Boolean Buchberger Algorithm とその並列化 (3) - 並列アルゴリズム*

佐藤 健, 毛受 哲, 岩山 登, 佐藤 洋祐†
(財) 新世代コンピュータ技術開発機構‡

1 はじめに

本稿では, Boolean Gröbner base [1] を求める逐次アルゴリズムから並列処理可能部分を取り出した並列アルゴリズムについて述べる. Boolean Gröbner base を求める逐次アルゴリズムにはいろいろな並列性がある. しかしながら, 通信のオーバーヘッドなどを考えると並列できる部分が大きいものでなければならない. この通信のオーバーヘッドは, 実装する並列マシンのアーキテクチャに影響される. そこで, 逐次アルゴリズムの実行を解析した結果 [2] をもとにして, 並列処理する部分を決定した.

2 逐次アルゴリズム内の並列性

PSI 上で implement されている逐次アルゴリズムの概略を以下に示す.

input a list of equation, EQlist.
GB := $\emptyset$
while EQlist$\neq$nil do
begin

1. Take the smallest equation, e in terms of maximal monomial from EQlist.
2. Reduce e by GB.
3. Add e to GB.
4. Take every equation in EQlist whose maximal monomial can be reduced by e.
5. Take every rule in GB whose maximal monomial can be reduced by e.
6. Reduce bodies of the rest rules in GB by GB.
7. Produce all critical pairs from e and GB.
8. Produce all self-critical pairs from e.
9. Reduce the maixmal monomial of each equation, each rule, each critical pair by GB which can be reduced by e and append non-zero equations to EQlist.

end
output GB as the Gröbner base.

このアルゴリズムをみるとわかるように以下のような並列処理できる部分があることがわかる.

1. 各項の書き換えの並列性 (ステップ 2,6,9)
   等式は項の和からなっており, それらは独立に GB による書き換えを行える.
2. EQlist 中の最大単項書き換え可能な等式の抜きだし (ステップ 4)
   EQlist 中の各等式ごとに e で書き換わるかどうかを独立にチェックできる.
3. GB 中の最大単項書き換え可能なルールの抜きだし (ステップ 5)
   GB 中の各ルールごとに e で書き換わるかどうかを独立にチェックできる.
4. GB 中のボディ書き換え可能なルールの書き換え (ステップ 6)
   GB 中のボディ書き換え可能な各ルールごとに独立に GB で書き換えを行える.
5. クリティカル・ペアの生成 (ステップ 7)
   e と GB のルールとのクリティカル・ペアの生成は各ルールごとに独立に行える.
6. セルフ・クリティカル・ペアの生成 (ステップ 8)
   e 中の変数に関するセルフ・クリティカル・ペアの生成は各変数ごとに独立に行える.

*Boolean Buchberger Algorithm and its Parallelization(3) - Parallel Algorithm

†Ken Satoh, Satoshi Menju, Noboru Iwayama, Yosuke Sato
‡Institute for New Generation Computer Technology

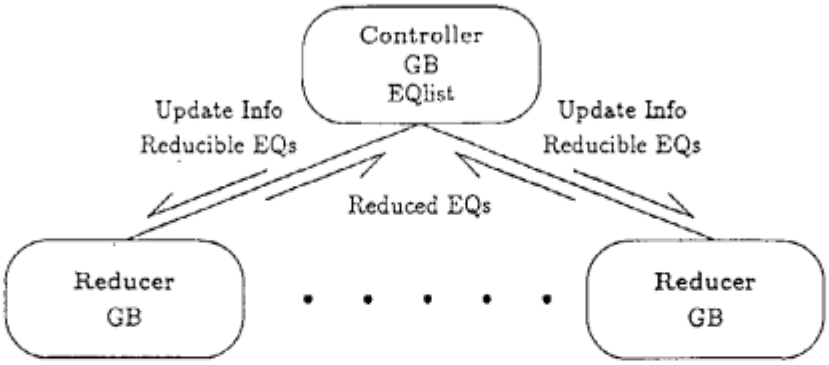

図 1: 並列アルゴリズムの構成

7. 書き換え可能な式の書き換え (ステップ 9)
書き換え可能な等式, ルール, クリティカル・ペア, セルフ・クリティカル・ペアの書き換えは独立に行える.

3 並列アルゴリズム

上のアルゴリズムのこうした並列処理可能部分をすべて並列化すればよいというわけではない. なぜなら, 実際の並列マシンで実装した場合には, 仕事を分割するための通信や, 答を集めるための通信などのオーバーヘッドがかかるからである. このオーバーヘッドが大きいと並列に処理したとしても, 逐次処理よりも遅くなる場合がありうる. さらに, これらの通信オーバーヘッドは各並列マシンで異なるのでこのアルゴリズムを実装するマシンの特性を考えながら並列処理部分を決めなければならない. われわれが実装した multiPSI は, 疎結合の分散メモリマシンであるので, できるだけ粒度の大きい並列処理可能部分を見つけだす必要があった. そこで逐次アルゴリズムの実行の解析結果 [2] を使うことによってステップ 9の部分が非常に大きく, それ以外は小さいことがわかった. このため, この部分を最大限に利用するような以下の並列アルゴリズムを考えた.

1 つのプロセッサを Controller とし, 残りのプロセッサを Reducer とする (図 1参照). Controller は, 逐次アルゴリズムにおけるステップ 1からステップ 8までを行う. Reducer は, GB の変更情報 (e の追加, ルールの消去, ルールのボディの変更) と書き換え可能な等式を Controller から受け取り, GB を変更してからステップ 9に対応する等式の書き換えを行う. もし, 0 に書き換わらない等式があれば, Controller にそれを送り返す. なお, Reducer から等式が送り返されるまで Controller が暇な状態になるので, その間は Reducer と同じタスクを実行するようにした. また, 負荷分散の方式としては, 各 Reducer に送られる書き換え可能な等式の数が均等になる方法をとった.

4 おわりに

逐次アルゴリズムの並列化について述べた. この並列化の方法で重要なのは, まず逐次アルゴリズムの並列処理可能部分を抽出し, つぎに実装する並列マシンの特性からどのような並列性を引き出せばよいかを考え, さらにその並列性にみあった部分を逐次アルゴリズムの解析によって決定することである. この方法で逐次アルゴリズムの並列性をマシンにあった形で最大限に引き出せると考える.

謝辞

本論文に貴重な御意見を頂いた ICOT の坂井氏に感謝します.

参考文献

[1] 佐藤 洋祐, 毛受 哲, 佐藤 健, 岩山 登, Boolean Buchberger Algorithm とその並列化 (1) – 基礎理論, 情報処理学会第 43 回全国大会論文集, 7N-3.

[2] 岩山 登, 佐藤 健, 毛受 哲, 佐藤 洋祐, Boolean Buchberger Algorithm とその並列化 (2) – 動作解析, 情報処理学会第 43 回全国大会論文集, 7N-4.

[3] 毛受 哲, 岩山 登, 佐藤 健, 佐藤 洋祐, Boolean Buchberger Algorithm とその並列化 (4) – 実現と評価, 情報処理学会第 43 回全国大会論文集, 7N-6.