T', T'' は、補正値 $e_0, e_1, \cdots e_n$ (木の葉) の一つであるか、あるいは、左又は右部分生成順序木である。

命題 1. n 個の間隙をもつデーター D に対して、$_{2n}C_n/(n+1)$ 個の相異なる生成順序木がある。

証明 n 個の頂点をもつ二分木の個数に関する組合せ理論のよく知られた結果 (例えば [10] 389頁参照) より直接に求まる。

命題 2. 任意の生成順序木は、条件式 (2.1) を十分規定できる。

証明 D は、最外部のカッコ対を残して他のすべてのカッコ対を取除く事により生成順序木から再構築できることによる。

定義 2. 生成順序木のその順序付組のすべての閉じカッコが右端に置かれた場合に、その生成順序木を R-リストと呼ぶ。(図1参照) また、すべての開きカッコが左端に置かれた生成順序木を L-リストと呼ぶ。

命題 3. $2n+1$ 組のデーター D は、R-リストと L-リストを唯一つ有する。
また、$n > 1$ であれば R-リストと L-リストは互に異なる。

証明 生成順序木の頂点の数 n に関する帰納法による。

定義 3. $T_1 = ((T \; g_i \; T') \; g_j \; T'')$ と $T_2 = (T \; g_i \; (T' \; g_j \; T''))$ を生成順序木、又は部分生成順序木であるとする。生成順序木変換 $T_1 = L(T_2)$ を L-変換と呼ぶ。(図2参照) 逆変換 $T_2 = R(T_1)$ を R-変換と呼ぶ。

命題 4. $2n+1$ 組のデータ D の $2nC_n/(n+1)$ 個の相異なるすべての生成順序木は、その一生成順序木に R−, L−変換を連続して適用することにより得られる。

証明　生成順序木の頂点の数 n に関する帰納法による。

各間隙 G_i は、アドレス g_i で指定される少なくとも1個のセルをもつという空でない事を仮定したので、各リストセルが2個のタグビットと2個のデーターをもつとするならば、各間隙中の1個のセルを使用して生成順序木の各頂点を表わすことができる。　タグビットを立てた1個のデーターで補正値（木の葉）を表わし、タグビットを降した1個のデーターで他の部分生成順序木へのポインターを表わす。　補正の対象となるポインター p が与えられると、(2.1)に対応する補正値は、片言 ALGOL* [1] で記述した次の帰納的手続きにより求められる。

```
procedure GETOFFSET (p,r):
if r が葉である then return r
else if r >g p then return GETOFFSET (p, r の左部分木)
                else return GETOFFSET (p, r の右部分木)        (2.2)
```

ここに $>_g$ は生成順序を検査する述語である。

この手続きは、容易に次の反復形式の手続きに変換することができる。

```
procedure GETOFFSET (p,r):
begin
L: if r が葉である  then return r;
   if r >g p  then r←r の左部分木
              else r←r の右部分木;
   go to L
end                                                          (2.3)
```

この手続きが生成順序木に適用された場合の正当性は、生成順序木の頂点の数 n に関する帰納法により容易に証明できる。

生成順序木の各頂点は2個のデーターのみをもつ。　手続き(2.2)((2.3)も同様)は、各頂点のアドレス自身を三番目の（隠れた）データーとして利用することで、生成順序木上の二分木探索を行う。　通常の二分木探索法では、各頂点について3個のデーターが必要である事に注意すべきである。

Wegbreit [17] は、補正値を求めるために退役セルの領域（間隙）に構築した木の上の二分探索を使用する事を提案しているが、その木は2個のポインターと頂

*"call-by-value" によるパラメターの引渡しと、ポインターと葉（補正値）の単一データー型表現を仮定しておく。

点の情報（Wegbreitの用語で言う'covering address'）のための記憶領域を必要とする。1個のセルのみから成る間隙が存在する場合、3個のデーターを格納するというWegbreitの方式では領域が不足する。　従って、1個のセルの間隙のみから成る場合には、外部記憶を使用せずに二分探索木を構築することは不可能である。

二分木探索は、AVL-木[2]の例にある様に、木の高さを均衡させる事により、その処理速度を最大にすることができる。　高さnのR-リストは記憶領域を一掃する事により容易に構築できる。　木の高さを均衡させるよく知られた確立した技法（例えば[1, 11]参照、）を利用することで，R-リストを高さ$\lceil \log_2 n \rceil$の均衡木に変換する次の手続き（2.4）を容易に得る。

```
procedure  BALANCETREE(t):
begin
1.      p←e_n;
2.  A:  s←t; r←tの右部分木 ;
3.  B:  u←tの右部分木 ;
4.      if u=p then if t=s then return t
           else
5.            begin p←t; t←r; go to A
              end;
6.      v←uの右部分木 ;
7.      if v=p then if u=r then return t
           else
8.            begin p←u; t←r; go to A
              end;
9.      sの右部分木をuに取換える;
10.     tの右部分木をuの左部分木に取換える ;
11.     uの左部分木をtに取換える;
12.     t←v; s←u; go to B
end
```

(2.4)

ここにもはR-リストの根に初期設定されているものとする。この手続きは得られた均衡木の根を結果として返す。

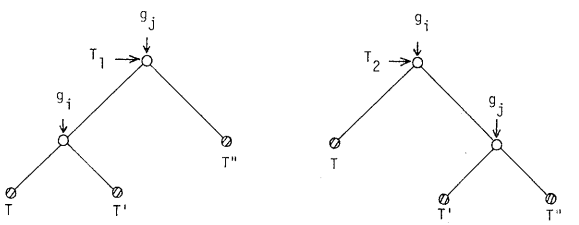


図2
生成順序木変換
（R-変換，L-変換）

R-transformation

$T_1 = ((T\ g_i\ T')\ g_j\ T'') \rightleftarrows T_2 = (T\ g_i\ (T'\ g_j\ T''))$

L-transformation

命題 5. R-リストの根rに対して、手続き(2.4)の結果は、(均衡)生成順序木である。

証明 手続き(2.4)中の木の変換は、9-11行目で行なわれる。この変換は、L-変換である。よって、得られた木は命題4により生成順序木である。

n個の頂点をもつ均衡生成順序木に手続き(2.2)を適用することにより、各ポインター補正は $O(\log n)$ 時間で行なえる。

2.2 高速ポインター補正技法

各ポインターpは、補正ビット累算器のための外部記憶とビット表(marking bit table)を利用する事により O(1) 時間で補正することができる。この技法は、補正値を効率良く算出するためにハードウェアインプリメンテーションに適していると考えられる。(図3参照)

ポインターpの補正値の算出は次の様にして行なわれる。

$\lfloor p/2^n \rfloor$ 番目のマーキングビット表の 2^n ビット列を $2^{\mathrm{mod}(p,2^n)}$ 値('1'ビットの $\mathrm{mod}(p,2^n)$ 長のビット列, $\mathrm{mod}(p,2^n) = \lfloor p/2^n \rfloor * 2^n$)でマスキングを行う。

マスキングを行なったビット列の退役状態ビット('1'で表わす)を累計する。この累計値は、$\lfloor p/2^n \rfloor * 2^n$ より pまでの退役セルの数である。

この累計値に $\lfloor p/2^n \rfloor$ 番目の補正ビット累算器の内容(この内容は、0から $\lfloor p/2^n \rfloor * 2^n - 1$ までの退役セルの総数である)を加える。各補正ビット累算器の内容は、ポインター補正段階の開始時にマーキングビット表の退役状態ビットを累計して設定する。

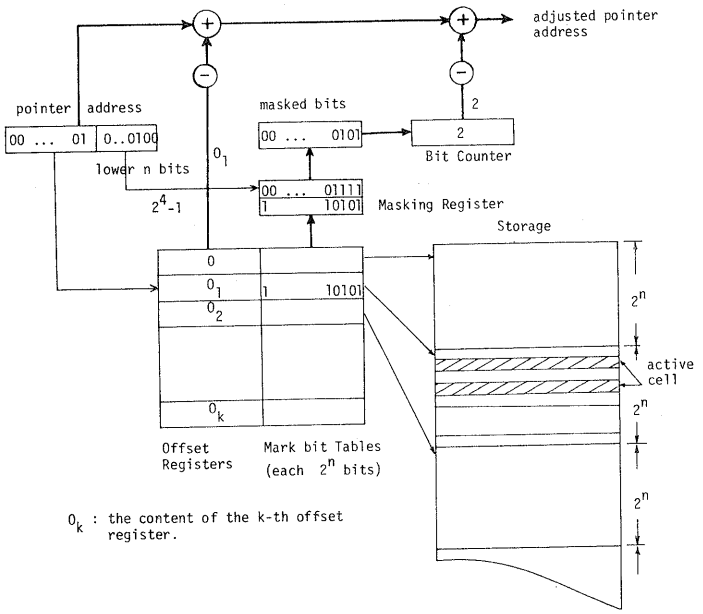


図3. ハードウェアポインター補正機構

補正ビット累算器とマーキングビット表にキャッシュ記憶が使用でき，i番目の補正ビット累算器とマーキングビット表を同時に読み取る事ができるならば，補正値の算出はキャッシュ記憶の1サイクルで実行できる。　さらに記憶の読み書きの命令のパイプライン制御を行なうことができるならば，ポインター補正に要する時間は，データーの再配置に要する時間に吸収されることになろう。

nを4にし，記憶領域が各64ビット長の65Kリストセルの場合には，各補正ビット累算器は16ビットで十分である。　よって，必要な外部記憶の量は，全記憶容量の3.125％である。

3. 遡行くず集め

システム作成記号（関数 gensym の実行により作られた記号）は，それがシステム演算器（system register(s)），スタック あるいはアトムの値部（value part）から参照されなければ退役の記号であり，すでに使用済のものである。　この退役のシステム作成記号を要素にもつ属性リスト**は，それを参照する事ができないために，使用済のものである。　この様な属性リストを回収する機能をもつものを遡行くず集めと呼ぶ。"遡行"と言うのは，退役のシステム作成記号を捜し出し，それが属性リストの要素であるか否かを検出する方式がリストを"遡行"する様な形態をとることによる。

4. 結論

生成順序の概念と生成順序保存処理系について述べ，生成順序の利用の有効性を生成順序木上の二分探索の例で示した。　生成順序の概念は，データーのある種の線型順序を利用する多くの検索，分類と併合の技法に適用できるものであると考える。　生成順序を保存する同時くず集め（concurrent garbage collection）は今後の興味ある研究課題であろう。

** システム作成記号を属性リストに使用する方法は，名前の混同を避ける一技法として用いられる。

References

1. Aho, A. V., Hopcroft, J. E. and Ullman, J. D. The Design and Analysis of Computer Algorithms (Addison-Wesley, Reading, MA, 1974) 113-169.

2. Adel'son-Vel'skii, G. M. and Landis, E. M. An Algorithm for the Organization of Information. Doklady Akademiia Nauk, SSSR 146 (1962) 263-226; English translation in Sov, Math. 3, 1259-1263.

3. Bobrow, D. G. ed. Symbol Manipulation Languages and Techniques (North-Holland, Amsterdam, 1971) 296.

4. Cheney, C. J. A Nonrecursive List Compacting Algorithm. Comm. ACM 13 (11) (Nov. 1970) 677-678.

5. Dijkstra, E. W., et al. On-the-fly Garbage Collection: an Exercise in Cooperation. Note for the 1975 NATO Summer School on Language Hierarchies and Interface (Lecture Notes in Computer Science 46, Springer-Verlag, N.Y., 1976)

6. Fitch, J. P. and Norman, A. C. A Note on Compacting Garbage Collection. University of Cambridge, Computer Laboratory, Cambridge (1976).

7. Goto, E. Monocopy and Associative Algorithms in an Extended LISP. Information Science Lab. Technical Report 74-03, University of Tokyo (Apr. 1974).

8. Gries, D. An Exercise in Proving Parallel Programs Correct. Comm. ACM 20 (12) (Dec. 1977) 921-930.

9. Hansen, W. J. Compact List Representation: Definition Garbage Collection and System Implementation. Comm. ACM 12 (9) (Sep. 1969) 499-506.

10. Knuth, D. E. The Art of Computer Programming, vol. 1, Fundamental Algorithms 2-nd ed. (Addison-Wesley, Reading, MA, 1973) 413-420.

11. Knuth, D. E. The Art of Computer Programming, vol. 3, Sorting and Searching (Addison-Wesley, Reading, MA, 1973) 422-471.

12. McCarthy, J. et al. LISP 1.5 Programmer's Manual. (MIT Press, Cambridge, MA 1965).

13. Müller, K. G. On the Feasibility of Concurrent Garbage Collection. Ph.D Thesis, Technical University, Delft, The Netherlands (1975).

14. Steel, G. L. Jr. Multiprocessing Compactifying Garbage Collection. Comm. ACM 18 (9) (Sept. 1975) 495-508.

15 Terashima, M. and Goto, E. Genetic Order and Compactifying Garbage Collectors Information Processing Letters 7 (1) (Jan. 1978) 27-32.

16. Terashima, M. Tabulative Computing, Principles and Portable Compilers. Ph.D Thesis, Faculty of Science, University of Tokyo (1978).

17. Wegbreit, B. A Generalized Compactifying Garbage Collector. The Computer J. 15 (3) (Aug. 1972) 204-208.

NEWS

1. LISPコンテスト速報

LISP性能コンテストには，外国から2件，国内で16件，計18件の応募がありました。幹事多忙のため，結果の整理はまだ終了しておりませんが，次回の研究会では，特別Sessionを開き，作成者に是非出席して頂いて結果の発表，反省，言い訳etc.を述べて頂くつもりです．

2. LIPQ 公開近し

武蔵野通研で開発した，PDP 11用のLISPであるLIPQが有償で公開される予定です。研究所内の手続にもたついているため，まだ正式の発表はできませんが，ソースリストまで含めて安価に提供できるよう、関係者は鋭意努力中です。LIPQは，PDP 11にメモリー28Kとディスクが一台あれば動きます（OSはDOS/BATCHです）。

3. TOSBAC 5600用LISP公開さる

TOSBAC 5600用LISPの1977度版が公開されました。特徴は，データとして，BIGNUM，TUPLE，SET，MULTISET，固定LISTが追加され，ユーザー定義のscannerと，くり返し関数としてのLOOPが導入されたことです。さらに，LISPで書かれたoptimizing compilerがついています。マニュアルetc.に関しては，電総研の推論機構研究室、または東芝総研の黒川までお問い合わせ下さい。

トピックス

1. Hearn教授来日決定

REDUCEでおなじみの，Utah大学計算機学科主任Hearn教授が，理研の招聘研究員として来日し，一ヶ月滞在される予定です。日程は未定ですが（11月頃ではないかと予測しています）決定しだいお知らせします。

2. 後藤教授 SIGSAM Bulletinのassociate editorに就任

後藤英一東大教授は，1978年4月より，SIGSAM Bulletinのassociate editorに就任されました。ちなみに，本年度のSIGSAM chairmanはP. S. Wangです。

春の人事移動

（かっこ内は移動前の所属）

1) 阿部芳彦氏・名大プラズマ研究所計算機センター（東北大工学部）

2) 金田泰正氏・名大プラズマ研究所計算機センター（東大後藤研）

3) 寺島元章氏・電通大計算機学科（東大後藤研）

4) 稲田信幸氏・理研情報科学研究室（小樽商大）