解　説

並列アルゴリズムの複雑さ†

宮　野　悟††

1. はじめに

並列計算に期待されているものは，多くのプロセッサを働かせることにより，従来の単一プロセッサよりもはるかに高速に問題を処理できることである．高速の並列計算が強く要求されている分野は，気象予報，エアロダイナミクス，人工知能，リモートセンシングなど広い範囲におよんでいるが[23]，そのためにさまざまの並列計算機アーキテクチャが考えられ商品化されてきている．またそれらの高速化を支えるものとして並列化アルゴリズムの研究が実用・理論の両面から盛んに行われている．

本稿では多様な並列計算の研究の中から「効率のよい並列アルゴリズム」をテーマに理論面から解説する．まずこの「効率のよい」とは何を意味するのだろうか．**図-1** のような並列計算のモデルを考えよう．これは有限個の（しかしいくらでも多くの）ランダムアクセス機械 (RAM) というプロセッサと共有メモリというランダムアクセスメモリとからなる．各 RAM はその RAM のみがアクセスできる局所メモリをもち，この局所メモリに対する通常の命令セットのほか共有メモリにアクセスするための命令をもっている．各プロセッサは任意の局所または共有メモリセルに 1 ステップでアクセスできる．共有メモリおよび局所メモリの容量は限定されていない．各プロセッサは共有メモリにアクセスすることで互いに通信を行い計算を実行する．このモデルは**並列ランダムアクセス機械** (parallel random access machine, PRAM) とよばれているもので，並列計算の標準的なモデルとして一般に理解されている．

PRAM モデルでは，入力のサイズに対して「あまり多くない」プロセッサを使って「とても速く」計算する並列アルゴリズムを「効率よい」といっている．並列計算量の理論では，入力のサイズを n とするとき，プロセッサの個数が n の多項式でおさえられるとき「あまり多くない」といい，計算に要する時間が $\log n$ の多項式，たとえば $(\log n)^2$ など，でおさえられるとき「とても速い」と理解されている．

並列計算量の理論は，種々の問題における並列性の発見とその限界を明らかにすることを目指している．すなわち，並列アルゴリズムの高速化・効率化のための方式の探求と，また同時に並列化困難な問題に対してはその困難さを示すことを目的としている．

本解説では，この PRAM モデルを念頭におき，**2.** で効率のよい並列アルゴリズムとは何かについて解説し，効率のよい並列アルゴリズムをもつ問題のクラス NC を紹介する．**3.** では効率よく並列化できる問題と推論システムの証明木のサイズとの関係を述べ，効率のよい並列化の原理を述べる．**4.** ではP完全な問題という多項式時間では解くことができるが，効率のよい並列化が困難な一連の問題について概観する．本稿の内容は，著者による解説[19]の流れに沿ってその内容をできるだけ多くの方々に理解していただけるようにと解説を試みたものである．

2. 効率のよい並列アルゴリズムとは

さまざまの並列計算のモデルのなかで，PRAM は並列アルゴリズムを評価するための標準的なモデルの一つと考えられている．

PRAM では，各プロセッサの命令実行サイクルが同期しており，1 サイクルで一つの命令が実行される．各プロセッサは同じプログラムを実行するが，プロセッサ番号というプロセッサに固有のデータを用いることができるため，異なるプロ

† Complexity Theory for Parallel Algorithms by Satoru MIYANO (Research Institute of Fundamental Information Science, Faculty of Science, Kyushu University).

†† 九州大学理学部基礎情報学研究施設

セッサが互いに異なる命令を同時に実行することもある．さらに，二つ以上のプロセッサが同時に読み出しのため共有メモリの同じメモリセルにアクセスすること (concurrent read, CR) を許したり，書き込みのため同じメモリセルに二つ以上のプロセッサがアクセスすることを禁じたり(exclusive write, EW) する．このような読み書きの競合により PRAM は大きく EREW PRAM, CREW PRAM, CRCW PRAM に分類される．ここで，ER は exclusive read を表し，同時読み出しを禁じたものである．また，CR は concurrent write を表し，複数のプロセッサが書き込みのために同じメモリセルにアクセスすることを許すものである．この場合，競合の解消法として

（a） プロセッサ番号の最小のものが書き込みに成功するという解消法 (PRIORITY 解消法)

（b） すべてのプロセッサが同じ値を書き込むときのみ書き込みに成功するという解消法 (COMMON 解消法)

（c） どのプロセッサが成功してもよいという解消法 (ARBITRARY 解消法)

などを導入しなければならない．

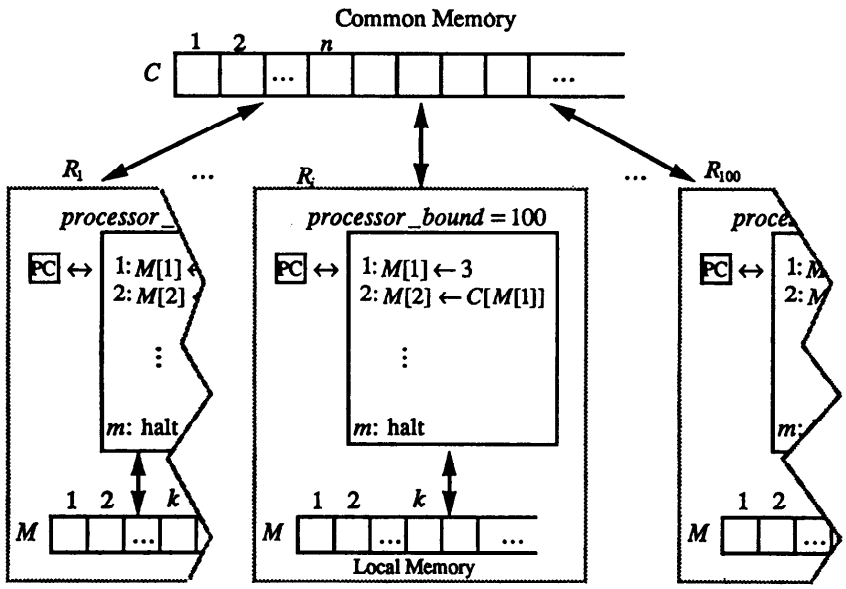
しかし，EREW PRAM, CREW PRAM と CRCW PRAM の間の差は，n 個のプロセッサをもつ CRCW PRAM の1ステップを EREW PRAM で $O(\log n)$ ステップで模倣できることが知られていることから問題にはならない．

もっと現実的なモデルとしてはプロセッサとメモリモジュールをネットワークで結んだものがある．たとえば，Hypercube[31], Cube-Connected Cycles[22], Ultracomputer[30] などがよく知られている．そして，n 個のプロセッサをもつ PRAM の1ステップを n 個のノードからなるこれらのネットワークモデルにより $O((\log n)^{O(1)})$ ステップで模倣できることが知られている．また Butterfly という多段ネットワークでは確率的なアルゴリズムにより CRCW PRAM を $O(\log n)$ ステップで模倣できることが示されている[24]．このアルゴリズムはとても単純で効率の面からも実用に適している．

このように，PRAM の1ステップを現実的なモデルで $O((\log n)^{O(1)})$ 時間で実現できるので，PRAM を並列計算の標準モデルと考えても「効率のよい並列化」の物理的な意味は失わない．

並列アルゴリズムの計算量をはかる二つの重要なメジャーに，そのアルゴリズムが使うプロセッサ数と計算を終了するまでに要する時間とがある．これらのメジャーを用いて効率よく並列化できる問題は次のように定義される．

定義 1 **効率よく並列化できる問題**とは，プロ



PC はプログラムカウンタで，プログラム（図の枠内）の第何行目を実行すべきかを指している．
processor_bound=100 は1番目から100番目までのプロセッサを起動することを表している．

図-1 PRAM

セッサ数が多項式でおさえられ，時間が $O((\log n)^k)$ ($k \geq 0$ は任意の定数) である PRAM 上の並列アルゴリズムによって解くことのできる問題である．

たとえば，ソーティングに対しては，$O(\log n)$ 時間の n プロセッサ EREW PRAM アルゴリズムが知られており[2]，ソーティングは効率よく並列化できる問題である．また文脈自由言語の認識問題に対しても，n を入力語長とするとき $O((\log n)^2)$ 時間の n^6 プロセッサ EREW PRAM アルゴリズムがあり[29]，この問題も効率よく並列化できる問題の一つである．一方，線形計画問題 (Linear Programming) についてはこのような効率のよい並列アルゴリズムが存在しないと予想されている．このことについては **4.** で解説する．

このような効率よく並列化できる問題のクラスは **NC** と記述されている．**NC** はそのクラスを同定した Nick Pippenger にちなんで Nick's Class を略したものである．この **NC** というクラスは，入力線の個数を n とするとき，深さが $O((\log n)^{O(1)})$，ゲートの個数が n の多項式でおさえられるある種の一様性をもったサイクルのない論理回路によって計算できる問題のクラスとして定義されたものである[21]．ここで，ある種の一様性をもった論理回路とは Turing 機械と呼ばれる計算モデルにより $O(\log n)$ 領域で生成される回路をさす．

その後の研究により **NC** はさまざまの並列計算のモデルによらず「効率よく並列化できる」問題のクラスであることが示されている．すなわち，上の定義 1 における PRAM を先ほどふれた Hypercube や Cube-Connected Cycles や Ultracomputer のモデルで置き換えても，**NC** は多項式個のプロセッサを用いた並列アルゴリズムによって $O((\log n)^{O(1)})$ 時間で解くことのできる問題のクラスといえる．

3. 並列化の原理

ここでは，次の例 1 にあるような推論規則を扱う．これは path system として導入されたが[3]，ここでは直観的なイメージに合うようにこれを推論システムと呼ぶことにする．そしてこの推論システムの証明木のサイズによって効率よく並列化できる問題クラス **NC** が特徴づけられることを述べる．

まず次の推論規則をみよう．

例 1

$A_1 \leftarrow$　　$A_9 \leftarrow$

$A_a \leftarrow$　　$A_b \leftarrow$

$A_c \leftarrow$　　$A_d \leftarrow$

$A_e \leftarrow$　　$A_f \leftarrow$

$A_2 \leftarrow A_1, A_9$　　$A_3 \leftarrow A_2, A_a$

$A_4 \leftarrow A_3, A_b$　　$A_5 \leftarrow A_4, A_c$

$A_6 \leftarrow A_5, A_d$　　$A_7 \leftarrow A_6, A_e$

$A_8 \leftarrow A_7, A_f$

ここで，A_t ($t \in \{1, \cdots, 9, a, \cdots, f\}$) を**文**と呼ぶ．$A_2 \leftarrow A_1, A_9$ という形の推論規則は A_1 と A_9 が成り立つならば A_2 が成り立つことを意味している．また $A_1 \leftarrow$ の形の推論規則は無条件に A_1 が成り立つことを表している．矢印の右側が空である推論規則の左側の文を**公理**ということにする．

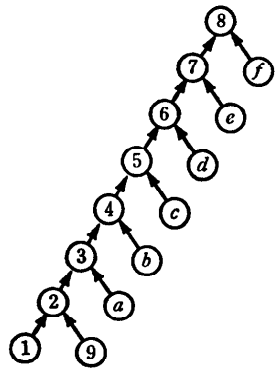


図-2 A_8 の証明木

ここでは，A_1, A_9, A_a, A_b, A_c, A_d, A_e, A_f が公理である．公理に推論規則を次々に適用して得られるものを**定理**というが，この例では，すべての A_t($t \in \{1, \cdots, 9, a, \cdots, f\}$) が定理である．$A_8$ が定理であることを示す**証明木**は**図-2** のようになるが，A_8 を導き出すために，7 段の推論が行われている．証明木のサイズはその木のノードの個数で与えるが，この例の場合 15 個の文に対して，A_8 の証明木のサイズは 15 である．このように，証明木に一つの文が高々 1 回しか現れていないとき証明木のサイズは文の個数に対して線形である．

もう一つの推論規則を例にとろう．

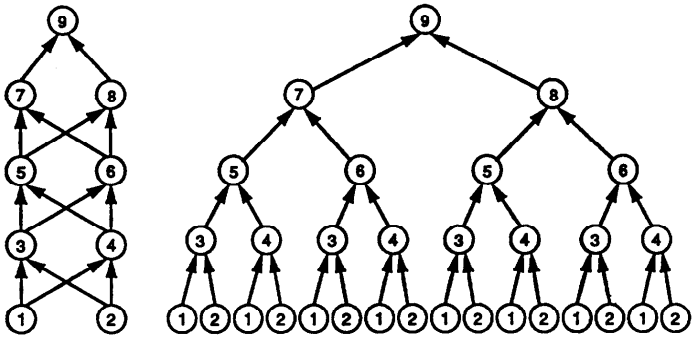


図-3　推論規則のグラフ表現と B_9 の証明木

例 2

$B_1 \leftarrow$　　$B_2 \leftarrow$

$B_3 \leftarrow B_1, B_2$　　$B_4 \leftarrow B_1, B_2$

$B_5 \leftarrow B_3, B_4$　　$B_6 \leftarrow B_3, B_4$

$B_7 \leftarrow B_5, B_6$　　$B_8 \leftarrow B_5, B_6$

$B_9 \leftarrow B_7, B_8$

この例では，図-3 のように文の間に推論関係がある．B_9 は定理であり，全体の文の個数が 9 であるのに対して，その証明木を構成してみるとそのサイズは 31 となる．それは，この証明木では一つの文が何回も重複して現れているからである．この例に見られるように，証明木のサイズは一般的には文の個数に対して指数関数的になる．

本節の目的は，並列アルゴリズムの効率を文の個数 n に関してはかるとき，**例 1** のように証明木のサイズが多項式であるときは効率のよい並列化が可能であることをみることである．一方，サイズが指数関数的であるとき一般には効率のよい並列化はのぞめない．ただし，**例 2** のようなバランスのよい証明木がある場合は効率よく並列化できることもある．

まず推論システムについての形式的な定義を与えよう．

推論システムは組 $Q=(X, R)$ で与えられる．X は**文**の有限集合，R は**推論規則**の集合である．推論規則は $\alpha \leftarrow \beta_1, \cdots, \beta_n$ の形をしている．ただし，α, $\beta_1, \cdots, \beta_n \in X (n \geq 0)$ である．特に $\alpha \leftarrow$ の形のとき，文 α を**公理**と呼ぶ．公理の集合を axiom(Q) とかく，公理を含み，推論規則に関して閉じている最小の文の集合，すなわち次の条件を満たす X の最小の部分集合を **TH**(Q) とかき，その要素を**定理**と呼ぶ．

1. axiom$(Q) \subseteq$ **TH**(Q).

2. $\beta_1, \cdots, \beta_n \in$ **TH**(Q) かつ $\alpha \leftarrow \beta_1, \cdots, \beta_n \in R$ ならば，$\alpha \in$ **TH**(Q) である．

Q の**証明木**とは，X の要素をノードのラベルとする有限木 T で次の条件を満たすものである．

（a） もしラベル α のついたノードが，ラベル $\beta_1, \cdots, \beta_n$ のついた n 個 $(n \geq 0)$ のノードを子としてもつならば $\alpha \leftarrow \beta_1, \cdots, \beta_n \in R$ である．

（b） 葉のラベルがすべて公理である．

証明木 T のノードの個数および高さをそれぞれ $size(T)$ および $height(T)$ で表す．

定理 $\gamma \in$ **TH**(Q) に対して，その定理の証明を与える最小の証明木のサイズを $size(\gamma)$ で表す．また推論システム Q のサイズを

$$size(Q) = \max \{size(\gamma) \mid \gamma \in \mathbf{TH}(Q)\}$$

で定義する．このときどの Q の定理に対してもサイズが $size(Q)$ 以下の証明木があることになる．

単純に推論システム $Q=(X, R)$ の推論を並列に行ったとしても，図-2 のような証明木の場合やはり 7 段のステップを要する．しかし，次のようにすると推論の高速化をはかることができる場合がある．まず $A_2 \leftarrow A_1, A_9$ および $A_1, A_9 \in$ axiom(Q) より $A_2 \in$ **TH**(Q) がいえる．また $A_8 \leftarrow A_7, A_7$ および $A_7 \in$ axiom(Q) より $A_8 \leftarrow A_7$ がいえる．同様にして，$A_7 \leftarrow A_6$, $A_6 \leftarrow A_5$, $A_5 \leftarrow A_4$, $A_4 \leftarrow A_3$, $A_3 \leftarrow A_2$ がいえる．これは 1 ステップで並列に行える．次のステップで，$A_8 \leftarrow A_7$ と $A_7 \leftarrow A_6$ から $A_8 \leftarrow A_6$ がいえる．同様にして，$A_6 \leftarrow A_4$, $A_4 \leftarrow A_2$ がいえる．また $A_4 \leftarrow A_2$ と $A_2 \in$ **TH**(Q) であることから $A_4 \in$ **TH**(Q) がいえる．さらにもう 1 ステップで $A_8 \leftarrow A_4$ と $A_4 \in$ **TH**(Q) から $A_8 \in$ **TH**(Q) がいえる．

アルゴリズム 1 は，このアイディアにより推論システム Q の定理の集合 **TH**(Q) を計算する CRCW PRAM の並列アルゴリズムである．ここでは推論規則の右側には高々二つの文しかないと仮定している．三つ以上の文が右辺に現れる場合は適宜新しい文と推論規則を導入すればよい．

配列 $P[1..n]$ の内容は，$P[i]=t \Longleftrightarrow p_i \in$ **TH**(Q) を満たすように計算される．配列 $D[1..n, 1..n]$ については，$p_i \leftarrow p_j$ が成り立つときに $D[j, i]=t$

```
   begin /* Q=(X, R), X={p1, ..., pn} とする */
   /* 配列 P[1..n] と D[1..n, 1..n] の初期化 */
1:   par(i, j): 1≤i, j≤n do
       begin
2:       if pi∈axiom(Q) then P[i]←t else P[i]←f;
3:       if i=j or pi←pj∈R
           then D[j, i]←t else D[j, i]←f
       end
   /* P[1..n] と D[1..n, 1..n] の計算 */
4:   repeat
       begin
5:       D←D・D/* ブール行列の積 */
6:       par(i, j, k): 1≤i, j, k≤n do
           begin
7:           if(P[j]=P[k]=t and pi←pj, pk∈R)
              or(P[j]=t and D[j, i]=t) then P[i]←t;
8:           if P[k]=t and pi←pj, pk∈R then D[j, i]←t
           end;
       end
   end
```

アルゴリズム 1: TH(Q) の計算

となる.

ライン1の **par**$(i, j): 1\leq i, j\leq n$ **do** という命令は, 各 (i, j) について並列に **do** 以下のことを実行することを表す. ライン6についても同様である. ライン5の $D\leftarrow D\cdot D$ の計算では, $p_i\leftarrow p_j$ と $p_j\leftarrow p_k$ から $p_i\leftarrow p_k$ を導くステップを計算している. ライン5は n^3 個のプロセッサを使うCRCW PRAMで $O(1)$ 時間で計算できる. ライン4の repeat は, 実際は $\lceil \log_{4/3}(size(Q))\rceil$ 回の繰り返しで十分なことが証明されている[14]. またライン7とライン8は $O(1)$ 時間で実行可能である. 配列の初期化も同様に $O(1)$ 時間で可能なことから, 全体として n^3 個のプロセッサを用いれば, CRCW PRAM上で $O(\log(size(Q)))$ 時間でこのアルゴリズムを実行できることが分かる.

以上をまとめると次の定理となる.

定理 1 推論システム $Q=(X, R)$ において, 各 $\alpha\in$**TH**(Q) がサイズが高々

$$2^{O((\log n)^{O(1)})}$$

の証明木をもっているとする. ここで n は文の個数とする. このとき, **TH**(Q) を計算する効率のよい並列アルゴリズムがある.

また逆に, 多少不正確な表現であるが, 次のことも成り立つ.

定理 2 効率のよい並列アルゴリズムをもつ問題, すなわち **NC** に属している問題は, 定理1の条件を満たす推論システムを構成し, アルゴリズム1の手法で効率よく並列化できる.

本章で述べたことは, 文献 14) を主に 8), 16), 17), 19), 27), 28) などを総合的にみたものである.

推論システムの定理の集合 **TH**(Q) を求めることによりどのように問題を解決できるかを例で説明しよう.

例 3 次の生成規則よりなる文脈自由文法 G を考える.

$$S\rightarrow AA,\ S\rightarrow AB,\ A\rightarrow AC,\ B\rightarrow DB,$$
$$A\rightarrow \mathbf{a},\ B\rightarrow \mathbf{b},\ C\rightarrow \mathbf{c},\ D\rightarrow \mathbf{d}$$

ここで, S, A, B, C, D は非終端記号, S は開始記号, **a**, **b**, **c**, **d** は終端記号である.

文脈自由言語の認識問題とは, 与えられた記号列 $w\in$ {**a**, **b**, **c**, **d**}* が文脈自由文法 G から生成されるかどうかを判定する問題である. たとえば, この文法からは記号列 $w=$**accddb** が

$$S\Rightarrow AB\Rightarrow ACB\Rightarrow ACCB\Rightarrow \mathbf{a}CCB\Rightarrow$$
$$\mathbf{ac}CB\Rightarrow \mathbf{acc}B\Rightarrow \mathbf{acc}DB\Rightarrow \mathbf{acc}DDB\Rightarrow$$
$$\mathbf{accd}DB\Rightarrow \mathbf{accdd}B\Rightarrow \mathbf{accddb}$$

のように導出される. 関係⇒の反射的推移閉包を⇒* とかくことにする. この問題を推論システムに以下のように帰着させる.

記号列 $w=x_1\cdots x_n$ に対して, 推論システム $Q_w=(X_w, R_w)$ を次のように定義する.

$X_w=\{Y[i, j]\mid 0\leq i<j\leq n, Y=S, A, B, C, D\}$.

R_w は次の推論規則からなる.

(1) $S[i, k]\leftarrow A[i, j], B[j, k]$ $(0\leq i<j<k<n)$.

(2) $A[i, k]\leftarrow A[i, j], C[j, k]$ $(0\leq i<j<k\leq n)$.

(3) $B[i, k]\leftarrow D[i, j], B[j, k]$ $(0\leq i<j<k\leq n)$.

各 $0\leq i<n$ に対して

(4) $A[i, i+1]\leftarrow$ $(x_{i+1}=\mathbf{a}$ のとき),

(5) $B[i, i+1]\leftarrow$ $(x_{i+1}=\mathbf{b}$ のとき),

(6) $C[i, i+1]\leftarrow$ $(x_{i+1}=\mathbf{c}$ のとき),

(7) $D[i, i+1]\leftarrow$ $(x_{i+1}=\mathbf{d}$ のとき),

とする. このとき非終端記号に対して

$$Y[i, j]\in\mathbf{TH}(Q_w)\Longleftrightarrow Y\Rightarrow^* x_{i+1}\cdots x_j$$

となる. このことから, 記号列 w が G から生成されるかどうかを判定するには, 開始記号 S から w が導出されるかをみればよいので, 文 $S[0, n]$ が推論システム Q_w の定理となっているかを調べればよいことが分かる. 記号列 $w=$**accddb** に対する推論システム Q_w の証明木は**図-4** のようにな

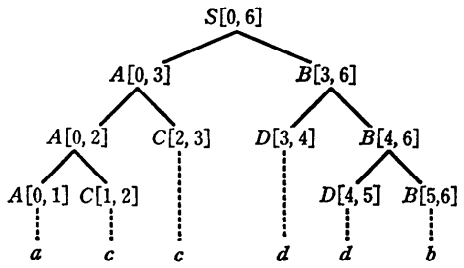


図-4 記号列 w=**accddb** に対する推論システム Q_w の証明木

る．この証明木のサイズは $O(|w|)$ である．

例3 の文脈自由言語の認識では，記号列の長さを n とするとき，一般に証明木は n 個の葉（各葉は与えられた記号列中の記号に対応する）をもつ2進木となり，そのサイズは $O(n)$ である．したがって多項式個のプロセッサを使う CRCW PRAM アルゴリズムにより $O(\log n)$ 時間で解けることが分かる．

4. 効率のよい並列化の困難な問題

逐次アルゴリズムにより多項式時間で解ける問題のクラス**P**のなかには，効率のよい並列アルゴリズムを見つけようと努力してもそのようなものが見つかっていない問題がある．そしてそれらの多くがP完全であることが示されてきている．この章では，P完全とは何か，またP完全な問題にはどのようなものがあるかについて述べる．

4.1 P完全性とは

P完全性を従来のように定義することは多少言葉の準備を必要とするので，ここでは以下のように説明しよう（P完全性の通常の定義については文献 3), 4), 12) を参照されたい）．まず，**問題**とは，ソーティング問題のように入出力関係を表す関数とする．文脈自由言語の認識問題では，入力が記号列，出力は与えられた記号列がその文法から生成されるかを表す1または0の記号となる．

定義 2 二つの問題 S と T をとる．次のような PRAM 並列アルゴリズム $\mathcal{A}$ が存在するとき，S は T に **NC 還元可能**であるという（**図-5** 参照）．

（1） $\mathcal{A}$ は T を解くサブルーチンを用いて S を解く．

（2） $\mathcal{A}$ が用いるプロセッサ数は入力サイズに関して多項式個でその時間は $O((\log n)^k)$ $(k \geq 0)$ である．ただし，$\mathcal{A}$ のなかで行われる1回のサブルーチンの呼び出しには1個のプロセッサが使われ，それに要する時間は1ステップであると勘定する．

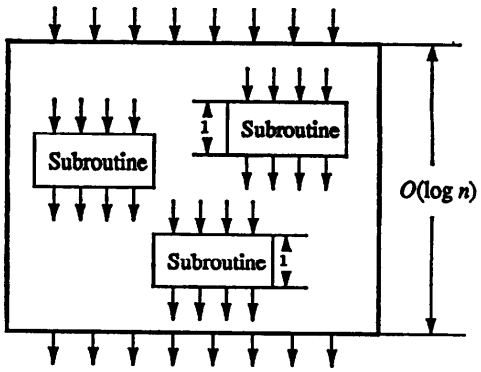


図-5 NC 還元可能性

このNC還元可能性を用いてP完全性を次のように定義する．

定義 3 問題 S_0 は次の二つの条件を満たすときP完全であるという．

（1） S_0 を解く多項式時間逐次アルゴリズムがある．

（2） 任意の $S \in \mathbf{P}$ に対して，S は S_0 に NC 還元可能である．

今，問題 S が T に NC 還元可能であるとしよう．このとき，もし T を解く効率のよい並列アルゴリズムがあるならば，それを定義2のサブルーチンとして使うことにより，S を解く効率のよい並列アルゴリズムを得ることができる．一方，$\mathbf{P} \neq \mathbf{NC}$ と強く信じられているが，このことを仮定するならば，P完全な問題は効率のよい並列アルゴリズムをもたないことになる．なぜならば，定義3と上に述べたことから，P完全な問題が効率のよい並列アルゴリズムをもつとすると，多項式時間で解けるすべての問題が効率のよい並列アルゴリズムをもつことになり，$\mathbf{P}=\mathbf{NC}$ となってしまうからである．

このことから，$\mathbf{P} \neq \mathbf{NC}$ の仮定のもとではP完全性は効率のよい並列アルゴリズムが存在しないことを導く．しかし **P** に属する問題のなかで効率よく並列化できない問題がすべてP完全になりそうにはない．実際P完全でもなくまた効率のよい並列アルゴリズムも存在しそうにない問題も見つかっている[15]．またP完全性は，たとえば，$O(n^3)$ 時間の逐次アルゴリズムによって解けるよ

うなP完全な問題に対して，$O(n)$ 時間の n^2 プロセッサ並列アルゴリズムの存在を否定するものではない．実際，P完全な問題に対して並列化により高速化を達成できた例もある[32]．

4.2 P完全な問題のいくつか

ある問題がP完全であることの証明は，その問題に効率のよい並列アルゴリズムが存在しないことの（数学的意味ではないが）証明を与えるものである．自然なP完全な問題は Cook[3] により初めて報告され，Jones と Laaser[12] はさらにいくつかのP完全な問題を見つけている．その後，NP完全な問題のように数は多くないがいくつかのP完全な問題が並列計算量との関連で発見され，P完全な問題のリストも作られている[20]．

P完全性を証明するためによく用いられる問題が次の**論理回路値問題** (Circuit Value Problem, CVP) である．これは次のように定義される．

CIRCUIT VALUE PROBLEM (CVP)

入力：2入力 AND ゲート，OR ゲートおよび NOT ゲートを使ったサイクルのない論理回路 α およびその入力線へのビット列．

問題：α の出力値が1か0かを判定せよ．

CVP は，**図-6** のような論理回路と入力が与えられたとき，その出力の値を求める問題である．AND ゲートと OR ゲートのみを使ったとき，MONOTONE CVP と呼ぶこともある．CVP および MONOTONE CVP はP完全であることが証明されている[9),13)]．すぐに分かるように，MONOTONE CVP の入力の論理回路から **3.** でとりあげた推論システムを構成し，論理回路の値を求めることとその推論システムの文が定理であるかを判定することが同値になる．そして，推論システムの定理を計算することもP完全となる．

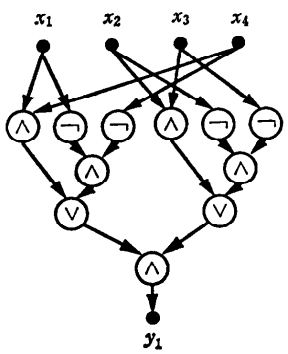


図-6 論理回路値問題

逐次アルゴリズムによって容易に計算される問題でP完全になる問題もある．$D=(V, A)$ をノードの集合が $V=\{1, \cdots, n\}$ である有向グラフとする．次のアルゴリズム2は D のノードに番号をつける**深さ優先探索アルゴリズム** (depth-first search algorithm) である．ただし，ノード $v \in V$ に対して，$\mathrm{ADJ}(v)=\{u \mid (v, u) \in A\}$ とし，$\mathrm{ADJ}(v)$ のノードには順番が与えられているとする．

```
begin/* V={1, ..., n} */
  i←0;
  for v=1 to n do visit(v)←0;
  for v=1 to n do DFS(v)
end
```

アルゴリズム 2：深さ優先探索アルゴリズム

ここで，DFS(v) は次のようにノードに番号をつけるものである．

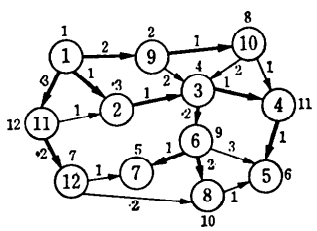
```
procedure DFS(v)
  begin
    if visit(v)=0 then
      begin
        local u;
        i←i+1;
        visit(v)←i;
        for each u∈ADJ(v) in given order
          do DFS(u)
          end
  end
```

図-7 はアルゴリズム2によりノードに番号をつけたものである．

このとき次の問題はP完全になる[25]．

DEPTH-FIRST SEARCH ORDER

入力：有向グラフ $D=(V, A)$ と $u, v \in V$．ただし，$V=\{1, \cdots, n\}$．



ノード内の番号がアルゴリズム2によりつけられた番号であり，ノードに接してつけている番号がもともとのノードの番号である．弧についている番号は ADJ(v) に順序を与えるものである．

図-7 深さ優先検索による番号づけ

問題：アルゴリズム2によって番号付けをするとき，uの番号はvの番号より小さいかを判定せよ．

アルゴリズム2のように逐次性の強いアルゴリズムによって計算される問題はP完全になることが多い．

以下，P完全な問題のいくつかを紹介しよう．さらに詳しくは文献20）を参照されたい．

（1） Linear Programming[5]：整数を要素とする$n\times d$行列Aとn-vector bとd-vector cが与えられたとき，$Ax\leq b$を満たしcxを最大にするような有理数を要素とするd-vector xを求める問題．この問題の特殊な場合である最大流問題(Maximum Flow Problem)もP完全である[10]．

（2） Lexicographically First Maximal Independent Set (LFMIS)[4]：グラフ$G=(V, E)$, $V=\{1, \cdots, n\}$，が与えられたときGの辞書式順序で最初の独立点集合を求める問題．このLFMISは次の逐次アルゴリズムによって容易に計算される．

```
begin
  U←φ;
  for i←1 to n
    if i が U のどの頂点にも隣接していない
      then U←U∪{i}
end
```

このLFMISの問題を拡張して，さまざまのグラフの問題に対応できるP完全性定理も示されている[18]．

（3） Unification[6),7),33]：二つの項（term）のmost general unifierを求める問題．

（4） Unit Resolution[12]：節（clause）の集まりが与えられたとき，unit resolutionにより空節□が導出されるかどうかを判定する問題．

（5） First Fit Decreasing Bin Packing[1]：有理数$1\geq v_1\geq\cdots\geq v_n\geq 0$が与えられているとする．容量が1の箱を必要なだけ使えるとする．この箱にv_1から順にいっぱいになるまで詰めてゆき．容量1をこえるときは次の箱に詰めていくことにする．こうしていくつかの箱を使ってv_nまで詰めたとき，各v_iが何番目の箱に入っているかを決定する問題．

（6） Context-Free Grammar Infinity Problem[12]：文脈自由文法を入力とし，その文法が無限個の記号列を生成できるかどうかを判定する問題．

5. おわりに

高度の並列化という関所はいずれ通らなければならないように思える．その問題の並列化にあたり，3. で述べた並列化の原理や，4. の効率のよい並列化が困難な一連の問題についての洞察は，並列アルゴリズムを考えていく際の羅針盤のような役割を果たすものといえるだろう．問題を並列化して計算機で高速化をはかろうとするとき，はたして期待しているほどの並列化が可能であるか．また高度の並列化が可能ならばその限界はどこにあるか．これらの問に，本稿で取り扱ったテーマは積極的に答えようとするものである．そして単にANDやORを並列化したようなヒューリスティックス[11,26]を理論的な支えのないまま扱い，台数効果だけでとらえて並列アルゴリズムの優劣を論じるようなことは避けようとしている．

並列アルゴリズムの理論面からの課題としては，P≠NC? といったきわめて難しい問題をはじめ，個々の問題に対する時間量の下限を示すことや，より少ないプロセッサ数の達成などがある．理論的な研究はますます深く複雑になり，牧歌的な雰囲気はなくなってきているが，そのもたらす知識や技法は高度の並列化へ着実に導くものと思う．

最後に，本稿の内容についてさまざまのご意見をくださった方々にこの場をかりてお礼申しあげます．また査読者の方には細やかなご配慮とご指摘をいただき深く感謝いたします．

参考文献

1) Anderson, R., Mayr, E. V. and Warmuth, M. K.: *Parallel Approximation Algorithms for Bin Packing*, Report No. STAN-CS-88-1200, Department of Computer Science, Stanford University (1988).
2) Cole, R.: *Parallel Merge Sort, Proc. 27th IEEE Symp. Foundations of Computer Science*, pp. 511-516 (1986).
3) Cook, S. A.: *An Observation on Time-Storage Trade Off, J. Comput. Syst. Sci.*, Vol. 9, pp. 308-316 (1974).
4) Cook, S. A.: *A Taxonomy of Problems with Fast Parallel Algorithms, Inf. Comput.* Vol. 64, pp. 2-22 (1985).
5) Dobkin, D., Lipton, R. J. and Reiss, S.: *Linear Programming is Log-Space Hard for P, Inf.*

Process. Lett., Vol. 9, pp. 96-97 (1979).
6) Dwork, D., Kanellakis, P. C. and Michell, J. C.: *On the Sequential Nature of Unification, J. Logic Programm.*, Vol. 1, pp. 35-50 (1984).
7) Dwork, D., Kanellakis, P. C. and Stockmeyer, L.: *Parallel Algorithms for Term Matching, SIAM J. Comput.*, Vol. 17, pp. 711-731 (1988).
8) Gibbon, A. and Rytter, W.: *Efficient Parallel Algorithms*, Cambridge University Press (1988).
9) Goldschlager, L. M.: *The Monotone and Planar Circuit Value Problems are Log Space Complete for P, SIGACT News*, Vol. 9, pp. 25-29 (1977).
10) Goldschlager, L. M., Shaw, A. and Staples, J.: *The Maximum Flow Problem is Log-Space Complete for P, Theor. Comput. Sci.*, Vol. 21, pp. 105-111 (1982).
11) Gupta, G.: *A Model for Combined And-Or Parallel Execution of Logic Programs, Proc. Int. Conf. Parallel Processing*, Vol. 2, pp. 260-263 (1989).
12) Jones, N. and Laaser, T.: *Complete Problems for Deterministic Polynomial Time, Theor. Comput. Sci.*, Vol. 3, pp. 105-117 (1977).
13) Ladner, R. E.: *The Circuit Value Problem is Log Space Complete for P, SIGACT News*, Vol. 7, pp. 18-20 (1975).
14) Mayr, E. W.: *The Design of Parallel Algorithms: Principles and Problems, Parallel Systems and Computation*, G. Paul and G. S. Almasi, eds., pp. 117-133, North-Holland (1988).
15) Mayr, E. W. and Subramanian, A.: *The Complexity of Circuit Value and Network Stability, Proc. 4th Conf. Structure in Complexity Theory*, pp. 114-123 (1989).
16) Miller, G. and Reif, J.: *Parallel Tree Contraction and its Application, Proc. 26th IEEE Symp. Foundations of Computer Science*, pp. 478-489 (1985).
17) Miyano, S.: *Parallel Complexity and P-Complete Problems, Proc. Int. Conf. Fifth Generation Computer Systems 1988*, pp. 532-541 (1988).
18) Miyano, S.: *The Lexicographically First Maximal Subgraph Problems: P-Completeness and NC Algorithms. Math. Syst. Theory*, Vol. 22, pp. 47-73 (1989).
19) 宮野 悟：並列化とその限界—理論的側面から，コンピュータソフトウェア，Vol. 7, pp. 2-15 (1990).
20) Miyano, S., Shiraishi, S. and Shoudai, T.: *A List of P-Complete Problems*, RIFIS-TR-CS-17, Research Institute of Fundamental Information Science, Kyushu University (1989).
21) Pippenger, N.: *On Simultaneous Resource Bounds, Proc. 20th IEEE Symp. Foundations of Computer Science*, pp. 307-311 (1979).
22) Preparata, F. P. and Vuillemin, J.: *The Cube-Connected Cycles: A Versatile Network for Parallel Computation, Comm. ACM*, Vol. 24, pp. 300-309 (1981).
23) Quinn, J.: *Designing Efficient Algorithms for Parallel Computers*, McGraw-Hill (1987).
24) Ranade, A. G.: *How to Emulate Shared Memory, Proc. 28th IEEE Symp. Foundations of Computer Science*, pp. 185-194 (1987).
25) Reif, J. H.: *Depth-First Search is Inherently Sequential, Inf. Process. Lett.*, Vol. 20, pp. 229-234 (1985).
26) Rothberg, E. and Gupta, A.: *Experiences Impremented a Parallel ATMS on a Shared-Memory Multiprocessor, Proc. IJCAI*, pp. 199-205 (1989).
27) Ruzzo, W. L.: *Tree-Size Bounded Alternation, J. Comput. Syst. Sci.*, Vol. 21, pp. 218-235 (1980).
28) Ruzzo, W. L.: *On Uniform Circuit Complexity, J. Comput. Syst. Sci.*, Vol. 22, pp. 365-383 (1981).
29) Rytter, W.: *On the Complexity of Parallel Parsing of General Context-Free Languages, Theor. Comput. Sci.*, Vol. 47, pp. 315-321 (1986).
30) Schwartz, J.: *Ultracomputers, ACM Trans. Prog. Lang. Syst.*, Vol. 2, pp. 484-521 (1980).
31) Seitz, C.: *The Cosmic Cube, Comm. ACM*, Vol. 28, pp. 22-33 (1985).
32) Shiloach, Y. and Vishkin, U.: *An $O(n^2 \log n)$ Parallel MAX-FLOW Algorithm, J. Algorithms*, Vol. 3, pp. 128-146 (1982).
33) Yasuura, H.: *On Parallel Computational Complexity of Unification, Proc. Int. Conf. Fifth Generation Computer Systems 1984*, pp. 235-243 (1984).

(平成2年4月16日受付)

宮野 悟（正会員）

昭和29年生．昭和52年九州大学理学部数学科卒業．昭和54年同大学院理学研究科数学専攻修士課程修了．同年，九州大学理学部基礎情報学研究施設助手．現在，同研究施設助教授．理学博士．昭和55年9月より10カ月間米国ノースウェスタン大学計算機科学科研究員．昭和60年4月～昭和62年1月西ドイツフンボルト財団研究員（パーダーボーン大学）のち昭和62年10月まで同大学研究員．並列計算の理論，計算量理論，計算学習に関する研究に従事．日本ソフトウェア科学会，日本人工知能学会，日本 OR 学会，日本数学会．EATCS（ヨーロッパ理論計算機科学会）各会員．