右逆関数の自動導出によるプログラムの並列化に関する研究

数理第七研究室 森田 和孝
指導教員 武市正人 教授

1 背景と目的

大規模な問題を解く上で, 並列計算は重要である. しかし並列プログラムを構成することは逐次プログラムに比べて難しいため, 並列プログラミングは敷居が高い. この問題を解決する手法のひとつとして, 逐次プログラムの並列化に関する研究がこれまで行われてきた.

プログラムの並列化に関する定理として, 第三準同型定理 [1] がある. この定理はある特定の形で定義された逐次プログラムが並列化可能であることを示している. しかし, 第三準同型定理は並列化可能であるということを示すのみで, その並列化をどのように自動的に行うかということについては知られていなかった.

そこで本論文では, 第三準同型定理に基づいた自動的な並列化手法を提案する. 我々の手法は元の関数の右逆元である右逆関数を用いるものである. その右逆関数を用いることによって, 第三準同型定理に基づいた並列化を自動的に行うことができる. 本論文では右逆関数を自動的に導出する手法を提案し, その上で第三準同型定理を用いた並列化が自動的に行えることを示す. また, 右逆関数の導出を用いた自動並列化システムの実装を与え, その有効性を示す.

2 リスト処理関数と並列化定理

本論文の記法は関数型言語 Haskell に基づく. まず, リストを処理する関数として, 3 種類の関数定義の方法について説明する.

定義 2.1 (左方向関数). 関数 h が, ある関数 f と演算子 $\oplus$ を用いて

$$\begin{aligned} h\ [a] &= f\ a \\ h\ ([a] +\!\!+\ x) &= a \oplus h\ x \end{aligned}$$

と定義できる時, h は左方向関数である. □

定義 2.2 (右方向関数). 関数 h が, ある関数 f と演算子 $\otimes$ を用いて

$$\begin{aligned} h\ [a] &= f\ a \\ h\ (x +\!\!+\ [a]) &= h\ x \otimes a \end{aligned}$$

と定義できる時, h は右方向関数である. □

ここで演算子 $+\!\!+$ はリストの連結を表す演算子である. これらの関数はパターンマッチを用いて定義されており, それぞれリストを先頭, および末尾から走査している. この走査は順番に行われる必要があるため, このままでは効率的にリストを並列処理できない.

定義 2.3 (リスト準同型). 関数 h が, ある関数 f と結合的演算子 $\odot$ を用いて

$$\begin{aligned} h\ [a] &= f\ a \\ h\ (x +\!\!+\ y) &= h\ x \odot h\ y \end{aligned}$$

と定義できる時, h はリスト準同型である. □

この定義は定義に用いられている演算子の結合性から, リストを任意の箇所で分割して処理することが可能である. そのため, 問題を部分問題に分割することができることから, 効率的に並列処理することができる. よって並列処理を行う上でリスト準同型の形で定義することが望まれるが, 通常リスト準同型の形で定義することは容易ではない.

これら 3 種類の関数定義の関係を示す定理として, 第三準同型定理がある.

定理 2.1 (第三準同型定理 [1]). 関数 h が左方向関数かつ右方向関数である時,

$$h\ (x +\!\!+\ y) \quad = \quad h\ x \odot h\ y$$

を満たす結合的演算子 $\odot$ が存在する. □

この定理はリスト準同型の定義の存在を示すが, どのようにその定義を構築するかということについては述べていない. 我々は第三準同型定理で存在が保証されている結合的演算子が, 右逆関数を用いて構築することができることに着目し, 次に示す並列化定理を提案する. ここで右逆関数とは, 元の関数の右逆元であり, 以下関数 h の右逆関数を h° と表記する.

定理 2.2 (並列化定理). 関数 h が左方向関数かつ右方向関数である時, 任意の h° に対して

$$\begin{aligned} h\ (x +\!\!+\ y) &= h\ x \odot h\ y \\ \textbf{where}\ a \odot b &= h\ (h^\circ\ a +\!\!+\ h^\circ\ b) \end{aligned}$$

が成立する.

この並列化定理を用いることによって, 左方向関数と右方向関数の 2 種類で定義されている逐次プログラムから, 並列処理可能なリスト準同型の定義を導出することが可能となる.

3 右逆関数の導出

結合的演算子の導出には右逆関数が必要である. しかし, 組化されている関数の右逆関数は, 返り値の各組の値の間に依存関係があるため, あまり自明ではない. そこで本節

ではこのような関数の右逆関数の導出を考える．右逆関数の導出は主に 4 段階の処理によって実行される．リストの先頭から始まる部分リストの最大要素和を求める関数

$$
\begin{aligned}
(mps \vartriangle sum)\ [a] &= (a \uparrow 0, a)\\
(mps \vartriangle sum)\ ([a] +\!\!+ x) &= (0 \uparrow (a + mps\ x), a + sum\ x)\\
(mps \vartriangle sum)\ (x +\!\!+ [a]) &= (mps\ x \uparrow (sum\ x + a), sum\ x + a)
\end{aligned}
$$

を例に, 右逆関数の導出アルゴリズムについて順に説明する．

関数の展開　本手法では右逆関数の返すリストの長さは, 組化されている関数の数であると仮定する．よって $(mps \vartriangle sum)^\circ$ が長さ 2 のリストを返すと仮定し, $(mps \vartriangle sum)^\circ\ (p, s) = [a, b]$ とする．ここで a, b を p と s の式で表すことができればよい．そのため, これらの関係式を求める．右逆関数の定義から

$$(mps \vartriangle sum)\ [a, b] \quad = \quad (p, s)$$

が成り立つので, 関数を展開して整理すると,

$$
\begin{aligned}
p &= 0 \uparrow a \uparrow (a + b)\\
s &= a + b
\end{aligned}
$$

となる．これを連立方程式と考え, a と b について解けばよい．

連立方程式の求解　連立方程式を解くことができるように, 場合分けを行う．

$$
\begin{aligned}
&\{0 \le a + b \ \wedge\ a \le a + b\} \Rightarrow p = a + b,\ s = a + b\\
&\{0 \le a \ \wedge\ a + b \le a\} \quad \Rightarrow p = a, \quad s = a + b\\
&\{a \le 0 \ \wedge\ a + b \le 0\} \quad \Rightarrow p = 0, \quad s = a + b
\end{aligned}
$$

こうして得られた各連立方程式を解くと,

$$
\begin{aligned}
\forall t.\ &\{0 \ge t \ \wedge\ 0 \ge s \ \wedge\ p = 0\} \Rightarrow a = t,\ b = s - t\\
&\{p \ge 0 \ \wedge\ p \ge s\} \quad \Rightarrow a = p,\ b = s - p\\
\forall t.\ &\{s \ge 0 \ \wedge\ s \ge t \ \wedge\ p = s\} \Rightarrow a = t,\ b = p - t
\end{aligned}
$$

となる．

効率化　冗長な分岐を削除することによって効率化が可能である．$(mps \vartriangle sum)^\circ$ の例ではふたつめの条件が他の条件より広い範囲を覆うので, 他の分岐は冗長であり, 除いてもよい．よって

$$(mps \vartriangle sum)^\circ\ (p, s) = \textbf{if}\ (p \ge 0 \wedge p \ge s)\ \textbf{then}\ [p, s - p]$$

となる．

定義域の正しさの検証　右逆関数は元の関数の値域についてのみ適切な値を返せばよい．一般に

$$\forall x.\ (mps \vartriangle sum)\ x = (p, s) \quad \Rightarrow \quad p \ge s \ \wedge\ p \ge 0$$

が成り立つので, 得られた右逆関数の定義域は十分であることがわかる．この判定は数学的帰納法により行うことが可能である．

4　自動並列化システムの実装と評価実験

我々は並列化定理に基づいた自動並列化システムを実装した．このシステムの処理はは大きく 3 つに分けられる．まず第一に, 入力プログラムの正しさの判定を行う．並列化定理を用いるためには, 左方向関数と右方向関数が等しい関数でなくてはならないので, これを証明する．このことは左方向関数と右方向関数の定義が任意の長さの入力リストに対して等しい値を返すということを, 数学的帰納法を用いて示すことにより可能である．第二に, 前節の手法により入力プログラムの右逆関数を導出し, 得られた右逆関数を用いて, 並列化定理から結合的演算子を得る．最後に, 得られた結合的演算子を用いて, 分割統治法に基づいた C++ 言語の並列プログラムを出力する．

我々はシステムが対象とする関数を, 実数のリストを受け取り, ひとつの実数を返す関数に制限した．これにより, 数式処理ライブラリを用いることにより本システムを実装することが可能となった．システムを記述するにあたり, メインルーチンとして関数型言語 OCaml を, 連立方程式の求解などの数式処理として Mathematica を, 並列処理ライブラリとして SkeTo を用いた．

また, 我々は最大部分列和問題や可視地点判定問題など, 様々な例に対して並列プログラムを導出し, 効率的な並列プログラムが現実的な時間内に導出できていることを確認した．

5　まとめと今後の課題

本論文では右逆関数を自動導出する手法を示し, 右逆関数を用いた新しいプログラムの自動的な並列化手法を提案した．さらに, 制限を加えた言語上で実際に右逆関数の導出を利用した並列化アルゴリズムの効率的な実装を行い, 評価実験によりその有用性を確認した．本研究の一部は [2,3] に含まれているものである．

今後の課題としては, 並列化可能な関数のクラスを拡張, また, 右逆関数を導出可能な関数のクラスを拡張することなどが挙げられる．

参考文献

[1] J. Gibbons. The third homomorphism theorem. *Journal of Functional Programming*, Vol. 6, No. 4, pp. 657–665, 1996.

[2] K. Morita, A. Morihata, K. Matsuzaki, Z. Hu, and M. Takeichi. Automatic inversion generates divide-and-conquer parallel programs. To be presented at *Programming Language Design and Implementation*, 2007.

[3] 森田和孝, 森畑明昌, 胡振江, 武市正人. 弱逆関数の自動導出によるプログラムの並列化. 日本ソフトウェア科学会 第 23 回大会論文集, 東京大学, 2006.