アルゴリズム 20－5
（1991. 3. 15）

多項式サイズの二分決定グラフで表現可能な論理関数のクラス

石浦菜岐佐　　矢島脩三
京都大学工学部情報工学教室

二分決定グラフ (BDD) は論理関数の一表現法であり、標準形が存在する、多くの実用的な論理関数が現実的な記憶量で表現できる、など優れた特長を有している。しかし、論理関数を表現するためには、最悪の場合、入力変数の数の指数に比例した記憶領域が必要であり、これはBDDに対しても例外ではない。本稿では、どのような論理関数が現実的な記憶量で表現できるのかという問題について考える。この目的のため、本稿では、多項式サイズの二分決定グラフで表現できる論理関数のクラスを定義し、そのチューリング機械との対応を議論する。このクラスは有限オートマトンに対数限定のワークテープを付加した計算機構に対応することを示し、これを利用して、このクラスのいくつかの性質を明らかにする。さらに、組合せ論理回路との関連を議論することにより、多項式サイズの BDD で表現可能な論理関数は、段数 $O(\log^2 n)$ 段、多項式サイズの組合せ論理回路で実現できることを示す。

A Class of Logic Functions Expressible by Polynomial-Size Binary Decision Diagrams

Nagisa ISHIURA　　Shuzo YAJIMA
Department of Information Science
Faculty of Engineering
Kyoto University

A binary decision diagram (BDD) is a data structure for representing logic functions. It has good properties that there is a unique canonical form for each function and that many practical functions are expressible within feasible memory space. However, in the worst case, the size of the representation is exponential to the number of input variables. In this paper we discuss the problem of what kind of logic functions are expressible within feasible memory space. For this purpose, we define a class of logic functions expressible by BDD's whose size is bounded by a polynomial of the number of input variables. We derive some properties of this class through discussion on the relation between polynomial-size BDD's and Turing machines. We also focus on the relation between polynomial-size BDD's and combinational circuits and show that polynomial size BDD's can be synthesized into $O(\log^2 n)$ depth combinational circuits.

1 はじめに

二分決定グラフ (BDD) [Ake78, Bry86] は論理関数の表現法である。この表現法には標準形が存在し、しかも多くの実用的な論理関数が現実的な記憶量で表現できるという性質が知られている。また、論理関数に対する論理否定や、論理積、論理和などの演算も表現の大きさ (二項演算の場合はその積) に比例する計算時間で行うことができる。これは、実用に現れる多くの論理式や組合せ回路の等価性判定問題や包含性判定問題が、現実的な時間で解けることを意味する。この優れた性質によって、BDD は論理設計検証 [Fuj88, Min89, Ish90] をはじめ、論理回路の故障検査 [Cho89, Tak90]、論理合成 [Mat89] など、多くの論理設計支援の応用に用いられるようになっている。しかし、論理関数を表現するためには、最悪の場合、入力変数の数の指数に比例した記憶領域が必要であり、これは BDD に対しても例外ではない。どのような論理関数が現実的な記憶量で表現できるのかという問題は、上記の応用プログラムの適用可能性や限界を知る上で、非常に重要であると考えられるが、これに関する研究はまだ少なく、[Bry90] が見られるのみである。これは BDD のサイズが入力変数の指数となる関数について論じたものであり、2 進数乗算の出力に現れる関数を表現する BDD のサイズは多項式ではおさえられないこと、および、BDD サイズが入力変数の指数に比例して大きくなる論理関数の一クラスとその AT^2 との関係を議論している。これに対し我々は、多項式サイズの BDD で表現可能な論理関数のクラスについて考える。本稿では、多項式サイズの BDD とチューリング機械の関係を議論することにより、このクラスの論理関数の性質を導く。また、組合せ論理回路との関係を議論することにより、このクラスの関数を組合せ回路として実現したときの段数の上界を導く。主な結果は次のとおりである。

(1) サイズが入力変数の多項式でおさえられる一様な BDD 族で表現される論理関数族 (言語) のクラスを *PolyBDD* とする。また、対数領域限定の一方向チューリング機械で、入力のサイズをテープを読む前に知ることができる能力をもつもの LSIA によって計算できる論理関数族のクラスを *LOGIREG* とする。このとき、$PolyBDD = LOGIREG$ が成立する。

(2) LSIA の性質から、全ての対象関数、重みが多項式でおさえられるしきい値関数、選択関数などは *PolyBDD* に属する。

(3) *PolyBDD* は NC^2 に属する。すなわち、多項式サイズの BDD 族で表現可能な論理関数族は、$O(\log^2 n)$ 段の組合せ回路族で実現可能である。

以下、まず 2 章で *PolyBDD* の定義をおこない、3 章では そのチューリング機械との関係を、4 章では組合せ回路との関係を議論する。

2 二分決定グラフとその族

2.1 二分決定グラフ (BDD)

$\mathcal{B} = \{0, 1\}$ 上の**二分決定グラフ** (BDD) を次のように定義する。

Def 2.1 $\mathcal{B}$ 上の二分決定グラフは 6 つ組 $B = (X, N, s, l, e^0, e^1)$ である。ただし、

$X = \{x_1, x_2, \cdots, x_n\}$ は**変数の全順序集合**で
$x_1 < x_2 < \cdots < x_n$ が成り立つ。
N は**節点の集合**。
$s \in N$ は**初期節点**。
$l : N \to (X \cup \mathcal{B})$ は**節点のラベル**。
$e^0, e^1 : N \to N$ はそれぞれ節点から出る 0 枝と 1 枝を表し、次を満たす。

$$\forall v \in N \text{ s. t. } l(v) \in X :$$
$$l(e^0(v)) \in \mathcal{B} \text{ or } l(v) < l(e^0(v)), \text{ and}$$
$$l(e^1(v)) \in \mathcal{B} \text{ or } l(v) < l(e^1(v)),$$
$$\forall v \in N \text{ s. t. } l(v) \in \mathcal{B} :$$
$$e^0(v) = e^1(v) = v.$$

□

N は二つの集合 N_V と N_R に分割される ($N = N_V \cup N_R$, $N_V \cap N_R = \phi$)。N_V は**変数節点**の集合、N_R は**値節点**の集合であり、それぞれ、$N_V = \{v \mid v \in N, l(v) \in X\}$、$N_R = \{v \mid v \in N, l(v) \in \mathcal{B}\}$ と定義される。N_R は、$l(r_0) = 0$ および $l(r_1) = 1$ を満たす 2 つの要素 r_0、r_1 からなる。r_0、r_1 をそれぞれ **0 節点**、**1 節点**と呼ぶ。変数 $x_i \in X$ のインデックスを $\iota(x_i)$ と表記することにする。すなわち、$\iota(x_i) = i$ である。節点の対 $(v, e^0(v))$、$(v, e^1(v))$ はそれぞれ節点 v の **0 枝**、**1 枝** と呼ばれる。BDD B に対し、$\mathrm{size}(B)$ は B のサイズを表す。BDD B のサイズは、B の節点数と定義される。すなわち、$|S|$ を集合 S の要素数とすると、$\mathrm{size}(B) = |N|$ である。

A を X に対する**割当て**の集合とする。ただし、X に対する割当て a とは、$a \in \mathcal{B}^{|X|}$ なるベクト

ルである。a の i 番目の要素を a_i と表記し、変数 x_i への割当てと呼ぶ。与えられた割当て a に対し、a のもとで初期節点 s から到達可能な節点の集合 $T(a)$ を次のように定義する。

$$v \in T(a) \text{ iff } v = s \text{ or } \exists u \in T(a) \text{ s. t. } e^{a_{i(l(u))}}(u) = v.$$

$T(a)$ には r_0、r_1 のいずれか一方だけが必ず含まれる。BDD によって表される論理関数は、$T(a)$ を用いて次のように定義される。

Def 2.2 次の論理関数 f_B を BDD B によって表現される論理関数と定義する。

$$f_B(a) = 1 \text{ iff } r_1 \in T(a). \qquad \square$$

変数節点 v が $e^0(v) = e^1(v)$ を満たすとき、v は**冗長な節点** と呼ばれる。また、二つの変数節点 v_1 と v_2 が $e^0(v_1) = e^0(v_2)$ かつ $e^1(v_1) = e^1(v_2)$ を満たすとき、v_1 と v_2 は**等価な節点**と呼ばれる。冗長な節点と等価な節点は、BDD が表現する論理関数を変更することなく除去することができる。この操作を**規約化** (reduction) と呼ぶ。

Def 2.3 次の (1)、(2) の操作を冗長な節点、等価な節点がなくなるまで繰り返すことを BDD の規約化という。

(1) v が冗長な節点であれば、v を削除する。$e^0(u) = v$ または $e^1(u) = v$ なる変数節点 u があれば、これをそれぞれ $e^0(u) = e^0(v)$、$e^1(u) = e^0(v)$ に変更する。

(2) v_1 と v_2 が等価な節点であれば、v_1 を削除する。$e^0(u) = v_1$ または $e^1(u) = v_1$ なる変数節点 u があれば、これをそれぞれ $e^0(u) = v_2$、$e^1(u) = v_2$ に変更する。

$\square$

与えられた変数の全順序の下で、ある論理関数 f を表す BDD は複数存在するが、これらに規約化を行なって得られるものは、唯一であることが知られている [Ake78, Bry86]。ただし、本稿の議論では、BDD は規約化されたものには限っていない。

次に BDD の記述を定義する。X および N の要素 e は識別子を持っているものとし、これを $\mathrm{id}(e)$ で表すことにする。4つ組 $D_v = (\mathrm{id}(v), \mathrm{id}(l(v)), \mathrm{id}(e^0(v)), \mathrm{id}(e^1(v)))$ を**節点 v の記述**と呼ぶ。BDD の全ての節点に対する記述を集めれば、BDD 全情報を記述することができる。

Def 2.4 $D_B = \{D_v \mid v \in N\}$ を BDD B **の節集合記述**と呼ぶ。 $\square$

2.2 BDD 族とその一様性

入力変数の数に対する BDD のサイズを議論するため、BDD 族を定義する。

Def 2.5 BDD 族 $\{B_n\}$ とは BDD の系列 $B_1, B_2, B_3, \cdots$ であり、かつ、各 $B_n = (X_n, N_n, s_n, l_n, e^0{}_n, e^1{}_n)$ に対して $|X_n| = n$ が成立するものをいう。 $\square$

$\{f_n\} = f_1, f_2, f_3, \cdots$ を $f_n : B^n \to B$ なる論理関数 (すなわち n 変数論理関数) の系列とする。次の対応を考えることにより、$\{f_n\}$ は B 上の言語 L を表していると考えることができる。

$$b_1 b_2 \cdots b_n \in L \text{ iff } f_n(b_1, b_2, \cdots, b_n) = 1.$$

同様にして、**BDD 族の表す言語** を定義することができる。

Def 2.6 BDD 族 $\{B_n\}$ の表す言語を $L_{\{B_n\}}$ と表記し、次のように定義する。

$$b_1 b_2 \cdots b_n \in L_{\{B_n\}} \text{ iff } f_{B_n}(b_1, b_2, \cdots, b_n) = 1. \qquad \square$$

本稿では、BDD 族とチューリング機械との対応を議論する。この目的のために、組合せ回路族の一様性 [Ruz81] と同様に、BDD 族の**一様性**を定義する。

Def 2.7 BDD 族 $\{B_n\}$ は、その n 番目の BDD B_n の節点集合記述が n の2進数記述から $O(\log \mathrm{size}(B_n))$ 領域限定のチューリング機械で生成できるとき、**一様**であるという。 $\square$

現実的なサイズの BDD の族で表現可能な言語のクラスとして、*PolyBDD* というクラスを定義する。

Def 2.8 一様な BDD 族 $\{B_n\}$ のうち、$\mathrm{size}(B_n) \leq \mathrm{poly}(n)$ ($\mathrm{poly}(n)$ は n の多項式) を満たすものによって表現される言語のクラスを *PolyBDD* と定義する。 $\square$

3 多項式サイズの二分決定グラフと対数領域オートマトンの関係

3.1 対数領域オートマトン

入力サイズ n に対して $O(\log n)$ のワークテープをもつ一方向オフラインチューリング機械を**対数領域オートマトン** (LSA) と呼ぶことにする。LSA

に対する入力は読み出しだけが可能な入力テープに与えられる。LSA は入力テープに書かれた値を一度だけ、かつ与えられた順序に読むことだけができる。ワークテープは任意回の読み出し / 書き込みが可能である。言い換えれば、LSA は有限オートマトンに読み書き可能な $O(\log n)$ のテープを付加したものである。次に、LSA を拡張したものとして、**入力サイズ先読み対数領域オートマトン (LSIA)** を定義する。LSIA は機構的には LSA と同じであるが、与えられた入力系列の長さを、入力テープを走査することなく知ることができる。入力系列の長さは、2 進数表現でワークテープの初期値として与えられる。LSA と LSIA は、入力テープを一回しか走査できないため、入力サイズを事前に知ることにより、能力に差が生じうる。

Def 3.1 LSA、LSIA により受理される言語のクラスをそれぞれ *LOGREG*、*LOGIREG* とする。 □

本論文の主要な結果は、*PolyBDD* が *LOGIREG* に一致することである。

Th 1 $PolyBDD = LOGIREG$.

($LOGIREG \subseteq PolyBDD$):
与えられた LSIA A に対し、各 BDD のサイズが多項式でおさえられており、かつ一様な BDD の族が構成できることを示す。長さ n の入力系列に対して A が使うワークテープの長さは $O(\log n)$ であるから、このときの A の状態数 p_n は n の多項式でおさえられる。A の状態遷移グラフを展開することにより、$p_n \times n$ 状態で、ループを持たない状態遷移グラフが得られる。i 番目の入力を読む状態を i 番目の変数をラベルとする変数節点に、初期状態を初期節点に、n 番目の入力による遷移先の状態のうち受理状態であるものを 1 節点に、非受理状態であるものを 0 節点に置き換え、入力 0、1 による遷移枝をそれぞれ 0 枝、1 枝に置き換えると、$|X| = n$ の BDD B_n が得られる。構成法より、B_n は A が $b_0 b_1 \cdots b_n$ を受理するときそのときに限り値が 1 となる論理関数を表現している。従って、この方法により BDD 族 $\{B_n\} = B_1, B_2, \cdots$ を構成すれば、$\{B_n\}$ は A と同じ言語を受理する。B_n の節点数は $p_n \times n$ であり、n の多項式でおさえられる。また、A の状態遷移関数が n の 2 進数表現 (ワークテープの初期値) から $O(\log n)$ 領域で計算できるので、B_n の記述も n の 2 進数表現から対数領域のチューリング機械で生成することができる。すなわち、$\{B_n\}$ は一様な BDD 族である。

($PolyBDD \subseteq LOGIREG$):
一様かつ多項式サイズの BDD 族 $\{B_n\}$ に対し、同じ言語を受理する LSIA を構成できることを示す。今、入力系列として $b_1 b_2 \cdots b_n$ が与えられたとする。入力系列の長さ n は、ワークテープに初期値として与えられるので、LSIA は $O(\log n)$ のワークテープを使うことにより、$\{B_n\}$ の n 番目の BDD B_n の記述を生成する $O(\log |B_n|)$ 領域限定のチューリング機械をシミュレートすることができる。これは、LSIA が B_n の節点 v の記述 $\mathrm{id}(v)$ に対して、v のラベルの記述 $\mathrm{id}(l(v))$、および v の 0 枝、1 枝の接続先の記述 $\mathrm{id}(e^0(n))$、$\mathrm{id}(e^1(n))$ を計算できることを意味する。従って、初期節点 s から始め、b_i の値に従って 0 枝または 1 枝をたどるという操作を n 回繰り返した後その節点のラベルを求めれば、$b_1 b_2 \cdots b_n$ が B_n で受理されるかどうかを判定できる。 □

3.2 Polybdd の性質

LSIA の性質から *PolyBDD* のいくつかの性質を導くことができる。

有限オートマトン、および対数領域限定決定性チューリング機械で受理できる言語のクラスをそれぞれ *REG*、*DLOGSPACE* とする。明らかに、$REG \subseteq LOGIREG \subseteq DLOGSPACE$ が成立する。*REG* と *LOGIREG* の差はワークテープの存在に、*LOGIREG* と *DLOGSPACE* の差は LSIA の入力テープが一方向であることに起因する。

パリティ関数や加算器のキャリー関数などの、有限の記憶で計算できる関数に対応する言語は *PolyBDD* に属している。LSIA は $O(\log n)$ の記憶を持っているので、これを用いてさらに複雑な言語を受理することができる。

(1) $\{0^n 1^n\}$ は *REG* には属さないが、*PolyBDD* に属する。
$O(\log n)$ のワークテープを用いることにより、LSIA は与えられた系列中の 0 と 1 の数を数え、比較することができる。

(2) 一様な組合せ回路で実現される対称関数に対応する言語は、全て *PolyBDD* に属する。
対称関数の値は入力の 1 の数に依存する。$O(\log n)$ のワークテープを用いてこれを数えれば、入力系列の受理 / 非受理が決定できる。上述の結果は *PolyBDD* が一様性を用いて定義されているためであり、一般に、全ての対称関数

は、サイズが n の多項式でおさえられる BDD で表現可能である。

(3) 一様な組合せ回路で実現されるしきい値関数に対応する言語は、各重みが n の多項式でおさえられていれば、*PolyBDD* に属する。

(4) $w \in \mathcal{B}^*$ を 2 進数とみたときの整数値を $\mathrm{int}(w)$、$\sigma \in \mathcal{B}^*$ 中の 1 の数を $\mathrm{weight}(\sigma)$、σ の k 番目のアルファベットを $\sigma[k]$ と現すことにする。すると、セレクタの関数に対応する言語 $\{w\sigma \mid |w| = \lceil \log|\sigma| \rceil, \sigma[|w| + \mathrm{int}\ (w) + 1] = 1\}$ は *PolyBDD* に属する。

これらの性質は、LSIA は $O(\log n)$ のワークテープを用いて $\mathrm{poly}(n)$ までの数を数えることができることに依っている。これにより、与えられた系列中の 1 の数の数え上げや、位置の同定を行えるからである。しかし、$O(\log n)$ のワークテープでは、系列そのものを記憶することはできない。これより、以下の性質が導かれる。

(5) $\{ww \mid w \in \mathcal{B}^*\}$ と $\{ww^R \mid w \in \mathcal{B}^*\}$ (但し、w^R は w の逆系列) は *PolyBDD* に属さない。入力系列を 1 回走査するだけで ww を受理するためには、系列の前半を記憶しなければならないが、これには $O(n)$ のワークテープが必要である。

(6) $x_i, y_i \in \mathcal{B}$ に対し、$(x_1x_2\cdots x_k)\circ(y_1y_2\cdots y_k) = x_1y_1x_2y_2\cdots x_ky_k$ とする。シフトを現す関数に対する言語 $\{sw \mid w = (x_1x_2\cdots x_k) \circ (0^{\mathrm{int}(s)}x_1x_2\cdots x_{k-\mathrm{int}(s)})\}$ は *PolyBDD* に属さない。

(4) のセレクタ関数に対応する言語は、σ と w の順序を入れ換えて $\{\sigma w \mid \cdots\}$ のように定義すると、*PolyBDD* に属さなくなる。これは、変数の順序が BDD のサイズに大きな影響を与える例でもある。

また、*PolyBDD* の集合演算に対する閉包性に関しては、次が成立する。これは、集合演算の結果に対応する LSIA を構成することにより証明できる。

(7) *PolyBDD* は補集合演算および有限回の集合積、集合和演算に関して閉じている。

順序回路の実現する論理関数が *PolyBDD* に属するかどうかに関しては、使用できるワークテープ量から、次がいえる。

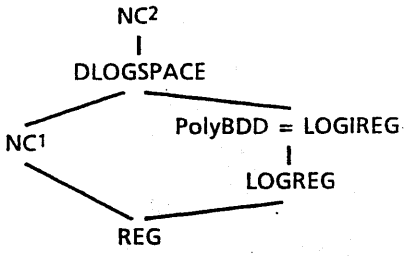

Figure 1: Relations among classes.

(8) 順序回路の出力が長さ n の入力系列に依存し、レジスタ (状態変数) の数が $O(\log n)$ でおさえられない場合には、組合せ回路部分の論理関数が *PolyBDD* に属していても、順序回路の実現する論理関数は *PolyBDD* に属さないことがある。

4 二分決定グラフと組合せ回路の関係

3 章では二分決定グラフとチューリング機械の関係について議論したが、すでに知られているチューリング機械と組合せ回路から、二分決定グラフと組合せ回路の関係を知ることができる。

Figure 1 は *PolyBDD* と種々の言語クラスの関係をまとめたものである。NC^k はサイズが多項式、段数が $O(\log^k n)$ 段でファンイン制限のある組合せ論理回路の族で受理できる言語のクラスである [Coo85]。*PolyBDD* は *DLOGSPACE* に含まれており、*DLOGSPACE* は NC^2 に含まれている。これより、ある論理関数が多項式サイズの BDD で表現できれば、それは $O(\log^2 n)$ 段の組合せ回路で実現できるということができる。

具体的には、BDD に対して以下の変換を行えば、目的の組合せ回路を得ることができる。

Lem 1 入力変数の数が n、節点数が $|V|$ の BDD に対し、同じ論理関数を実現する、段数 $O(\log n \log |V|)$、サイズ $O(n|V|^3)$ の組合せ回路が構成できる。

2 節で定式化したように、BDD が表現する論理関数は、BDD 上での到達可能性により定義される。ここでは、この到達可能性問題を解く組合せ回路を構成する。

与えられた BDD B に対して $|N| \times |N|$ 行列 $A_B = [a_{i,j}]$ を以下のように定義する。直観的には、与えら

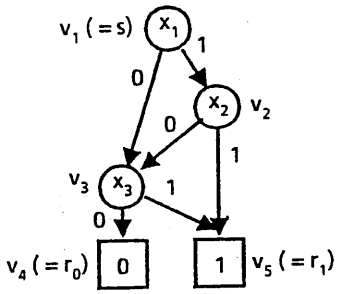

Figure 2: A BDD.

れた割当の下で、接点 v_i から v_j へ直接到達できる時 $a_{i,j}=1$ となる。

$a_{i,j}: \mathcal{B}^n \to \mathcal{B}$, where
$a_{i,j}=\overline{x_k}$ if $l(v_i)=x_k, e^0(v_i)=v_j$,
$a_{i,j}=x_k$ if $l(v_i)=x_k, e^1(v_i)=v_j$,
$a_{i,i}=1$ if $l(v_i)\in R$,
$a_{i,j}=0$ otherwise.

例えば、Figure 2 の BDD B に対する行列 A_B は、

$$A_B=\begin{bmatrix} 0 & x_1 & \overline{x_1} & 0 & 0 \\ 0 & 0 & \overline{x_2} & 0 & x_2 \\ 0 & 0 & 0 & \overline{x_3} & x_3 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

である。${A_B}^n$ (A_B の n 乗) の (i,j) 要素を ${a^n}_{i,j}$ と表記することにする。また、$v_s=s$ (初期節点)、$v_r=r_1$ (1 節点) とする。すると定義より、$f_B \equiv {a^n}_{s,r}$ であり、f_B の計算は、ブール行列の n 乗を求める問題に帰着できる。$|V|\times|V|$ ブール行列の乗算は段数 $O(\log|V|)$、サイズ $O(|V|^3)$ の組合せ回路で計算できるので、${A_B}^n$ は 段数 $O(\log n \log|V|)$、サイズ $O(n|V|^3)$ で計算できる。 □

$|V|=\mathrm{poly}(n)$ であれば、上記の段数、サイズはそれぞれ $O(\log^2 n)$、$O(\mathrm{poly}(n))$ となるので、次の定理が成り立つ。

Th 3 *PolyBDD* に属する言語は、段数 $O(\log^2 n)$、サイズ $O(\mathrm{poly}(n))$ の組合せ回路族により受理できる。 □

また、*PolyBDD* と NC^1 の関係については、NC^1 には属するが *PolyBDD* には属さない言語が存在することがわかっている。これは、n ビットの2進乗算の第 n ビットは、$O(\log n)$ の一様な組合せ回路族により計算できるが、多項式サイズの BDD 族では表現できない [Bry90] ことによる。従って、$PolyBDD \subset NC^1$、あるいは *PolyBDD* と NC^1 が比較不能であるという 2 通りが考えられるが、いずれであるかはまだ不明である。もし $PolyBDD \subset NC^1$ であれば、ある論理関数族が多項式サイズの BDD で表現できれば、$O(\log n)$ 段の多段組合せ回路に合成できることになり、実用的な意味が大きい。

5 むすび

多項式サイズの BDD で表現可能な論理関数のクラスを定義し、対数領域オートマトンとの関連の議論を通じて、このクラスの性質を調べた。また、BDD と組合せ回路の関係を調べ、多項式サイズの BDD から $O(\log^2 n)$ 段の組合せ回路を構成する方法を示した。

今後の課題としては、次があげられる。

(1) 変数の順序を考慮した BDD のサイズを議論すること。BDD のサイズは変数の順序によって大きく変わるが、本稿では、変数の順序も関数の定義に含まれており、一方向のチューリング機械を用いた議論では、変数の順序が考慮できない。変数の順序が考慮できる枠組を考えることが次の重要な課題である。

(2) *PolyBDD* と NC^1 の関係を明らかにすること。(但し、この解決は困難であることが予想される)。

(3) BDD から組合せ回路を合成する更に効率的なアルゴリズムを開発すること。4 章の構成法では、$O(\log n)$ 段で実現できる関数に対しても $O(\log^2 n)$ 段の組合せ回路が生成されてしまう。また、回路のサイズも多項式でおさえられているとはいえ、実用には大きすぎると考えられる。

謝辞

本研究にあたり、数々の方から御助言をいただきました。本学電子工学教室の安浦寛人助教授には、本研究全面にわたって御討論いただき、4 章の BDD と 組合せ回路の対応に関する御助言をいただきました。本学情報工学教室の岡部寿男氏には、組合せ回路族、対数領域オートマトンの性質に関して、廣瀬勝一氏には組合せ回路族とチューリング機

械の関係に関して御助言をいただきました。また、東道徹也氏には BDD の一様性と対数領域オートマトンの性質に関して御討論いただきました。このほか、矢島研究室の諸氏からも種々のコメントをいただきました。ここに、感謝の意を表します。

参考文献

[Ake78] S. B. Akers: "Binary Decision Diagrams", *IEEE Transactions on Compututers*, vol. C-27, no. 6, pp. 509–516 (Jun. 1978).

[Bry86] R. E. Bryant: "Graph-Based Algorithms for Boolean Function Manipulation", *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691 (Aug. 1986).

[Bry90] R. E. Bryant: "On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication", *private communication* (to appear in *IEEE Transactions on Computers*).

[Cho89] K. Cho and R. E. Bryant: "Test Pattern Generation for Sequential CMOS Circuits by Symbolic Fault Simulation", *Proc. ACM/IEEE 26th Design Automation Conference*, pp. 418–423 (Jun. 1989).

[Coo85] S. A. Cook: "Taxonomy of Problems with Fast Parallel Algorithms", *Information and Control* 64, pp. 2–22 (1985).

[Fuj88] M. Fujita, H. Fujisawa and N. Kawato: "Evaluations and Improvements of a Boolean Comparison Method Based on Binary Decision Diagrams", *Proc. IEEE International Conference on Computer-Aided Design (ICCAD-88)*, pp. 2–5 (Nov. 1988).

[Ish90] N. Ishiura, Y. Deguchi and S. Yajima: "Coded Time-Symbolic Simulation Using Shared Binary Decision Diagram", *Proc. ACM/IEEE 27th Design Automation Conference*, pp. 130–135 (Jun. 1990).

[Mat89] Y. Matsunaga and M. Fujita: "Multi Level Logic Optimization Using Binary Decision Diagrams", *Proc. IEEE International Conference on Computer-Aided Design (ICCAD-89)*, pp. 556–559 (Nov. 1989).

[Min89] S. Minato, N. Ishiura and S. Yajima: "Fast Tautology Checking Using Shared Binary Decision Diagram – Benchmark Results –", *Proc. IFIP International Workshop on Applied Formal Methods for Correct VLSI Design*, vol. 2, pp. 580–584 (Nov. 1989).

[Ruz81] Ruzzo: "On uniform circuit complexity", *JCSS* 22, pp. 365–383 (1981).

[Tak90] N. Takahashi, N. Ishiura and S. Yajima: "Fault Simulation for Multiple Faults Using Shared Binary Decision Diagrams", *Proc. Synthesis and Simulation Meeting and International Interchange*, pp. 157–164 (Oct. 1990).