KBMS．Cランナにおけるメモリ管理方式

5G-7

串間和彦　村山隆彦　山下竹見
NTT　情報通信処理研究所

1．はじめに

ｴｷｽﾊﾟｰﾄｼｽﾃﾑ(ES)の言語環境としてLISPとCの2つが多く用いられている。この内LISPは高い記述能力と優れたﾃﾞﾊﾞｯｸﾞ環境によりESの早期開発に適している反面、完成したESのﾃﾞﾘﾊﾞﾘ*を考えると①Cで記述した場合に比べて処理性能が悪い。特にｶﾞｰﾍﾞｰｼﾞｺﾚｸｼｮﾝ(GC)が頻繁に発生し、かつ1回毎に時間がかかる。②作成したESと既存の情報処理ｼｽﾃﾑとの結合が困難といった欠点がある。一方Cはその逆の性質を有しており、一長一短の関係にある。我々はLISPの高い生産性とCの高い処理性能を有効利用するために、LISP環境で作成したESをC環境で処理可能な形式に変換するCﾗﾝﾅを実現した。

ESをCで直接記述する場合に比べて、LISP上のESをC変換する場合はGCの回避が主要な問題である。LISP上は利用者によるﾒﾓﾘ管理が行われないため、ﾒﾓﾘは処理系が汎用的に管理せざるを得ない。このため、変換後も不要域の回収のためにGCを必要とする場合が多い。本稿ではESが必要とするﾒﾓﾘを特性毎に分割管理し、領域単位に有効ﾃﾞｰﾀを決定することでGCの頻度、所用時間を実用上問題ない範囲まで削減する方式を提案する。

*)ESを配布し、ｻｰﾋﾞｽとして運用する環境。

2．GCに関するLISP処理系の問題点

LISP上でのESの実行ではﾒﾓﾘ使用量が一定に達するとLISP処理系が自動的にGCを行う。GCはまずﾃﾞｰﾀの参照関係をもとに有効ﾃﾞｰﾀを決定(「印付け」)し、次に有効ﾃﾞｰﾀ以外の領域を「回収」することで行われる。

ESの実行時には、ﾙｰﾙ、WM等の知識の記述、LISPによる手続きの記述、推論ｴﾝｼﾞﾝ等がﾒﾓﾘ上に展開される。各々の性質に着目すると領域単位に有効ﾃﾞｰﾀを決定できる場合があるが、LISP上では用途別の領域管理は困難であり、全有効ﾃﾞｰﾀについて印付けが行われる。このため、不要ﾒﾓﾘの回収に時間がかかる。

3．Cﾗﾝﾅの概要

3.1　機能概要

LISP上のESでは知識の記述以外にLISPによる手続きの記述が多く含まれている。Cﾗﾝﾅでは変換時のESのﾘﾗｲﾄを最小化する目的で①IF-THEN形式のﾙｰﾙ記述、②WMによる事実の記述、③LISPによる手続きの記述を変換の対象としている。特に③については実用ESに必須な約120種のCommonLISP関数とMMI作成用の40種の関数について変換を保証している。

Cﾗﾝﾅを用いた変換は図1の手順で行う。すなわち、

a)ESのｿｰｽﾌﾟﾛｸﾞﾗﾑをCのﾌﾟﾛｸﾞﾗﾑに変換する。変換の単位は①から③の記述を含む1つのﾙｰﾙｾｯﾄ(後述)とする。

b)変換後のCﾌﾟﾛｸﾞﾗﾑを、推論ｴﾝｼﾞﾝ、LISPの組み込み関数等とﾘﾝｸして実行形式ﾌﾟﾛｸﾞﾗﾑを生成する。Cで記述したｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑもこの段階でﾘﾝｸする。

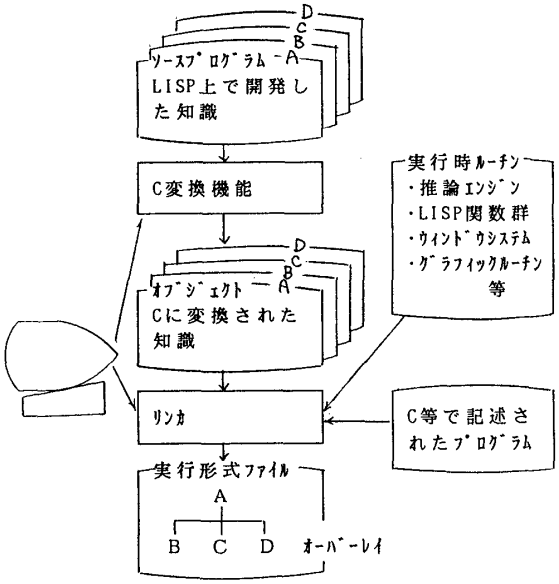

図1　Cﾗﾝﾅを用いた変換の流れ

3.2　実現方式の概要

使用目的毎に最適なﾒﾓﾘ管理を行うため、①から③の記述について以下に示す変換、実行方式を採用している(図2)。

①ﾙｰﾙ記述:条件部はreteﾂﾘｰを表現する構造体ﾃﾞｰﾀに変換し、共通の推論ｴﾝｼﾞﾝで解釈実行する。結論部はC関数に変換し、関数呼び出しで起動する。

②WM記述:構造体ﾃﾞｰﾀに変換し、共通のWM管理機能で参照、更新を行う。

③LISP記述:関数定義はCの関数定義に、関数呼び出しはC上での関数呼び出しに変換する。LISP上で使用するﾃﾞｰﾀ(文字列、ﾘｽﾄ、数字)は型情報を付加したC上のﾃﾞｰﾀに変換する。

4．ﾒﾓﾘ管理方式

実行時には使用ﾒﾓﾘを特性毎の領域に分割して管理

Memory management of KBMS C-runner
Kazuhiko Kushima,　Takahiko Murayama　and Takemi Yamashita
NTT Communications and Information Processing Laboratries

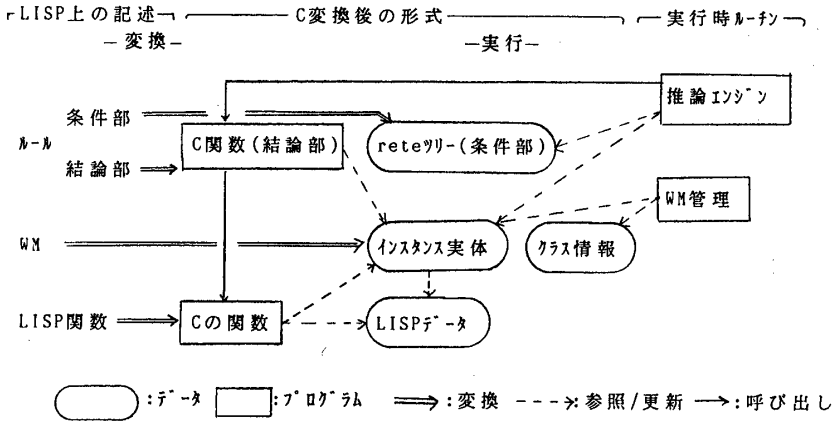

図2 ESのCへの変換と実行

し、領域単位に確保／解放を行うことでGCによらずに不要域を回収する。具体的には、以下のとおり。

4.1 ルールセット単位のメモリ管理

KBMSでは知識のモジュール化機能としてルールの集合をルールセットとして定義し、ルールセットを単位とした起動、終了機能を提供している[1]。Cランナではモジュール化のメリットを活かし、ルールセット終了後に使用メモリを一括回収することでGCによらずに不要メモリを回収する。

(1)プログラム部

ルールセット毎のLISP記述は1つのオブジェクトファイルに変換する。ESを構成する複数のルールセットをリンクする段階で、同時に実行されないルールセット間にオーバーレイを指定する機構を提供する。オーバーレイ指定されたルールセットは起動時に共有エリアにロードされる。

(2)データ部

各ルールセットを変換して得られるreteツリーはルールセット起動時にメモリ上にロードし、ルールセット終了時に解放する。reteツリー内の作業用データ(メモリノードに蓄えられるトークン)についてもルールセット毎に専用のエリアを順次切り出して使用し、ルールセット終了時に一括解放する。

4.2 WM

インスタンスはクラス単位に大きさが決まっているため、フリーリストで管理する。すなわち削除されたインスタンスのエリアはクラス毎の大きさに応じたフリーリストにチェーンし、再利用する。

4.3 LISPデータ

LISP関数が生成する文字列、リスト等に関してGCを完全に抑止するためにはLISP上で個々のデータについて解放の契機を記述する必要があり、利用者の負担が増大する。このため、不要域はGCにより回収する。GCの処理では大域変数とWMスロットの内容のみを保証し他のLISPデータ域はすべて回収する。

処理の特性に応じた上記メモリ管理により、実行時に必要なGCの回数を極小化するとともに、全データを対象とするLISP上のGCに比べて処理時間を大幅に短縮できる。なお、長時間の連続運転を目的とするシステム等、使用メモリの整理にGCを必要とするシステムのためには、システム側からGCが制御可能なよう、使用メモリ量の過去のピーク値を取得する機能とGCの起動機能をCのインタフェースで提供している。

図3にCランナのメモリマップを示す。

5. 評価

LISP上のESとC変換後のESのメモリ諸元を表1に示す。対象ESはC変換を意識せずに作成された診断型のESであり、ルールにより故障診断を行い、診断結果をマルチウインドウ、グラフィック等により表示する

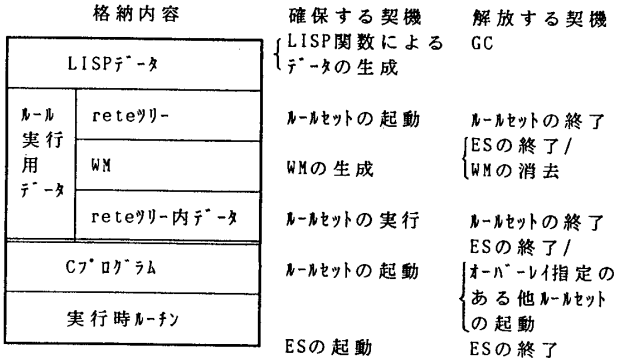

図3 Cランナのメモリマップ

表1 LISP環境と比較した使用メモリ

	LISP環境	C環境
システム全体の使用メモリ量*1	1 (LISP処理系を含む)	0．1
GCの対象となるメモリの割合*2	1 (ほぼシステム全体)	0．01 (LISPデータのみ)

*1)LISP環境でのシステム全体の使用メモリを1として場合の値。
*2)GCの対象となるメモリ量 ／ LISP環境でのシステム全体の使用メモリ量。

ものの規模は約2KSである。測定はデリバリ環境であるパーソナルコンピュータ上で行った。表1からES全体の使用メモリを1/10に削減でき同一のメモリ環境ではGCがほぼ不要になること、GCを行う場合でも回収のためにサーチするメモリはLISPに比べて1/100程度であり、GCの所用時間を大幅に削減できることが分かる。なおGCの実際の発生回数は使用可能メモリ量、LISP処理系の実現方式に強く依存するため単純比較は困難だが、同一の処理においてLISP環境で1~2回程度発生していたGCがC環境では完全に不要になることを確認している。

[参考文献]1)森原他:"推論制御機能を強化したES構築支援ツール:KBMS"、人工知能学会研究会、SIG-KBS-8801-7(1988).