IEICE TRANS. INF. & SYST., VOL.E91–D, NO.4 APRIL 2008

PAPER

Tree-Shellability of Restricted DNFs

Yasuhiko TAKENAGA[†a)], *Member and* Nao KATOUGI[†*], *Nonmember*

SUMMARY A tree-shellable function is a positive Boolean function which can be represented by a binary decision tree whose number of paths from the root to a leaf labeled 1 equals the number of prime implicants. In this paper, we consider the tree-shellability of DNFs with restrictions. We show that, for read-k DNFs, the number of terms in a tree-shellable function is at most k^2. We also show that, for k-DNFs, recognition of ordered tree-shellable functions is NP-complete for $k = 4$ and tree-shellable functions can be recognized in polynomial time for constant k.

key words: *Boolean function, shellability, prime implicant, binary decision tree*

1. Introduction

Prime implicant is a very important concept in the theory of Boolean functions. An irredundant disjunctive normal form (DNF) Boolean formula is given as a sum of prime implicants. In particular, for a positive Boolean function, its irredundant DNF representation is unique and given as the sum of all prime implicants. A tree-shellable function [10] is a positive Boolean function defined by the relation between its prime implicants and binary decision tree (BDT) representations. A Boolean function is called tree-shellable if there exists a BDT representation such that the number of prime implicants equals the number of paths from the root to a leaf labeled 1 in the BDT. An ordered tree-shellable function is a special case of a tree-shellable function such that the BDT must be an ordered BDT.

A tree-shellable function is a kind of shellable function. Shellable Boolean functions play an important role in many fields. The notion of shellability was originally used in the theory of simplicial complexes and polytopes (for example, in [5], [6]). More recently, it is studied for its importance to reliability theory (for example, in [1], [2], [9]). The notion of tree-shellability makes it possible to give another characterization of some subclasses of shellable functions [10].

If a Boolean function is shellable and the order of terms to make it shellable is given, the following problem on computing the reliability of some kind of systems can be solved easily.

[Union of Product Problem] ([2])
Input: $Pr[x_i = 1](1 \leq i \leq n)$, $f(x_1, \ldots, x_n)$
Output: $Pr[f(x_1, \ldots, x_n) = 1]$

$Pr[A]$ represents the probability of event A. Each variable represents the state of a subsystem. A subsystem is operative if the variable has value 1. If f is shellable, one can easily compute the exact value of $Pr[f = 1]$ using the orthogonal DNF representation of f.

In addition, if the BDT representation of a Boolean function f is given, it is easy to compute the BDT representation of its dual f^d. The dual of a Boolean function $f(x_1, \ldots, x_n)$ is defined by $f^d = \overline{f(\overline{x_1}, \ldots, \overline{x_n})}$. From the BDT representation of f, the BDT representation of f^d can be obtained by simply exchanging a 1-edge and a 0-edge for every variable node and exchanging label 1 and label 0 for every leaf node. In spite of the importance of dualization, it is not yet known if the DNF representation of the dual f^d can be computed from the DNF representation of f in time polynomial to the input and output size. Therefore, the problem still interests many researches (for example, in [3], [7], [8]).

To utilize the good properties of tree-shellable functions, it is important to clarify the class of Boolean functions for which tree-shellability can be tested in polynomial time.

In this paper, we consider the properties and the recognition problem of tree-shellable and ordered tree-shellable functions when their DNF representations have some restrictions. As restricted DNFs, we consider read-k DNFs and k-DNFs. A read-k DNF is a DNF in which each variable appears at most k times. A k-DNF is a DNF all of whose terms consist of at most k literals.

First, we consider read-k DNFs. In the Union of Product problem, the restriction corresponds to the case when the influence of each subsystem is limited. We show that, in this case, a tree-shellable function has at most k^2 prime implicants with two or more literals, which is a tight upper bound. It follows that tree-shellable functions can be recognized in polynomial time when k is a constant.

Next, we consider k-DNFs. It is shown in [4] that it is NP-complete to decide if a function has lexico-exchange property (equivalent to ordered tree-shellability) for $k \geq 5$. On the other hand, for quadratic functions, that is when $k = 2$, ordered tree-shellable and tree-shellable functions are equivalent and their recognition can be executed in polynomial time [1], [10]. A quadratic function is tree-shellable iff its graph representation is a cotriangulated graph. In this paper, we improve the proof of [4] and prove that it is NP-complete for $k = 4$. We also show that tree-shellable func-

Manuscript received May 31, 2007.
Manuscript revised October 24, 2007.

[†]The authors are with the Department of Computer Science, the University of Electro-Communications, Chofu-shi, 182–8585 Japan.
[*]Presently, the author is with NEC Corporation.
a) E-mail: takenaga@cs.uec.ac.jp

DOI: 10.1093/ietisy/e91–d.4.996

tions can be recognized in polynomial time when k is a constant.

This paper is organized as follows. In Sect. 2, we give basic definitions on a Boolean function and a binary decision tree. In Sect. 3, we define a tree-shellable function and show its basic properties. In Sect. 4, we consider tree-shellability of read-k DNFs. In Sect. 5, we consider tree-shellability of k-DNFs. Conclusions and future works are noted in Sect. 6.

2. Preliminaries

2.1 Boolean Function

Let $f(x_1, \ldots, x_n)$ be a Boolean function. We denote $f \geq g$ if $f(x) = 1$ for any assignment $x \in \{0,1\}^n$ which makes $g(x) = 1$. An *implicant* of f is a product term $\bigwedge_{i \in I} x_i \bigwedge_{j \in J} \overline{x_j}$ which satisfies $\bigwedge_{i \in I} x_i \bigwedge_{j \in J} \overline{x_j} \leq f$, where $I, J \subseteq \{1, 2, \ldots, n\}$. An implicant which satisfies $\bigwedge_{i \in I - \{s\}} x_i \bigwedge_{j \in J} \overline{x_j} \not\leq f$ for any $s \in I$ and $\bigwedge_{i \in I} x_i \bigwedge_{j \in J - \{t\}} \overline{x_j} \not\leq f$ for any $t \in J$ is called a *prime implicant* of f.

An expression of the form $f = \bigvee_{k=1}^{m} \left(\bigwedge_{i \in I_k} x_i \bigwedge_{j \in J_k} \overline{x_j} \right)$ is called a *disjunctive normal form Boolean formula* (DNF), where $I_k, J_k \subseteq \{1, 2, \ldots, n\}$ and $I_k \cap J_k = \emptyset$ for $k = 1, \ldots, m$. A *positive* DNF (PDNF) is a DNF such that $J_k = \emptyset$ for all k. If f can be represented as a PDNF, it is called a positive Boolean function. For simplicity, we call that I_k is an implicant or a term of a positive function. A PDNF is called *irredundant* if $I_k \subseteq I_l$ is not satisfied for any k, l $(1 \leq k, l \leq m, k \neq l)$. For an irredundant PDNF, let $PI(f)$ be the set of all I_k. $PI(f)$ represents the prime implicants of f. In the following of this paper, we consider only positive functions and we assume that a function is given as an irredundant PDNF $f = \bigvee_{k=1}^{m} \bigwedge_{i \in I_k} x_i$.

In the following of this paper, we also use sum (+) and product ($\cdot$) instead of $\vee$ and $\wedge$.

2.2 Binary Decision Tree

A *Binary Decision Tree* (BDT) is a labeled tree that represents a Boolean function. A leaf node of a BDT is labeled by 0 or 1. Any other node is labeled by a variable and called a variable node. Let $label(v)$ be the label of node v. Each variable node has two outgoing edges, which are called a *0-edge* and a *1-edge*. Let $edge_0(v), edge_1(v)$ denote the nodes pointed to by the 0-edge and the 1-edge of node v respectively. The value of the function is obtained by traversing from the root node to a leaf node. At a variable node, one of the outgoing edges is chosen according to the value of the variable. The value of the function is 0 if the label of the leaf is 0, and 1 if the label is 1. Let the right (left resp.) subtree of node v be the BDT whose root is $edge_1(v)$ ($edge_0(v)$ resp.).

A path from the root node to a leaf node labeled 1 is called a *1-path*. On every 1-path, each variable appears at most once. A 1-path P of a BDT is represented by a sequence of literals. For simplicity, we denote $\tilde{x}_i \in P$ when $\tilde{x}_i$ is included in the sequence representing P, where $\tilde{x}_i$ is either x_i or $\overline{x_i}$. Let the *street* of a BDT T be the sequence of nodes that can be reached from the root of T by using only 0-edges (including the root itself).

When the 0-edge and the 1-edge of node v point to the nodes representing the same function, v is called to be a *redundant node*. A BDT which has no redundant node is called a *reduced BDT*. In the following of this paper, a BDT means a reduced BDT.

A BDT is called an *ordered* BDT (OBDT) if variables appear in the order consistent with a total order of variables on any path from the root to a leaf. The total order of variables for an OBDT is called the *variable ordering*. A variable ordering of variables $x_1, x_2, \ldots, x_n$ is represented by a permutation π on $\{1, 2, \ldots, n\}$.

3. Tree-Shellable Boolean Functions

Definition 1: A positive Boolean function f is *tree-shellable* when it can be represented by a BDT with exactly $|PI(f)|$ 1-paths.

If a BDT T represents $f = \bigvee_{k=1}^{m} \bigwedge_{i \in I_k} x_i$ and has exactly m 1-paths, we say that T witnesses that f is tree-shellable.

Definition 2: A positive Boolean function f is *ordered tree-shellable with respect to* π if it can be represented by an OBDT with variable ordering π which has exactly $|PI(f)|$ 1-paths. f is *ordered tree-shellable* if there exists π such that f is ordered tree-shellable with respect to π.

Proposition 1: [10] Let $f = \bigvee_{k=1}^{m} \bigwedge_{i \in I_k} x_i$ be tree-shellable. Then a BDT that witnesses that f is tree-shellable satisfies the following.

· Each 1-path P_k corresponds to a term I_k by the rule that $i \in I_k$ iff $x_i \in P_k$.

Figure 1 is an example of a tree-shellable function. Note that the leaf nodes labeled 0 and the edges that point to them are omitted in the figures of this paper. f of Fig. 1 is also ordered tree-shellable with respect to variable ordering $x_1 x_2 x_3 x_4$.

The following corollaries show the relation between a tree-shellable function f and the BDT which witnesses that f is tree-shellable.

Corollary 1: [10] Let a BDT T witness that f is tree-shellable and 1-path P_k correspond to term I_k for any k $(1 \leq k \leq |PI(f)|)$. Then, for any 1-path P_k of T and any $\overline{x_s} \in P_k$, there exists l which satisfies $I_l \subseteq I_k \cup \{s\}$ $(l \neq k)$.

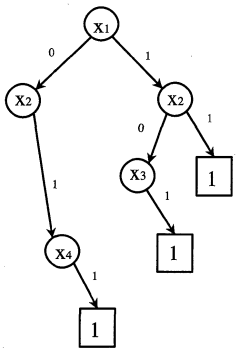


Fig. 1 An example of a tree-shellable function.

Corollary 2: [10] Let T be a BDT such that T has $m = |PI(f)|$ 1-paths $P_1, P_2, \ldots, P_m$ and $i \in I_k$ iff $x_i \in P_k$ for any $k(1 \leq k \leq m)$. Then T witnesses that f is tree-shellable if for any 1-path P_k of T and any $\overline{x_s} \in P_k$, there exists l such that P_k and P_l diverge at a node labeled x_s and $I_l \subsetneq I_k \cup \{s\}$.

When Corollary 1 holds, we say that P_l (I_l resp.) compensates P_k (I_k resp.) for x_s.

4. Tree-Shellability of Read-k DNFs

In this section, we consider the tree-shellability of Boolean functions represented by read-k DNFs. From the next lemma, we assume in this section that the given DNF consists of terms with at least two literals.

Lemma 1: Let f be represented by a DNF which has terms with only one literal. Let g be the function obtained from f by removing terms with only one literal. Then, f is (ordered) tree-shellable iff g is (ordered) tree-shellable.

This lemma is obvious because the variable used in a term with one literal does not appear in the other terms.

Theorem 1: A Boolean function represented by a read-k DNF with more than k^2 terms is not tree-shellable.

Proof Let f be a tree-shellable Boolean function and a BDT whose root is labeled by x_r witness that f is tree-shellable. Consider a term I_t including x_r. Obviously, the literals in $I_t \backslash \{r\}$ can appear at most $(k-1)$-times in the terms not including x_r. It means that each term including x_r can compensate at most $k-1$ terms not including x_r. As there exist at most k terms including x_r, at most $k(k-1) = k^2 - k$ terms not including x_r can be compensated by them. Therefore, f has at most $k + (k^2 - k) = k^2$ terms. □

Corollary 3: Let f be a tree-shellable function represented by a read-k DNF with m terms. Then the root of the tree which witnesses that f is tree-shellable is labeled by a variable that appears at least m/k times in the DNF.

Proof Assume that x_s appears q ($< m/k$) times in the DNF. If the root of a BDT is labeled by x_s, as in the proof of Theorem 1, the number of terms of f must be at most $q + q(k-1) = qk$. As $q < m/k$, $qk < m$. It is a contradiction. □

Though the maximum number of terms in a tree-shellable function is k^2, the number of nodes in the street of a BDT which witnesses the tree-shellability is much smaller. In the following, we consider the maximum number of nodes in the street.

Theorem 2: Let f be a tree-shellable function represented by a read-k DNF. Then the tree which witnesses that f is tree-shellable has at most $2k-1$ nodes in its street.

Proof We classify the nodes in the street by whether the following condition is satisfied or not.
Condition: There exists a 1-path in its right subtree which is compensated by a 1-path in the right subtree of the root that includes the positive literal of the variable of the node in the street.
For simplicity, we call the nodes in the street that satisfy the condition to be the nodes of type A, and the nodes that do not satisfy the condition to be the nodes of type B. Also, we call the right subtrees of the nodes of type A (type B resp.) to be the subtrees of type A (type B resp.). Note that the root is classified as a node of type B.

First, we show that there exist at most k nodes of type A. Assume that there exist p nodes of type A and their labels are $x_{s_1}, x_{s_2}, \ldots, x_{s_p}$ in order of appearance in the street. Let the right subtrees of the nodes be $S_1, S_2, \ldots, S_p$. For each $i(1 \leq i \leq p)$, there exists a 1-path P_i in the right subtree of the root that includes x_{s_i} and compensates a 1-path in S_i for the variable of the root. In addition, for any $q(2 \leq q \leq p)$, P_q is different from P_j for all $j(1 \leq j \leq q-1)$. It is because if $x_{s_j} \in P_q$, any 1-path compensated by P_q also has to include x_{s_j}. That is, the 1-paths are in S_j, not in S_q. Thus, $P_i(1 \leq i \leq p)$ are all different. Therefore, there exist at least p 1-paths in the right subtree of the root. It means that there exist at least p terms that include the label of the root. As each variable can appear in at most k terms, $p \leq k$ holds.

Next, we show that there exist at most k nodes of type B. Assume that there exist $k+1$ nodes of type B and their labels are $x_{t_1}, x_{t_2}, \ldots, x_{t_{k+1}}$ in order of appearance in the street. Let the right subtrees of the nodes be $T_1, T_2, \ldots, T_{k+1}$. Note that x_{t_1} is the label of the root. Let I_{k+1} be a term that corresponds to a 1-path in T_{k+1}. There exists a 1-path I_k in T_k that compensates I_{k+1} for x_{t_k}. Similarly, we can inductively define $I_j(1 \leq j \leq k)$ as the 1-path in T_j that compensates I_{j+1} for x_{t_j}. That is, $I_j \subsetneq I_{j+1} \cup \{t_j\}$ holds for each $j(1 \leq j \leq k)$. Therefore, $I_1 \subsetneq I_{k+1} \cup \{t_1, \ldots, t_k\}$ holds.

However, actually, $I_1 \subseteq \{t_1, \ldots t_k\}$ holds. To prove this, assume that $p \in I_1$ for some $p \in I_{k+1}$. Then $p \in I_2$ because $I_1 \subsetneq I_2 \cup \{t_1\}$ and $p \neq t_1$. Similarly, if $p \in I_j$, $p \in I_{j+1}$ holds for all j. Thus, x_p is included in I_j for all $j(1 \leq j \leq k+1)$. That is, there must be $k+1$ terms that include x_p. It contradicts the fact that f is represented by a read-k DNF. Therefore, $I_1 \subseteq \{t_1, \ldots t_k\}$.

Let r be the smallest integer satisfying $t_r \in I_1 \backslash \{t_1\}$. Then, $I_1 \subseteq \{t_1, t_r, \ldots, t_{k+1}\}$. As $I_1 \subsetneq I_r \cup \{t_1, \ldots, t_{r-1}\}$ from the definition and $t_2, \ldots, t_{r-1} \notin I_1$, $I_1 \subsetneq I_r \cup \{t_1\}$ holds. It means that I_r is compensated by I_1. As $t_r \in I_1$, it contradicts

the assumption that the node labeled x_{t_r} is type B. Therefore, there can be at most k nodes of type B.

From the above discussions, there can be at most $2k$ nodes in the street. In addition, we can show that if there exist k nodes of type B, there exist at most $k-1$ nodes of type A. If there exist k nodes of type B, as seen from the above proof, there exists a variable which appears in all the subtrees of type B. As the variable appears in k terms corresponding to the 1-paths in the subtrees, it cannot appear in any subtree of type A. Therefore, the 1-path in the right subtree of the root that includes the variable cannot compensate any 1-path in the subtrees of type A. As there exist at most $k-1$ 1-paths in the right subtree of the root that can compensate a 1-path in subtrees of type A, there can be at most $k-1$ nodes of type A. Thus, the total number of nodes in the street is at most $2k-1$. □

The next theorem gives more detailed structure of the tree that witnesses the tree-shellability of a function f whose DNF representation has k^2 terms.

Theorem 3: Let f be a tree-shellable function represented by a read-k DNF with k^2 terms. Let T be the BDT that witnesses that f is tree-shellable. Then, in any right subtree of the node in the street of T, no pair of 1-paths include the same positive literal.

Proof We consider the right subtree of the root. The similar discussion holds for the other right subtrees. As f has k^2 terms, there exist k 1-paths in the right subtree T_r of the root. Assume that there exist m (> 1) terms in T_r which include both the label of the root and x_a. By each 1-path in T_r not including x_a, at most $k-1$ 1-paths can be compensated. By 1-paths in T_r including x_a, at most $k-m$ 1-paths can be compensated due to the restriction on the number of appearances of x_a. In total, T can have at most $k + (k-1)(k-m) + (k-m) = k^2 + (1-m)k$ 1-paths. As f has k^2 terms, m must be 1. □

Theorem 3 means that only one term whose 1-path ends in T_r can have more than two variables. It is because if there exist two terms with more than two variables, they must include two same variables.

We have shown the upper bound on the number of terms and the length of the street. Next, we show that the upper bounds are tight.

Theorem 4: There exist tree-shellable functions s.t.
1) the function is represented by a read-k DNF,
2) the number of terms is k^2, and
3) a tree which witnesses the tree-shellability has $2k-1$ nodes in the street.

Proof We show that $f_k(x_1, \ldots, x_{2k-2}, x_{2k-1}, \ldots, x_n) = (x_1 + x_2 + \cdots + x_{k-1} + A)(x_k + x_{k+1} + \cdots + x_{2k-2} + B)$ satisfies the conditions of the theorem. Here, A and B are products of positive literals not including $x_1, x_2, \cdots, x_{2k-2}$ s.t. there exists no common variable in A and B, and x_{2k-1} is included in either A or B. Obviously, f_k has k^2 terms. Construct a tree representing f by using $x_1, x_2, \ldots, x_{2k-1}$ in the street in this order. Then we can obtain the tree with $2k-1$ nodes in the street which witnesses that f_k is tree-shellable. The tree for $k = 3$ is shown in Fig. 2. □

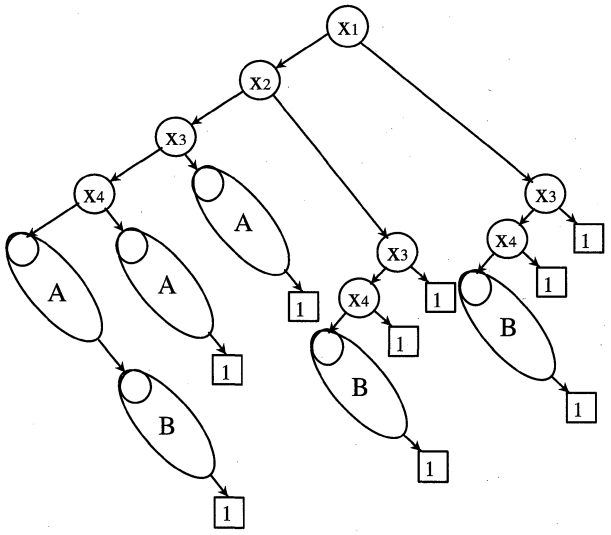


Fig. 2 The BDT representing f_k for $k = 3$.

For read-k DNFs, tree-shellable functions with $2k-1$ nodes in the street have very restricted forms, though above f_k is not the only one example. However, there seems to be more variations if the number of nodes in the street can be smaller.

The above results are also important when we consider the complexity of recognizing tree-shellable functions. As we have seen, the tree that witnesses the tree-shellability has at most $2k-1$ nodes in the street and each right subtree of the nodes has at most k terms. That is, any 1-path of the tree has at most $3k-2$ nodes at which two 1-paths diverge. Therefore, when k is a constant, exhaustive search of all such trees can be executed in polynomial time.

Corollary 4: For Boolean functions represented by read-k DNFs for a constant k, recognition of tree-shellability can be executed in polynomial time.

5. Tree-Shellability of k-DNFs

In this section, we consider the tree-shellability of Boolean functions represented by k-DNFs and clarify the complexity of recognizing tree-shellable and ordered tree-shellable functions from the DNF representation.

5.1 NP-Completeness of Recognizing Ordered Tree-Shellable Functions

NP-completeness of testing lexico-exchange property, which is equivalent to ordered tree-shellability, is proved in [4]. Though the length of terms is not considered in [4], the proof shows the NP-completeness for $k = 5$. In this paper, by improving the proof, we show that the NP-completeness result holds for $k = 4$.

Theorem 5: Given a k-DNF representing f, it is NP-complete to check if f is ordered tree-shellable for $k \geq 4$.

In the remaining of Sect. 5.1, we give the proof of this theorem. The problem is in NP because when we guess a variable ordering, it is easy to check if the given function is ordered tree-shellable with respect to the variable ordering.

To prove NP-hardness, we give the reduction from 3 HITTING SET problem.

3 HITTING SET

Input : Subsets $S_1, S_2, \ldots, S_m$ of $X = \{1, 2, \ldots, n\}$ which satisfy $|S_i| = 3$ for all i $(1 \leq i \leq m)$.

Question : Does there exist $A \subset X$ which satisfies $|A \cap S_i| = 1$ for all i $(1 \leq i \leq m)$?

Using our terms, Lemma 5.3 of [4] can be described as follows.

Lemma 2: If f is ordered tree-shellable with respect to π, then $f|_{x_i=0}$ is also ordered tree-shellable with respect to π.

Lemma 3: Let $f(a, b, c, y) = abc + ya + yb + yc$. Then f is not ordered tree-shellable with respect to any variable ordering $\pi = \pi_1\pi_2\pi_3\pi_4$ satisfying $\pi_3 = y$ or $\pi_4 = y$. On the other hand, f is ordered tree-shellable with respect to any variable ordering satisfying $\pi_2 = y$.

Proof As a, b and c are symmetric in f, we have only to consider the cases of $\pi = abyc, abcy$ and $aybc$. We can easily see that the lemma holds by constructing the trees representing f with the variable orderings. □

Lemma 4: Let $g(a, b, c, y) = ya+ab+ac+bc$. Then g is not ordered tree-shellable with respect to any variable ordering $\pi = \pi_1\pi_2\pi_3\pi_4$ satisfying $\pi_1 = y$. On the other hand, g is ordered tree-shellable with respect to any variable ordering satisfying $\pi_2 = y$.

Proof As b and c are symmetric in g, we have only to consider the cases of $\pi = ybac, ybca, aybc, byac$ and $byca$. We can easily see that the lemma holds by constructing the trees representing g with the variable orderings. □

Lemma 5: [4] Let $f_1, f_2, \ldots, f_r$ be ordered tree-shellable with respect to the same variable ordering $\pi = x_1 \cdots x_n$. Then $F(t_1, \ldots, t_r, x_1, \ldots, x_n) = \bigvee_{1 \leq i < j \leq r} t_i t_j \vee \bigvee_{i=1}^{r} t_i f_i(x_1, \ldots, x_n)$ is ordered tree-shellable with respect to variable ordering $t_1 t_2 \cdots t_r\ x_1 \cdots x_n$.

Now we give the reduction from 3 HITTING SET by using above lemmas. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ be the given instance of 3 HITTING SET. $n + 1 + 2m$ variables $x_i (i = 1, 2, \ldots, n), y$ and $t_j (j = 1, 2, \ldots, 2m)$ are used in the reduction. Define $2m$ DNFs ψ_{k1}, ψ_{k2} as follows: for all $S_i = \{i_1, i_2, i_3\} (i_1 < i_2 < i_3)$, $\psi_{i1}(i_1, i_2, i_3, y) = f(x_{i_1}, x_{i_2}, x_{i_3}, y)$ and $\psi_{i2}(i_1, i_2, i_3, y) = g(x_{i_1}, x_{i_2}, x_{i_3}, y)$. f and g are as defined in Lemma 3 and Lemma 4. Let $F(t_1, \ldots, t_{2m}, x_1, \ldots, x_n, y) = \bigvee_{1 \leq i < j \leq 2m} t_i t_j \vee \bigvee_{1 \leq i \leq m} (t_{2i-1}\psi_{i1} \vee t_{2i}\psi_{i2})$. Each term of F includes at most 4 literals.

It remains to show that F is ordered tree-shellable iff the instance of 3 HITTING SET problem has a solution A. First, we show that the 3 HITTING SET problem has a solution if F is ordered tree-shellable. If F is ordered tree-shellable with respect to π, $A = \{i \mid x_i$ appears before y in $\pi\}$ is the solution of the 3 HITTING SET problem. To prove this, consider $S_i = \{i_1, i_2, i_3\}$. Let π^i be the variable ordering of $\{x_{i_1}, x_{i_2}, x_{i_3}, y\}$ in π. As $t_{2i-1}\psi_{i1}$ or $t_{2i}\psi_{i2}$ is obtained by assigning 0 to all but one variables t_j, they are ordered tree-shellable with respect to π from Lemma 2. Thus ψ_{i1} and ψ_{i2} are ordered tree-shellable with respect to π^i. From Lemma 3 and 4, $\pi^i_2 = y$ because otherwise either ψ_{i1} or ψ_{i2} is not ordered tree-shellable with respect to π^i. It means that, for all $S_i = \{i_1, i_2, i_3\}$, y appears second among $\{x_{i_1}, x_{i_2}, x_{i_3}, y\}$. Therefore, A defined above satisfies $|A \cap S_i| = 1$.

Next, we show that F is ordered tree-shellable if the 3 HITTING SET problem has a solution. W.l.o.g. let $A = \{1, 2, \ldots, l\}$ be the solution and let $B = \{l + 1, \ldots, n\}$. Let $\pi = t_1, t_2, \ldots, t_{2m}, x_1, x_2, \ldots, x_l, y, x_{l+1}, \ldots, x_n$. Then, for all $S_i = \{i_1, i_2, i_3\}$, $\pi^i_2 = y$ in π^i. It is because π is defined so that the variables corresponding to the elements of A appear before y and all the other variables appear after y. As A is the solution of the 3 HITTING SET problem, $|A \cap S_i| = 1$. That is, $\pi^i_2 = y$ holds. From Lemma 3 and 4, ψ_{i1} and ψ_{i2} are ordered tree-shellable with respect to π. Hence, from Lemma 5, F is ordered tree-shellable with respect to π.

5.2 Complexity of Recognizing Tree-Shellable Functions

In this section, we show an algorithm for recognizing tree-shellable functions and evaluate its time complexity for k-DNFs. In the algorithm, when we check the tree-shellability by searching all the possible trees, it is not necessary to use backtracking on the variables in the street.

To begin with, we give some definitions used in this section. For a Boolean function f and a variable x_i of f, let $right(f, x_i)$ and $left(f, x_i)$ be the functions that satisfy $PI(right(f, x_i)) = \{I \backslash \{i\} \mid I \in PI(f), i \in I\}$ and $PI(left(f, x_i)) = \{I \mid I \in PI(f), i \notin I\}$. Note that $left(f, x_i) = f|_{x_i=0}$ holds.

Lemma 6: If f is tree-shellable, then $f|_{x_i=0}$ is also tree-shellable for any i.

Proof As f is tree-shellable, there exists a BDT T that witnesses that f is tree-shellable. By assigning 0 to x_i in the DNF representation of f, the terms that include x_i disappear. The BDT obtained by assigning 0 to x_i in T represents $f|_{x_i=0}$. From Proposition 1, the number of disappeared 1-paths equals the number of terms that include x_i. Therefore, the obtained BDT witnesses that $f|_{x_i=0}$ is tree-shellable. □

Lemma 7: A Boolean function f is tree-shellable and a BDT whose root is labeled by x_i witnesses that f is tree-shellable iff the following conditions hold.

(1) $left(f, x_i)$ is tree-shellable.

(2) $right(f, x_i)$ is tree-shellable.

(3) For any $I \in PI(left(f, x_i))$, there exists $I' \in PI(right(f, x_i))$ that satisfies $I' \subseteq I$.

Proof (if) Consider BDTs whose root is labeled by x_i.

From condition 1, there exists a BDT whose left subtree of the root has $|PI(left(f, x_i))|$ 1-paths. When condition 3 holds,

$$\begin{aligned} f &= left(f, x_i) + x_i right(f, x_i) \\ &= \overline{x_i} left(f, x_i) + x_i left(f, x_i) + x_i right(f, x_i) \\ &= \overline{x_i} left(f, x_i) + x_i right(f, x_i). \end{aligned}$$

It means that $right(f, x_i) = f|_{x_i=1}$. Therefore, from condition 2, there exists a BDT whose right subtree of the root has $|PI(right(f, x_i))|$ 1-paths. As $|PI(left(f, x_i))| + |PI(right(f, x_i))| = |PI(f)|$, there exists a BDT which witnesses that f is tree-shellable.

(only if) BDT T that witnesses that f is tree-shellable satisfies the property described in Corollary 1. We show that the conditions of this lemma are satisfied if the property of Corollary 1 is satisfied. Condition 1 holds from Lemma 6. As Corollary 1 holds when $x_s = x_i$, condition 3 holds. As $f = \overline{x_i}\ left(f, x_i) + x_i\ right(f, x_i)$ holds and the right subtree of the root has $|PI(right(f, x_i))|$ 1-paths, condition 2 holds. □

From Lemma 6 and Lemma 7, the next theorem is immediate.

Theorem 6: Let f be tree-shellable. Then there exists a BDT whose root is labeled by x_i that witnesses that f is tree-shellable iff the following conditions are satisfied.
(1) $right(f, x_i)$ is tree-shellable.
(2) For any $I \in PI(left(f, x_i))$, there exists $I' \in PI(right(f, x_i))$ that satisfies $I' \subsetneq I$.

This theorem means that if f is tree-shellable, we can construct a BDT using recursion only to the right subtrees. On the other hand, if f is not tree-shellable, there exists no variable which satisfies the conditions of Theorem 6 or, if it exists, condition (1) of Lemma 7 is not satisfied. From Theorem 6, we can test if a Boolean function $f(x_1, \ldots x_n)$ is tree-shellable or not by the following algorithm.

CheckTS(f)

1. $K = PI(f)$, $R = \emptyset$.
2. If $|K| = 1$, return 'YES'.
3. For $i = 1$ to n, repeat Step4 and 5.
4. If $i \notin R$, check if the following a) and b) are satisfied.
   a) For any $I \in PI(left(f, x_i))$, there exists $I' \in PI(right(f, x_i))$ that satisfies $I' \subsetneq I$.
   b) $CheckTS(right(f, x_i)) = YES$.
5. If the conditions are satisfied, $R = R \cup \{i\}$, remove the terms including i from K and go to Step2.
6. If the conditions are not satisfied for any i, return 'NO'.

When f is not tree-shellable, CheckTS(f) clearly returns 'NO'. When f is tree-shellable, from Theorem 6, an arbitrary variable which satisfies the conditions can be chosen as the label of the root. Thus, the tree that witnesses tree-shellability can be constructed without backtracking on the street.

We evaluate the complexity of recognizing tree-shellable functions for k-DNFs. Note that, when f is represented by a k-DNF, $right(f, x_i)$ is represented by a $(k-1)$-DNF.

Consider the number of times conditions of Step4 are tested. In Step4, it takes $O(km^2)$ time to check a) for one variable, where m is the number of terms. For a node in the street, the conditions are tested for at most n variables, and there are at most n nodes in the street. Thus, step4 a) takes $O(kn^2m^2)$ time in total. In addition, step4 is executed in each recursive call of $CheckTS$. As Step4 b) calls $CheckTS$, $CheckTS(right(f, x_i))$ is called at most n^2 times. For each call, conditions of Step4 are tested at most n^2 times. Thus, in the first level of recursive calls, step4 a) takes $O(kn^4m^2)$ time in total. Similarly, we can see that in the t-th level of recursive call, step4 a) takes $O(kn^{2t+2}m^2)$ time in total. As tree-shellability is checked in $O(n^2)$ time for quadratic functions, $(k-2)$-nd level of recursive calls takes $O(n^{2k})$ time. $(k-3)$-rd level of recursive calls takes $O(kn^{2k-4}m^2)$ time. Either of them dominates the computation time of $CheckTS$. Therefore, when k is a constant, the following corollary holds.

Corollary 5: For Boolean functions represented by k-DNFs for constant k, recognition of tree-shellable functions can be executed in polynomial time.

6. Conclusion

In this paper, we considered the (ordered) tree-shellability of Boolean functions represented by restricted DNFs.

First, we considered the Boolean functions represented by read-k DNFs. We showed some properties of tree-shellable functions with the restriction. In particular, we showed that a tree-shellable function has at most k^2 prime implicants with two or more literals.

Next, we considered the Boolean functions represented by k-DNFs. We showed that the recognition of ordered tree-shellable functions is NP-complete for $k = 4$ and, on the other hand, tree-shellable functions can be recognized in polynomial time for constant k. On ordered tree-shellable functions, it remains open to clarify the complexity of the recognition problem for $k = 3$. On tree-shellable functions, the complexity of recognition is not clarified even for general positive functions.

It is also our future work to clarify the number of tree-shellable and ordered tree-shellable functions, especially under the restrictions we considered in this paper.

Acknowledgements

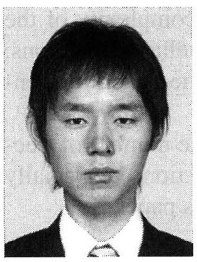
This research was partially supported by the Scientific Grant-in-Aid from Ministry of Education, Science, Sports and Culture of Japan.

References

[1] M.O. Ball and G.L. Nemhauser, "Matroids and a reliability analysis problem," Math. Oper. Res., vol.4, pp.132–143, 1979.
[2] M.O. Ball and J.S. Provan, "Disjoint products and efficient computation of reliability," Oper. Res., vol.36, no.5, pp.703–715, 1988.
[3] J.C. Bioch and T. Ibaraki, "Complexity of identification and dualization of positive Boolean functions," Inf. Comput., vol.123, no.1, pp.50–63, 1995.

[4] E. Boros, Y. Crama, O. Ekin, P.L. Hammer, T. Ibaraki, and A. Kogan, "Boolean normal forms, shellability and reliability computations," SIAM J. Discrete Math., vol.13, no.2, pp.212–226, 2000.
[5] H. Brugesser and P. Mani, "Shellable decompositions of cells and spheres," Math. Scand., vol.29, pp.199–205, 1971.
[6] G. Danaraj and V. Klee, "Shellings of spheres and polytopes," Duke Math. J., vol.41, pp.443–451, 1974.
[7] C. Domingo, N. Mishra, and L. Pitt, "Efficient read-restricted monotone CNF/DNF dualization by learning with membership queries," Mach. Learn., vol.37, pp.89–110, 1999.
[8] M.L. Fredman and L. Khachiyan, "On the complexity of dualization of monotone disjunctive normal forms," J. Algorithms, vol.21, no.3, pp.618–628, 1996.
[9] J.S. Provan and M.O. Ball, "Efficient recognition of matroids and 2-monotonic systems," in Applications of Discrete Mathematics, eds. R. Ringeisen and F. Roberts, pp.122–134, SIAM, Philadelphia, 1988.
[10] Y. Takenaga, K. Nakajima, and S. Yajima, "Tree-shellability of Boolean functions," Theor. Comput. Sci., vol.262, no.2, pp.633–647, 2001.

Yasuhiko Takenaga received the B.E., M.E. and Ph.D degrees in information science from Kyoto University, Kyoto, Japan, in 1989, 1991 and 1995, respectively. From 1991 to 1997, he was an instructor at the Department of Information Science, Graduate School of Engineering, Kyoto University. He joined the Department of Computer Science, the University of Electro-Communications, Tokyo, Japan in 1997 as an assistant professor, where he is currently an associate professor. His research interest includes graph algorithms and graph representations of Boolean functions.



Nao Katougi received the B.E. and M.E. degrees from Department of Information Science, the University of Electro-Communications, Tokyo, Japan, in 2004 and 2006, respectively. He is currently with NEC Corporation 1st Financial Systems Division.