GCを視覚化したLisp

4V－10

田中 良夫　鈴木 佐江子
慶応義塾大学理工学研究科　野村総合研究所

1. はじめに

Lispのようなリスト処理言語では，**ガーベッジコレクション**(GC)によって自動的にメモリ管理が行なわれるため，アプリケーションプログラマはメモリ管理に煩わされることなくプログラムを書くことができる．GCの効率は処理系自体の効率に大きく影響するため，様々なGCアルゴリズムが研究，開発されている．システムプログラマが処理系を製作する際にはアーキテクチャなどの様々な要素を考慮してGCアルゴリズムを選択する必要がある．しかし，処理系作成の初心者にとっては，各種のGCアルゴリズムの特徴を理解するのは決して易しいことではない．それは，GCをはじめとする計算機内部で行なわれる処理は，実際に目で見ることができないからである．「計算機内部の処理の様子を実際に目で認識できるような環境」があれば，そのような処理を容易に理解できるようになる．現在すでにApple LispでGCの様子を目で見ることができ，実際にLispやGCの教育のために役立っている[1]．本研究においては，汎用ワークステーション上でLispインタプリタにおけるセル空間の使われ方やスタックの使用状況を視覚化する[2]．GCという目には見えない処理を見せることにより，GCに対する理解を深め，スタックの使われ方を見せることにより，再帰に対する理解を深めることができる．システムプログラマを育てる際に有効な学習環境を提供することができる．また，アプリケーションプログラマにとっても，セル消費の様子やスタックの伸び縮みを実際に目で見ることによりそのアプリケーションの性質を知ることができる．

Visual GC Lisp

Yoshio TANAKA
Saeko SUZUKI

Graduate School of Science and Technology, Keio University
Nomura Research Institute, Ltd.

2. ガーベッジコレクション

GCのアルゴリズムは，大きく分けると以下の3種類に分けられる[3]．

- 参照カウンタを用いる方法
- マーク アンド スイープ法
- 複写法

以上の方法をベースとして，GCにオブジェクトの寿命の概念を取り入れた世代別GC，CONSによってセルが消費されるたびに少しずつGCの処理を行なう逐次型GC，リスト処理とGCの処理を並列に行なう並列GCなどの様々なGC手法が存在する．

3. 実装

本研究においては，参照カウンタを用いるGC，停止型マーク アンド スイープGC，停止型複写GC，停止型世代別GC，逐次型マーク アンド スイープGC，並列型マーク アンド スイープGCの6種類のGCを視覚化する．また，再帰の深さを示すスタックレベルメータを作成し，再帰の様子も視覚化する．本システムはLUNA-88Kワークステーション上でX-Windowシステムを用いて実装された．LUNA-88Kは4つのCPUを持ち，CMUで開発されたMACHオペレーティングシステムをOSとして用いている．既存のLisp処理系になるべく手を加えずに視覚化を実現するため，Lispインタプリタとは別にユーザインタフェース用のスレッド(**描画スレッド**)を生成して実際の描画処理を行なった．

3.1 視覚化の方法

今回視覚化するLispはセルの総数が250000個なので，500x500の描画領域を作成し，セルとピクセルを1対1対応させた．各セルの状態(フリーセル，生きている，死んでいるなど)に応じてセルの色を色分けし，ひと目でセル空間の様子が理解できるようにした．また，「インフォメーションボタン」をつけ，

GC が終了した時点での生きているセルの数と死んでいるセルの数を表示できるようにした．また，100個の目盛を持つスタックレベルメータを作成し，再帰の時にシステムが使用できるスタックと対応づける．1つの目盛がスタックの大きさの100分の1を表すことになり，スタックの使用状況に応じて目盛を色づけする．

3.2 インタプリタとの結合

セルの色を変更するのは，CONS によってフリーセルから生きているセルに変わった時，印が付けられた時，印がはずされた時，フリーセルとして回収された時，複写された時などである．基本的には，インタプリタがセルに対して処理を行なう(生成，印づけ，回収など)時に描画スレッドに対して描画要求を出し，その要求を受けた描画スレッドが実際の描画を行なうという方法を用いる．しかし，セル1つ1つに関してインタプリタが処理を行なうたびに描画要求を出すのでは，描画にかなり時間がかかってしまう．そこで，実際にはインタプリタがバッファリングを行ない，ある程度セルの数がたまった時点で描画スレッドに描画要求を出すようになっている．また，スタックレベルメータに関しては，eval が呼ばれた時点でその時に使っているスタックがスタックレベルメータのどの目盛に対応するのかを調べ，現在描画されている目盛から変更があった場合に描画スレッドに対して描画要求を出すようになっている．

4. 評価

実装されたシステムを用いていくつかのアプリケーションを動かした結果次にあげるようなことが容易に確認された．

- 各 GC 方法の特徴(GC にかかる時間，オブジェクトの局所性など)
  マーク アンド スイープ法に比べ，複写法は極めて短時間で GC が終了すること．ただし，複写法は GC に入る回数がかなり多いこと(あるアプリケーションでは，マーク アンド スイープ法では2回，複写法では13回 GC に入った)．
- オブジェクトの寿命(ある程度の GC を生きぬいたセルはずっと生き続ける)
  はじめの GC の時に生きていてコピーされたセルのほとんどが，その後も生き続けてコピーされること．
- アプリケーションのセル消費の様子生きているセルが徐々に増えてゆくものや，生成されたセルのほとんどがゴミになってしまうものなど，アプリケーションに応じたセル消費の様子．
- 再帰の様子アプリケーションごとの再帰の様子．

5. 結論

GC を視覚化したことにより，GC アルゴリズムを容易に理解できるようになった．また，スタックレベルメータにより，再帰の様子を視覚的に理解できるようになった．その結果，理解することが困難であった，計算機内部で行なわれる目には見えない処理を具体的なものとして捉えることができ，また，様々なアプリケーションの実行によって，それぞれのアプリケーションの性質を知ることができた．本システムはアプリケーションプログラマやシステムプログラマに対し，「目に見える」形で各種の情報を提供することができる．今まで計算機内部の処理を見ることができるシステムはあまりなく，このようなシステムは計算機教育の分野においても様々な形で利用，応用することができる．

参考文献

[1] 中西 正和.: “目で見る *GC*”，情報処理学会，ゴミ集め（ガーベジコレクション）の基礎と動向–チュートリアル資料, pp.23-35, 1992

[2] 鈴木 佐江子.: “*Visual Garbage Collection*”，慶應義塾大学学士論文，1993

[3] Wilson, Paul R.: “Uniprocessor Garbage Collection Techniques”, Lecture Notes in Computer Science, 637, Springer-Verlag (1992).