構造をもったデータの高速マッチング方式

7D-2

和田良一・青木　豊・本間真人・江村里志

(松下電器産業株式会社・無線研究所)

1．はじめに

LISP で扱うデータはリストであらわされた構造を持つデータであり、計算機の内部ではその構造部分をポインタで表現されたリストセルで、要素をアトムとして別々に管理している。このうちリストセルは基本的には1セル毎に操作され、またセルは複数のデータにおいて共用されるため以下の問題がある。

(1) RPLACA,RPLACD 等、直接リスト操作を行うと他のデータも変更してしまうという思いがけない副作用が生じる。
(2) 並列処理時、変数のロックが困難である。
(3) パタンマッチング処理がリストたぐりとなり非効率となる。
(4) ガーベージコレクションが困難である。
(5) メモリ参照の局所性が悪く、キャシュのヒット率が下がる。

我々はこれらの問題点を解決するためリストセルを共用せず、主にパタンマッチングを高速に処理するような構成を持つ LISP 処理システムの開発を行っているのでその内容につき報告する。

2．処理系の構成

本システムにおいては構造データに関しても通常の文字や数値データと同じようにメモリからレジスタへ転送して処理が行われる。図1において Structure register がこれに対応する。実際には Structure Register にはＶＲ (Virtual Register) へのポインタが入り、実際の構造データはＶＲへたくわへられる。
メモリとＶＲの間の転送はＴＣ (Transfer Circuit) によってバーストモードで行われる。ＴＣは転送時にデータ表現の変換を同時に行う。ＶＲはＣＡＭ (Content Addressable Memory) により構成されオーバフロー時メインメモリとの間で仮想化処理がなされる。構造体データの切り出し、比較、結合等の各種演算はＳＰＵによって行われる。

3．リストの内部表現

本処理系においてはリストセルは図2における List Area へ一つの構造データにつき連続したエリヤに格納されている。また、他の構造データと共用部分を持たない構成になっている。リストエリヤは Tree Table によって管理され、Tree No. によってアクセスされる。
リストセルはメモリ内においては cdr coding で表現されていてメモリ領域の節約を行っている。

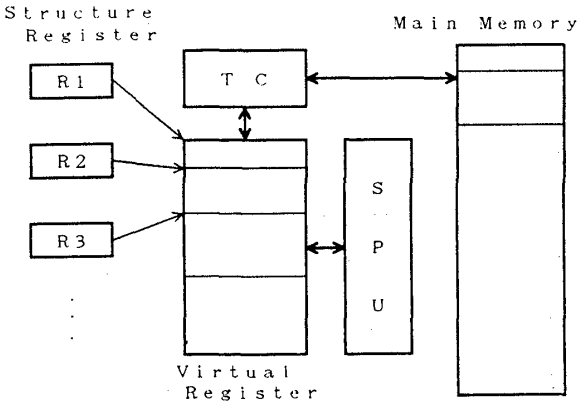

図１．処理系の構成

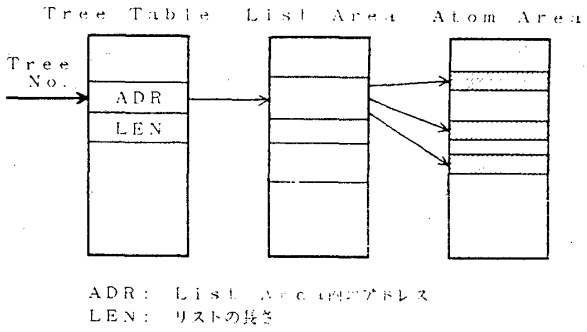

図２．ツリー表現

A Fast Pattern Matching Algorism for List Structure

Ryoichi WADA,Yutaka AOKI,Masato HOMMA,Satosi EMURA

Matsushita Electric Industrial Co.,Ltd. Wireless Research Lab.

また、そのポインタ部分の指す領域は同一構造内の他のセルかアトム領域である。

ＶＲ内においては構造データは図３に示すように要素のテーブルで表現される。これはリストの長さ方向に順次番号を付け、深さ方向に順次項目を割り当てて、すなわち、ノードの位置を一次元ベクトルで表現し、その中で要素（ＮＩＬかアトム）を指している部分を抜き出してテーブルとしたものである。

このテーブルと実際の構造とはユニークに対応する。この様なリスト表現方式をとる事により前述の様な欠点を無くすことが出来る。すなわち、リストの共用が無いので直接リスト操作に対して他のデータは保護されるし、したがってツリーのロックも可能である。また、リストを順次たぐらなくても所望の要素にアクセスが可能である。例えば、図３において要素Ｂにアクセスするためには従来はトップから２回リストをたぐるのに対し、この方式においてはノードベクトルが２－１－１の要素をそのままアクセスすれば良い。

ガーベージコレクションについては即時型であり前記の性質によりきわめて容易に行う事ができる。

４．リストのマッチング処理

リストの比較はテーブルの各要素を順次比較していく事により行われる。この時従来の方法のようにリストを分解する必要は無い。この動作はＶＲとそれに接続されたＳＰＵによって行われる。ＶＲはＣＡＭにより構成されているので実際には比較する一方の要素を順次ＶＲに問い合わせる事により比較処理が進行する。そしてすべての要素についてヒットすれば２つのリストは同値であることがわかる。比較する一方が部分木の場合でもノードベクトルを修飾する事により同一の方法で比較処理を行うことが出来る。例えば、一方のリストに関数ＣＡＤＲが作用している場合、他方のノードベクトルの先頭に２を付加するだけでそのまま比較処理が出来る。また，ＡＳＳＯＣ関数の実行は第一引数のノードベクトルの先頭にすべての数とマッチするワイルドナンバーを付加して問い合わせを行うことにより、一度で実行出来る。

５．おわりに

リストのテーブル表現とその処理系の概要、および、比較操作の内容につき述べた。本方式によれば、前述の比較処理以外でも種々のリスト処理を高速に行う事が出来るが、メモリーＶＲ間のデータトラフィックが大きくなる欠点がある。

特に並列処理を行う場合、ブロック転送を使用してもメモリバンド幅がボトルネックとなる可能性がある。現在この部分につきシミュレーションによる検証を行っている。

最後にこの研究に対して御指導いただいた京大工学部情報工学科の大野豊教授、阿草清滋助教授、および、大野研のメンバーの方々に感謝します。

参考文献

[１] Gurindar S. Sohi et al : An Efficient LISP-execution Architecture with a New Representation for List Structure , Conf. Proc. Annu. Int. Symp. Computer Arch. 12th 1985 .

[２] John Allen : Anatomy of LISP , McGrow-Hill, 1978

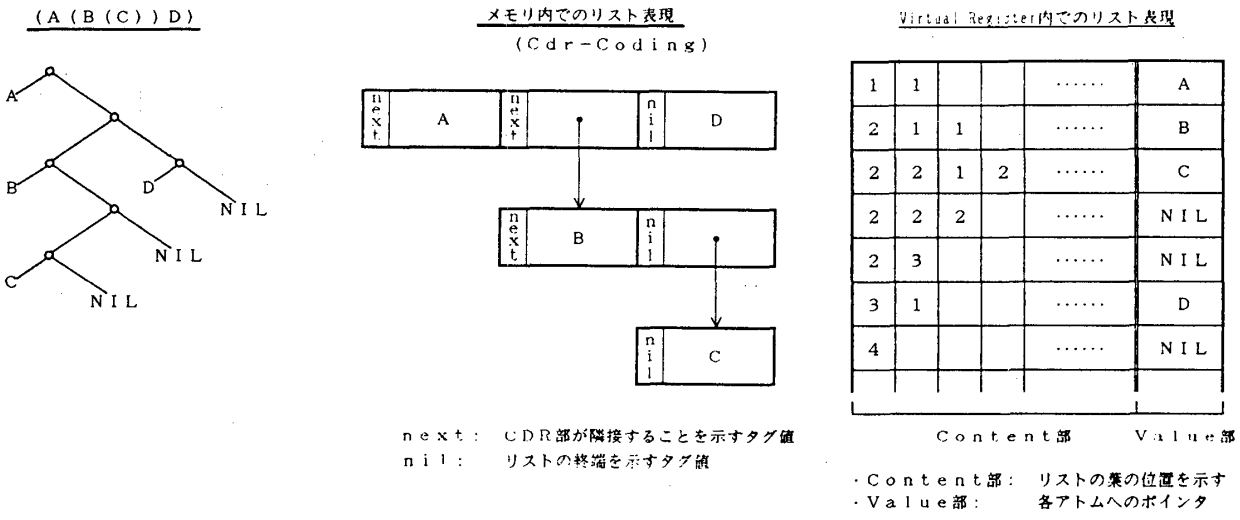

図３．リスト表現