3E−10 オブジェクト指向言語ＣＯＢにおける自動メモリ管理

久世和資　上村務
日本アイ・ビー・エム(株)　東京基礎研究所

1．はじめに

ＣＯＢは，Ｃベースの安全性を重視したオブジェクト指向言語で，プログラムの記述性や再利用性が高い．ＣＯＢに自動メモリ管理機構を導入するにあたって，メモリ管理システムの高い移植性と，多くのアプリケーションプログラムにおいて10%以内のオーバーヘッドとなることを目標にした．遅延参照カウント方式，コピー方式，および，マークスイープ方式の三種類の自動メモリ管理機構を，効率向上のための改良を施してＣＯＢに実現した．本論文では，Ｃベースの言語における各メモリ管理手法の問題点，評価および効率に影響する要因を述べる．さらに，各メモリ管理手法の組合せや選択による使用の可能性についても考察する．

2．ＣＯＢの言語仕様と処理系の性質

まず，ＣＯＢの言語仕様および処理系からは，以下の考慮点が挙げられる．

1）スタック上のオブジェクトポインタの識別
2）グローバル領域上のオブジェクトポインタの識別
3）型変換されたオブジェクトの先頭アドレスの計算
4）型変換によるオブジェクトポインタの位置
5）オブジェクトの消去時ルーチンｆｉｎａｌの実行

ＣＯＢは，Ｃの上位互換言語であり，クラス以外に，Ｃで許される型を持つデータが使用できる．したがって，ＣＯＢのオブジェクトを管理するためには，スッタクとグロバール領域上のオブジェクトポインタを識別する必要がある．ＣＯＢのコモン変数は，グローバル領域にとられるので，グローバルなオブジェクトポインタも多いことが予想される．

ＣＯＢは，オブジェクトの上位クラスおよび下位クラスへの型変換を明示的に行なう機能を持っている．このため，オブジェクトへのポインタが，その先頭を指しているとは限らないし，あるオブジェクトへの複数のポインタが異なる位置を指す可能性もある．オブジェクトに付加されたメモリ管理用の情報を参照したり，オブジェクトを消去するには，オブジェクトの先頭を計算しなければならない．

ＣＯＢでは，オブジェクトの生成時と消去時に自動的に実行されるｉｎｉｔとｆｉｎａｌ関数が用意され，その内容はプログラマが定義することができる．オブジェクトの生成時と消去時には，これらの関数を実行しなければならない．

3．ＣＯＢプログラムの特性

次に，ＣＯＢで記述したアプリケーションプログラムの特性を以下に述べる．

1）オブジェクトが，環状に結合されやすい．
2）オブジェクトへのポインタ数が２以上のことが多い．

遅延参照カウンタ方式が有効に動作するには，「ほとんどの場合，参照数が１である」という仮定がある．しかし，オブジェクト指向プログラムでは，オブジェクト間でメッセージをやりとりするために，各オブジェクトは，単純な木構造ではなく，ネットワーク状に結合されている可能性が高い．テストプログラムで調べた結果，参照数２が多いことがわかった．

また，同じような理由で，オブジェクトの結合が環状になっていることも多い．テストに用いた５つのプログラムのうち，２つに環状の構造があった．

4．自動メモリ管理の効率

メモリ管理の効率を比較するには，以下の要因が考えられる．

1）メモリ管理の時間
2）ごみ回収回数
3）ごみ回収１回あたりの中断時間
4）使用メモリ総量
5）ページフォルトの影響
6）生きているオブジェクトの割合による影響

実現した三方式の比較を，同じメモリ容量を与えて行なった．遅延参照カウント方式は，ゼロ参照オブジェクトの数が一定数を越えたときにごみ回収を開始する機構になっている．そこで，その数を変化させて実行した時の使用メモリ容量を，それぞれコピー方式とマークスイープ方式に与えて性能を測定した．コピー方式では，２分割でメモリを使用するので，オブジェクトが最大使用できるのは参照カウント方式の半分のメモリである．評価は，lispインタプリタとCOBパーサーを用いて行なった．

この評価結果からは，マークスイープが，もっともオーバーヘッドが少ないことがわかる．しかし，ごみ回収のオーバーヘッドに着目すれば，使用できるメモリ量が半分にもかかわらず，コピー方式のオーバーヘッドが小さいことがわかる．ごみ回収１回あたりの中断時間は，遅延参照カウント方式が有利である．なお，遅延参照カウント方式によるparserの実行では，環状のポインタも回収できるようにプログラムの一部を変更した．

表１と表２では，遅延参照カウント方式のオーバーヘッドが大きいが，これは，すべてのアプリケーションプログラムに単純にあてはまらない．グラフは，簡単なプログラムで，各ごみ回収時点での生きているオブジェクトの割合を変化させた時のオーバーヘッドを示す．ごみ回収時で，生きているオブジェクトの割合が大きい時は，遅延参照カウンタ方式のオーバーヘッドは小さくなる．遅延参照カウンタ方式は，生きているオブジェクトの割合に対する影響が少ないこともわかる．また，ごみ回収による中断時間も，カウント方式が小さく，応答性が要求されるアプリケーションには適している．

Automatic Memory Management for an Object-Oriented Programming Language COB
Kazushi KUSE and Tsutomu KAMIMURA
IBM Research, Tokyo Research Laboratory

使用ﾋｰﾌﾟ(Kbyte)		214	300	505	880
参照ｶｳﾝﾄ方式	合計	17	15	14	13
	ｶｳﾝﾄ	9	9	9	9
	回収	8	6	5	4
コピー方式	合計	15	14	14	14
	間接	12	13	13	13
	回収	3	1	1	1
ﾏｰｸｽｲｰﾌﾟ	合計	4	3	1	1

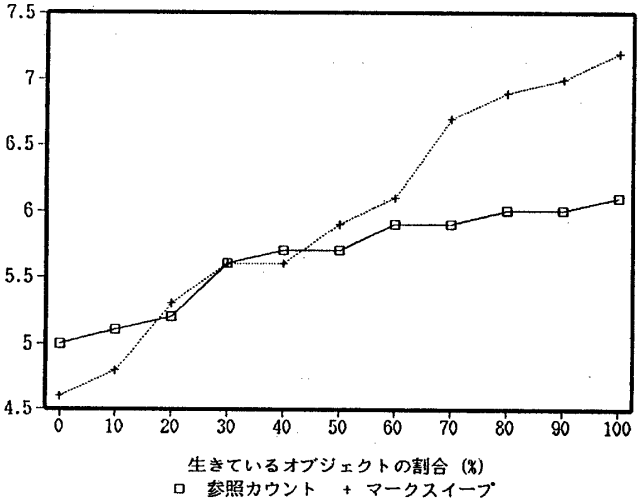
(単位：%)

表1　lispインタプリタにおける自動メモリ管理のオーバーヘッド

使用ﾋｰﾌﾟ(Kbyte)		711	1300	2292	3351
参照ｶｳﾝﾄ方式	合計	33	28	25	24
	ｶｳﾝﾄ	13	13	13	16
	回収	20	15	12	8
コピー方式	合計	10	8	4	7
	間接	3	3	3	3
	回収	7	5	1	4
ﾏｰｸｽｲｰﾌﾟ	合計	5	1	2	5

(単位：%)

表2　COBパーサにおける自動メモリ管理のオーバーヘッド



グラフ　生きているオブジェクトの割合とメモリ管理のオーバーヘッド

5．自動メモリ管理機構の移植性

遅延参照カウント方式とマークスイープ方式の移植性は高い．コピー方式は，ＯＳのメモリ割付けの方法に依存するため移植性が低い．

遅延参照カウンタ方式とマークスイープ方式は，PS/2とRT/PCのAIX上とPS/2のOS/2上に移植されている．２つのAIXでは，ヒープ領域の上限値と下限値の定数を変更するだけで対応できる．OS/2とAIXでは，ヒープとグローバルの領域のチェックが変わる．OS/2のメモリは，６４Ｋバイトずつのセグメント単位でとられるため，チェック用のコードが数行，余分に必要になる．

コピー方式をOS/2に移植するには，この６４Ｋバイトに適した調整をする必要がある．

6．議論

表3は，2．から5．までに述べた要因に対して各方式の適応度を示したものである．たとえば，効率2では，ごみ回収回数に関するもので，マークスイープ，コピー，カウントの順に多くなる．移植性は，5．でも述べたが，AIX用のコピー方式のメモリ管理システムをOS/2に移植するのは難しい．AIXの場合，連続するヒープ領域を単純に２分割して使用できるが，OS/2の場合，６４Ｋバイトのセグメントを連結して使用する必要がある．

項目		参照ｶｳﾝﾄ	ｺﾋﾟｰ	ﾏｰｸｽｲｰﾌﾟ
言語処理系	1	△	△	△
	2	○	×	×
	3	△	△	△
	4	△	×	△
	5	○	×	○
COBﾌﾟﾛｸﾞﾗﾑ	1	×	○	○
	2	△	○	○
効率	1	×	△	○
	2	×	△	○
	3	○	×	×
	4	△	△	○
	5	△	△	×
	6	○	×	×
移植性		○	×	△

表3　各メモリ管理方式の比較

7．おわりに

オブジェクト指向言語ＣＯＢの自動メモリ管理についてその性能評価を述べた．今回のテストプログラムでは，どの方式が最良であるか判断が難しい．しかしながら，各方式でどの部分にオーバーヘッドがかかっているのかを明らかにすることができた．特にオブジェクトの間接参照については，メソッド表へのポインタを間接参照領域に設けたり，ごみ回収が起こらない部分でのメンバの参照は直接行うなどの改良の余地が考えられる．

さらに，エディタなどの会話型アプリケーションを含む多くのテストプログラムで性能評価を行うとともに，自動メモリ管理システムのオーバーヘッドをさらに削減するのが，今後の課題である．

参考文献

[1] Onodera,T., Kamimura,T.: "Increasing Safety and Modularity in C-Based Objects", ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語研究会資料, No.25, (1990).

[2] 久世，上村:"ｵﾌﾞｼﾞｪｸﾄ指向言語COBにおける自動メモリ管理", ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語研究会資料, No.25,(1990).