数理解析研究所講究録
943巻 1996年 56-62

二分決定グラフに基づく計算複雑さに関する未解決問題について

奈良先端大 高木一義 (Kazuyoshi Takagi)

1 はじめに

近年の計算機の進歩にはめざましいものがあるが、さらなる高速かつ大規模計算への要求は増大するばかりである。特に、計算機内部で扱うことが要求される論理関数の規模はますます大きくなってきている。

二分決定グラフ (BDD あるいは OBDD : (Ordered) Binary Decision Diagram)[1, 2] は、有向非巡回グラフによる論理関数の表現である。二分決定グラフは多くの実用上重要な関数に対して必要な記憶領域が少なく、変数の順序を固定すると表現が一意に定まり、演算が高速であるという特長を持つ。このため、論理関数処理のためのデータ構造として、論理回路設計支援システムや組合せ問題の求解などの分野で多く用いられている。

BDD を一種の計算モデルとして扱い、その表現能力を計算複雑さの理論に基づいて明らかにする研究がなされてきた [3, 4]。また、手におえないサイズの BDD を分割して扱う手法を分析するために、BDD に存在限量変数を導入した計算モデル (∨BDD) の表現能力に関する研究も進められてきた [5]。

また、組合せ集合の表現にに適したゼロサプレス型 BDD が提案され [6]、BDD を集合の表現として用いる手法が示されている。ゼロサプレス型 BDD を積項集合の表現のために用いると、これを積和形論理式と見て改めて論理関数の表現として扱うことができる。[7] の 3 分決定グラフ (TDD : Ternary Decision Diagram) による表現は本質的にこれと等価であり、積項集合の表現に重点をおいて効率化を図ったものと考えられる。ゼロサプレス型 BDD や TDD によるこのような形での表現能力に関する研究も行われた [8]。

これらの研究の目的は、実際に用いられている論理関数処理の手法を定量的に解析し、その有用性と限界を解明することにあった。これまでの研究により、BDD 等による計算と Turing 機械による計算との対応付けが明らかになってきた。その過程で定義されてきた計算複雑さのクラスは、能力が非常に限定された計算モデルの振る舞いを特徴付けるも

のであり、それ自体理論的見地からも興味深いものである。しかし、これらのクラスの相互の関係については、未解決な点がいくつか残されている。本稿では、特にこれらのクラスの包含関係に注目し、未解決問題を整理して今後の進むべき方向を探る。以下、2章では基本的な定義を与える。3章では、各々の課題を列挙する。

2 準備

本稿では、計算モデルとして、Turing 機械、組合せ回路、及び二分決定グラフ等を扱う。組合せ回路、二分決定グラフ等を Turing 機械と比較するときには、一様性が問題になるが、本稿では Turing 機械が適当に非一様化されていると考えて以下では言及しない。また、本稿で述べる範囲の結果は、特に断わらない限り一様性を仮定しても成立する。

また、$\{0,1\}^*$ 上の言語のみを扱い、言語とその特性関数を同一視し、関数の計算複雑さとして記述する。

2.1 二分決定グラフ

二分決定グラフ (BDD : Binary Decision Diagram)[1, 2] は、有向非巡回グラフである。BDD には開始節点が 1 個存在する。BDD の各節点の出次数は 0 あるいは 2 である。出次数 2 の節点は、入力ビットの一つ $x_i(i = 1, 2, \ldots, n)$ をラベルとして持ち、その節点から出ている 2 本の辺はそれぞれ 0 と 1 をラベルとして持つ。節点のラベルはすべての経路において高々1 回ずつ添字の昇順に現われる。出次数 0 の節点は終端節点であり、0 または 1 をラベルとして持つ。n ビットの入力が与えられたとき、開始節点から、各節点のラベルの変数の値を持つ枝をたどって到達する終端節点のラベルが出力となる。

サイズ $n^{O(1)}$ の BDD の族で計算できる関数のクラスを *PolyBDD* と書く [3, 4]。

$\vee$BDD は、出次数 2 の節点 のいくつかが $\vee$ をラベルとして持つ BDD である [5]。ラベル $\vee$ は任意に現れ得るが、他のラベルの出現に関する制限は BDD と同様である。$\vee$BDD の計算は、$\vee$ をラベルとして持つ節点に到達したときにその節点から出ている 2 本の枝の両方を非決定的にたどること以外は分岐プログラムと同様である。少なくとも 1 つの経路が 1 をラベルとする節点に到達すれば出力は 1 となる。

サイズ $n^{O(1)}$ の $\vee$BDD の族で計算できる関数のクラスを *Poly*$\vee$*BDD* と書く。

$\oplus$BDD は、$\vee$BDD の受理条件を「奇数個の経路で受理節点へ到達すれば受理」としたモデルである。サイズ $n^{O(1)}$ の $\oplus$BDD の族で計算できる関数のクラスを *Poly*$\oplus$*BDD* と書く。

3 分決定グラフ (TDD : Ternary Decision Diagram)[7] は、有向非巡回グラフであり、BDD と異なるのは、終端節点以外の各節点の出次数が 3 であり、枝がそれぞれ 0, 1, $*$ のラベルを持つことである。入力が与えられたとき、各節点では、ラベルの変数の値を持つ枝と、$*$ を持つ枝を非決定的にたどって、1 を持つ終端節点に到達する経路が存在すれば出力は 1 となる。1 を持つ終端節点へ至る各経路を 1 つの積項に対応させると、全体として TDD は積和形 (Sum-of-Product Form) の論理式を表現しているとみなすことができる。

サイズ $n^{O(1)}$ の TDD の族で計算できる関数のクラスを *PolyTDDsop* と書く [8]。

TDD の受理条件を「奇数個の経路で受理節点へ到達すれば受理」とした場合に、サイズ $n^{O(1)}$ で計算できる関数のクラスを *PolyTDDsop* と書く。このとき、TDD の 1 を持つ終端節点へ至る各経路を積項に対応させ、全体として TDD は積-環和形 (Exclusive-or Sum-of-Product Form) の論理式を表現しているとみなすことができる。

2.2 Turing 機械

対数領域限定決定性 (あるいは非決定性) Turing 機械で計算できる関数のクラスを L (あるいは NL) と書く。

非決定性 Turing 機械で、受理条件を「奇数個の計算パスで受理状態へ到達すれば受理」とした場合に、対数領域限定で計算できる関数のクラスを$\oplus L$ と書く。

関数のクラス $\mathcal{C}$ に対し、Turing 機械の入力ヘッドの移動を 1 方向に限定したモデル、即ちオンライン Turing 機械によって定義されるクラスを *1-*$\mathcal{C}$ と書く。

2.3 組合せ回路

組合せ論理回路は、有向非巡回グラフである。入次数 0 の節点は入力素子であり、入力変数 $x_i(i = 1, 2, \ldots, n)$ あるいは定数 0, 1 が割り当てられる。入次数 d の節点は d 変数論理関数を計算する素子であり、d をファンインという。回路の素子の個数を回路のサイズという。経路の長さの最大値を、回路の深さという。回路による計算は自然に定義される。$\{0,1\}^*$ 上の関数を対象とするために、入力数 n の回路 c_n からなる族 $\{c_n\}_{(n=1,2,\ldots)}$ を一つの計算モデルとして扱う。

サイズ $n^{O(1)}$, 深さ $O(\log^k n)$ のファンイン定数のゲートから成る回路の族で計算できる関数のクラスを NC^k と書く。サイズ $n^{O(1)}$, 深さ $O(\log^k n)$ のファンイン無制限の AND,OR ゲートから成る回路 (NOT は入力でのみ現われる) の族で計算できる関数のクラスを AC^k と書く。

BDD のサイズは Turing 機械の計算領域と対応するため、組合せ回路の幅と関連づけられる。∨BDD の場合は、BDD の場合のように直接的ではないが、サイズが組合せ回路のグラフとしてのカット幅と関連する [5]。

有向非巡回グラフ $G=(V,E)$ の線形配置とは、全単射 $L:V\to\{1,2,\cdots,|V|\}$ である。全ての線形配置 L に関する、

$$\max_{1\le i<|V|}(|\{(u,v)\in E \mid L(u)\le i<L(v)\}|+|\{(u,v)\in E \mid L(v)\le i<L(u)\}|)$$

の最小値を、G の (双方向) カット幅という。組合せ回路を有向非巡回グラフとしてみたときのカット幅を回路のカット幅として定義する。

カット幅が入力のサイズ n に対し $O(\log n)$ である多項式サイズ回路族で計算できる関数のクラスを *LogCut* と書く。また、各入力に対する入力素子を高々1個づつに限定し、かつ、入力素子が定められた順に並ぶ線形配置のみに限定して、同様に定義したクラスを *1-LogCut* と書く。

図 1に示すような包含関係が知られている。実線は包含関係、斜線つきの実線は真の包含関係を表す。

3 これまでの結果と未解決の課題

3.1 *PolyTDDsop* vs *1-NL*

言語 *TAGAP* は、節点 1 から節点 n への、節点番号が昇順に並んでいる有向経路が存在する、有向グラフの記述の集合である。[8] で、*1-L PolyTDDsop* $\subseteq$ *1-NL* であることと、*TAGAP* が *PolyTDDsop* に属していることが証明されている。*TAGAP* は帰着 $\le_{1\text{-}L}$ の下で *1-NL*完全であることから、当初は *PolyTDDsop* = *1-NL* であることが予想された。しかし、これは証明されていないだけでなく、どうやら成立しないのではないかと考えられている。

PolyTDDsop が帰着 $\le_{1\text{-}L}$ で閉じていることが証明されれば、*PolyTDDsop* = *1-NL* が言える。[8] では、以下の 2 つの方針でこの問題に近づこうとしている。

- $\le_{1\text{-}L}$ より弱いある帰着では、*PolyTDDsop* は閉じている。
- *PolyTDDsop* の、別のある帰着による閉包をとると、*1-NL* に等しくすることができる。

これらの議論に改善の余地はあるだろうか。また、等号が成立しないのなら、*PolyTDDsop* $\neq$ *1-NL* である証拠となる言語が存在するだろうか。

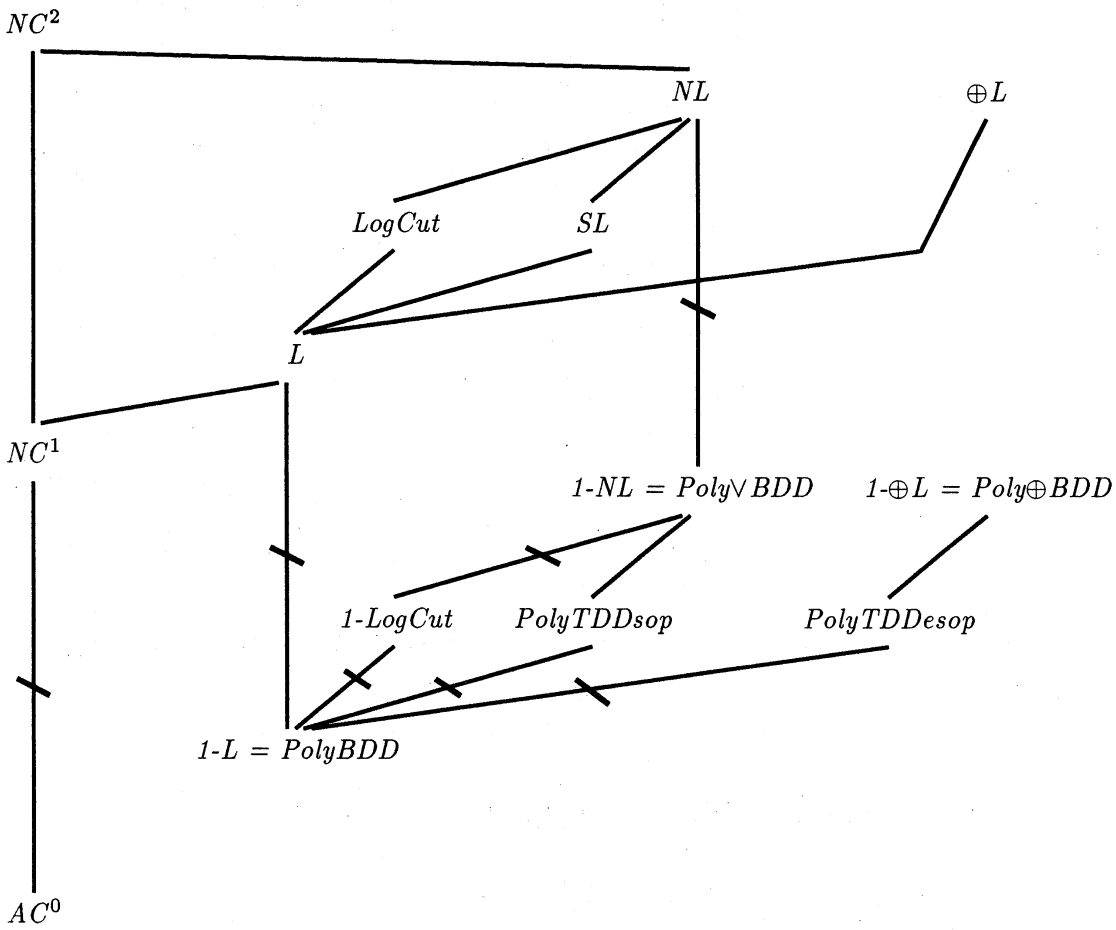

図 1: 関数のクラスの包含関係

3.2 *1-LogCut* vs *PolyTDDsop*

1-LogCut もまた、*1-L* を真に含み、*1-NL* に含まれるクラスである [5]。*PolyTDDsop* と異なり、*1-LogCut* に *TAGAP* が属さないことが証明できるので、*1-LogCut* *1-NL* である。*1-LogCut* と *PolyTDDsop* の関係は不明である。

3.3 *LogCut* vs *NL*

LogCut は、*L* を真に含み *NL* に含まれるが [5]、*NL* に真に含まれるかどうかはわかっていない。

同じく *L* と *NL* の間に存在する、自然なクラスとして知られている *SL* とは、対数領域の対称 Turing 機械で受理される言語のクラスである [9]。対称 Turing 機械とは、非決

定性 Turing 機械であり、すべての状態遷移δ に対して、その逆遷移 δ^{-1} もまた正しい状態遷移であるものである。言語 *UGAP* は、節点 1 から節点 n への経路が存在する、無向グラフの記述の集合である。*UGAP* は *SL*完全であることが知られている。

SL と *1-LogCut* の間の関係もわかっていない。

3.4 *1-NL* vs *1-*$\oplus L$

[10] では、非一様の条件の下では $NL \subseteq \oplus L$ であることが示されている。

これをそのままオンライン計算モデルに持ってくることができるだろうか。即ち、*1-NL* $\subseteq$ *1-*$\oplus L$ が言えるだろうか。これが言えると、*PolyTDDsop* と *PolyTDDesop* との何らかの関係の状況証拠が得られるかも知れない。

3.5 *PolySOP* vs *PolyESOP*

TDDsop, *TDDesop* は、それぞれ積和形、積-環和形の論理式の、リテラルを共有した形と見ることができる。では、元の論理式のサイズに関しては何が言えるだろうか。

AC^0_k は、深さ $k+1$ (経路長は枝の数で測る) の AC^0 回路で計算できる関数のクラスである。多項式サイズの積和形、和積形の論理式で計算される関数のクラスをそれぞれ *PolySOP*,*PolyPOS* と書くと、定義より、$AC^0_1 = PolySOP \cup PolyPOS$ である。これらと *PolyESOP* との関係の解明が今後の課題となるかも知れない。特に、積和形と積-環和形との表現能力の比較については古くから研究されているはずだが、計算複雑さの枠組での一般的な解析結果は知られていないようである。

4 おわりに

本稿では、二分決定グラフとその派生物に基づく計算複雑さに関する諸々の未解決の課題について述べた。本来、この一連の研究の目標は、実際に用いられている手法の定量的な解析であったが、本稿では理論的興味を主とする立場から改めて概観した。ここでは、各々のモデルに対し「多項式サイズ」という限定を基準としたが、より詳細な分析が必要となる場合もあるかも知れない。また、二分決定グラフなどに基づく複雑さを考える場合、入力の順序が大きな問題になるが、本稿では考慮しなかった。この点を考慮したモデルでも、同様の結果が導かれると思われる。

謝辞

有益なご助言、ご討論をいただく京都大学矢島研究室の皆様に感謝いたします。

参考文献

[1] S. B. Akers. Binary decision diagrams. *IEEE Trans. Comput.*, Vol. C-27, No. 6, pp. 506–516, Jun 1978.

[2] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, Vol. C-35, No. 8, pp. 677–691, August 1986.

[3] N. Ishiura and S. Yajima. A class of logic functions expressible by a polynomial-size binary decision diagram. In *Proc. Synthesis and Simulation Meeting and Int. Interchange (SASIMI '90)*, pp. 48–54, October 1990.

[4] Hiroshi Sawada, Yasuhiko Takenaga, and Shuzo Yajima. On the computational power of Binary Decision Diagrams. *IEICE Trans. Information and Systems*, Vol. E77-D, No. 6, pp. 611–618, Jun 1994.

[5] Kazuyoshi TAKAGI and Shuzo YAJIMA. On the expressive power of OBDDs with parametric variables and bounded cutwidth circuits. Technical Report KUIS-95-0005, Kyoto University, April 1995.

[6] S. Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *Proc. of ACM/IEEE DAC '93*, pp. 272–277, Jun 1993.

[7] Kouichi Yasuoka. Ternary decision diagrams as a representation of sets of products. In *RIMS kokyuroku, Kyoto University*, 1995.

[8] Koyo NITTA. Expressive power of binary decision diagrams representing boolean formulas, 1995. Master Thesis, Department of Information Science, Kyoto University.

[9] H. Lewis and C. H. Papadimitriou. Symmetric space-bounded computation. *Theoretical Computer Science*, Vol. 19, pp. 161–187, 1982.

[10] A. Wigderson. NL/poly $\subseteq \oplus$L/poly. In *Proceedings of the 9th Annual Symposium on Structure in Complexity Theory*, 1994.