記 号 処 理 45－4
（１９８８．３．２３）

印付けの対象領域を動的に管理するごみ集め

小川貴英
津田塾大学数学科

ごみ集めの対象となる領域を任意の大きさで管理し、1回のごみ集めの処理時間の上限を制御する一括型ごみ集めを提案し、その動的特性を解析する。プログラムの中でrplaca, rplacdを使ってポインターを操作しなければ、このアルゴリズムを採用したことによるオーバーヘッドはまったくない。さらに、ごみ集めによるオーバーヘッドは作業領域の大きさによってきまり、セルの占有率には依存しないので、残りの空きセルが少なくなったときにも有効である。

アルゴリズムは単純で容易に実現できる。本報告では、マイクロコンピュータのLispの上にアルゴリズムを実現し、その動的特性について解析する。

Pseudo real-time garbage collection

Takahide OGAWA
Department of Mathematics, Tsuda College.
2-1-1, Kodaira-shi, Tokyo, 187, Japan

Algorithm of pseudo real-time garbage collection, which limit the number of cells scanned in marking phase is presented. Using proposed algorithm we can shorten the suspension of a program execution by the garbage collector. Additional operations are only executed when rplaca or rplacd is evaluated. As the overhead of the garbage collector depends on the number of cells to be scanned, programs can be executed efficiently when only small number of cells are left.

This algorithm is implemented on small lisp system running on the micro computer and the dynamic behavior is analyzed.

１．はじめに

実時間処理を行なうにはごみ集めによる中断は許されない。これに対処するために実時間ごみ集めの手法がいくつか提案されているが、一般にそのアルゴリズムは複雑でオーバーヘッドも大きい。本論文では、ごみ集めの対象となるセルの個数を一定の数に制限することで、 1回あたりのごみ集めの処理時間を短縮し、疑似的に実時間ごみ集めに近い効果を上げられるアルゴリズムを提案する。アルゴリズムの特徴は次の通りである。

(1)計算に使う作業領域を制限し、ごみ集めの対象を作業領域に限定することで、1回のごみ集めに要する処理時間を短縮できる。これにより実時間で使った場合でもごみ集めによる中断が気にならない程度に抑えることができる。
(2)作業領域の大きさはプログラムの中から自由に変更でき、 中断時間の上限はプログラムごとに制御できる。さらに、作業領域を未使用のconsセル全体に一致させれば、このアルゴリズムを適用していることによるオーバーヘッドは無視できる。したがって、中断時間を短かくする、あるいはごみ集めをのオーバーヘッドを最小にするといった制御をプログラムごとに最適化できる。
(3)セルの使用率が高くなったとき、 すなわち空いているセルの数が残り少なくなったときでも、ごみ集めばかりをやっていて計算ができないという状態は発生しない。

このアルゴリズムは単純で、小さなシステムでも実現は容易である。本論文の後半では、マイクロコンピュータ上で動いているLispに実際にインストールして、その効果を調べ、ごみ集めの動的特性についても考察を行なう。

２．アルゴリズム

アルゴリズムの基本となる考え方は、ごみ集めの対象とする領域を小さくすることにある。そのような管理をしたとき、どのような影響があるのかを最初に考察し、次に具体的にどのようにして領域管理をすればよいのか、具体的な領域管理法を示す。

２．１ 作業領域

以下ではごみ集めの対象領域を制限したときの影響と、その対策について考察する。領域の制限は次の2つの考察が基本的になる。

(1)計算に使用する空きセルの個数を、 実際の空きセルの個数より小さくして、ごみ集めの間隔を短かくする。
(2)ごみ集めの対象を限定してごみ集め1回あたりの処理時間の短縮する。

(1)で計算に使用するセルの領域を作業領域と呼ぶ。(1)の結果、実際には空きセルがまだ存在する状態でごみ集めを開始するので、ごみ集めの回数は多くなる。作業領域の大きさをｎとすればごみ集めの回数はｎに反比例して増加する。 しかし、1回あたりのごみ集めの処理時間をこれに見合うだけ短縮できれば、ごみ集めの回数の増加によるオーバーヘッドを小さく抑えることができる。作業領域だけがごみ集めの対象となるように管理すれば、1回のごみ集めの時間は作業領域の大きさｎに比例し、 ごみ集めの回数の増加による損失は相殺できる。実際には、ごみ集めの初期設定の手間などは回数に比例するので、回数の増加による損失を完全に相殺することはできない。

プログラムｐを実行したとき、ごみ集めが起こらないときに問題を解くのに要する時間をＣ（ｐ）、ごみ集めの時間をＧ（ｐ）とすれば、問題を解くのに要した計算時間Ｔ（ｐ）は次のように表すことができる。

$$T(p) = C(p) + G(p) \qquad (1)$$

問題を解くのに使った延べのセル数をN（p）、作業領域の大きさをnとすれば、ごみ集めの回数gc（n）は次のように表される。

$$gc(n) = |N(p)/n| \qquad (2)$$

ただし、ここで｜x｜は切り上げ計算を表す。ごみ集めの初期設定などに要する時間をI、1セルあたりの処理時間をαとすると1回のごみ集めの処理時間g（n）、ごみ集めの時間G（p）は次のように表される。

$$g(n) = I + \alpha n \qquad (3)$$

$$G(p) = gc(n)(I + \alpha n) \qquad (4)$$

2．2 閉じた領域

次にごみ集めの対象を作業領域に限定するための領域管理法を示す。ごみ集めの対象を作業領域に限定することの利点は、印付けをするセルを作業領域に限定できることである。これにより印付けの手間は大幅に減少する。

ごみにならないことが分かっている部分を静的領域として、ごみ集めの対象から除外することは、ごみ集めの効率をよくするための1つの手法である。本論文のアルゴリズムは、この静的領域に相当する部分を動的に管理するものである。通常、静的領域は実行中その状態がまったく変更されることがない、ごみのない領域を指す。しかし、実際に計算するときには、領域内にごみが含まれていても、それらが再利用されないだけのことで、計算に支障はない。

ここでは、静的領域の条件を緩め次のような領域を考える。consセルの領域で、その中にある有効なポインターがすべてその領域内のセルを指しているとき、その領域を閉じた領域と呼ぶ。以下で考察の対象とするのはconsセルの領域だけである。閉じた領域の内部にはごみがあってもかまわない。ごみのセルの値は問題にしない。consセル全体は1つの閉じた領域とみなすことができる。

閉じた領域の定義から明らかなように、印付けの途中でポインターが閉じた領域のセルを指せば、その先をどれだけ手繰ってもポインターが領域外のセルを指すことはない。したがって、印付けの途中で閉じた領域のセルを指せば、その先を印付けする必要はない。すなわち、閉じた領域をごみ集めの対象からはずしても差し支えはない。作業領域以外にある有効なセルはすべて閉じた領域にあるようにconsセルの領域を管理すれば、ごみ集めは作業領域だけを対象にして行なえばよいことになる。

consセル全体は明らかに閉じた領域になる。ここでは特にconsセルの中で共通部分を持つ閉じた領域について考察する。共通領域を持つ閉じた領域は、その定義から一方が他方を完全に包含するものしか存在しない。閉じた領域の階層は何層になってもよいが、以下ではconsセル全体を4つの領域の階層としてとらえる。それらを小さい方からRi, i=1, 2, 3, 4と呼び、領域RiからRi-1の領域を除いた部分をRiの固有領域と呼ぶ。以下では、文脈から明らかなときはRiの固有領域にあるセルのことを単にRiのセルと呼ぶことにする。これに対し、固有領域に限定しないときはRiの中のセルという。

R1は主にプログラムを、R2はデータを、R3は作業領域を、R4は空きセルだけを固有

領域としてもつ。R4の固有領域は空きセルだけで、このセルを直接使用することはない。R1は静的領域に相当する。 R2は比較的ごみを多く含むが、R4に余裕がある間は静的領域と考えてよい。 各領域が常に閉じた領域の条件を満足するように管理すれば、R3だけをごみ集めの対象にしても不都合は生じない。

以下では、R3の固有領域だけをごみ集めの対象とするごみ集めを部分ごみ集め、R1からR3までを対象するごみ集めを全域ごみ集めと呼ぶ。見掛け上R3だけを対象にしている場合でも、R1とR2が空領域のときは全域ごみ集めと見なす。

原則として部分ごみ集めだけですまそうというのが、本アルゴリズムの目的である。R1とR2のセルの数は作業領域のセルの数より1桁程度多いと考えてよい。 したがって、部分ごみ集めだけですますことができれば、1回のごみ集めの処理時間を1桁程度短縮するのは容易である。

2．3 ごみ集めのアルゴリズム

本アルゴリズムで計算に使うセルはR3のセルだけである。R3に空きセルがなくなるとごみ集めが起動される。ごみ集めは次の手順で行なう。

1. 部分ごみ集めを行なう。
2. R1とR2がともに空のときは、次の処理をする。
   2-1 セルが回収できなかったときは、これ以上回収できるセルは存在しないので実行時エラーとする。
   2-2 使用中のセルをすべてR1に移し、回収した未使用のセルをR4に移し、R1＝R2＝R3とする。
3. R4にn個のセルがあるときは、R4からn個のセルをR3に移し、計算を再開する。
4. R2の固有領域を調べ次のいずれかの処理を行なう。
   4-1 R2≠R1のときはR2のセルをR3に移す。
   4-2 R2＝R1のときはR1のセルをR3に移す。
5. R4が空でなければ、R4のセルをR3に移し計算を再開する。R4が空のときは処理1からやりなおす。

上記のアルゴリズムが正しくいことを確認するには、R1、R2、R3、R4が最初に閉じた領域になっていれば、上の各操作で変更した後も各領域はやはり閉じていることを確かめる必要がある。R4が閉じていることは明らかなので以下ではR4については言及しない。

(1)処理1はR3だけを対象にしており、 ごみ集めの後もR1、R2、R3が閉じていることは明らかである。

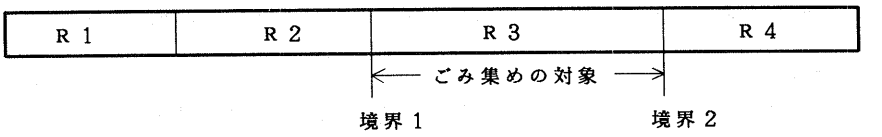
(2)(1)より処理2の前でR1からR4は閉じている。 処理2-2の結果すべての使用中のセルはは1つの領域にはいるので閉じているのは明らか。

(3)(1)(2)より処理3の前でR1からR4は閉じている。処理3は未使用のセルを移動するだけであり、R3は処理3後も閉じている。

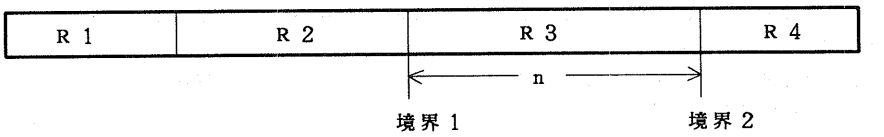
(4)(1)(2)(3)より処理4の前でR1からR4は閉じている。

処理4-1ではR2の固有領域をすべて移動するので、R3は閉じている。 R2=R1となるのでR2も閉じている。

処理4-2ではすべての使用中のセルはR3にはいるのでR3は閉じている。R1とR2は空になるのでこれも閉じている。(5)処理5は閉じている条件に影響を与えないので、R1からR4は閉じている。



（ａ）ごみ集め開始前



（ｂ）ごみ集め終了後

図１．ごみ集めによる境界の移動

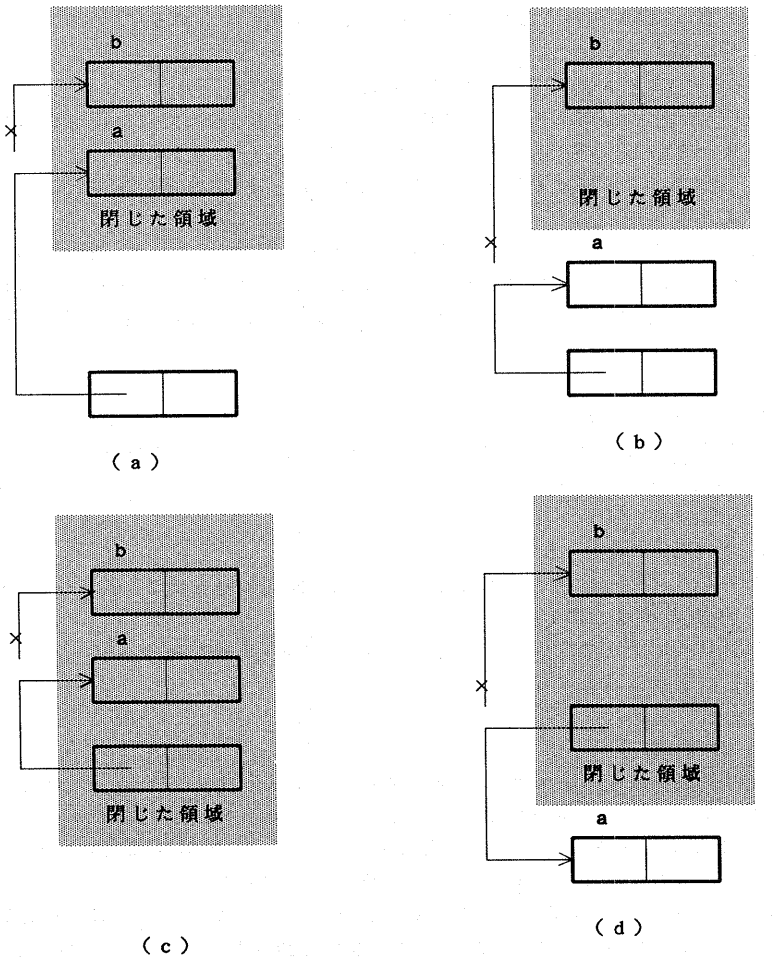


図２．ポインターの付け替え

以上の考察から各領域は常に閉じた領域の条件を満たしていることが分かる。

２．４ 領域の縮退

計算の途中でポインターの付け替えがなければ2.3までの処理で十分である。ポインターの付け替えを行なうと、閉じた領域の条件を満たさなくなる場合が起きる。本節では、どのような場合に領域が条件を満たさなくなるかを考察し、その処理方法を示す。閉じた領域に影響を与えるポインターの付け替えは次の4つの場合が考えられる(図2)。

(a)閉じた領域内のセルを指していた領域外のポインターを、領域内の別のセルを指すように変更
(b)閉じた領域内のセルを指していた領域外のポインターを、領域外のセルを指すように変更
(c)閉じた領域内のポインターを閉じた領域内の別のセルを指すように変更
(d)閉じた領域内のポインターを外部のセルを指すように変更

上記の各場合の内、(a)(b)(c)は閉じた領域はポインターの付け替え後も閉じた領域の条件を満たしているが、(d)の変更をした後ではその領域は閉じなくなる。このような場合は領域の再編成が必要になる。

仮に、ＲiからＲjへのポインターの付け替えが起きたとする。ただし、i<jとする。この場合にはＲiからＲj-1の固有領域をＲjに移し、Ｒiから Ｒj-1の領域の固有領域を空にする。すなわちＲi=Ｒi+1=...=Ｒj-1=Ｒi-1とする。ここでＲ0は空の領域とする。この操作を領域の縮退と呼ぶ。この領域の再編成によってＲiからＲjまでの各領域は閉じた領域の条件を満たすようになる。

Lispの関数でこのようなポインターの操作をするのは、rplaca、rplacdの2つの操作だけである。したがって、これらの操作をするときに新しいポインターの値が領域内にあることを確認する。もし、上の(d)に当てはまる場合は閉じた領域の縮退を行なう必要がある。具体的には次の2つの場合がある。

(1)Ｒ2のセルを付け替えた場合:
縮退が起きるのは付け替えたポインターがＲ3を指す場合である。このときはＲ2を縮退してＲ2の固有領域をＲ3に移す。
(2)Ｒ1のセルを付け替えた場合:
この場合には次の2つの場合がありうる。
(2-1)Ｒ2を指すポインターを代入したときは、Ｒ1の固有領域をＲ2に移し、Ｒ1を縮退させる。Ｒ1は空の領域となる。
(2-2)Ｒ3を指すポインターを代入したときは、Ｒ1、Ｒ2の領域を縮退させる。この結果Ｒ1、Ｒ2ともに空の領域となる。

３．アルゴリズムの評価

アルゴリズムの効果を確かめるために、マイクロコンピュータの上で動いているTSE Lispのごみ集めにこのアルゴリズムを採用して実験を行なった。TSE Lispのセルは12000で、これを図3に示すようにconsセル、スタックとして使っている。実験は8MHzの80286の上で、8-クイーンの解をすべて求めるプログラムの実行時間を計測した。

表1は単純な一括方式のごみ集めと、ここに示したアルゴリズムで作業領域を管理した場合の実行結果である。表1のｎ=0は作業領域の大きさを指定しなかったことを意味する。これは空きセルを全部作業領域に使った場合と考えてよい。TSE Lispはスタックとconsセ

ルの領域は1つの領域を使うので、スタックとセルの境界を何処に引くかは簡単ではない。ここでは、空き領域を2分する所に仮の境界をひき、スタックあるいはセルのどちらかの領域がなくなったときには、残りの領域の大きさを計算しなおし、ある値より大きければその領域を2分して計算を続ける方法を採用した。

表1を見るとセルが残り少なくなったときに、単純な一括型のごみ集めと比べて計算時間は1/6程度である。作業領域を管理した場合には、残りセルが300でも82330msで計算できた。残りセル200ではセル不足で計算できなくなるので、ぎりぎりまで十分計算に使うことができる。中断時間もn=1000で1/5から1/10程度になる。この管理アルゴリズムが有効であることが確認できる。

次に計測したデータから式(1)-(4)の各パラメータの値を求めて見ると、
N(p)=71700, C(p)=29500ms, I=51ms, α=0.0085msとなる。これを式(3)に代入すると、

$$g(n) = 0.0085n + 51$$

がえられる。(2)、(4)式から

$$0.601+51gc(n) \leqq G(p) < 0.601+51gc(n) + \alpha n$$

がえられる。αnを無視して、G(p)の値を(1)式に代入すれば、

$$T(p) = 30109+51gc(n)$$

となる。表2にこれらをもとにした計算結果を示す。

4. おわりに

本論文で示した領域管理のアルゴリズムは、ある程度中断時間を短かくできるが、Iの値以下にはできない。3章の場合であれば51ms以下にはできない。しかし、(1)アルゴリズムが単純で容易に実現できる、(2)nの値が極端に大きくなければIの値がごみ集めのオーバーヘッドを決定するので、高々10数%のオーバーヘッドを覚悟すればI時間の中断に抑えられる、(3)ごみ集めのオーバーヘッドはセルの占有率には影響されないので、Knuth[11]の指摘にあるように空き領域が少なくなっても著しい効率の低下はおきない、などの利点がある。

ここでいう作業領域として常に空きセル全体を使うようにすれば、ポインターの付け替えがないプログラムではオーバーヘッドがまったくない。したがって、中断時間が問題にならないのであれば、ごみ集めの対象領域の管理を行なうことで、ごみ集めのオーバーヘッドはかなり削減できる。

中断時間を一定の時間内に抑えるときは、Iの値を求め適当なnの値を計算し、領域を管理するだけでよい。3章の評価ではTSE Lispの特殊事情で複雑な処理をしたが、consセルの領域が固定されている場合は特別な処理は不用である。

ここでは、セル数が10K程度の小さなシステムで実験したが、セル数が多くなったとき、Iの値がどのような影響をうけるのかは今後さらに検討する必要がある。

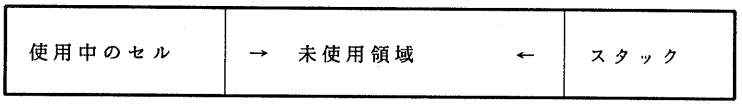


図3 TSE Lispの記憶管理

表1. ごみ集めの方式による処理時間の比較

空きセルの数	一括ごみ集め			作業領域 n=1000			作業領域 n=0		
	a	b	c	a	b	c	a	b	c
9699	32860	14	240	33800	71	61	30380	7	126
7699	36260	18	376	34680	71	73	30500	9	111
5699	41870	25	495	32680	71	45	30700	13	92
3699	54040	40	614	34680	71	73	31240	22	79
1703	96280	91	734	38710	149	62	34830	90	59
1203	129830	133	754	43270	277	50	37750	147	56
703	229810	253	792	55430	520	50	45210	294	53
503	343600	391	803	52310	453	50	54350	454	55

a:経過時間(ms), b:ごみ集めの回数, c:平均中断時間(ms)

表2. 作業領域の大きさと中断時間

空きセルの数	実測値		計算値		
	経過時間	中断時間	経過時間	中断時間	T／C
0	30380	125.7	30393	127.5	1.03
4000	30920	83.5	30945	85.0	1.05
2000	31880	68.0	31880	68.0	1.08
1000	33800	60.6	33725	59.5	1.14
500	37350	54.9	37401	55.3	1.27
300	42090	52.9	42245	53.6	1.43
200	48140	52.1	48367	52.7	1.64

時間の単位はms

参考文献

[1]Knuth, D.E., The art of computer programming, vol. 1: Fundamental algorithms, Addison-Wesley, Reading, Mass., 1973.

[2]Hoare, C.A.R., Optimization of store size for garbage collection, Inf. Process. Lett. Vol.2, No. 6, April, 1974, pp.165-166.

[3]Jacques Cohen, Garbage Collection of Linked Data Structures, Computing Surveys, Vol. 13, No. 3, September, 1981, pp.341-367.