Discrete Applied Mathematics 126 (2003) 305–312

DISCRETE APPLIED MATHEMATICS

Note

Efficient dualization of O(log n)-term monotone disjunctive normal forms[☆]

Kazuhisa Makino

Division of Systems Science, Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka 560, Japan

Received 18 May 2000; received in revised form 19 February 2001; accepted 27 August 2001

Abstract

This paper shows that O(log n)-term monotone disjunctive normal forms (DNFs) φ can be dualized in incremental polynomial time, where n is the number of variables in φ. This improves upon the trivial result that k-term monotone DNFs can be dualized in polynomial time, where k is bounded by some constant.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Dualization; Monotone Boolean function; k-term DNF; Hypergraph; Transversal computation; Combinatorial enumeration; Polynomial total time algorithm

1. Introduction

A *Boolean function* is a mapping $f:\{0,1\}^n \to \{0,1\}$. This paper considers monotone (also called positive) Boolean functions—those Boolean functions f satisfying that $v \leqslant w$ (i.e., $v_i \leqslant w_i$ for all i) always implies $f(v) \leqslant f(w)$. Monotone functions have a unique prime disjunctive normal form (DNF) expression

$$\varphi = \bigvee_{I\in\mathscr{F}} \left(\bigwedge_{i\in I} x_i\right), \tag{1}$$

where $\mathscr{F}$ is *Sperner* (i.e., $I \not\subseteq J$ and $I \not\supseteq J$ holds for $I,J \in \mathscr{F}$ with $I \neq J$). It is well-known that $\mathscr{F}$ corresponds to the set of all prime implicants (i.e., minimal true vectors) of f.

[☆] The author gratefully acknowledges the partial support of a Scientific Grant in aid from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

E-mail address: makino@sys.es.osaka-u.ac.jp (K. Makino).

The dual of a Boolean function f is defined by

$$f^d(x) = \bar{f}(\bar{x}). \tag{2}$$

We can easily see that the dual of a monotone function is also monotone.

This paper is concerned with the following problem.

Problem Dualization

Input: The prime DNF φ of a monotone Boolean function f.
Output: The prime DNF ψ of f^d.

For example, consider the monotone function f given by $\varphi = \bigvee_{i=1}^{n} x_i y_i$. Then $\psi = \bigvee_{z_i \in \{x_i, y_i\},\ i=1,2,\ldots,n} \left(\bigwedge_{i=1}^{n} z_i\right)$ is the prime DNF of f^d; E.g., for $n = 2$, we have $\varphi = x_1y_1 \vee x_2y_2$ and $\psi = x_1x_2 \vee x_1y_2 \vee y_1x_2 \vee y_1y_2$. This example shows that the output size $|\psi|$ can be exponentially large in the input size $|\varphi|$. For this reason, the complexity of this type of problem is customarily measured in the input and output sizes. We say that a problem can be solved *in polynomial total time* if it has an algorithm which runs in time polynomial in the input size and output size [17]. In particular, we say that a problem can be solved *in incremental polynomial time* if it has an algorithm in which the time between consecutive outputs is bounded by a polynomial in the input size and the output size so far [1] [17]; E.g., for problem Dualization, if the kth term t_k can be output in polynomial in $|\varphi|$ and $\sum_{i=1}^{k-1} |t_i|$, where $\psi = \bigvee_{i=1}^{p} t_i$, then we say that it is solvable in incremental polynomial time. Clearly, incremental polynomiality implies polynomial totality.

As noted in [2,11], problem Dualization is polynomially equivalent to many other interesting problems encountered in various fields such as hypergraph theory [11], database theory [5,11,22], theory of coteries (used in distributed systems) [13,15], artificial intelligence [26] and learning theory [2,20]. Unfortunately, no polynomial total time algorithm for problem Dualization is known [2,11,17]. We only have incremental quasi-polynomial time algorithms [12,14,27]. This was first proposed by Fredman and Khachiyan [12], by giving an incremental $m^{o(\log m)}$ time algorithm, where m is the sum of the number of terms in φ and ψ. Recently, Tamaki [27] extended it to the algorithm that runs in $(N+M)^{O(\log N)}$ time and uses $O(N \log N)$ space, where N denotes the length of φ and M denotes the number of terms in ψ.

In this paper, we show that problem Dualization can be solved in incremental polynomial time if a given DNF φ contains $O(\log n)$ terms. This kind of setting is for example studied in computational learning theory (see e.g. [3]). Note that Dualization is clearly solvable in (input) polynomial time, if a given DNF φ contains the constant number of terms (See Section 2 for more details). Our result is a nontrivial generalization of the constant case. It computes the prime DNF of f^d by considering the *Horn minorant* of f (See the definition in Section 3.). Our result can also be seen as the

[1] The first (resp., last) output occurs in time polynomial in the input size (resp., in the input and output sizes) after the start (resp., before halt) of the algorithm.

generalization of the result in [9], where a polynomial time algorithm was presented to decide whether $f = f^d$ for a monotone function f given by an $O(\log n)$-term DNF. We remark here that the result in [12] does not lead to incremental polynomiality of problem Dualization, even if $|\mathscr{F}| = O(\log n)$.

Problem Dualization can be efficiently solved for many other subclasses of monotone functions. For example, if φ is a k-DNF (i.e., the size of each term in φ is limited by a constant k), then problem Dualization can be solved in incremental polynomial time (see e.g. [4,11]). In the quadratic case, i.e. when $k = 2$, even more efficient algorithms are known (see e.g. [17,19,28]). Efficient algorithms exist also for 2-monotonic, threshold, matroid, read-bounded, acyclic and so on (see e.g. [1,6,7,10,21,24,25,30]). Therefore, the result obtained in this paper enlarges solvable subclasses of monotone functions for problem Dualization.

We also mention that natural generalizations of problem Dualization, which arise in data-mining, machine learning and mathematical programming, are investigated in [5,29].

The rest of the paper is organized as follows. Section 2 recalls elementary concepts and provides basic notations. Section 3 gives an incremental polynomial time algorithm for Dualization when the input DNF φ contains $O(\log n)$ terms.

2. Definitions and basic properties

A *Boolean function*, or a *function* in short, is a mapping $f : \{0,1\}^n \to \{0,1\}$, where $v \in \{0,1\}^n$ is called a *Boolean vector* (a *vector* in short). If $f(v) = 1$ (resp., 0), then v is called a *true* (resp., *false*) vector of f. The set of all true vectors (resp., false vectors) is denoted by $T(f)$ (resp., $F(f)$). A function f is *monotone* (also called *positive*) if $v \leqslant w$ (i.e., $v_i \leqslant w_i$ for all i) always implies $f(v) \leqslant f(w)$, and *negative* if $v \leqslant w$ always implies $f(v) \geqslant f(w)$. In order to apply the results in [8,16,18] to problem Dualization, we consider negative functions instead of monotone functions. A true vector v of f is *maximal* if there is no other true vector w such that $w > v$ (i.e., $w \geqslant v$ and $w \neq v$), and let $\max T(f)$ denote the set of all maximal true vectors of f. A *minimal* false vector is symmetrically defined and $\min F(f)$ denotes the set of all minimal false vectors of f.

If functions f and h satisfy $h(v) \leqslant f(v)$ for all $v \in \{0,1\}^n$, then we denote $h \leqslant f$. If $h \leqslant f$ and there exists a vector v satisfying $h(v) = 0$ and $f(v) = 1$, we denote $h < f$. The variables $x_1, x_2, \ldots, x_n$ and their complements $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ are called *literals*. A *term* (resp., *clause*) is a conjunction (resp., disjunction) of literals such that at most one of x_i and $\bar{x}_i$ appears for each variable. A term t (resp., clause c) is called an *implicant* (resp., *implicate*) of a function f if $t \leqslant f$ (resp., $c \geqslant f$). An implicant t (resp., implicate c) of a function is called *prime* if there is no implicant $t' > t$ (resp., no implicate $c' < c$). As is well-known, if a function f is negative, then all prime implicants and implicates are negative (i.e., containing only negative literals $\bar{x}_i$), and there is one-to-one correspondence between prime implicants (resp., prime implicates) and maximal true vectors (resp., minimal false vectors).

A *disjunctive normal form* (DNF) (resp., conjunctive normal form (CNF)) is a disjunction of terms (resp., conjunction of clauses). A *negative* DNF (resp., CNF) is a DNF (resp., CNF) consisting of only negative literals. It is well-known that a function f is negative if and only if it has a negative DNF representation and that every negative function has a unique shortest DNF representation, called *prime* DNF, formed by the disjunction of all its prime implicants. Similar statement holds for CNF expressions, i.e., a function f is negative if and only if it has a negative CNF and every negative function has a unique shortest CNF representation, called *prime* CNF, formed by the conjunction of all its prime implicates. For example, a negative function f defined by $T(f) = \{(0011), (0001), (0010), (1000), (0000)\}$ can be represented by the prime DNF $\varphi = \bar{x}_1\bar{x}_2 \vee \bar{x}_2\bar{x}_3\bar{x}_4$ and the prime CNF $\psi = \bar{x}_2(\bar{x}_1 \vee \bar{x}_3)(\bar{x}_1 \vee \bar{x}_4)$, and it has prime implicants $\bar{x}_1\bar{x}_2$ and $\bar{x}_2\bar{x}_3\bar{x}_4$, which respectively correspond to maximal true vectors (0011) and (1000), and prime implicates $\bar{x}_2$, $(\bar{x}_1 \vee \bar{x}_3)$ and $(\bar{x}_1 \vee \bar{x}_4)$, which respectively correspond to minimal false vectors (0100), (1010) and (1001).

Recall that the *dual* of a function f, denoted f^d, is defined by

$$f^d(x) = \bar{f}(\bar{x}),$$

where $\bar{f}$ and $\bar{x}$ denote the complement of f and x, respectively. By definition, we have $(f^d)^d = f$. From De Morgan's law, a formula defining f^d is obtained from that of f by exchanging $\vee$ and $\wedge$ as well as the constants 0 and 1. For example, if f is given by $\varphi = x_1x_2 \vee \bar{x}_1(\bar{x}_3 \vee x_4)$, then f^d can be represented by $\psi = (x_1 \vee x_2)(\bar{x}_1 \vee \bar{x}_3x_4)$. From this observation, f^d can be represented by

$$f^d = \bigwedge_{I \in \mathscr{F}} \left(\bigvee_{i \in I} x_i \right), \tag{3}$$

if a monotone function f is given in (1), where the corresponding negative functions are easily obtained by replacing each x_i by $\bar{x}_i$. This immediately implies the following simple algorithm for problem Dualization.

Algorithm. Dualize

```
Step 1: Apply the distributive law to (3) to obtain a DNF representation $\rho$ of $f^d$.
/* Note that $\rho$ is monotone, but not prime in general.*/
Step 2: Remove redundant terms from $\rho$ to obtain the prime DNF $\psi$ of $f^d$.
```

Note that ρ consists of at most $n^{|\mathscr{F}|}$ terms and Step 1 can be executed in $O(n^{|\mathscr{F}|+1})$ time, where $\mathscr{F}$ is given in (1). Moreover, a simple implementation of Step 2 (Remove t from ρ if there exists t' in ρ such that $t \leqslant t'$ (i.e. $t(x) = 1$ implies $t'(x) = 1$)) requires $O(n^{2|\mathscr{F}|+1})$ time.[2] Therefore, if the number of terms in a given DNF φ is bounded by some constant k (i.e. $|\mathscr{F}| \leqslant k$, where $\mathscr{F}$ is given in (1)), then Dualization can be solved in (input) polynomial time. However, this argument cannot be extended to the nonconstant cases, e.g. $|\mathscr{F}| \leqslant \log\log n$, or $\log n$.

[2] More careful analysis shows that algorithm DUALIZE requires $O(|\varphi| n^{|\mathscr{F}|})$ time.

In this paper, we consider the following problem to solve Dualization.

Problem Negative DNF Conversion

Input: The prime DNF φ of a negative Boolean function f.
Output: The prime CNF ψ of f.

From $(f^d)^d = f$, we can identify the prime CNF of a negative function f with the prime DNF of f^d, and hence problem Negative DNF Conversion is linearly equivalent to Dualization. Therefore, the rest of the paper discusses problem Negative DNF Conversion and shows that it can be solved in incremental polynomial time if a given DNF φ contains $O(\log n)$ terms.

3. Problem Negative DNF Conversion

A CNF ρ is called *Horn* if each clause contains at most one positive literal; e.g., $\rho = (\bar{x}_1 \vee \bar{x}_2 \vee x_3)(\bar{x}_2 \vee \bar{x}_4)(x_5)$ is Horn, while $\rho = (\bar{x}_1 \vee x_2 \vee x_3)(\bar{x}_2 \vee \bar{x}_4)(x_5)$ is not. A function f is called *Horn* if it has a Horn CNF representation.

For a set of vectors $S\,(\subseteq \{0,1\}^n)$, let $Cl_\wedge(S)$ denote the *intersection closure* of S, i.e.,

$$Cl_\wedge(S) = \left\{ \bigwedge_{v \in S'} v \mid S' \subseteq S,\ S' \neq \emptyset \right\},$$

where $\bigwedge_{v\in S} v$ denotes the componentwise intersection of all vectors in S. For example, if $S = \{(0011), (0110), (1100)\}$, then $Cl_\wedge(S) = \{(0011), (0110), (1100), (0010), (0100), (0000)\}$. We point out the following well-known characterization of Horn functions.

Proposition 1 (McKinsey [23]; Dechter and Pearl [8]). *A function f is Horn if and only if $T(f) = Cl_\wedge(T(f))$ (i.e., $T(f)$ is closed under intersection).*

For a negative function f, the *Horn minorant* f_H of a negative function f is defined by

$$T(f_H) = Cl_\wedge(\max T(f)). \tag{4}$$

By Proposition 1, f_H is a Horn function, and the negativity of f implies $f_H \leqslant f$. Moreover, we have the following lemma.

Lemma 2. *Let f be a negative function. Then a negative clause c is a prime implicate of f if and only if it is a prime implicate of f_H.*

Proof. To show the only-if part, let us assume that a negative clause $c = \bigvee_{i\in I} \bar{x}_i$ is a prime implicate of f, i.e. (i) $c \geqslant f$ and (ii) $c_j \not\geqslant f$ for every $j \in I$, where $c_j = \bigvee_{i \in I\setminus\{j\}} \bar{x}_i$. Then (i) together with $f \geqslant f_H$ implies $c \geqslant f_H$, i.e., c is an implicate of f_H. We can see that (ii) is equivalent to the condition that for each $j \in I$, there exists a vector $v \in T(f)$ such that $c_j(v) = 0$. Since f and c are both negative, (ii) can be written as follows. For each $j \in I$, there exists a vector $v \in \max T(f)$ such that $c_j(v) = 0$. Now since $T(f_H) \supseteq \max T(f)$ by the definition of f_H, this implies that c is a prime implicate of f_H, which completes the only-if part.

On the other hand, let us assume that $c = \bigvee_{i \in I} \bar{x}_i$ is a prime implicate of f_H, i.e., (I) $c \geqslant f_H$ and (II) $c_j \not\geqslant f_H$ for every $j \in I$, where $c_j = \bigvee_{i \in I \setminus \{j\}} \bar{x}_i$. If c is not an implicate of f, then there exists a vector $v \in T(f)$ such that $c(v) = 0$. Since f and c are both negative, we can choose such a v from $\max T(f)$. Since $T(f_H) \supseteq \max T(f)$, this v is an evidence that c is not an implicate of f_H, a contradiction. Hence, c is an implicant of f. Moreover, similarly to the only-if part, we can see that (II) is equivalent to the condition that, for each $j \in I$, there exists a vector $v \in \max T(f_H)$ such that $c_j(v) = 0$. Since $T(f) \supseteq \max T(f_H)(=\max T(f))$, this implies that c is a prime implicate of f_H, showing the if part. □

Therefore, Lemma 2 immediately implies the following algorithm for problem Negative DNF Conversion.

Algorithm. Negative DNF Convert

Input: The prime DNF φ of a negative Boolean function f.

Output: The prime CNF ψ of f.

Step 1: Compute $T(f_H)$.

Step 2: Compute from $T(f_H)$ a Horn CNF ρ representing f_H.

Step 3: Compute the prime CNF ψ of f by generating from ρ all negative prime implicates of f_H.

Let

$$\varphi = \bigvee_{I \in \mathscr{F}} \left(\bigwedge_{i \in I} \bar{x}_i \right)$$

be an input prime DNF. For $I \subseteq \{1, 2, \ldots, n\}$, let v^I denote the vector in $\{0,1\}^n$ such that $v^I_j = 0$ if $j \in I$, and 1 otherwise. Then we have $\max T(f) = \{v^I \mid I \in \mathscr{F}\}$. It is not difficult to see that $T(f_H)$ can be computed from $\max T(f)$ in $O(n|T(f_H)|^2)$ time [18, Corollary 2]. Since $|T(f_H)| \leqslant 2^{|\mathscr{F}|}$, Step 1 requires $O(n \cdot 2^{2|\mathscr{F}|})$ time.

As for Step 2, the following result is known.

Lemma 3 (Dechter and Pearl [8, Lemmas A.2 and A.3]). *Given a set of vectors S, closed under intersection, we can compute a Horn CNF ρ describing S in* $O(n^2|S|^2)$ *time. Moreover, such a ρ contains at most $n^2|S|$ clauses.*

From Lemma 3, Step 2 can be executed in $O(n^2|T(f_H)|^2) = O(n^2 2^{2|\mathscr{F}|})$ time, and the obtained Horn CNF ρ contains at most $n^2|T(f_H)| = n^2 2^{|\mathscr{F}|}$ clauses.

Finally, by making use of input resolution, we can efficiently generate all negative prime implicates of a Horn function, if we have its Horn CNF.

Lemma 4 (Ibaraki et al. [16, Lemma 3.4]). *Let g be a Horn function. Given a Horn CNF ρ of g, we can generate all prime negative implicates of g in* $O(n^2|Clause(\rho)|^2 |NPI(g)|)$, *where $Clause(\eta)$ denotes the set of all clauses in a CNF η and $NPI(h)$ denotes the set of all negative prime implicates of a function h. Furthermore, this can be done in incremental polynomial time.*

Hence, since $Clause(\psi) = Neg(f_H)$ by Lemma 2, Step 3 can be executed in $O(n^2|Clause(\rho)|^2|Neg(f_H)|) = O(n^6 2^{2|\mathscr{F}|}|Clause(\psi)|)$.

In total, Algorithm Negative DNF Convert requires $O(n^6 2^{2|\mathscr{F}|}|Clause(\psi)|)$ time, and it is incremental polynomial if $|\mathscr{F}| = O(\log n)$.

Theorem 5. *Problem* Negative DNF Conversion *can be solved in incremental polynomial time if a given DNF φ contains* $O(\log n)$ *terms.*

From the discussion in Section 2, we have the following corollary.

Corollary 6. *Problem* Dualization *can be solved in incremental polynomial time if a given DNF φ contains* $O(\log n)$ *terms.*

Remark 7. The results in [2,11] imply that several problems are solvable in incremental polynomial time if a monotone function f can be represented by an $O(\log n)$-term DNF. For example, the identification problem (i.e. given a membership oracle of f, compute all minimal true vectors and maximal false vectors of f) is solvable in incremental polynomial time.

Acknowledgements

The author thanks the anonymous referees for their helpful and constructive comments which improved the presentation of this paper.

References

[1] P. Bertolazzi, A. Sassano, An $O(mn)$ time algorithm for regular set-covering problems, Theoret. Comput. Sci. 54 (1987) 237–247.

[2] J.C. Bioch, T. Ibaraki, Complexity of identification and dualization of positive Boolean functions, Inform. Comput. 123 (1995) 50–63.

[3] A. Blum, S. Rudich, Fast learning of k-term DNF formulas with queries, in: Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 1992, pp. 382–389.

[4] E. Boros, V. Gurvich, P.L. Hammer, Dual subimplicants of positive Boolean functions, Optim. Methods and Software 10 (1998) 147–156.

[5] E. Boros, V. Gurvich, L. Khachiyan, K. Makino, Dual-bounded generating problems: partial and multiple transversals of a hypergraph SIAM J. Comput. 30 (2001) 2036–2050.

[6] E. Boros, P.L. Hammer, T. Ibaraki, K. Kawakami, Polynomial time recognition of 2-monotonic positive Boolean functions given by an oracle, SIAM J. Comput. 26 (1997) 93–109.

[7] Y. Crama, Dualization of regular Boolean functions, Discrete Appl. Math. 16 (1987) 79–85.

[8] R. Dechter, J. Pearl, Structure identification in relational data, Artif. Intell. 5 (1992) 237–270.

[9] C. Domingo, Polynomial time algorithms for some self-duality problems, Proceedings of the Italian Conference of Algorithms, Rome (Italy), Springer Lecture Notes in Computer Science, 1203, (1997) 171–180.

[10] C. Domingo, N. Mishra, L. Pitt, Efficient read-restricted monotone CNF/DNF dualization by learning with membership queries, Mach. Learning 37 (1999) 89–110.

[11] T. Eiter, G. Gottlob, Identifying the minimal transversals of a hypergraph and related problems, SIAM J. Comput. 24 (1995) 1278–1304.

[12] M.L. Fredman, L. Khachiyan, On the complexity of dualization of monotone disjunctive normal forms, J. Algorithms 21 (1996) 618–628.

[13] H. Garcia-Molina, D. Barbara, How to assign votes in a distributed system, J. ACM 32 (1985) 841–860.

[14] V. Gurvich, L. Khachiyan, On generating the irredundant conjunctive and disjunctive normal forms of monotone Boolean functions, Discrete Appl. Math. 96–97 (1999) 363–373.

[15] T. Ibaraki, T. Kameda, A theory of coteries: Mutual exclusion in distributed systems, IEEE Trans. Parallel Distribut. Systems 4 (1993) 779–794.

[16] T. Ibaraki, A. Kogan, K. Makino, Inferring Functional dependencies in Horn and g-Horn theories, Rutcor Research Report, RRR 35-2000, Rutgers University, 2000, to appear in Annals of Mathematics and Artificial Intelligence.

[17] D.S. Johnson, M. Yannakakis, C.H. Papadimitriou, On generating all maximal independent sets, Inform. Process. Lett. 27 (1988) 119–123.

[18] D. Kavvadias, C.H. Papadimitriou, M. Sideri, On Horn envelopes and hypergraph transversals, in: K.W. Ng, et al., (Eds.), ISAAC'93 Algorithms and Computation, Springers Lecture Notes in Computer Science, Vol. 762, Springer, Berlin, 1993, pp. 399–405.

[19] E. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Generating all maximal independent sets: NP-hardness and polynomial-time algorithms SIAM J. Comput. 9 (1980) 558–565.

[20] K. Makino, T. Ibaraki, The maximum latency and identification of positive Boolean functions, SIAM J. Comput. 26 (1997) 1363–1383.

[21] K. Makino, T. Ibaraki, A fast and simple algorithm for identifying 2-monotonic positive Boolean functions, J. Algorithms 26 (1998) 291–305.

[22] H. Mannila, K.J. Räihä, Design by example: an application of Armstrong relations J. Comput. System Sci. 22 (1986) 126–141.

[23] J.C.C. McKinsey, The decision problem for some classes of sentences without quantifiers, J. Symbolic Logic 8 (1943) 61–76.

[24] U.N. Peled, B. Simeone, Polynomial-time algorithm for regular set-covering and threshold synthesis, Discrete Appl. Math. 12 (1985) 57–69.

[25] U.N. Peled, B. Simeone, An $O(nm)$-time algorithm for computing the dual of a regular Boolean function, Discrete Appl. Math. 49 (1994) 309–323.

[26] R. Reiter, A theory of Diagnosis from first principles, Artif. Intell. 32 (1987) 57–95.

[27] H. Tamaki, Space-efficient enumeration of minimal transversals of a hypergraph, IPSJ-AL 75 (2000) 29–36.

[28] S. Tsukiyama, M. Ide, H. Ariyoshi, I. Shirakawa, A new algorithm for generating all maximal independent sets, SIAM J. Comput. 6 (1977) 505–517.

[29] E. Boros, K. Elbassioni, V. Gurvich, L. Khachiyan, K. Makino, On generating all minimal integer solutions for a monotone system of linear inequalities, ICALP2001, edited by F. Orejas et al., Crete, Greece, Springer Lecture Notes in Computer Science 2076 (2001) pp. 92–103.

[30] D. Gaur and R. Krishnamurti, Self-duality of bounded monotone Boolean functions and related problems, ALT2000, Sydney, Australia, eds. H. Arimura et al., Springer Lecture Notes in Computer Science 1968 (2000) pp. 209–223.