論理型プログラムの並列計算複雑さと組合せ論理回路による超並列計算†

岡部寿男†† 矢島脩三††

論理型プログラムの並列計算複雑さを組合せ論理回路モデルに基づいて論じ，論理型プログラムを並列計算機で処理したときの計算速度の理論的限界を与える．組合せ論理回路モデルでは，与えられた問題をできるだけ高速に解くために，並列度の最も高い専用のハードウェアを考える．組合せ回路モデル上での最高速のアルゴリズムが，その問題を解くために必要な計算時間の最小値を与えるという意味で，最高速の超並列計算モデルであるといえる．本論文では，Shapiroにより導入された論理型プログラムの計算複雑さの尺度を基に，論理型プログラムと交替Turing機械の計算量の関係について新しい結果を示す．これにより，論理型プログラムと，組合せ回路モデルを始めとする種々の理論上の並列計算モデルとの関係が明らかになる．特に，論理型プログラムを超並列に処理する組合せ回路が構成可能であることを示す．回路の段数，素子数と，論理型プログラムの複雑さの間には美しい関係が成立する．さらに，文脈自由言語などのよく知られた計算の複雑さのクラスを，論理型プログラムの複雑度で特徴付ける．ここで示された結果により，並列システムにおける論理型プログラムの計算時間や並列度の理論限界を，定量的に論じることが可能となる．

1. はじめに

Prologに代表される論理型言語は，第5世代コンピュータ用の言語として盛んに研究が行われている．論理型言語の実行の過程は並列計算とみなすことができることから，最近特に，並列処理を記述することを目的とした論理型言語が種々提案されている[1),2)]．一方，論理型プログラムを並列計算機により処理しようとする試みも各所で行われており，様々なアーキテクチャが提案され，そのいくつかは実用化されている．

並列計算の目的は，問題をできるだけ高速に解くことにある．ある問題が原理的にどこまで高速に計算できるのかを明らかにするために，計算の複雑さの理論では，計算モデルを決めてその上でアルゴリズムを評価することが行われる．並列計算の理論上のモデルとしては，並列ランダムアクセス機械 (PRAM)[3)]，交替Turing機械[4)]など様々なモデルが提案されているが，近年の集積回路技術の進歩に伴い，なかでも組合せ論理回路モデル[5)]が特に重要視されるようになってきた．組合せ論理回路モデルは，ある問題をできるだけ高速に解くために並列度の最も高い専用ハードウェアを考える立場であり，これ以上の高速計算は不可能という意味で計算速度の理論的限界を与える．「ハードウェアをいくらでも使ってよいという条件の下でどこまで高速化が可能か」という並列計算の1つの極限的なモデルであり，この上での計算は超並列計算と呼んでもよいであろう．さらに，物理的に実現可能な回路を基礎にしているので，種々の理論的な並列計算モデルの中でも最も現実的かつ安定なものの1つである．現在様々な並列システムの研究実用化が進められているが，組合せ論理回路の複雑さの理論は，こうした並列処理による高速化の限界を明らかにするといえる．

論理型プログラムは述語論理に基づいており，従来の手続き型プログラムとは種々の点で異なっている．従来から研究されている計算モデルの上での諸成果が論理型プログラムとどのような関係にあるかを明らかにするには，論理型プログラムの計算複雑さについて数学的基礎付けを与えることが必要である．特に，並列処理による論理型プログラムの効率良い実行系を考え，あるいはその高速化の限界を明らかにする上で，組合せ論理回路のような理論的かつ現実的な並列計算モデル上で，計算時間や並列度を定量的に評価することが不可欠である．このような研究はまた，組合せ回路モデル上での超並列ハードウェアアルゴリズムを，論理型プログラムを用いて記述し，汎用の並列計算機上でソフトウェア的に実現するための基礎ともなる．

本論文では，Shapiro[6)]により導入された論理型プログラムの計算複雑さの尺度を基に，種々の並列計算モデル，特に組合せ論理回路モデルと，論理型プログラムの並列計算複雑さとの関係について明らかにする[7)]．Shapiroによれば，論理型プログラムは一種の言語受理系とみなされる．ゴール節は入力系列に対応

† Parallel Complexity of Logic Programs and Highly Parallel Computation by Logic Circuits by YASUO OKABE and SHUZO YAJIMA (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 京都大学工学部情報工学教室

し，プログラムがそのゴールを証明可能なとき，入力系列が受理されるものとする．論理型プログラムの行う計算とは，証明の過程そのものである．Shapiro は，論理型プログラムの計算複雑さの尺度として，ゴール幅複雑度，深さ複雑度，長さ複雑度の3つの概念を提案し，これらがそれぞれ交替 Turing 機械における領域複雑度，時間複雑度，木幅複雑度と関連付けられることを示した[6]．

本論文では Shapiro の定義したゴール幅の定義を，領域複雑度が入力長より小さい場合も扱えるよう修正する．その結果，Shapiro の示した，論理型プログラムと交替 Turing 機械の関係が，実用的に重要な問題のクラスの場合に拡張される．さらに，並列度の低いプログラムが，並列度を高めた等価なプログラムに常に変換可能であることを示す．

この結果，Ruzzo の示した交替 Turing 機械と組合せ論理回路との間の対応関係[8]により，論理型プログラムを超並列に計算する組合せ論理回路が構成可能であることが示される．論理型プログラムの認識する言語のクラスの階層は，よく知られた並列計算複雑さのクラスの階層に対応付けられる．例えば，ゴール幅複雑度 $O(\log n)$，深さ複雑度 $\log^{O(1)}n$ の論理型プログラムで認識できる言語のクラスが，NC，すなわち多項式オーダの素子数かつ $\log^{O(1)}n$ 段の組合せ論理回路で計算できる問題のクラスに一致することが示される．

以下，2章では準備として組合せ回路と交替 Turing 機械について述べる．3章で論理型プログラムの計算複雑さの定義を述べる．4章では論理型プログラムと交替チューリング機械の相互シミュレーションのアルゴリズムを示す．5章では組合せ回路と論理型プログラムの関係を示し，論理型プログラムを用いて種々の計算の複雑さのクラスを特徴付ける．

2. 準　　備

2.1 組合せ回路

組合せ回路は，与えられた種類の論理素子を組み合わせて構成される，フィードバックループを持たない論理回路である，形式的には次のような有向非巡回グラフとして定義できる．各節点はゲートに対応し，その入次数は高々定数以下である．入次数が d の節点には d 変数の論理関数がラベルとして割り当てられ，この節点がラベルの関数を計算する論理素子に対応する．入次数0の節点は入力端子に対応し，入力変数 x がラベルとして割り当てられる．出次数0の節点は出力端子に対応する．

組合せ回路 c の素子数 (size) とは，c 中の節点の総数である．c の段数 (depth) とは，入力点から出力点にいたる最長の経路の長さである．本論文では段数によって組合せ回路の並列計算時間を評価する．

次に組合せ回路により認識される言語を定義する．回路族 (circuit family) $C=(c_1, c_2, \cdots)$ とは，各 c_n が n 入力1出力であるような組合せ回路の無限列である．c_n が入力 $x_1, x_2, \cdots, x_n \in \{0,1\}$ に対して1を出力するときかつそのときに限り系列 $x_1x_2\cdots x_n \in A (\subset \{0,1\}^*)$ であるとき，C は言語 A を認識するという．

組合せ回路と他の計算モデルとの関係を考える上で重要な概念に，回路の均質性 (uniformity) がある．ここでは Ruzzo により提案された「U_{E^*}-均質」の概念を均質性の定義として採用する[9]．これは回路族 C において，n を与えたとき $O(\log n)$ 領域限定 Turing 機械で c_n の記述を生成できるということにほぼ相当する．

2.2 交替チューリング機械

決定性 Turing 機械 (Deterministic Turing Machine, DTM)，非決定性 Turing 機械 (Nondeterministic Turing Machine, NTM) については既知とする．

交替 Turing 機械 (Alternating Turing Machine, ATM)[4] は非決定性 Turing 機械を一般化したものであり，次のように述べられる．ATM の状態は "existential" な状態 (存在状態) と "universal" な状態 (全称状態) に分けられる．NTM の場合と同じように，ATM の計算過程を状相 (configuration) の木と考える．

[定義] (交替 Turing 機械の計算木)

ATM M の入力系列 w に対する計算木 (computation tree) とは，各節点が M の w に対する状相でラベル付けられている有限または無限の木であって，以下の性質を満たすものである．

(1) 葉節点は，次動作の集合が空である状相である．
(2) 全称状態である状相の子節点は，その状相の1動作後にとりうる状相すべてである．
(3) 存在状態である状相の子節点は，その状相の1動作後にとりうる状相のうちの1つである． □

すべての葉節点が受理状相であるような有限の計算木を受理的 (accepting) であると呼ぶ．根節点が初期状相であるような受理的な計算木が存在するとき，ATM M は入力系列 w を受理すると定義する．

NTM は ATM において全称状態が存在しない場合に相当する．NTM の計算過程を OR グラフとみなすのと同様に，ATM の計算過程は AND/OR グラフと考えることができる．

指標付き交替 Turing 機械 (indexing ATM, IATM)[9] とは，ATM に次のようなダイレクトアクセスの機能を付け加えたものである．IATM は入力 head のかわりに指標テープ (index tape) と呼ばれる特別のテープを持つ．指標テープに2進数で値 "i" を書き込み，"読み出し状態" と呼ばれる特別な状態に入ると，入力系列の i 番目の記号が読み出される．したがって IATM は長さ n の入力系列中の任意の記号を step 数 $O(\log n)$ で読むことができる．

ATM が計算に要する時間 (time) が $T(n)$ であるとは，長さ n の入力を受理する計算木の高さ $\leqq T(n)$ が成り立つことである．領域 (space) が $S(n)$ であるとは，計算木中に現れる状相の作業用テープの使用量 $\leqq S(n)$ が成り立つことである．木幅 (tree size) が $Z(n)$ であるとは，解グラフの節点数 $\leqq Z(n)$ が成り立つことである．交替 (alternation) が $A(n)$ であるとは，証明木を根節点から終端節点まで辿ったときの OR 節点と AND 節点の入れ替わりの数 $\leqq A(n)$ が成り立つことと定義する．

$O(S(n))$ 領域限定・$O(T(n))$ 時間限定 ATM で認識できる言語のクラスを $A\text{-}SpTi(S(n), T(n))$ と略記する．同様に，$A\text{-}SpSz(S(n), Z(n))$ は $O(S(n))$ 領域限定・$O(Z(n))$ 木幅限定 ATM で認識できる言語のクラス，$A\text{-}SpAl(S(n), A(n))$ は $O(S(n))$ 領域限定・$O(A(n))$ 交替限定 ATM で認識できる言語のクラスを表す．

3. 論理型プログラムと計算の複雑さ

3.1 論理型プログラム[10]

F を関数記号の集合，V を変数記号の集合とする．

[定義] (項)

$F \cup V$ 上の項 (term) を次のように帰納的に定義する．

(1) 変数 $X \in V$ は項である．

(2) $t_1, \cdots, t_n$ が項であり，f が F 中の n 変数関数記号であるとき，$f(t_1, \cdots, t_n)$ は項である．

(3) (1)，(2)によって得られるものだけが項である． □

項の集合を T で表す．0変数関数記号のことを定数 (constant) と呼ぶ．

[定義] (基本論理式)

$p \in F$ を n 変数述語記号とする．$t_1, \cdots, t_n \in T$ に対し，項 $p(t_1, \cdots, t_n)$ を基本論理式 (atomic formula) と呼ぶ．(述語記号は関数記号の一種である．) □

置換 $\theta: V \to T$ は，有限集合

$$\{\langle X_1, t_1\rangle, \langle X_2, t_2\rangle, \cdots, \langle X_n, t_n\rangle\}$$

で表される．ここに X_i はすべて異なる変数，t_i は項であり，X_i を t_i に置換することを意味する．基本論理式 A に対し，$A\theta$ で，A の中に現れる各変数 X_i を同時に項 t_i に置換して得られる基本論理式を表す．$A\theta$ を A のインスタンス (instance) と呼ぶ．

2つの基本論理式 A_1, A_2 に対し，$A_1\theta = A_2\theta$ となる置換 θ が存在するとき，A_1 と A_2 は単一化可能 (unifiable) であるといい，θ を単一化置換 (unifier) と呼ぶ．さらに A と A_2 の任意の単一化置換 θ_1 に対し，$A_1\theta_1$ が $A_1\theta$, $A_2\theta_1$ が $A_2\theta$ のインスタンスになっているとき，θ を A_1 と A_2 の最大単一化置換 (most general unifier) と呼ぶ．2つの基本論理式 A_1, A_2 が単一化可能であれば，最大単一化置換が一意に存在する．

[定理] (Horn 節，論理型プログラム)

$A, B_1, \cdots, B_k$ $(k \geqq 0)$ を基本論理式とする．

$$A \leftarrow B_1, \cdots, B_k$$

の形を Horn 節(Horn clause) と呼ぶ．$k=0$ のときは

$$A \leftarrow$$

と書く．論理型プログラム P は Horn 節の有限集合である． □

論理型プログラムの例を**図1**に示す．

論理型プログラム P の計算をつぎのように定義する．基本論理式 $A_1, \cdots, A_m$ $(m \geqq 0)$ の論理積 "$A_1, \cdots, A_m$" をゴール節 (goal clause) と呼ぶ．特に $m=1$ のとき基本ゴール節 (unit goal) という．また $m=0$ のときは空ゴール節 (empty goal) と呼び，□で表す．

ゴール節

$$N = \text{"}A_1, \cdots, A_m\text{"}$$

と P 中の Horn 節

$$C = \text{"}A \leftarrow B_1, \cdots, B_k\text{"}$$

に対し，A と N 中の A_i が最大単一化置換 θ によって単一化可能であるとき，ゴール節

$eq(0, 0) \leftarrow$ (i)
$eq(s(X), s(Y)) \leftarrow eq(X, Y)$ (ii)
$sum(X, 0, Y) \leftarrow eq(X, Y)$ (iii)
$sum(X, s(Y), Z) \leftarrow sum(X, Y, W), eq(Z, s(W))$ (iv)

図1 論理型プログラムの例
Fig. 1 An example of logic programs.

$N'=$"$A_1,\cdots,A_{i-1},B_1,\cdots,B_k,A_{i+1},\cdots,A_m$" θ が N と C から θ により導出される．ゴール節 N のプログラム P からの導出 (derivation) D とは，有限または無限の列

$D=\{\langle N_i,C_i,\theta_i\rangle \mid i=0,1,2,\cdots$
N_i はゴール節，C_i は P 中の Horn 節，
$N_0=N$，N_{i+1} は N_i と C_i から θ_i により導出される$\}$

である．導出 D が $N_l=\square$ となる $\langle N_l,C_l,\theta_l\rangle$ を含むとき，D は P からの N の証明という．すなわち証明は長さ l の有限長の導出である．ゴール節 N のプログラム P からの証明が存在するとき，P は N を証明するという．

基本ゴール節 A_0 の P からの証明 R を木とみなしたものを，証明木という．

[定義] (証明木)

以下の条件を満たす木を，証明Rの証明木という．

(1) 各節点は基本ゴール節

(2) 根節点は A_0

(3) 葉節点は空ゴール節

(4) $R=\{\langle N_i,C_i,\theta_i\rangle\}$ の各ステップにおいて
$C_i=$"$A_i\leftarrow B_1,\cdots,B_k$"
で，N_i 中の A_{ij} が A_i と θ_i によって単一化されるとき，節点 A_{ij} から各 $B_1\theta_i,\cdots,B_k\theta_i$ への有向枝がそれぞれ存在する． □

図1のプログラムによる証明と証明木の例を図2に示す．

プログラム P の Herbrand 領域 $H(P)$ とは，P 中に現れる関数記号から構成される変数を含まない項全体の集合である．P 中の各述語記号 $p(t_1,\cdots,t_n)$ の各引数 $t_1,\cdots,t_n$ に，Herbrand 領域 $H(P)$ の要素を代入したものの集合 $HB(P)$ を，P の Herbrand 基底集合と呼ぶ．

論理型プログラム P の解釈とは，次のように定義される変数を含まない基本ゴール節の集合 $I(P)$ である．

$I(P)=\{A\in HB(P) \mid P$ は A を証明する$\}$．

基本ゴール節 A を前置記法 (prefix notation) で表しアルファベット F 上の系列とみたとき $I(P)$ の表す言語を，論理型プログラム P が認識する言語と呼ぶ．

3.2 論理型プログラムの計算複雑さ

Shapiro は，論理型プログラムの計算複雑さを次のように定義した[6]．

P を論理型プログラム，A_0 を基本ゴール節とする．A_0 の P からの証明を R とするとき，証明 R の証明木中の節点の数を R の長さ (length)，証明木の高さを R の深さ (depth) と呼ぶ．

証明木中の節点のゴール節の幅 (size) とは，そのゴール節にRで行われるすべての置換を順次施した基本ゴール節に現れる関数記号の数である．証明Rの証明木中のゴール節の幅の最大値を R のゴール幅 (goal size) と呼ぶ．

P において長さ n の任意の $A_0\in I(P)$ に対しゴール幅 $G(n)$ 以下の証明が存在するとき，P はゴール幅複雑度 (Goal-Size Complexity) $G(n)$ であるという．深さ複雑度 $D(n)$，長さ複雑度 $L(n)$ も同様に定義する．

直観的には，ゴール幅複雑度は Turing 機械における領域計算量に対応し，深さ複雑度，長さ複雑度は時間計算量に対応する複雑さの尺度であると考えられる．実際，論理型プログラムのゴール幅複雑度，深さ複雑度，長さ複雑度がそれぞれ交替 Turing 機械の領

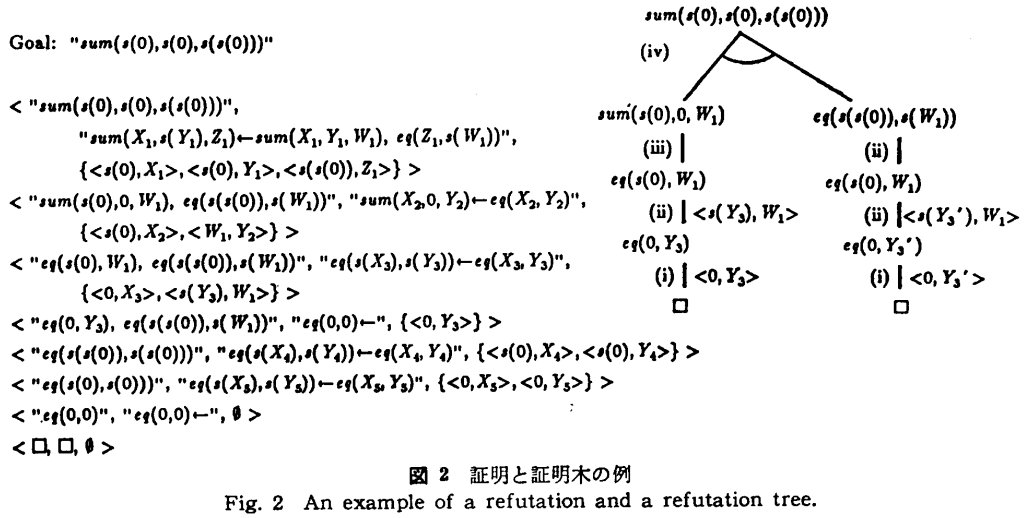


図 2 証明と証明木の例
Fig. 2 An example of a refutation and a refutation tree.

域複雑度，時間複雑度，木幅複雑度に関連付けられることを Shapiro は示した．

しかしながら，Shapiro のゴール幅の定義では，入力として与えられるゴール節と，証明において“作業領域”として生成されるゴール節とを区別していないため，Turing 機械における対数領域計算量に相当する複雑度を扱うことができない．そこで本論文では論理型プログラムのゴール幅複雑度の定義を以下のように修正する．

まず，読み出し (read only) 型の項を定義する．証明木の節点のゴール節 A に，R で行われるすべての置換を順次施した基本ゴール節 $A\theta^*$ を考える．$A\theta^*$ に現れる項 t と同じ項が初期ゴール節 A_0 に現れるとき，t は読み出し型であるという．明らかに，読み出し型の項の部分項は読み出し型である．他の読み出し型の項の部分項でない読み出し型の項を，基本読み出し型と呼ぶ．

初期ゴール節 A_0 の長さを n とする．A_0 の証明 R に現れる基本読み出し型の項 t の幅を，t に現れる関数記号の総数と $\lceil \log_2 n \rceil$ の小さいほうと定義する．証明木の節点 A のゴール節の幅とは，$A\theta^*$ 中に現れる関数記号の数の和である．ただし，基本読み出し型の項に対しては上述のようにして幅を評価するものとする．この修正されたゴール幅定義に基づいて，ゴール幅複雑度を Shapiro と同様に定義する．

上の定義は入力系列 A_0 の部分系列を A_0 へのポインタで記憶していることに対応し，Shapiro の定義によるゴール幅複雑度の自然な拡張になっている．

4. 論理型プログラムと交替 Turing 機械

4.1 ATM による論理型プログラムのシミュレーション

Shapiro は，交替 Turing 機械と論理型プログラムの関係について，次の結果を示した．

[命題 1][6] $G(n)=\Omega(n)$ のとき，ゴール幅複雑度 $G(n)$，深さ複雑度 $D(n)$，長さ複雑度 $L(n)$ の論理型プログラムは，$O(D(n)G(n))$ 時間限定・$O(G(n))$ 領域限定・$O(L(n)G(n))$ 木幅限定 ATM でシミュレートできる．

本節では，命題 1 をゴール幅複雑度 $G(n)=o(n)$ の場合に拡張した次の結果を示す．

[定理 1] $G(n)=\Omega(\log n)$，$D(n)=\Omega(\log n)$ のとき，ゴール幅複雑度 $G(n)$，深さ複雑度 $D(n)$，長さ複雑度 $L(n)$ の論理型プログラムは，$O(D(n)G(n))$ 時間限定・$O(G(n))$ 領域限定・$O(L(n)\cdot(G(n)+n\log n))$ 木幅限定・$O(D(n))$ 交替限定 ATM でシミュレートできる． □

以下，指標付き交替 Turing 機械により論理型プログラムをシミュレートし，論理型プログラム P が基本ゴール節 A_0 を証明するか否かを判定するアルゴリズムを示す．これは Shapiro が命題 1 の証明に用いた方法を修正し，シミュレーションに必要な領域が小さくなるようにしたものである．

[アルゴリズム 1]
Given : 論理型プログラム P;
Input : 基本ゴール節 A_0;
Output : P が A_0 を証明するかどうかを判定;
function UNIFY (t : 変数を含まない項, t' : 項) : 置換;
  begin
    t と t' の最大単一化置換 ϕ を推定 (存在分岐);
    UNIFY := ϕ;
    ϕ が正しいかを確認 (全称分岐);
  end {of function UNIFY};
procedure DERIVE (A : 基本ゴール節);
  begin
(R1) P から Horn 節 $C=$“$A'\leftarrow B_1,\cdots,B_k$” ($k\geq 0$) を 1 つ選ぶ (存在分岐);
(R2) σ := UNIFY (A, A');
(R3) $C\sigma$ に含まれる変数が，すべて変数を含まない項に置換されるような置換 θ を推測する (存在分岐);
(R4) **for** all $i\in\{1,\cdots,k\}$ **parallel do** (全称分岐)
      DERIVE ($B_i\sigma\theta$);
  end {of function DERIVE};
begin {of main routine}
  DERIVE (A_0) が正常終了すれば，“YES” を出力;
end. □

シミュレーションにおいては，ゴール節は作業テープ上に置かれ，項は前置記法で表現されているものとする．基本読み出し型の項は入力テープへのポインタの形で表現することが可能である．

関数 UNIFY は，入力として項 t' と変数を含まない項 t を受け取り，t と t' の最大単一化置換 ϕ を出力する．実際には，存在分岐を用いて ϕ の推定値を求

め，呼び出し側に返すとともに，全称分岐を用いて推定した置換が正しいかを確認する．(交替 Turing 機械では，このように推定した値を用いて計算を進めると同時に，並列に推定を確かめることが可能である．)

DERIVE は基本ゴール節 A の P からの導出を行う手続きである．入力である基本ゴール節 A に対し，まずプログラム P から非決定的に Horn 節 C を選択し (R1)，関数 UNIFY を呼び出して節 A と A' の最大単一化置換 σ を求める (R2)．A と A' が単一化可能であるとする．節 A は変数を含まないので，最大単一化置換 σ より A' に現れるすべての変数に変数を含まない項が代入される．つぎに C に現れる変数のうち σ で置換されない変数の値を非決定的に推測し，それを置換 θ とする (R3)．すなわち C に現れる変数はすべて，置換 $\sigma\theta$ により変数を含まない項が代入される．最後に，C の右辺に現れるすべての節 B_i について $B_i\sigma\theta$ が P から導出されるかを，全称分岐を用いて再帰的に確かめる (R4)．

A_0 を $\{0,1\}$ で符号化した長さを n とし，証明 R のゴール幅，深さ，長さをそれぞれ $G(n)$, $D(n)$, $L(n)$ とする．手続き DERIVE の1回の呼び出しにおいて必要な領域，時間，木幅が $O(\log n+G(n))$ であり，また交替が高々定数回であることに注意すれば，シミュレーションに必要な IATM の領域，時間，木幅，交替が，それぞれ $O(G(n)+\log n)$, $O(D(n)G(n)+\log^2 n)$, $O(L(n)(G(n)+n\log n))$, $O(D(n)+\log n)$ となることがわかる．以上により定理が示された．

4.2 論理型プログラムによる ATM のシミュレーション

交替 Turing 機械を論理型プログラムによりシミュレートするアルゴリズムを示す．このアルゴリズムは，Ruzzo が，$S(n)$ 領域限定・$Z(n)$ 木幅限定 ATM が，$O(S(n)\log Z(n))$ 時間限定・$O(S(n))$ 領域限定 ATM でシミュレートできることを示した際に用いたものである[9]．

$S(n)$ 領域限定・$Z(n)$ 木幅限定 ATM M を考える．r を M の領域 $\leqq S(n)$ なる状相，L をそのような状相の集合とする．(r,L) を M のフラグメント (fragment) という．フラグメント (r,L) が実現可能 (realizable) であるとは，r を根とする M の計算木で，その葉がすべて，受理状態であるかまたは L に含まれるようなものが存在することをいう．ATM M が系列 w を受理するための必要十分条件は，系列 w に対する初期状相を r_0 として，フラグメント (r_0,ϕ) が実現可能であることである．

フラグメント (r,L) が実現可能であるかどうかを判定するアルゴリズムは以下のとおりである．

[アルゴリズム 2]
Given : 指標付き交替 Turing 機械 M;
Input : 系列 $w\in\Sigma^*$;
Output : M が w を受理するかどうかを判定;
function REAL (r : ATM M の状相,
L : ATM M の状相の集合) : Boolean;
begin
if (フラグメント (r,L) が木幅 $\leqq 3$ で実現可能)
then REAL := TRUE;
else begin
ATM M の状相 s と，$L'\subseteq L$, $L''\subseteq L$ かつ $L=L'\cup L''$ であるような M の状相の集合 L', L'' を推定 (存在分岐);
REAL := (REAL(r, $L'\cup\{s\}$) $\wedge$ REAL(s, L)) (全称分岐);
end
end {of REAL};
begin
入力 w に対する M の初期状相 r_0 に対し
REAL(r_0,ϕ)=TRUE ならば "YES" を出力;
end. □

このアルゴリズムについて，Ruzzo は次のことを示した．

[補題][9] ATM M が入力系列 w を木幅 $\leqq Z(n)$ で受理するならば，REAL(r_0,ϕ) は，再帰呼び出しの深さ $O(\log Z(n))$ で停止し，TRUE を出力する．□

さらに，非決定的遷移によって選ばれるすべてのフラグメント (r,L) について $|L|\leqq 3$ という条件を加えても同じ結果が成り立つ．

次に，関数 REAL をシミュレートする論理型プログラムを記述する．ATM M の長さ n の入力系列 $w=x_1x_2\cdots x_n(\in\Sigma^*)$ を次のような高さ $\lceil\log_2 n\rceil$ の完全二分木で表すことにする．

$$w=\cdot(\cdots(\cdot(\cdot(x_1,x_2),\cdot(x_3,x_4)),\cdots,(\cdot(x_n,\$),\cdot(\$,\$))\cdots).$$

ここで "$\cdot$" は2変数関数記号，"$" は定数記号である．ATM M の状相を次のような項で表す．

$q(w_{cur}, t_{left}, t_{right})$.

ここで，q は状態に対応する関数記号，w_{cur} は，入力系列wを完全二分木とみて，指標テープの内容に従い根からたどって得られる w の部分木，t_{left}, t_{right} はそれぞれ作業用テープのヘッド位置からみて左部分（ヘッド位置を含む），右部分を表す項である．

以上の準備の下で，**図3**に示す論理型プログラム P_{real} を考える．P_{real} に，M の各状態遷移に対応する Horn 節を加えたものは関数 REAL と同じふるまいを示す．命題3より，M が w を領域 $\leqq S(n)$，木幅 $\leqq Z(n)$ で受理するならば，

$accept(q_0(w, \$, \$))$ （q_0 は M の初期状態）

の P_{real} からの深さ $O(\log Z(n))$，ゴール幅 $O(S(n))$，長さ $O(Z(n))$ の証明が存在する．このことより，次の定理を得る．

［定理2］ $S(n)$ 領域限定・$Z(n)$ 木幅限定 IATM は，ゴール幅複雑度 $O(S(n))$，深さ複雑度 $O(\log Z(n))$，長さ複雑度 $O(Z(n))$ の論理型プログラムでシミュレートできる． □

すなわち，ゴール幅複雑度が IATM の領域複雑度に比例し，長さ複雑度が木幅複雑度に比例する．また深さ複雑度は幅複雑度の対数に比例する．

IATM の時間計算量 $T(n)$ と**木幅計算量** $Z(n)$ との間に，$T(n) \geqq O(\log Z(n))$ の関係が常に成り立つことに注意すれば，定理2より次の命題2が直ちに導かれる．すなわち，定理2は命題2より強い結果である．

［命題2］[6] $T(n)$ 時間限定・$S(n)$ 領域限定・$Z(n)$ 木幅限定 ATM M は，ゴール幅複雑度 $O(S(n))$，深さ複雑度 $O(T(n))$，長さ複雑度 $O(Z(n))$ の論理型プログラムでシミュレートできる． □

命題2から，論理型プログラムのゴール幅複雑度，深さ複雑度，長さ複雑度はそれぞれ交替 Turing 機械の領域複雑度，時間複雑度，木幅複雑度に関連付けられることがわかる．

Program P_{real}

accept(*W*)←*realizable*(q_0(*W*, \$, \$), *set*(\$, \$, \$))

realizable(*Y*, *set*(*Y*1, *Y*2, *Y*3))←*realizable*(*Y*, *set*(*Z*, *Y*2, *Y*3)), *realizable*(*Z*, *set*(*Y*1, *Y*2, *Y*3))

realizable(*Y*, *set*(*Y*1, *Y*2, *Y*3))←*realizable*(*Y*, *set*(*Y*2, *Y*1, *Y*3))

realizable(*Y*, *set*(*Y*1, *Y*2, *Y*3))←*realizable*(*Y*, *set*(*Y*3, *Y*2, *Y*1))

realizable(*Y*, *set*(*Y*1, *Y*2, *Y*3))←*realizable*(*Y*, *set*(*Y*1, *Y*2, *Y*3))

図3 交替 Turing 機械をシミュレートする論理型プログラム
Fig. 3 A program for simulating ATMs.

5. 論理型プログラムと組合せ論理回路

5.1 論理型プログラムが認識する言語

定理1および定理2を用いて，従来知られている様様な計算の複雑さのクラスと，論理型プログラムで認識できる言語のクラスとを関連付けることができる．

ゴール幅複雑度 $O(G(n))$，深さ複雑度 $O(D(n))$ の論理型プログラムで認識できる言語のクラスを *L-GzDp*($G(n), D(n)$)，ゴール幅複雑度 $O(G(n))$，長さ複雑度 $O(L(n))$ の論理型プログラムで認識できる言語のクラスを *L-GzLn*($G(n), L(n)$) と略記する．

［定理3］ $D(n)=\Omega(\log n)$，$G(n) \leqq 2^{O(D(n))}$ のとき，ゴール幅複雑度 $O(G(n))$，深さ複雑度 $O(D(n))$ の論理型プログラムで認識できる言語のクラス，ゴール幅複雑度 $O(G(n))$，長さ複雑度 $O(2^{O(D(n))})$ の論理型プログラムで認識できる言語のクラス，$O(G(n))$ 領域限定・$O(2^{O(D(n))})$ 木幅限定 ATM で認識できる言語のクラスは一致する．すなわち，

$$L\text{-}GzDp(G(n), D(n)) = L\text{-}GzLn(G(n), 2^{O(D(n))}) = A\text{-}SpSz(G(n), 2^{O(D(n))})$$

（証明） 命題1および定理1より

$$L\text{-}GzLn(G(n), 2^{O(D(n))}) \supset A\text{-}SpSz(G(n), 2^{O(D(n))} \cdot (G(n)+n\log n)) = A\text{-}SpSz(G(n), 2^{O(D(n))})$$

定理2より

$$A\text{-}SpSz(G(n), 2^{O(D(n))}) \supset L\text{-}GzDpLn(G(n), D(n), 2^{O(D(n))})$$

一般に証明の長さは，証明木の深さを $D(n)$ として $2^{O(D(n))}$ を越えない．ゆえに

$$L\text{-}GzDp(G(n), D(n)) \supset L\text{-}GzDpLn(G(n), D(n), 2^{O(D(n))})$$

よって，定理は示された．（証明終） □

定理より，長さ複雑度 $Z(n)$ のプログラムに対し，ゴール幅が等しく深さ複雑度が $O(\log Z(n))$ の等価なプログラムが存在する．このことは，細長く幅が狭い証明木を持つプログラムが，浅く幅の広い証明木を持つ等価なプログラムに変換できることを示している．すなわち，並列度を高めて計算の高速化が可能であることを保証している．このことはまた，論理型プログラムの深さ複雑度と長さ複雑度が，問題の複雑さを測る尺度としてある意味で同じものであることを意味する．

5.2 並列計算複雑さのクラスの階層

前節の結果を利用して，論理型プログラムを計算す

る並列度の高い組合せ回路が構成できることを示す.

[定義] $(NC)^{11}$ 段数 $O(\log^k n)$, 素子数 $n^{O(1)}$ の均質な組合せ回路族により計算できる問題のクラスを NC^k と呼ぶ. $NC=\cup_k NC^k$ □

NC は, 比較的現実的なプロセッサ数を持つ並列計算機構により非常に高速に解ける問題のクラスである. Ruzzo は NC^k と指標付き交替 Turing 機械との密接な関係を示した.

[命題3][8] $O(\log^k n)$ 時間限定・$O(\log n)$ 領域限定指標付き交替 Turing 機械で認識できる言語のクラスは, NC^k に一致する. すなわち

$$NC^k = A\text{-}SpTi(\log n, \log^k n). \quad \square$$

さて, 定理3において $G(n)=O(\log n)$, $D(n)=O(\log^k n)$, の場合,

$$L\text{-}GzDp(\log n, \log^k n) = L\text{-}GzLn(\log n, 2^{O(\log^k n)}) = A\text{-}SpSz(\log n, 2^{O(\log^k n)})$$

が成立する. ここで,

$$A\text{-}SpTi(\log n, \log^k n) \subset A\text{-}SpSz(\log n, 2^{O(\log^k n)}) \subset A\text{-}SpTi(\log n, \log^{k+1} n)$$

であること[9]を用いると, 次の系が導かれる.

[系1]

(1) 長さ複雑度 $2^{O(\log^k n)}$, ゴール幅複雑度 $O(\log n)$ の論理型プログラムを計算する, 段数 $O(\log^{k+1} n)$, 素子数 $n^{O(1)}$ の組合せ回路が存在する.

$$L\text{-}GzLn(\log n, 2^{O(\log^k n)}) \subset NC^{k+1}.$$

(2) 段数 $O(\log^k n)$, 素子数 $n^{O(1)}$ の組合せ回路をシミュレートする, 深さ複雑度 $O(\log^k n)$, ゴール幅複雑度 $O(\log n)$ の論理型プログラムが存在する.

$$NC^k \subset L\text{-}GzDp(\log n, \log^k n).$$

(3) ゴール幅複雑度 $O(\log n)$, 深さ複雑度 $\log^{O(1)} n$ の論理型プログラムで認識できる言語のクラスは, NC に一致する.

$$L\text{-}GzDp(\log n, \log^{O(1)} n) = L\text{-}GzLn(\log n, 2^{\log^{O(1)} n}) = NC. \quad \square$$

(1)は, 論理型プログラムの計算が, 組合せ回路により高速に並列計算できることを意味する. 一方(2)は, 組合せ回路の並列計算アルゴリズムを, 論理型プログラムを用いて自然に記述することができることを示していると考えることができる.

[系2] ゴール幅複雑度 $O(\log n)$, 深さ複雑度 $O(\log n)$ の論理型プログラムで認識できる言語のクラスは, LOGCFL (文脈自由言語に対数領域還元可能な言語のクラス) に一致する.

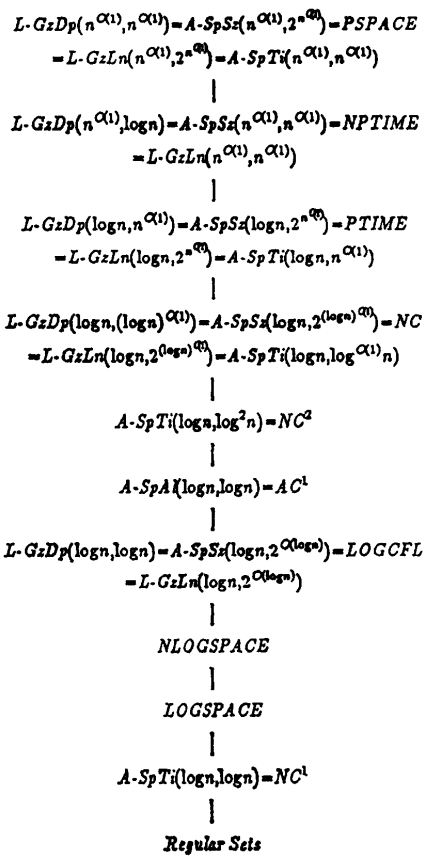


図4 論理型プログラムの計算の複雑さのクラスの階層
Fig. 4 Hierarchy of complexity classes of problems with respect to logic program complexity.

$$L\text{-}GzDp(\log n, \log n) = A\text{-}SpSz(\log n, n^{O(1)}) = \mathrm{LOGCFL}. \quad \square$$

同様にして, 論理型プログラムとよく知られた計算複雑さのクラスとの次のような関係が得られる.

$$L\text{-}GzDp(\log n, n^{O(1)}) = A\text{-}SpSz(\log n, 2^{n^{O(1)}}) = \mathrm{PTIME}.$$

$$L\text{-}GzDp(n^{O(1)}, \log n) = A\text{-}SpSz(n^{O(1)}, n^{O(1)}) = \mathrm{NPTIME}.$$

$$L\text{-}GzDp(n^{O(1)}, n^{O(1)}) = A\text{-}SpSz(n^{O(1)}, 2^{n^{O(1)}}) = \mathrm{PSPACE}.$$

ここで, PTIME, NPTIME, PSPACE は, 多項式時間限定 DTM, 多項式時間限定 NTM, 多項式領域限定 DTM でそれぞれ認識される言語のクラスである.

図4に計算複雑さのクラスの階層を示す[11].

6. おわりに

論理型プログラムの計算複雑さと交替 Turing 機械の計算複雑さの関係について新しい結果を示した. そ

れに基づき，組合せ回路と論理型プログラムの並列計算複雑さの関係について論じた．

論理型プログラムに対応する組合せ回路を示したことにより，論理型プログラムを超並列計算により計算したときの計算時間が明らかになった．論理型プログラムの高速処理のための並列システムに対し，計算速度の1つの理論的限界を与えたといえる．

論理型プログラムと種々の並列計算モデルとの関係を明らかにしたことにより，論理型プログラムで記述されたアルゴリズムを他のモデルで評価することが容易になった．論理型プログラムはいわばANDとORが非対称な交替Turing機械とみることができ，並列計算の理論的なモデルとして重要な位置を占めると考えられる．

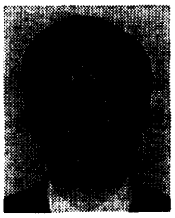
NCは，多項式素子数の組合せ回路で高速に計算できる問題のクラスであり，実際的にも重要である．Ullmanらは，logical query programと呼ばれる特別な形の論理型プログラムを用いてNCを特徴付けた[12]．これは論理型プログラムを，形式言語理論における生成文法とみなす立場と考えることができる．論理型プログラムを言語受理系（オーマトン）とみなす本論文の立場と好対照をなし，大変興味深い．この方面の研究がさらに進むことが期待される．

謝辞 ご討論いただいた本学平石裕実助教授，安浦寛人助教授，高木直史博士，石浦菜岐佐氏に感謝します．

参考文献

1) Shapiro, E. Y.: A Subset of Concurrent Prolog and Its Interpreter, ICOT Tech. Report, TR-003 (1983).
2) Ueda, K.: Guarded Horn Clauses, *Proc. Logic Programming Conference '85*, pp. 168-179 (1985).
3) Fortune, S. and Wyllie, J.: Parallelism in Random Access Machines, *Proc. 10th ACM Symp. Theory of Comput.*, pp. 114-118 (1978).
4) Chandra, A. K., Kozen, D. C. and Stockmeyer, L. J.: Alternation, *J. ACM*, Vol. 28, No. 1, pp. 114-133 (1981).
5) 安浦寛人：論理回路の複雑さの理論，情報処理，Vol. 26, No. 6, pp. 575-582 (1985).
6) Shapiro, E. Y.: Alternation and the Computational Complexity of Logic Programs, *J. Logic Programming*, Vol. 1, pp. 19-33 (1984).
7) Okabe, Y. and Yajima, S.: Parallel Computational Complexity of Logic Programs and Alternating Turing Machines, *Proc. International Conference on Fifth Generation Computer Systems*, pp. 356-363 (1988).
8) Ruzzo, W. L.: On Uniform Circuit Complexity, *J. Comput. Syst. Sci.*, Vol. 22, No. 3, pp. 365-383 (1981).
9) Ruzzo, W. L.: Tree-Size Bounded Alternation, *J. Comput. Syst. Sci.*, Vol. 21, No. 2, pp. 218-235 (1980).
10) 安浦寛人：論理型プログラムの並列計算複雑さについて，信学技報，AL83-69, pp. 9-16 (1984).
11) Cook, S. A.: A Taxonomy of Problems with Fast Parallel Algorithms, *Inf. Control*, Vol. 64, pp. 2-22 (1985).
12) Ullman, J. D. and van Gelder, A.: Parallel Complexity of Logical Query Programs, *Proc. 27th IEEE Symp. of Foundation of Computer Science*, pp. 438-454 (1986).

（平成元年8月30日受付）
（平成2年4月17日採録）



岡部 寿男（正会員）

昭和61年京都大学工学部情報卒業．昭和63年同大大学院修士課程修了．同年京都大学工学部助手．並列計算の理論，計算機基本ソフトウェア等の研究に従事．電子情報通信学会，ソフトウェア科学会，IEEE, ACM, EATCS各会員．

矢島 脩三（正会員）

昭和8年生．昭和31年京都大学工学部電気工学科卒業．同大学院博士課程修了．工学博士．昭和36年より京大工学部に勤務，昭和46年情報工学科教授．昭和35年京大第一号計算機KDC-1を設計稼動，以来，計算機，論理設計，オートマトン等の研究教育に従事．著書は「電子計算機の機能と構造」（岩波，57年）等，本学会元常務理事，元会誌編集委員（地方），元JIP編集委員．電子情報通信学会元評議員およびオートマトンと言語研専元委員長，North-Holland出版IPL編集委員，IEEE Senior Member．