5B-1

データフローマシンTOPSTAR-IIにおける並列LISPシステム

鈴木達郎　田中英彦　元岡達
(東京大学　工学部)

§1. はじめに

データフロー処理の1つの制御方式である,(注1)要求駆動制御を用いて,TOPSTARにLISPシステム(TOPLISP)を実装した。[1] TOPLISPはpure LISPに関数定義機能を追加したものである。本論文では,これまでの実装,使用経験をもとに,LISPシステムを要求駆動で実現する時の問題点(環境問題,cond式の扱いなど)を明確にし,構成法を整理する。またこれまでに得られた実測値にもとづいて,実用性についての検討を行なう。

§2. LISPの環境問題

通常LISPは,変数とその値,入変数の結合関係などを環境として持っている。prog, setqなどに代表される副作用は,この環境(即ち共有変数)に基づいている。この様な副作用を使いたくなる時はよくある。例えば,

(cond ((F X) (F X)) ---)　(1)

という式で,条件をみたすものの存在を確かめる動作と,その値を求める動作とが,ほとんど同じ内容(例えばトリーサーチ)ならば,関数Fの中で,その値を変数にsetしておき,二度手間を省きたくなる。

(cond ((F x y) y) ---)　(2)

(注1) 要求駆動方式は,「要求」によりデータフローグラフを動的に展開し,それを(値を返すことで)逆にたたんでルートノードに戻る。このたたむ過程は,データ駆動と共通である。違いは,パイプライン処理の表現などに見られる様に,動的なノードの生成という点に集約できる。

これまで,pure LISPの様な理論的言語に,各種の(副作用ともいえる)機能を追加してきた。その理由は,主として(ノイマン型計算機における)効率を意識したからだと言える。

しかしこの様な環境は,関数相互の独立性をさまたげ,見通しを悪くすると同時に,並列性を取り出す上での障害となる。もし理想的な並列処理(プロセッサの価格が0,数が無限大,オーバヘッドがない)を仮定すると,(1)式における二度手間は効率上問題でなくなる。現実の並列処理は理想からまだ遠いが,プログラム(関数)の見通しを良くすることに伴なう効率の悪化を,償える可能性を持っている。

TOPLISPは極端な例として,環境のまったくないLISPである。環境のないLISPは置き換え規則によるリダクションシステムとして扱える。(注2)また関数定義も関数名と関数本体との置き換え規則と見ることができる。

§3. CONDの並列処理

リダクションシステムでは原理的にデータ依存性以外のすべての並列性を取り出せる。しかし実装においては,いくつかの問題点がある。ここでは,cond式について述べる。COND式は,

(cond (P1 e1) (P2 e2) ---(Pn en))　(3)

と書け,n組の引数から成る関数と,

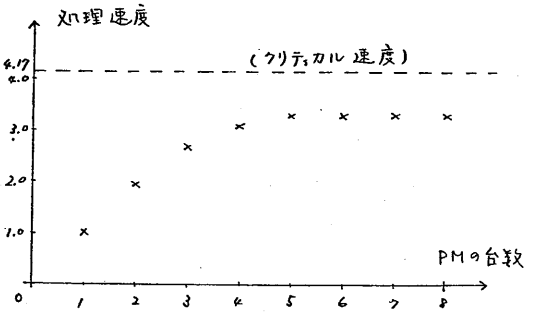
(注2) LABEL,関数引数などがなければ,最も内側から置き換えを実行できる。ここでは外側から置き換えを行なうため,名前の衝突などの問題を含む。名前の衝突は,内側に含まれる置き換え規則そのものを,外側から変えてしまったことに相当する。

見ることができる。ただしPiを左から順に調べて最初に真になった組のeiを値とするという逐次性が含まれている。従って各Pi及びeiを一度に(並列に)要求するといくつかの問題を生じる。第一にエラーになる場合がある。例えば (cond ((atom X) (…)) (… (car X) …)) において (car X) はXがアトムでないことを仮定している。この場合は値をとりあえずエラーとしておく。プログラムが正しければ、このエラーは最終的な値に影響しない。第二に、並列に要求した部分式の最も遅いもの(場合によっては無限)を待つことである。これは lazy eval で解決できる。lazy eval はすべてのデータ(要求に対する返答)が揃わなくても、実行を進める方法である。[2]

§4 TOPLISPの実測値

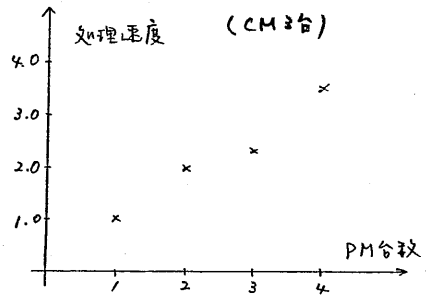
TOPLISPは現在実装上いくつかの欠点を持っているが、(注3)とりあえず得られた実測値について述べる。

図1は、フィボナッチ関数の値を求める計算、図2は、Jacobian を求める数式微分及び簡単化についてである。(ただし処理時間にトレースも含める)



[図1. CM1台での(FIB 6)の計算]

(注3) ①部分結合を扱う relay 機能が未実装 ②従来のLISPシステムを改造して作製したため、同じ親ノードにつながる範囲しか一度に要求を出さない。③システムプログラムのオーバヘッドが大きい。④ cond 式は逐次的に処理する。



[図2. (JACOB '(PLUS X Y) '(PLUS Y X) 'X 'Y) の計算]

§5 実用性についての考察

TOPLISPの問題点としては、データ依存性による逐次性以外の並列性を取りきれていないこと及びシステム構成が洗練されていず、オーバヘッドがかなり大きいことである。

一般にデータフロー処理の高速化を限定する要因は、次の二つである。

(1) 応用プログラムの持つクリティカルパス。

(2) システムのオーバヘッド(S式の転送、要求と返答の結合、実行可能ノードのサーチなど)

クリティカルパスとは、データ依存性によって決まる逐次的パスの最も長いものである。プロセッサの数を増やしても、これ以上は時間短縮できない。またシステムオーバヘッドが大きいと、ノードの数がある値以上又はノードの処理時間がある値以上でないと、並列効果が出なくなる。

実用的な応用は、これらの制約にかかってはいけない。また現在までのソフトウェアやノウハウの蓄積も大きな問題だ。

(1) 従来実用にされていない(できない)分野で並列性の大きい応用を扱う。

(2) データフローの考え方(独立性)を従来のシステムに徐々に取り入れる。

の二つの方向が考えられる。

<参考文献>

[1] 駒田他,「データフローマシン TOPSTARによる並列LISPシステム」情処大会 55年度

[2] 横井,「記号処理とデータフローマシン基本計算機構の紹介」情処大会 55年度