5P-8

データ依存解析による関数型プログラムのコンパイル法
-- Datarolプログラムの抽出 --

立花 徹　　谷口 倫一郎　　雨宮 真人
(九州大学 総合理工学研究科)

1. はじめに

動的なデータフロー・アーキテクチャではプロセス・スイッチがハードウェアで直接行われるので多重処理機構を効率的に実現できる．しかし，データフロー・アーキテクチャをそのままハードウェアで実現しようとする場合，通信のオーバヘッド，フロー制御のオーバヘッド，メモリレス概念の弱点，といった問題が生じる．データフロー・アーキテクチャのそのような問題点を解決するために，関数型言語からデータ依存関係を基に抽出されたマルチスレッド・コントロール・フロー(Datarol)に沿って並列演算実行を制御するプロセッサの機構が提案されている[1][2]．Datarolプログラムは，データフロー・プログラムからペア・オペランドの存在チェックやゲート演算などの余分なデータフロー制御を取り除いて最適化したマルチスレッド・コントロールフロー・プログラムである．本稿ではDatarolプログラムの実行メカニズムと関数型言語から最適なDatarolプログラムを抽出するアルゴリズムについて述べる．

2. Datarolプログラムの実行メカニズム

2.1 コントロールフローの表現

Datarolの一般的な形式を示すと次のようになる．

lx : [*](x op y z) --> E

lx はこの命令のラベルであり，op はオペレーション名，x は op の結果を書き込むレジスタ名，y,zは op のオペランドが格納されているレジスタ名，E はこの命令の次に発火される命令の集合である．* が付されている命令はオペランドのマッチングが必要な命令で，両方のオペランドの到着が確認されないと発火されない．また x のかわりに # の記号が書かれている命令は op の結果をレジスタに書かなくてよい．また，y または z の代わりに # が書かれている命令はデータをレジスタから引き出さなくてよく命令トークンに含まれるデータを第1または第2オペランドとする．

2.2 関数リンケージ

関数適用によって新たな関数活性体が生成されるとその関数活性体は他の活性体と並列に実行される．呼び側の活性体の call 命令が発火されると新たな活性体にオペランドデータを保持するための作業領域（レジスタファイル）が割り当てられる．関数本体に関しては，同一定義の関数本体を持つ関数活性体と関数本体を共有する．引数の受渡しは，link 命令により呼び出し側のレジスタファイルから新しく生成された関数活性体のレジスタファイルへデータを転送することでなされる．link 命令で受け側へ引き数が渡されると，その引き数に対応する receive 命令が発火し，receive 命令の次命令へ実行の制御が移り，関数本体の計算が行なわれて行く．結果が生成されると受け側の関数活性体は return命令で呼び側に結果を返す．return 命令の実行が終わると eins 命令によって受け側の関数活性体のレジスタファイルは解放される．呼び側は rlink 命令で結果を受け取り rlink 命令の次命令へと実行の制御を移す．

2.3 条件実行

条件命令は (sw b) --> (Et,Ef) という形をしている．b は真偽を表わす bool 値，Et は b が真の場合の次命令の集合．Ef は b が偽の場合の次命令の集合である．

例を用いてその実行をデータフローと比較してみると，余分なゲートを通らない分だけパスが少なくてすむことが解る（図1）．

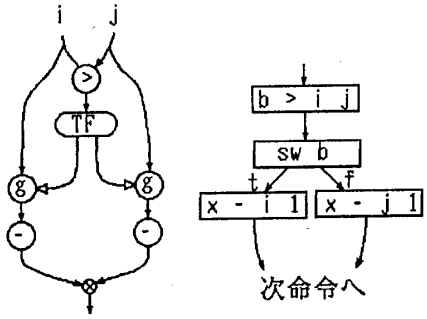


図1．条件文実行の例

2.4 ループ実行

ループ構造のプログラムは図2のように実行される．

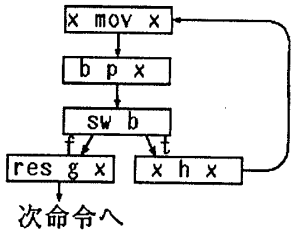


図2．ループ実行の例

3. Datarolプログラム抽出アルゴリズム

3.1 Datarolプログラムの抽出

関数型言語からDatarolプログラムを抽出するコンパイラの性能は，Datarolプロセッサの効率に強く影響する．コンパイラは関数型言語に内在する並列性を最大限にひきだし，Datarolプロセッサの資源を有効に利用するDatarolプログラムを抽出することが望ましい．

関数型プログラムは，次の3つのステップを経てDatarolプログラムに変換される．

Compiling method of functional program by dataflow analysis
-- Extraction of Datarol program --
T.Tachibana R.Taniguchi M.Amamiya
(Kyushu university)

(A)関数型プログラムを中間言語表現に変換する．中間言語表現は，(w op u v) といった命令の集合である．ここで，u，v，w はこの変換によって新しく生成された変数の名前である．
(B)ステップ(A)で得られたデータフローからマルチスレッド・コントロールフローを抽出する．
(C)ステップ(A)で得られた命令の集合に含まれるそれぞれの変数にレジスタを割り付ける．

このなかで，(A)に関しては既に存在する字句解析，構文解析等の方法で可能であるのでここでは述べず，以下(B)と(C)について述べる．

3.2 データ依存関係を表す概念

ここでは命令間の依存関係を解析するために必要となるいくつかの概念について述べる．以下，命令 (w op u v) を単に w と表記する．

- 命令間の直接依存関係[<]
  ある命令集合Sの中に (p op q w) または (p op w q) という命令があったとすると p を q の直接ディペンデントと呼び，p<q で表わす．
- 命令間の依存関係[<<]
  (1) すべての命令pについて，p<<p
  (2) 任意の命令p,q,rについて，p<qかつq<<rならばp<<r
- ディペンデント　Dep(p)={q|q<<p}
- ガバナ　Gov(p)={q|p<<q}
- 最小上限ディペンデント[Lud]
  命令 q の実行後，命令 p を参照する命令がすべて終了しているならば，q は命令 p の最小上限ディペンデントLud(p)に含まれる．
- 最小上限ガバナ[Lug]
  命令 q が命令 p の実行に必要かつ十分なデータを供給するならば，q は p の最小上限ガバナに含まれる．
- 式の頭[head(E)]
  Eの中にガバナを持たない命令の集合．
- 式の尾[tail(E)]
  Eの中にディペンデントを持たない命令の集合．

3.3 データ依存グラフからのマルチスレッド・コントロールフロー抽出

(1)単純式命令 p の次命令

p の全ての直接ディペンデント w について，p と w 以外の命令を通ったデータのパスが存在しないならば，w を p の次命令とする．

つまり w<p であり，かつ，w が p の間接的なディペンデントであれば w は p の次命令とはならない．このことにより無駄なパスを省き，オペランドマッチングを減らすことができる（図３）

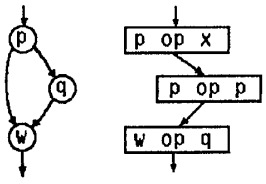


図３．無駄なパスの削除

(2)条件分岐命令 p の次命令

p が (sw b) --> (Et,Ef) という形をしているとし，tail(Et)=m，tail(Ef)=n であるとすると，b が真の場合の p の次命令は Lug(m) であり，偽の場合の p の次命令は Lug(n)である．

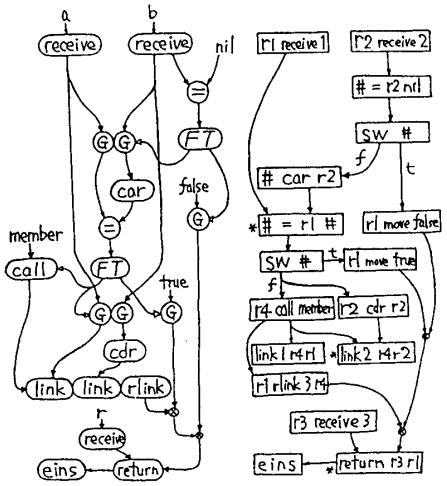
3.4 レジスタ割付の最適化

レジスタ割付はレジスタファイルを有効に利用するためにできるだけ少ないレジスタで行なわれるべきである．ある変数に割り付けられたレジスタはその変数を参照する命令の実行が全て終了したら解放することができる．そして，そのレジスタは新たな変数に割り付けることができる．レジスタを解放できるか否かの判定は Lud の概念を用いる．

4．Datarol及びその抽出アルゴリズムの評価

上記のアルゴリズムを用いて 和集合，N-Queen，行列の演算，フィボナッチ数列，素数列，階乗等を求める関数型プログラムから Datarolプログラムを抽出した．また，それらの問題を解く Datarol，データフロー，コントロールフロー（ノイマン型）の各プログラムを比較した．ここでは簡単な例を用いて Datarolとデータフローのプログラムを比較する．図３に関数 member に対するデータフローとData-rolプログラムを示す．

```
function member(a,b) return(bool)
= if b=nil then false
    elseif a=car(b) then true
    else member(a,cdr(b));
```



(A) Dataflow　　(B) Datarol

図４．関数memberのデータフロー・プログラムとDatarolプログラム

ノード数は(A)が21，(B)が17，オペランドマッチング数は(A)が8，(B)が3でともにDatarolプログラムの方が少なくなっている．また，(B)についてはDatarolプログラムのレジスタ割付を行ったので，7個の変数に対して必要なレジスタは4個となった．

一般的に，ノード数（命令数），オペランドマッチング数は Datrarolプログラムの方が少なくてすむので，関数型言語を実行する上で効率がよい．

5．おわりに

本稿では，関数型プログラムからマルチスレッドコントロールフローを抽出するアルゴリズムと，その評価について述べた．今後はこのアルゴリズムを用いたコンパイラを完成させ，より多くの問題についてDatarolプログラムの評価を行なっていく．

[参考文献]
[１] 雨宮：超多重並列処理のためのプロセッサアーキテクチャ，情報処理学会「コンピュータ・アーキテクチャ・シンポジウム」昭和６３年５月．
[２] 上田，谷口，雨宮：Datarolプロセッサのアーキテクチャについて，情報処理学会第３８回全国大会講演論文集，昭和６４年．