リスト処理系設計のための記憶管理に関する考察†

濱田眞美†† 野口正一†††

リスト処理系では動的に記憶割付を行うので，記憶管理の良否が処理速度に大きく関係する．本稿では，統計的モデルから導かれる結果と Lisp 1.5 インタプリタ系の測定値を比較しながら，固定長セル型リスト処理系の記憶管理効率について論じる．特に，処理速度に大きな影響を与える，ガーベージコレクションに費やす時間 T_G とポインタの局所性について考える．

リスト処理のオーバヘッドの多くを T_G が占める．単位時間当りの固定セル数が一定である処理に対して，始めに与えられるセル数の関数として T_G を計算する．この結果は実測値と一致した．ある許されるオーバヘッド時間比の下で処理を行う場合に必要なセル数を予測する目的に利用できる．

多数のセルを確保する手段としてページ方式によるセル空間の仮想化を考えた場合，ポインタに局所性があれば処理速度の低下は少ない．局所性の示標としてポインタのスパンを考え，セル使用形態モデルから静的スパン分布を与える．この結果は，仮想空間上で効率的にリスト処理系が動作する時のセル空間に対するページサイズを決定するのに役立つ．ガーベージコレクション時に実行されるセル空間圧縮がスパン分布に与える影響についても考察を加えている．

1. まえがき

リスト処理系が扱うデータはポインタを使った不定長データであって，記憶の動的割付け，不要領域の回収が要求される．処理系が使用できる記憶領域が大きければ，計算実行中に費される不要領域回収時間を縮小できることが示せる．仮想記憶化すれば使用可能な記憶空間が大幅に拡張される．しかし，リスト処理系の代表例である Lisp 処理系に対し，ポインタに局所性が有る結果が得らている[4]一方，ページング形の仮想記憶方式で動作させると実行時間の 90% 程度が補助記憶とのデータ転送に使われ実用的でないともいわれている[5),10)]．

Lisp 処理系を高速化するアルゴリズム的解析は行われている[6),8)]．しかし，仮想記憶化の問題を解決するには系の物理的動作に関する測定解析が必要となる．文献 4) ではポインタの局所性が実験的に考察された．本稿では，不要セル回収時間およびポインタのスパン分布に関し，経験的測定値と合致する統計的モデルを与える．処理系が持つべきセル数，仮想記憶化する時のページサイズなどの記憶管理動作に対して，一般のリスト処理系に適用できる結果を得た．

2. 測　定

動作モデルの基礎を成す測定データを下記の対象から得た．

2.1 m-Lisp 処理系

m-Lisp は筆者が作成した Lisp 1.5 インタプリタである．ミニコン FACOM・U-200 のオペレーティングシステム DIMOS 下で動く．インタプリット方式は文献 9) 付録Bの処理系 (MIT-Lisp) と全く同じであるが，空き領域の散在を無くす目的から，次の項目に示す実現上の小さな違いがある．測定値に関与する主な実現方式は，

- a-list を使った deep bind
- フルワード領域はリスト用セルで代用
- ガーベージコレクタ (GCR) はスタックを利用したマークビット方式
- セル消費抑制対策は施さない．

m-Lisp では，MIT-Lisp 同様，各再帰関数の引数がリストに調整される．このリストは短時間後に不要となる．インタプリタ内部で使う関数は，evlis, evcon, pair, sassoc 以外展開して書いてあるが，供給したセルの延べ数中 80% が引数受渡し用，10% が a-list 用としてインタプリタ自身の作業用に消費される．インタプリタの実現方法を工夫すると両者ともほぼ零にできるが[6)]，この抑制を行っていない．

セル領域内は解析の便宜上，システムアトム，プログラム，作業用セルの順に割付ける (図 1)．システムアトムには SUBR, FSUBR, APVAL 属性を持つ

† Considerations on the Storage Management of List Processing Systems by MASAYOSHI HAMADA (Nippon Electric Co., Ltd.) and SHOICHI NOGUCHI (Research Institute of Electrical Communication, Tohoku University).

†† 日本電気(株)
††† 東北大学電気通信研究所

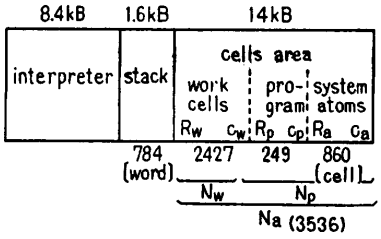

図 1 m-Lisp 処理系の記憶領域配置
Fig. 1 Storage assignment in m-Lisp.

アトムが含まれる.

2.2 測定時に使用したプログラム

一般に広く知られ, Lisp 処理系の速度比較によく利用される関数 BITA, BITB, n-Queens, SORT[8] を実行して諸データを測定した.

測定時に使うプログラムは可能な限り大きい方が, 実行される関数の頻度, セルの消費方法, 引数の利用方法等のプログラムに依存する効果が平均化されて, 処理系自体の評価に適する. にもかかわらず, 計算過程が既知, 測定時に使用した計算機からの制約, セル数が極度に少ない処理系の動作をも測定する理由から, 本稿では短いプログラムを採用した.

3. ガーベージコレクション時間

Lisp では処理系動作時間の大きな部分をガーベージコレクション (GC) に費やす. 手元のフリーセルを供給し尽した時使用済みの不要セルを回収再生する記憶管理時間であるから, 短いことが望ましい.

3.1 ガーベージ回収率

処理系測定結果の多くは使用したプログラムの影響を受ける. GC ごとの回収率 γ_i', すなわち, i 回目の GC で回収されたセル数 N_G と作業用セル数 N_w の比 N_G/N_w で表わされるセル消費過程もプログラムごとに異なる (図 2). 中間結果用にセルを固定する傾向の強いプログラムは γ_i' が小さく, 主に作業用に使うものでは γ_i' が大きい.

プログラムが扱うデータの大きさ k に対しては不変で, 必要数のセルを集めた時実行が終わる. また, 初期回収率 γ_0' で正規化した γ_i' の変化は使用可能セル個数 N_a にかかわらず同一曲線となる. 更に, 横軸として供給されたセルの延べ数をとると γ_i' はほぼ 1 次関数に従う. 以上の実験結果を総合して, 計算の進行ごとに一定の割合でセルが固定されるといえる.

供給された 1 セル当りの固定化される確率を定数 α とし, $\beta=-\log(1-\alpha)$ と置く. 初期固定セル数 $(1-\gamma_0')N_w$ をプログラムセル分に組入れて正規化すると, 第 i 回目の GC の回収率 γ_i' は

$$\gamma_i'=\exp(-\beta i) \qquad (1)$$

である. 明らかに γ_i' は N_a に依存しない. Lisp プログラム自身が計算の目的に使用するセル数に比べ, インタプリタが作業用に消費するセル数が多い場合には特にこの指数的変化に成り易い.

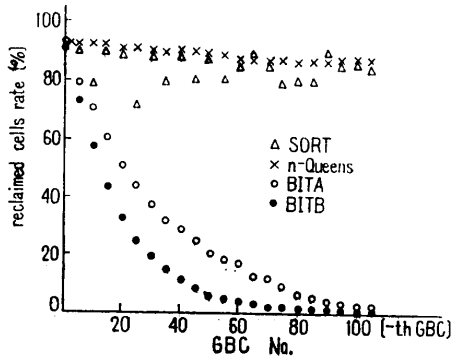

図 2 ガーベージコレクションごとのセル回収率
Fig. 2 Reclamation rate of each GBC.

n-Queens, BITB は α の大きさが両極端に近い単純な固定化経緯を持つプログラムである. 通常のプログラムでは GC 周期より大きな間隔で固定解除を繰返すと考えられる. この回収率経過を 1 次関数で近似する. この時, 固定確率が N_a に依存して異なった N_a に対しては 1 次関数に従わず, N_a を変化させた回収時間の議論はできない.

3.2 セル当りの回収時間

GCR で回収されたセル当りの回収時間 t_G/N_G を考察する. 回収率が小さいほど回収効率が悪くなることが示される.

GC 動作はマーク段階, 回収段階の順に進められる. マーク段階では, 棚上げされている中間結果リストを構成する各セルへマークを付ける. 回収段階で全セルに対しマークの有無を検査し, マークの無い不要セルをフリーセルリストへ接続する. 各操作で扱われるセル数を評価することにより, 1 回の GC に費やした時間 t_G が(2)式で, 回収されたセル当りの回収時間が(3)式で表わせる (**表 1**, **図 3**).

表 1 1 回のガーベージコレクションに費す時間
Table 1 The time consumed by a garbage collection.

$$t_G=m(1-\gamma)N_a+sN_a+l\gamma N_a+d \qquad (2)$$

$$t_G/N_G=(m+s+d/N_a)/\gamma-(m-l) \qquad (3)$$

ここで, γ: 回収比 $(=N_G/N_a=\gamma'(N_a-N_p)/N_a)$

GCR 定数 m: マーク時間/セル　s: セル走査時間/セル
l: セル接続時間/セル　d: GCR 起動時間

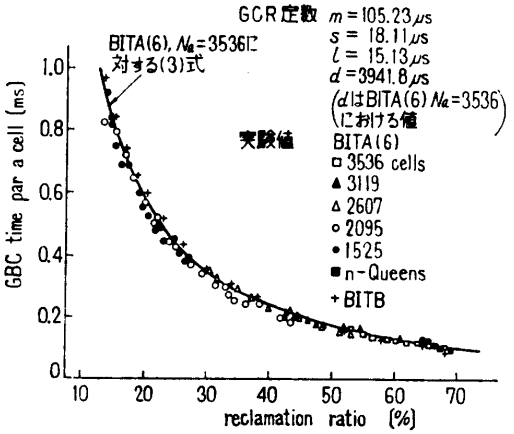

図 3 1つのセルを回収するに要す時間
Fig. 3 GBC time par a cell.

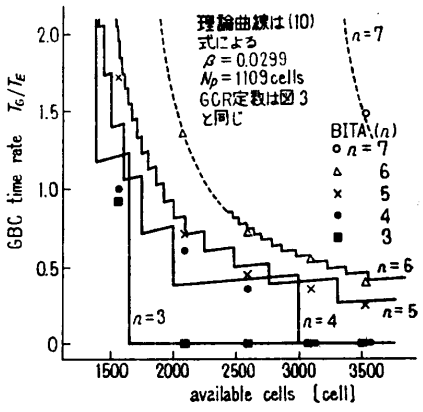

図 4 ガーベージコレクションに費やすオーバヘッドの割合と利用可能セル数との関係
Fig. 4 GBC time overhead ratio vs. available cells.

GCR 定数値は，GCR プログラムの実行命令を追跡して機械語命令の実行時間から計算する．走査および接続時間は GCR のみで決定される．マーク時間はリスト構造の影響を受ける．アトムヘッダ，文字コード，nil などポインタ以外の属性を持つデータがセル中に存在する，ポインタが同一セルを多重に指示することに因る．しかし，実際のデータ属性比は約 90% がポインタである．またセル当りの再マーク時間が初回マーク時間の 22.7%，多重度が 1.25～1.29 でありマーク時間を 5% 程増加させるに過ぎない．従って初回マーク時間を m とみなす．起動時間は GCR が起動される時に棚上げされているデータ数に依存して変わるが，t_G への寄与は少ない．

3.3 総ガーベージ回収時間

GC 回ごとの回収率が実験式として与えられるから，GC 回数だけ(2)式の和をとることにより，プログラム実行完了までに費やされる GC 時間 T_G が求まる．回収率の変化が指数的，1次関数的の2つの場合に対し計算する（**表 2**）．

指数的回収率変化の場合には，回収率が使用可能セル数に依存しないので，GCR 定数 m, s, l, d を求め，セル固定確率 α および固定的セルの数 N_p とプログラム終了までに供給すべきセルの延べ数 N_0 を測定すれば，任意の使用可能セル数 N_a に対する総 GC 時間 T_G が得られる．関数評価のためだけに使用された真の演算時間を T_E とした時，GC オーバーヘッド時間の割合を表わす T_G/T_E 比を測定値と比較する（**図 4**）．理論式による結果と m-Lisp からの測定値が良く合う．

GCR はセル空間内のリスト形式には依存しないパラメータで表わせたから，リスト処理系一般の不要セル回収動作に対し上の結果を適用できる．

表 2 実行終了までにガーベージコレクションに費やす時間
Table 2 The total garbage collection time.

回収率変化形	指数的		1次関数的	
GC_i の回収率 $\gamma i'$	$e^{-\beta i}$	(1)	$b-ai$	(4)
総供給セル数 N_a からの制約	$\sum_{i=1}^{i_0-1}(N_a-N_p)\gamma i'+N_a-N_p < N_0 \leq \sum_{i=1}^{i_0}(N_a-N_p)\gamma i'+N_a-N_p$	(5)		
GC 回数 i_0	$\left\lceil -1-\frac{1}{\beta}\log\left(1-\frac{N_0(1-e^{-\beta})}{N_a-N_p}\right)\right\rceil$	(6)	$\left\lceil \frac{b}{a}-\frac{1}{2}-\sqrt{\left(\frac{b}{a}-\frac{1}{2}\right)^2+\frac{2}{a}\left(1-\frac{N_0}{N_a-N_p}\right)}\right\rceil$	(7)
実行完了に必要な N_a の条件	$N_a>(1-e^{-\beta})N_0+N_p$	(8)	$N_a \geq \frac{N_0}{(2b-a)^2/8a+1}+N_p$	(9)
条件を与える理由	(6)式中の対数の定義域		(7)式で i_0 が実数となる範囲	
総 GC 時間 T_G	$\{(m+s)N_a+d\}i_0+\frac{(m-l)(N_a-N_p)(1-e^{-\beta i_0})}{1-e^{-\beta}}$	(10)	$(a/2)(m-l)(N_a-N_p)i_0^2+\{(m+s)N_a+d-(m-l)(N_a-N_p)(b-a/2)\}i_0$	(11)

4. ポインタスパン分布

処理系の使用可能なセル数が多い時に処理効率が高い事実が前章の議論から明らかとなった．主記憶量は限定されているので，補助記憶を活用した仮想記憶がセル数を大幅に増加させる方法として考えられる．データを論理的境界で分割するセグメンテーションと物理的境界で分割するページングの2方式がある．Lisp 処理系に前者を適用する場合 S-表現単位に区切ることになる．転送ごとにリストの転写または S-表現とリスト構造間の変換が必要である．また，セルが多重に指示される可能性があるから，転送後の非使用セルは GC で回収する必要がある．後者の場合，ページサイズを適切に選ばないと処理速度が大幅に低下する．ページサイズに直接関係するパラメータは動的参照系列に基づく参照番地間隔であるが，ここでは静的なポインタの性質から Lisp ポインタの局所性を検討しようと試みる．

番地 a に在るセルに含まれるポインタ P が番地 b のセルを指示する時，$(b-a)$ を P のスパンと呼ぶ．セル領域中の全ポインタに対してスパン値の分布を測定考察する．GC により分布形の変化が不連続となるため，その直前直後を測定時点に選ぶ．

4.1 ポインタスパン分布形

セル領域を低番地側からシステムアトム領域 R_a，プログラム領域 R_p，作業セル領域 R_w に分割する．また指示先データの種類により，アトムを指示するポインタ(AP)とリストを指示するポインタ(LP)に類別する．Lisp 処理系の構造上の性質として，ポインタの類別とそれを含む領域から指示先領域がほぼ定まる．

数値演算を行わず実行中にアトムが創出されなければ，AP は R_a, R_p を指示する．R_w 中に置かれる中間結果が含むアトムは主に R_p 中にある．R_p 中の AP は大部分 R_a を指す．R_a 中の AP は少数で無視できる．領域 R_w, R_p 中の AP は対象域を無作意に指すと考えられる．

領域 R_a 中の LP は，システムの作り方から ±10 セル以内のスパンを持つ．またアトムヘッダを改良した処理系[1]では本領域が存在しない．スパン分布の変化にも寄与しないから，R_a 中の LP に関するスパン分布は ±10 セル内の一様分布と仮定し $\bar{g}$ で表わす．

領域 R_p, R_w 中の LP は自領域内にあるセルを指示する．セルは番地順に供給され，時間的に近接して使われたデータが順次セルへ納められる．論理的には，定確率で占有されているセル領域に対しデータの位置の近くから空きセルを探して格納する幾何分布のモデルでポインタの指示密度を表現できる．

領域 R_s 中にあるポインタが領域 R_d を指示する時のスパン分布モデルを**表3**に示す．R_d と R_s 間に d の距離を持つ AP のスパン分布が $h_{q,l_s,l_d}(x+d)$ で表わせるので，フルワード領域等のセル領域外へアトムヘッダを置く処理系を含めて議論できる．

各領域 R_a, R_p, R_w のセル数を c_a, c_p, c_w とする．セルに含まれるデータ属性の割合を，領域 $R_x(x\in\{a,p,w\})$ に対してAPが a_x，LP が l_x，非ポインタが n_x とする．1セル中には2個のポインタが含まれるから，系全体のスパン分布が

$$h_{a_pc_p,c_p,c_a}+h_{a_wc_w,c_w,c_p}+\bar{g}+gl_{p\,c_p,c_p}+gl_{w\,c_w,c_w} \tag{15}$$

と表わされる（**図5**）．長スパン部分は AP のスパン

表3 スパン分布のモデル
Table 3 A model of the span distribution.

	アトムポインタ	リストポインタ
ポインタ指示密度	$1/l_d$ の一様密度	$e^{-\delta x}$ の指数減少的密度
R_s, R_d の位置関係	R_d が R_s の直後に隣接	R_d が R_s と下端を共有し，R_s を含む
スパン分布の一般形	$h_{q,l_s,l_d}(x)=\begin{cases}f_{q,l_s,l_d}(x) & l_s\geq l_d\\ f_{q,l_d,l_s}(x) & l_s<l_d\end{cases}$ (12) ここで $f_{q,u,v}(x)$ $=\begin{cases}0 & x<-(u+v)\\ (2q/uv)x+2q(u+v)/uv & -(u+v)\leq x<-u\\ 2q/u & -u\leq x<-v\\ -(2q/uv)x & -v\leq x<0\\ 0 & 0\leq x\end{cases}$ (13)	$\hat{g}_{q,\ l_s,l_d}(x)$ $=\begin{cases}0 & x<-l_d\\ 2q\nu(l_d+x)e^{\delta x} & -l_d\leq x<-l_d+l_s\\ 2q\nu l_s e^{\delta x} & -l_d+l_s\leq x<0\\ 2q\nu(l_s-x)e^{-\delta x} & 0\leq x<l_s\\ 0 & l_s\leq x\end{cases}$ (14)
スパン分布	$h_{q,l_s,l_d}(x)$	$g_{q,l}(x)=\hat{g}_{q,l,l}(x)$ ($R_s=R_d$ の時の $\hat{g}$)

l_s: 領域 R_s の幅　l_d: 領域 R_d の幅　R_s: ポインタが位置する領域　R_d: ポインタが指示する領域
l: $R_s=R_d$ の時の幅 l_s　q: R_s 中のポインタ数の 1/2　ν: 正規化定数 $(=\delta^2/\{2\delta l_s-1+2e^{-\delta l_d}-e^{-\delta(l_d-l_s)}\})$

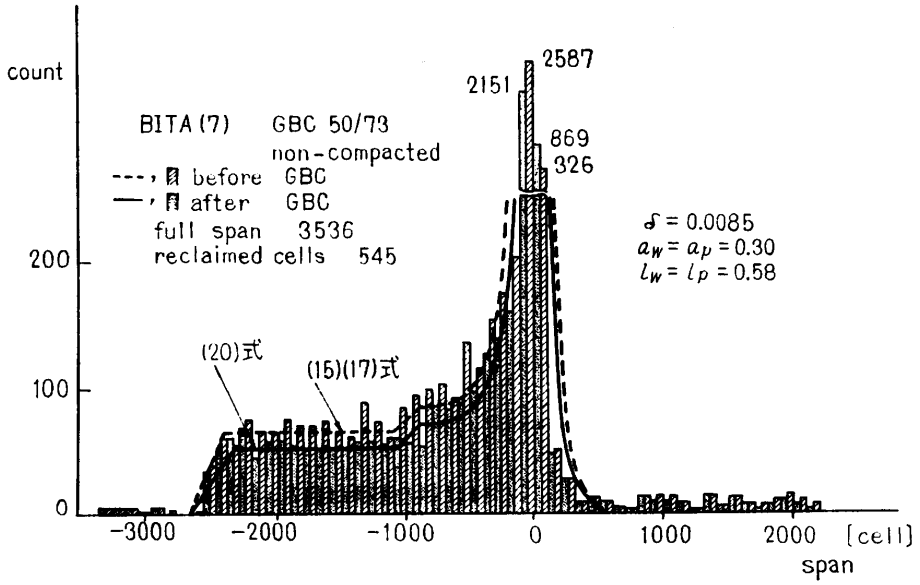

図 5 スパン分布（圧縮を行わない系）
Fig. 5 Span distribution of pointers.

分布で決定される．その密度は属性比 a_w と一致し，広がりは $-c_w$ から $-(c_w+c_p)$ にかけて減少する．

セル領域をページ分割した時，無作意に選んだセル c 中に含まれるポインタが c の属するページ内にある番地を指示する確率を同一ページ参照率と呼ぶ．スパン分布から同一ページ参照率が求まる．幅 l の領域 R を幅 p 単位のページに分割した時の LP だけに関する同一ページ参照率を π_R とする．R が仮想化されている時，δ に対応する幅，ページ幅よりも全セル領域幅が大きく，$\delta l \gg 1$，$\delta l \gg \delta p$ の関係が利用でき

$$\pi_R = \{-1+\delta p+\exp(-\delta p)\}/\delta p \qquad (16)$$

と書ける．AP はページ内を指さない．$c_a \ll c_w$ として良いから R_a 中の LP は無視できる．また $c_p \ll c_w$ であれば R_p が無視でき，c_p と c_w が同程度の大きさであれば $\pi_{Rp}=\pi_{Rw}=\pi_R$ である．a_p と a_w を同一値 a と考えて良く，処理系全体としての同一ページ参照率 π は $(1-a)\pi_R$ となる．π_R が 0.9, 0.95 のページ幅 δp は約 10, 20 である．$\delta p \to \infty$ の時 $\pi_R \to 1$ であってページ幅を大きくしても π は $(1-a)$ 未満である．すなわち AP がポインタの局所性を制限する．AP が実際に参照される比率を r と置く時，π は $(1-ar)\pi_R$ と書ける．m-Lisp ではポインタの指示先を見て初めてアトムか否かを知るが，領域設定の改良によってポインタ値からアトムの性質が得られる[2]．この時データアトムに対し $r \ll 1$ にできるのでかなり改良される．

4.2 スパン分布の変化

任意時点での参照可能セルに対するスパン分布は，GC 直後の分布とほぼ等しい．GC 直前のスパン分布には，2つの GCの間で作業用に追加された参照不可能なセルのポインタも数えられている．GC 位置を考慮せずに測定したスパン分布と，参照可能セルのみを対象とした分布の差異が，GC 前後の分布形変化機構から類推できる．

GCR は R_w 中の平均的構造を持つリストを短スパン後方指示のフリーセルリストに変える．回収率が γ' の時属性比が $a_w'=a_w(1-\gamma')$，$l_w'=l_w(1-\gamma')+\gamma'$，$n_w'=n_w(1-\gamma')+\gamma'$ に変わる．l_w'，n_w' の2項目がフリーセル分である．各領域の幅は変わらない．フリーセル分を除外した属性比を(15)式に適用して GC 直後のスパン分布を得る（表4）．

計算の進展に伴い γ' が減少した時分布形変化は次の2要因により生じる．1つは，作業用データと固定されたデータの性質が異なる場合ポインタの属性比が変わるためである．もう1つは，供給するセルの平均番地間隔 s の増加による．GC 前では処理系のセル使用法とプログラムで指示密度減少係数値 δ_0 が決まる．第 $i-1$ 回の GC(GC_{i-1}) と GC_i の間で作られるリストの減少係数を δ_i' とする．$s=1/\gamma_i'$ であるから $\delta_i'=\gamma_i'\delta_0$．$GC_n$ での分布を指数で近似した時の減少係数，すなわち GC ごとに蓄積されるポインタ数の対数を重みとして平均した δ_i' の値を δ_n とする．$\gamma_n' < \gamma_{n-1}'$ であれば $\delta_0 > \delta_{n-1} > \delta_n$ である．実際，計算の進行と共に γ_i' が減少し LP の分布広がりが増すが顕著ではない．

4.3 使用領域圧縮の効果

記憶領域の圧縮 (compaction) 方法は

(i) cdr 消去を行う compacting[3),7)]

(ii) cdr 消去を行わない compactifying

表 4 スパン分布に及ぼす圧縮の影響
Table 4 Effects of the compaction to the span distribution.

	圧縮を行わない系 S_{nc}	圧縮を行う系 S_c
GC_i 直前のスパン分布仮定式	$h_{a_w c_w, c_w, c_p+gl_w c_w, c_w}$ (17)	
(17)式を仮定した時の GC_{i+1} 直後のスパン分布	$h_{a_w(1-\gamma_{i+1}')c_w, c_w, c_p+gl_w(1-\gamma_{i+1}')c_w, c_w}$ (18)	$h_{a_w(1-\gamma_{i+1}')c_w,(1-\gamma_{i+1}')c_w, c_p}$ $+gl_w(1-\gamma_i')c_w,(1-\gamma_i')c_w$ $+\hat{g}l_w(\gamma_i'-\gamma_{i+1}')c_w,(\gamma_i'-\gamma_{i+1}')c_w,(1-\gamma_{i+1}')c_w$ (19)
GC_n 直後のスパン分布一般式	$h_{a_w(1-\gamma_n')c_w, c_w, c_p+gl_w(1-\gamma_n')c_w, c_w}$ (20)	$h_{a_w(1-\gamma_n')c_w,(1-\gamma_n')c_w, c_p}$ $+\sum_{j=1}^{n}\hat{g}l_w(\gamma_{j-1}'-\gamma_j')c_w,(\gamma_{j-1}'-\gamma_j')c_w,(1-\gamma_j')c_w$ (21)

GC_i: i 回目の GC

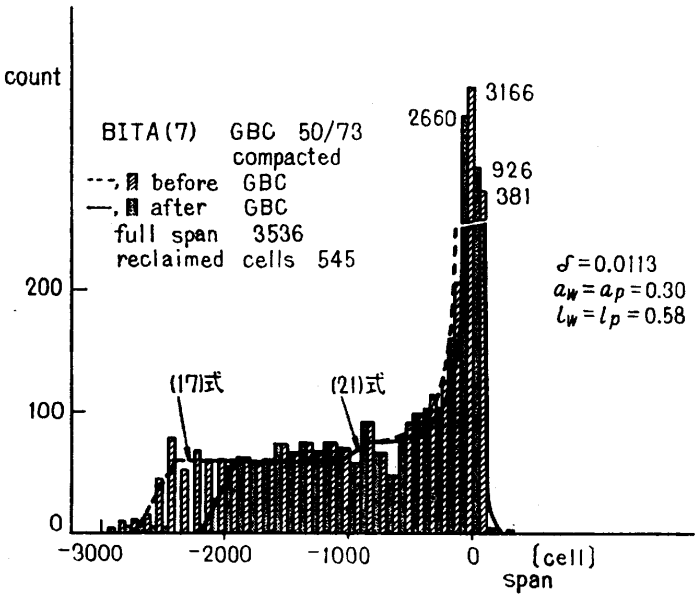

図 6 スパン分布（ガーベージコレクションごとに圧縮を行う系）
Fig. 6 Span distribution with compactifying GBC.

（a） 圧縮前後でセルの番地順序が保存される方法[11]

（b） 圧縮によりセルの番地順序が変わる方法[1]

に分類できる．ここでは，GC 時に（a）の方法でセル領域の低番地側へ使用中のセルを集める圧縮を考える．圧縮時のスパン分布集中化は，LP に対しては分布の細分化，減少係数 δ の増加によって，AP に対してはセル間隔の近接によって生じる．

GC による効果は R_w 中のポインタだけに働くから，R_w のスパン成分のみを扱う．圧縮を行う系 S_c と行わない系 S_{nc} のスパン分布形を漸化的に計算し，圧縮の効果を表式化できる（表 4）．S_{nc} では最後の回収率で分布が決定されるのに対し，S_c では以前の回収率の影響が残る．LP の分布幅は前方が $(\gamma_j'-\gamma_{j+1}')c_w$，後方が $(1-\gamma_{j+1}')c_w$ で決められ c_w より小さい．回収率変化分は小さいから，特に前方スパンは短いものだけになる．AP に対しては分布密度が変化せずに，$(1-\gamma_r')$ に比例してスパンが短くなる（図 6）．

減少係数 δ の値にも圧縮の影響が及ぶ．S_c では供給されるセルの平均番地間隔は 1 で，GC 後に圧縮を受けるから $\delta_i'>\delta_0$ である．前章の S_{nc} に対する議論と同様にし，δ_n の値は δ_0 より大きくなる．実際 GC_{50} 直後において BITA（7）の δ 値は S_{nc} で 0.0085，S_c で 0.0113 である．

5. むすび

リスト処理系設計のための基本的問題であるポインタ空間の動的記憶管理について考察した．使用可能セル数の増加に従いガーベージコレクション時間が減少する関数形を明らかにした．任意の仕事をある能率で処理するために要求されるセル数が決定できる．また，より正確に論じるにはポインタ参照列を対象に研究する必要があるが，静的スパン分布から仮想記憶化する場合のページサイズを論じた．Lisp 処理系においては，リストへのポインタは適当なページサイズを選べば効率良く処理できるが，アトムへの参照動作が全体の効率を制限する．不要リスト回収時の使用域圧縮については，実行すれば確かに局所性が増すが，それに費やされる時間との比較で実施の可否が判定されるべきである．

参考文献

1) Bobrow, D. G., ed.: Symbol Manipulation Languages and Techniques. North-Holland, Amsterdam, p. 296 (1971).
2) Bobrow, D. G. and Murphy, D. L.: Structure of a LISP System Using Two-Level Storage. Commun. ACM, Vol. 10, No. 3, pp. 155-159 (1967).
3) Cheney, C. J.: A Nonrecursive List Compacting Algorithm. Commun. ACM, Vol. 13, No. 11, pp. 677-678 (1970).
4) Clark, D. W. and Green, C. C.: An Empirical Study of List Structure in Lisp. Commun. ACM, Vol. 20, No. 2, pp. 78-87 (1977).
5) Cohen, J.: A Use of Fast and Slow Memories in List-Processing Languages. Commun. ACM, Vol. 10, No. 2, pp. 82-86 (1967).
6) 後藤英一: LISP 入門, bit, Vol. 6, No. 1-Vol. 7, No. 2 (1976).
7) Hansen, W. J.: Compact List Representation: Definition, Garbage Collection and System Implementation. Commun. ACM, Vol. 12, No. 9, pp. 499-506 (1969).
8) 情報処理学会プログラミングシンポジウム委員会編: 記号処理シンポジウム報告集 (1974).
9) McCarthy, J., Abrahams, P. W., Edwards, D. J., Hart, T. P., and Levin, M. I.: LISP 1.5 Programmer's Manual. MIT Press, p. 106 (1962).
10) 長尾 真, 中村和雄: 仮想記憶の概念を使用したミニコン用リスト処理言語 LISP, 電子通信学会技報, AL 73-70 (1973).
11) Wegbreit, B.: A Generalized Compactifying Garbage Collection. Comput. J., Vol. 15, No. 3, pp. 204-208 (1972).

(昭和52年11月7日受付)
(昭和53年8月31日採録)