7W-2

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

Nagisa ISHIURA, Tetsuya TOHDO and Shuzo YAJIMA

Kyoto University

1. Introduction

Binary Decision Diagram (BDD) [1] is an expression form of logic functions. Besides the existence of a unique canonical form for each logic function, many practical functions can be expressed by a BDD of feasible size. It is easily shown that the size of a BDD to express a logic function is in the worst case exponential to the number of input variables, but there have been very few discussions on the problem what kind of logic functions are expressed by a BDD of feasible size. In order to make this point clear, in this paper, we define a class of logic functions which is expressible by a BDD whose size is bounded by a polynomial of the number of input variables, and we investigate the properties of the logic functions of this class. Especially we focus on relations between BDD's and Turing machines.

2. Family of Binary Decision Diagrams

2.1 Binary Decision Diagram (BDD)

We define a binary decision diagram (BDD) as follows.

Def 2.1 A binary decision diagram B is a 7-tuple $B = (V, R, N, n0, l, e0, e1)$, where
V is an ordered set of variables,
R is a domain ($R = \{0, 1\}$ in this paper),
N is a set of nodes,
$n0 \in N$ is an initial node,
$l: N \rightarrow (V \cup R)$ represents a label of a node,
$e0, e1$: $N \rightarrow (N \cup \{nil\})$ represents a 0-edge and a 1-edge of a node, respectively, where
$\forall n \in N$ s.t. $l(n) \in R$ $e0(n) = e1(n) = nil$,
$\forall n \in N$ s.t. $l(n) \in V$ $l(e0(n)) \in R \vee l(e0(n)) < l(n)$
$\wedge l(e1(n)) \in R \vee l(e1(n)) < l(n)$. □

The size of a BDD B, denoted as $size(B)$, is defined as the number of nodes. Namely, $size(B) = |N|$, where $|N|$ is size of the set N.

Let A be a set of assignments for V, where an assignment a for V is a mapping a: $V \rightarrow \{0, 1\}$. We define a mapping t: $A \rightarrow ((N \cup R) \rightarrow (N \cup R))$ and f_B: $A \rightarrow R$ as follows.

$t_a(n)$ = if $l(n) \in R$ then n
　else if $a(l(n)) = 0$ then $e0(n)$ else $e1(n)$,
$f_B(a) = l(t_a{}^{|V|}(n0))$,
　where $t_a{}^1 = t_a$ and $t_a{}^m = t_a{}^{m-1} \circ t_a$.

Def 2.2 We define f_B as a logic function expressed by a BDD B. □

2.2 BDD Family and Its Uniformity

In order to discuss the size of BDD's with respect to the number of input variables, we define families of BDD's.

Def 2.3 A BDD family $\{B_i\}$ is a sequence of BDD's $B_1, B_2, B_3, \cdots$, where $|V_i| = i$ holds for every $B_i = (V_i, R_i, N_i, n0_i, l_i, e0_i, e1_i)$. □

Let $\{f_i \mid f_i: \{0, 1\}^i \rightarrow \{0, 1\}\}$ be a sequence of logic functions. We can consider that $\{f_i\}$ expresses a language L over $\{0, 1\}$ by the following correspondense:

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

Similarly we define a language for a BDD family.

Def 2.4 A language accepted by a BDD family $\{B_i\}$ is denoted as $L_{\{B_i\}}$ and defined as folows.

$b_1 b_2 \cdots b_n \in L_{\{B_i\}}$ iff $f_{B_n}(a) = 1$,

where a is an assignment for V and $a(v_i) = b_i$ ($v_i \in V$ and $b_i \in \{0, 1\}$). □

In this paper, we discuss correspondense between BDD families and Turing machines. For this purpose we define uniformity of a BDD family following after the uniformity of a combinational circuit family [2].

Def 2.5 A BDD family $\{B_i\}$ is uniform if the description of the n-th BDD B_n can be generated from a binary description of n by an $O(log\ size(B_n))$ space bounded off-line Turing machine. □

As a class of languages accepted by a feasible size of a BDD family, we define a class of PolyBDD.

Def 2.6 PolyBDD is a class of languages accepted by a uniform BDD family $\{B_i\}$, which satisfies $size(B_n) \leqq poly(n)$, where $poly(n)$ is an arbitrary polynomial of n. □

多項式サイズの二分決定図で表現可能な論理関数のクラス
石浦菜岐佐, 東道徹也, 矢島脩三
京都大学工学部情報工学教室

3. PolyBDD and Log-Space Automata (LSA)

3.1 Equivalent Class of PolyBDD

We will refer a one-way off-line Turing machine with $O(\log n)$ bounded working tape as a log space automata (LSA). We define an abstract machine referred to as a log space input-size-look-ahead automata (LSIA). LSIA is an LSA with the ability to know the length of an input sequence without scanning an input sequence. The length of an input sequence is given as an initial value on a working tape.

Def 3.1 LOGREG and LOGIREG are classes of languages which can be accepted by a LSA and a LSIA, respectively. □

The main result of this paper is that a class PolyBDD is equivalent to LOGIREG.

Th 1 PolyBDD = LOGIREG
[Proof Schetch]

(LOGIREG⊂PolyBDD): Since an LSIA has only *log n* memory, all the states of an LSIA can be represented by p nodes where $p \leqq poly(n)$. Then using $p \times n$ nodes we can construct a state transition diagram with no loop, which is the very n-th BDD. Since transiticns are computable by an $O(\log n)$ space bounded Turing machine, the BDD family is uniform.

(PolyBDD⊂LOGIREG): Using an $O(\log n)$ working tape, LSIA can simulate the $O(\log n)$ space bounded Turing machine to generate uniform BDD. So LSIA can compute the next node of a node for a given assignment. Since the number of nodes in a BDD is bounded by a polynomial of n, $O(\log n)$ space is enough to reach final node from initial node $n0$. □

3.2 Properties of PolyBDD

We can lead the following properties of PolyBDD directly from the properties of LSIA.

(1) $\{0^n 1^n\}$ belongs to PolyBDD.

(2) $\{ww \mid w \in \{0, 1\}^*\}$ does not belong to PolyBDD.

(3) Symmetric functions belong to PolyBDD.

(4) Threshould functions belong to PolyBDD if the magnitude of each weight is bounded by a polynomial of n.

(5) $\{w\sigma \mid |w| = \lceil \log|w\sigma| \rceil,\ int(w) = weight(\sigma)\}$ and $\{w\sigma \mid |w| = \lceil \log|w\sigma| \rceil,\ \sigma[|w| + int(w)] = 1\}$ belongs to PolyBDD (=LOGIREG) but not to LOGREG. Here, $int(w)$ is an integer value of a binary representation w, $weight(\sigma)$ is a number of 1's in the sequence σ, and $\sigma[k]$ is the k-th alphabet of σ. This property shows that LOGREG is truely included by PolyBDD.

Prop 3.1 LOGREG $\subsetneq$ PolyBDD. □

4. Relation between Other Classes

We also investigated the relation between PolyBDD and other classes of logic functions. Fig. 1 is a summary of the results. REG and LOGCFL are the classes of languages which can be accepted by a finite automata and a push-down automata with $O(\log n)$ bounded work tape, respectively. NC^k is a class of logic functions which can be expressed by a $(\log n)^k$-depth combinational circuit using gates with fan-in restriction [3].

As for the relation between PolyBDD and NC^1, there exist a logic fanction family which belongs to NC^1 but not to PolyBDD, but we have not yet obtained further results. Since PolyBDD is at least included by NC^2, we can synthsize a combinational circuit of $(\log n)^2$-depth from a polynomial size BDD.

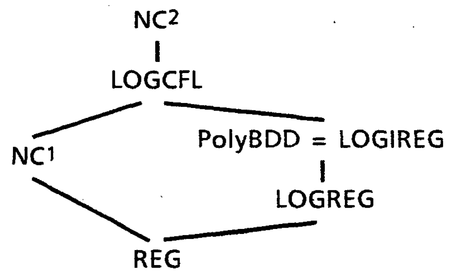

Fig. 1 Relations among classes.

5. Conclusion

We have defined a class of logic functions expressible by a polynomial-size BDD, and have investigated its property. To clarify the relation between polyBDD and NC^1 remains as a future work.

Acknowledgments

Authors would like to express their sincere appreciation to Prof. K. Iwama, Prof. H. Yasuura, Mr. Y. Okabe, Mr. S. Hirose and all the members of Yajima Lab. at Kyoto University for their discussions and valuable comments.

References

[1] R. E. Bryant: "Graph-based algorithms for Boolean function manipulation", *IEEE Trans. Comput.*, vol. C-35, no. 8, pp. 677~691 (1986).

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

[3] S. A. Cook: "Taxonomy of problems with fast parallel algorithms", *Information and Control* 64, pp. 2~22 (1985).