4V－4

共有メモリ型並列計算機上への関数型言語処理系の実装

金子 裕之

慶應義塾大学理工学研究科 計算機科学専攻

1. はじめに

遅延評価型関数型言語は参照の透明性などの好ましい性質や記述力の高さから近年注目されているが, 高階関数や遅延評価などの実現のために本質的に実行効率が良くない. そこで効率の良い処理系の構築は重要な研究課題であるといえる. また, 並列化の研究も盛んに行なわれている. 本研究では, 関数型言語 Gofer[5] のコンパイラを共有メモリ型並列計算機 Luna88K 上に実装した.

2. 関数型言語 Gofer

Gofer は Yale 大学の Mark P.Jones によって開発された遅延評価型関数型言語である. Gofer は Haskell とある程度の互換性を保ち, 処理系の大きさも比較的コンパクトである. ワークステーションからパーソナルコンピュータまでさまざまな計算機上で動作する.

3. グラフ簡約

グラフ簡約 [2] は関数プログラムをグラフに変換し, そのグラフを簡約していき弱頭部正規形 (Weak Head Normal Form) になったら簡約を終了する計算機構である. 遅延評価には

- 値は必要となるまで評価しない.
- 一度評価をおこなったものは二度と評価しない.

という特徴があるが, これらはグラフ簡約においてはサブグラフの共有および簡約の結果をグラフに破壊的に書き込むことによって実現する.

グラフ簡約は並列に実行する事に適した以下のような特徴がある [4].

- プログラムカウンタという逐次的な概念がなく, グラフ簡約は分散である.
- リダクションは並列にグラフの多数の場所で行なうことができ, スケジューリングの順番は結果に影響しない.
- 全ての通信と同期はグラフを通して行なう.

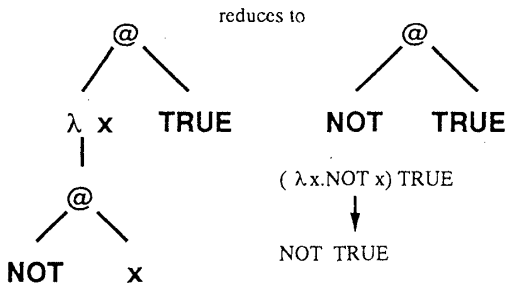

図 1: 拡張ラムダ算法のグラフ簡約の例

4. G-machine

G-machine は関数プログラムをグラフ簡約に基づき,G-Code といわれる逐次的な中間言語へ変換する抽象機械である. G-Code は逐次型言語へ変換して実行することができる.

Gofer は G-machine をベースにした抽象機械を使用している. 関数型言語のコンパイラには,G-Code をターゲットマシンのアセンブラのコードに直接変換して出力する方式 [6] をとるものと G-Code を C 言語のコードに変換して出力する方式をとるものがある. Gofer はポータビリティを高めるために後者の方式を採用している.

本研究においては並列処理用に G-machine を拡張した並列 G-machine を使用する. 並列 G-Code は Mach 上で動作する C プログラムへ変換され実行される.

The Implementation of Lazy Functional Language on Shared-Memory Multiproccesor

Hiroyuki KANEKO

Department of Computer Science, Keio University, 3-14-1 Hiyoshi, Kanagawa Pref., 223, Japan

5. 並列性の導入

一般に言語に並列性を与えるには陽 (explicit) の並列性の導入と暗黙 (implicit) の並列性の導入 [1] という 2 つのアプローチがある.

本研究では Luna88K において使用できるプロセッサ数が最大 3 である事および並列性の爆発などを考慮して陽の並列性を導入し, 注記 (annotation) を用いて Gofer プログラム中に並列実行する部分をコンパイラに指示する方法を採用した.

本研究における注記 (annotation)[3] の例

```
dsum lo hi | hi == lo = hi
           | otherwise = (dsum lo mid)
           !+! (dsum (mid + 1) hi)
           where mid = (lo + hi) / 2
```

本研究においては, ! を注記として採用した.

!+!は関数+の引数の評価を並列に実行することをプログラム中に示している.

6. 相互排除

グラフ簡約においてサブグラフは共有されているので, 複数のタスクが同時に同じサブグラフを簡約する事がありえる. 同じ結果が得られる事は明らかであるので, 一つのタスクに簡約を行なわせ残りのタスクは待っていた方が良い. 以上の理由でタスクの相互排除を行なう.

相互排除は簡約するグラフのノードへマークする事によって行なう. 簡約しようとするグラフがマークされているならばタスクは簡約を行なっているタスクがマークをはずすまで待ちに入る.

相互排除の例

```
let x = * 4 5
in + x x
```

上のプログラムにおいて+を並列に実行すると 2 つのタスクがサブグラフ (* 4 5) を同時に簡約しようとして相互排除が起こる.

7. 実装

Gofer コンパイラのソース

- gofc.c
  コンパイラのメインルーチン
- cbuildin.c
  コンパイラモジュール
- cmachine.c
  G-Code を C プログラムへ変換するルーチン

を変更, 追加した. 構文解析時に注記の付加されている関数を記憶しておき, 注記の付加している関数の引数の評価時に C スレッドライブラリを呼び出す C 言語のコードを生成し評価を並列に行なう.

参考文献

[1] Benjamin Goldberg. Multiprocessor execution of functional programs. *International Journal of Parallel Programming, Vol.17,No.5*, pp. 425–473, 1988.

[2] Simon L Peyton Jones. The implementation of functional programming languages. 1987.

[3] Simon L Peyton Jones. Parallel implementations of functional programming languages. *The Computer Journal, Vol. 32, No.2*, pp. 175–186, 1989.

[4] Simon L Peyton Jones. Implementing functional languages. 1993.

[5] Mark P.Jones. The implementation of gofer functional programming system. *Research Report*, 1994.

[6] E.G.J.M.H.Nocker S.Smetsers. Generating efficient code for lazy functional languages. *5th Functional programming languages and computer architecture, LNCS 523*, pp. 592–617, 1991.