数理解析研究所講究録
第381巻 1980年 42-51

A FAST ALGORITHM FOR COMPUTING THE MAXIMUM NUMBER OF PRIME IMPLICANTS OF SYMMETRIC BOOLEAN FUNCTIONS

Yoshihide Igarashi,
Department of Computer Science,
Gunma University, Kiryu, 376 Japan

1. Introduction

Prime implicants play an important role in the simplification problem of Boolean formulae or the minimization problem of switching circuits. In order to implement an algorithm for the minimization of Boolean formulae on a computer, it is necessary or desirable to evaluate the maximum number of prime implicants for a given number of variables. This value is typically used to estimate the size of arrays to store some intermediate results when implementing such an algorithm [7]. We therefore need a fast algorithm for computing the maximum number of prime implicants from such a practical viewpoint.

Dunham and Fridshal [3] have given $n!/(\lfloor n/3\rfloor!\,\lfloor(n+1)/3\rfloor!\,\lfloor(n+2)/3\rfloor!)$ as a lower bound, and Chandra and Markowsky [2] have given $\binom{n}{\lfloor(2n+1)/3\rfloor}2^{\lfloor(2n+1)/3\rfloor}$ as an upper bound on the maximum number of prime implicants of n-variable Boolean function. This lower bound is the number of prime implicants of a formula that is the disjunction of all the cubes consisting of $\lfloor(n+2)/3\rfloor$ nonnegated variables and $\lfloor(n+1)/3\rfloor$ negated variables. The upper bound of Chandra and Markowsky has been derived as a corollary of a result on the maximal sized antichains in partial orders given by Kleitman et al [8]. Igarashi has obtained a marginal improvement on the previous lower bound [5,6]. It is given as a recursive procedure, $n!/(\lfloor n/3\rfloor!\,\lfloor(n+1)/3\rfloor!\,\lfloor(n+2)/3\rfloor!) + \hat{g}(n,\lfloor(n+1)/3\rfloor-2) + \hat{g}(n,\lfloor(n+2)/3\rfloor-2)$, where $\hat{g}(n,r)$

is defined as follows: $\hat{g}(n,r) = 0$ for $r < 0$ and $\hat{g}(n,r) = n!/(\lfloor r/2 \rfloor ! (n-r)! \lfloor (r+1)/2 \rfloor !) + \hat{g}(n, \lfloor (r+1)/2 \rfloor - 2)$ for $0 \leq r$.

It is conjectured that for each $n \geq 1$ the maximum number of prime implicants of n-variable Boolean function might be equal to the maximum number of prime implicants of n-variable symmetric Boolean functions [5,6]. It is also conjectured that for each $n \geq 1$ the new lower bound might be optimal. We have not found any Boolean function which has more the number of prime implicants than the new lower bound [5,6].

Under these circumstances it is interesting to find a fast algorithm for computing the maximum number of prime implicants of symmetric Boolean functions. It seems that no published literature explicitly describes a polynomial time algorithm for computing the maximum number of prime implicants of symmetric Boolean functions. We introduce integer representation in n+1 bits to uniquely express each n-variable symmetric Boolean function. It is closely related to the set of prime implicants of the symmetric Boolean function. From this way of representation we devise a fast algorithm for computing the maximum number of prime implicants of symmetric Boolean functions in which a dynamic programming technique is used. The total logarithmic computing time cost and the total uniform computing time cost of the algorithm are $O(n^4)$ and $O(n^3)$, respectively.

2. Preliminary

In the main we employ definitions and notations used in standard texts on switching theory such as [4,9]. When discussing an n-variable Boolean formula or Boolean function in this paper, we assume that the set of its variables is $\{x_1, \ldots, x_n\}$. For notational convenience the i-th variables x_i $(1 \leq i \leq n)$ is occasionally denoted by x_i^1. The complement of x_i is denoted by $\bar{x}_i$ or x_i^0, and is occasionally called the negation of x_i.

$\vee$ and $\wedge$ denote disjunction and conjunction, respectively. Conventionally $\wedge$ may be omitted if there is no confusion. For example, $(x_1 \wedge \bar{x}_2) \vee x_3$ may be written $x_1\bar{x}_2 \vee x_3$. For convenience we often identify a Boolean formula with the Boolean function expressed as the formula.

$x_{i_1}^{e_1} \ldots x_{i_{n-r}}^{e_{n-r}}$ is called an r-cube or r-implicant over n variables, where $i_s \neq i_t$ if $s \neq t$, and for each j $(1 \leq j \leq n-r)$ $1 \leq i_j \leq n$ and

$$x_{i_j}^{e_j} = \begin{cases} \bar{x}_{i_j} & \text{if } e_j = 0 \\ x_{i_j} & \text{if } e_j = 1. \end{cases}$$

$\lfloor d \rfloor$ is the floor of d (i.e., the greatest integer k such that $k \leq d$), and $\lceil d \rceil$ is the ceiling of d (i.e., the least integer k such that $k \geq d$). A function S(n) is said to be O(T(n)) if there exists a constant c such that $S(n) \leq cT(n)$ for all but some finite (possibly empty) set of nonnegative integers for n.

The universal upper bound and universal lower bound of the Boolean algebra are denoted by I and $\emptyset$, respectively. These are Boolean constants. Relations $\lessdot$ and $\leqslant$ on n-variable Boolean formulae are defined as follows: $\emptyset \lessdot I$. For a pair of a and b in $\{\emptyset, I\}$, $a \leqslant b$ means $a \lessdot b$ or $a = b$. For a pair of n-variable Boolean formulae P and Q, $P \leqslant Q$ means that for all $(a_1, \ldots, a_n) \in \{\emptyset, I\}^n$ $P(a_1, \ldots, a_n) \leqslant Q(a_1, \ldots, a_n)$, where $S(a_1, \ldots, a_n)$ is the evaluation of Boolean formula S when for each i $(1 \leq i \leq n)$ x_i is set to be a_i. $P \lessdot Q$ means that $P \leqslant Q$ and for at least one $(a_1, \ldots, a_n) \in \{\emptyset, I\}^n$ $P(a_1, \ldots, a_n) \lessdot Q(a_1, \ldots, a_n)$.

A cube P is said to be a prime implicant of an n-variable Boolean function E if and only if $P \leqslant E$ and there does not exists a cube Q such that $P \lessdot Q \leqslant E$. NPI(E) denotes the number of prime implicants of E. g(n) is defined as $\max\{NPI(E) \mid E \text{ is an n-variable Boolean function}\}$. θ_n is a function from n-variable Boolean formulae to sets of n-tuples of 0's and 1's defined as $\theta_n(E) = \{(i_1, \ldots, i_n) \mid x_1^{i_1} \ldots x_n^{i_n} \leqslant E\}$. w_n is

Hamming weight function. That is, for each $(a_1, \ldots, a_n) \in \{0, 1\}^n$ $w_n(a_1, \ldots, a_n)$ is the number of 1's of $(a_1, \ldots, a_n)$, and for each $A \subseteq \{0, 1\}^n$ $w_n(A) = \{w_n(a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in A\}$.

An n-variable Boolean formula E is said to be (n,m,r)-regular if and only if E is the disjunction of all (n-m)-cubes consisting of r nonnegated variables and m-r negated variables. When n is understood, an (n,m,r)-regular formula is said to be (m,r)-regular for short. If E is (m,r)-regular for some (m,r), then E is said to be regular. If E is a disjunction of regular formulae, then E is said to be semi-regular. A Boolean function is said to be symmetric if and only if it can be expressed as a semi-regular formula. In general, a semi-regular formula for a symmetric Boolean function is not unique. $\Gamma(n)$ is defined as $\max\{NPI(E) \mid E$ is an n-variable symmetric Boolean function$\}$. It is an open problem whether $g(n)$ is equal to $\Gamma(n)$ for all n. It is conjectured that $g(n)$ might be equal to $\Gamma(n)$ for all n [5,6].

In order to estimate the efficiency of algorithms we shall consider two cost criteria of computing time of a random access machine. One is the logarithmic cost criterion,and the other is the uniform cost criterion. The definitions of these criteria can be found in [1].

3. A Canonical Representation of a Symmetric Boolean Function

A binary representation of a symmetric Boolean function is introduced. The representation is closely related to the set of prime implicants of the Boolean function that is expressed as it.

Suppose that E is an n-variable symmetric Boolean function. If for an $(a_1, \ldots, a_n) \in \{0, 1\}^n$ $x_1^{a_1} \ldots x_n^{a_n} \leq E$, then for any $(c_1, \ldots, c_n) \in \{0, 1\}^n$ such that $w_n(a_1, \ldots, a_n) = w_n(c_1, \ldots, a_n)$ $x_1^{c_1} \ldots x_n^{c_n} \leq E$. Therefore, there is a one-one correspondence ζ_n between the

set of binary sequences of length n+1 and the set of n-variable symmetric Boolean functions as follows: Let $b_0 \ldots b_n$ be a binary sequence of length n+1. Then $\zeta_n(b_0 \ldots b_n)$ is an n-variable symmetric Boolean function E such that $x_1^{a_1} \ldots x_n^{a_n} \leq E$ if and only if $b_{w_n(a_1, \ldots, a_n)} = 1$.

A block, denoted by b(i,j), of a binary sequence is defined as follows: b(i,j) is a block of a binary sequence $b_0 \ldots b_n$ if and only if for each k ($i \leq k \leq j$) $b_k = 1$, $b_{i-1} = 0$ when $i \neq 0$, and $b_{j+1} = 0$ when $j \neq n$. BLOCK($b_0 \ldots b_n$) is defined as $\{b(i,j) \mid b(i,j)$ is a block of $b_0 \ldots b_n\}$. $\xi_{b(n)}$ is a one-one correspondence between the set of blocks of binary sequences of length n+1 and the set of regular fomulae over n variables defined as $\xi_{b(n)}(b(i,j)) = (n,n-j+i,i)$-regular formula. $\xi_{B(n)}$ is a one-one mapping from the set of binary sequences of length n+1 to the set of semi-regular formulae over n variables defined as $\xi_{B(n)}(b_0 \ldots b_n) = \bigvee_{b(i,j) \in \mathrm{BLOCK}(b_0 \ldots b_n)} (n,n-j+i,i)$-regular formula. From the definitions of $\xi_{b(n)}$ and $\xi_{B(n)}$ for each symmetric Boolean function E over n variables $\xi_{B(n)}(\zeta_n^{-1}(E))$ is uniquely defined. This representation is called the canonical semi-regular formula of E.

Theorem 1. Let E be the canonical semi-regular formula of an n-variable symmetric Boolean function. Then P is a prime implicant of E if and only if P appears in E as its cube.

Proof. Let P be a cube that appears in E. If P = I, then E = I and the theorem holds obviously. We suppose $P \neq I$. Let P be a conjunction of t nonnegated variables and s negated variables. Then b(t,n-s) is in BLOCK($\zeta_n^{-1}(E)$).

Case 1: t = 0 or s = 0.

We first assume t = 0. Since we suppose $P \neq I$, $s \neq 0$. Suppose that P is not a prime implicant of E. Then there exists an (n-s+1)-cube Q such that $P \lneq Q \leq E$. Since t = 0, Q is a conjunction of s-1 negated variables. Therefore, there exists an $(a_1, \ldots, a_n) \in \{0, 1\}^n$ such that

$x_1^{a_1} \ldots x_n^{a_n} \leq E$ and $w_n(a_1, \ldots, a_n) = n-s+1$. Then the $(n-s+1)$-th component of $\zeta_n^{-1}(E)$ should be 1. This is contrary to the fact that $b(t,n-s)$ is in $BLOCK(\zeta_n^{-1}(E))$.

The proof in the case where $s = 0$ is analogous to the above proof.

Case 2: $t \neq 0$ and $s \neq 0$.

Suppose that P is not a prime implicant of E. Then there exists an $(n-t-s+1)$-cube Q such that $P \lessdot Q \leq E$. Q is a conjunction of t non-negated variables and s-1 negated variables, or a conjunction of t-1 non-negated variables and s negated variables. Therefore, there exists an $(a_1, \ldots, a_n) \in \{0, 1\}^n$ such that $x_1^{a_1} \ldots x_n^{a_n} \leq E$ and $w_n(a_1, \ldots, a_n)$ $= t-1$ or $n-s+1$. This is contrary to the fact that $b(t,n-s)$ is in $BLOCK(\zeta_n^{-1}(E))$. □

From the above theorem the canonical semi-regular formula E is the disjunction of all the prime implicants of E. The corresponding binary sequence (i.e., $\zeta_n^{-1}(E)$) is a compact representation of the symmetric Boolean function.

The next lemma is immediate from the definition of a regular formula and Theorem 1.

Lemma 1. Let E be an (n,m,r)-regular formula. Then $NPI(E) = \binom{n}{m}\binom{m}{r}$.

Theorem 2. Let E be an n-variable symmetric Boolean function. Then

$$NPI(E) = \sum_{b(i,j) \in BLOCK(\zeta_n^{-1}(E))} \left(\binom{n}{j-i}\binom{n-j+i}{i} \right).$$

Proof. $\xi_{b(n)}(b(i,j))$ is an $(n,n-j+i,i)$-regular formula. From Lemma 1 $NPI(\xi_{b(n)}(b(i,j))) = \binom{n}{j-i}\binom{n-j+i}{i}$. Therefore, from Theorem 1

$$NPI(E) = \sum_{b(i,j) \in BLOCK(\zeta_n^{-1}(E))} \left(\binom{n}{j-i}\binom{n-j+i}{i} \right).$$ □

Theorem 3. $\Gamma(n) = \max\{ \sum_{b(i,j) \in BLOCK(k_{(n+1)})} \left(\binom{n}{j-i}\binom{n-j+i}{i} \right) \mid 0 \leq k \leq 2^{n+1}-1\} = \max\{ \sum_{b(i,j) \in BLOCK(k_{(n+1)})} \left(\binom{n}{j-i}\binom{n-j+i}{i} \right) \mid 0 \leq k \leq 2^n-1\}$, where $k_{(n+1)}$ is the binary expansion of k in n+1 bits.

Proof. The first equality is from Theorem 1 and Theorem 2, and the second equality is from the following fact: Let $E^{(c)}$ be the conjugate of E (i.e., $E^{(c)}$ is obtained from E by replacing x_i by $\bar{x}_i$ and $\bar{x}_i$ by x_i for all i ($1 \leq i \leq n$). Then $NPI(E) = NPI(E^{(c)})$. □

4. A Fast Algorithm for Computing $\Gamma(n)$

A polynomial time algorithm for computing $\Gamma(n)$ is devised. The algorithm is based on Theorem 2, and a tabular method called dynamic programming is used in it. We define $r_n(i,j)$ as $\max\{NPI(E) \mid E$ is a semi-regular formula over n variables such that $w_n(\theta_n(E)) \subseteq \{i, i+1, \ldots, j\}\}$. For technical convenience we define $r_n(i,j) = 0$ when i is greater than j. Then for each k ($0 \leq k \leq n$) $r_n(k,k) = \binom{n}{k}$, and $\Gamma(n) = r_n(0,n)$.

Lemma 2. For each $0 \leq i \leq j \leq n$ $r_n(i,j) = r_n(n-j,n-i)$.

Proof. Let E be an n-variable Boolean formula, and let $w_n(\theta_n(E)) \subseteq \{i, i+1, \ldots, j\}$. Then $w_n(\theta_n(E^{(c)})) \subseteq \{n-j, \ldots, n-i\}$. Since $NPI(E) = NPI(E^{(c)})$, the lemma therefore holds. □

Theorem 4. For each $0 \leq i \leq j \leq n$ $r_n(i,j) = \max\{\binom{n}{j-i}\binom{n-j+i}{i}, \max\{r_n(i,k-1) + r_n(k+1,j) \mid i \leq k \leq j\}\}$.

Proof. Let E be a semi-regular formula over n variables, and let $\zeta_n^{-1}(E) = b_0 \ldots b_n$. $w_n(\theta_n(E)) \subseteq \{i, i+1, \ldots, j\}$ if and only if E satisfies one of the following two conditions:

(i) For each t such that $t < i$ or $t > j$ $b_t = 0$, and for each k ($i \leq k \leq j$) $b_k = 1$.

(ii) For each t such that $t < i$ or $t > j$ $b_t = 0$, and there exists k such that $i \leq k \leq j$ and $b_k = 0$.

If E satisfies (i) above, from Lemma 1 $NPI(E) = \binom{n}{j-i}\binom{n-j+i}{i}$. If E satisfies (ii) above and k is such an integer, then from Theorem 2 $NPI(E) = r_n(i,k-1) + r_n(k+1,j)$. Thus the theorem holds. □

From the above theorem a dynamic programming technique can be used to compute $\Gamma(n)$.

Algorithm 1. A fast algorithm for computing $\Gamma(n)$.

```
procedure FAST(n);
    comment  In the following statements for each (i,j) such that
             n < i+j  r_n(i,j) should be read r_n(n-j,n-i) (Notice that
             r_n(i,j) = r_n(n-j,n-i) by Lemma 2);
    begin  for t:=0 until n do
             for i:=0 until ⌊(n-t)/2⌋ do
               begin  j:=i+t;
                      r_n(i,i-1):= r_n(j+1,j):= 0;
                      r_n(i,j):= max{ C(n, j-i) C(n-j+i, i), max{r_n(i,k-1) +
                                    r_n(k+1,j) | i ≤ k ≤ j}};
               end
             FAST:= r_n(0,n)
    end
```

Theorem 5. The total logarithmic computing time cost and the total uniform computing time cost of Algorithm 1 by a random access machine are $O(n^4)$ and $O(n^3)$, respectively.

Proof. Let $F(n,r) = n(n-1) \ldots (n-r+1)$, and let $T(n,r)$ be the logarithmic computing time cost of $F(n,r)$. We suppose that the following procedure is used to evaluate $F(n,r)$:

```
procedure F(n,r);
    begin  if r = 1 then F:= n else F:= F(n,⌊r/2⌋) F(n-⌊r/2⌋,⌊(r+1)/2⌋)
    end
```

Then $T(n,r) \leq \log_2 n$ if $r = 1$ and

$T(n,r) \leq r \log_2 n + T(n,\lfloor r/2 \rfloor) + T(n-\lfloor r/2 \rfloor, \lfloor (r+1)/2 \rfloor)$ if $r \geq 2$.

Solving this recurrence we have $T(n,r) = O(r \log_2 r \log_2 n)$. Therefore

the logarithmic computing time cost of $\binom{n}{j-i}\binom{n-j+i}{i}$ is $O(n(\log_2 n)^2)$. Since an upper bound on $g(n)$ is $O(3^n/n^{1/2})$ [2], for each $0 \leqq i \leqq j \leqq n$ $r_n(i,j) \leqq O(3^n/n^{1/2})$. When the values of each $r_n(i,k-1)$ and each $r_n(k+1, j)$ $(i \leqq k \leqq j)$ are available in a table, the logarithmic computing time cost of $\max\{ r_n(i,k-1) + r_n(k+1,j) \mid i \leqq k \leqq j\}$ is $O((j-i)\log_2(3^n/n^{1/2}))$. Hence the logarithmic computing time cost at the inside of the innermost loop of the algorithm is $O(n^2)$. Thus the total logarithmic computing time cost of the algorithm is $O(n^4)$.

The uniform computing time cost at the inside of the innermost loop is $O(n)$. Therefore the total uniform computing time cost of the algorithm is $O(n^3)$. □

A Fortran version of Algorithm 1 was tested on FACOM 230/38 System at Gunma University. The CPU times for 10 variables and for 20 variables were 0.10 seconds and 0.57 seconds, respectively. Algorithm 1 does not enumerate symmetric Boolean functions nor their prime implicants. This is the reason why the algorithm is very efficient.

For each j $(0 \leqq j \leqq n)$ all $r_n(i,i+j)$ $(0 \leqq i \leqq n-j)$ can be evaluated in parallel when for each $0 \leqq b \leqq j-1$ and each $0 \leqq a \leqq n-b$ the value $r_n(a,b)$ is available. We therefore have the next theorem.

Theorem 6. The total logarithmic computing time cost and the total uniform computing time cost of Algorithm 1 by a parallel computer with $O(n)$ processors are $O(n^3)$ and $O(n^2)$, respectively.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addison-Wesley, Reading, MA, 1974).

[2] A.K. Chandra and G. Markowsky, On the number of prime implicants, Discrete Math. 24 (1978) pp. 7-11.

[3] B. Dunham and R. Fridshal, The problem of simplifying logic expressions, J. Symbolic Logic 24 (1959) pp.17-19.

[4] M.A. Harrison, Introduction to Switching and Automata Theory (McGraw-Hill, New York, 1965).

[5] Y. Igarashi, Analysis of Dunham and Fridshal's formulas consisting of large numbers of prime implicants, Tech. Report No. 98, Centre for Computer Studies, University of Leeds, 1977.

[6] Y. Igarashi, An improved lower bound on the maximum number of prime implicants, Trans. IECE of Japan, E62 (1979) pp.389-394.

[7] Y. Igarashi, The size of arrays for a prime implicant generating algorithm, to appear in The Computer Journal, 23 (1980).

[8] D.J. Kleitman, M. Edelberg and D. Lubell, Maximum sized antichains in partial orders, Discrete Math. 1 (1971) pp.47-53.

[9] R.E. Miller, Switching Theory: Vol. 1 (John Wiley, New York, 1965).