論理式簡単化の一手法：MINI-LN

宮腰 隆[†] 松田秀雄[†] 畠山豊正[††]

多値入力二値出力関数を表現する論理式簡単化の一手法：MINI-LN を提案する．本方法の特徴は，(1)項の拡大時に関数全体の否定をそのまま使わずに，その一部の否定を使用することである．すなわち，まず拡大したい項の拡大可能領域を周辺の肯定項および don't care 項より求め，次いで，その限定領域内の否定項(局所否定)だけで拡大操作を行い，否定項の記憶保持の軽減を図っている．さらに，(2)必須主項のほか，MINI，MINI-II で使っていない擬似必須主項を検出して，順次解に採用している．四値のランダム関数(入力変数の数 n=4～7)，算術関数および多変数関数(n=10, 25, 50)について，MINI および MINI-II との比較実験では，本方法は解の項数が少なく，かつ真理値表濃度 0.5 以下の関数で高速に求まった．また，局所否定しか使わないので，MINI や MINI-II では記憶容量の制限で解が求まらない関数でも，本方法では求まった．

A Simplification Method for Logical Expressions: MINI-LN

Takashi Miyagoshi,[†] Hideo Matsuda[†] and Toyomasa Hatakeyama[††]

MINI-LN is a method to simplify logical expressions for a binary function with multiple-valued inputs. The method differs from wellknown MINI and MINI-II, in two aspects. First, in the expansion procedure each term is enlarged without generating all terms of the complement of the given function but only terms of the complement in the peripheries. Second, in addition to essential prime implicants, quasiessential prime implicants are also adopted into the simplified expression. Consequently, by using MINI-LN, we can minimize even functions that cannot be handled by MINI and MINI-II because of the required storage, and obtain the logical expression with fewer terms than other two methods.

1. はじめに

近年 VLSI 内で多用される PLA (Programmable Logic Array) の設計問題[1]は，多値入力二値出力関数を表現する論理式の簡単化問題に帰着される[2]．本来，論理式簡単化問題は最簡形式を得ることが望ましいが，それには多くの計算時間と記憶容量が必要となる．また，高集積化技術の進歩により，大規模回路の取り扱いが要求されるので，発見的手法を使った近似解を得る高速，高性能なアルゴリズムの開発がより一層望まれている．

二値論理関数の簡単化の方法としては，われわれは，部分マップ法を提案し，A5 アルゴリズムやその他の方法と比較検討している[3]．

多値論理簡単化アルゴリズムについても，すでに種々の方法が提案されており，それらの代表例による性能比較もなされている[4]．それらの中で，孤立最小項に着目し，その最小項を優先的に選出する方法では，良い近似解が得られる．しかし，最小項に基づいている欠点として，変数が増えるとその最小項の数が指数関数的に増大するため，実際には 10 変数程度までの関数しか取り扱うことができない[5]．われわれの部分マップ法も考え方としてはその方法に属する．一方，実用向きには数十変数までの処理が必要といわれており，MINI[6]，MINI-II[7] および ESPRESSO-MV[8] 等が優れている．

これらのうち，後の二方法は発見的手法の先駆的研究である MINI の原理に基づいたものであり，それをさらに改良し，発展させた高速かつ高性能なアルゴリズムである．これらの方法の共通点は，与えられた関数 f 全体の否定項(これを後で出てくる局所否定に対して，全体否定と呼ぶ)をまず導出し，導出した否定項を使って，各変数の方向に項を否定項にぶつかるまで拡大している．この手法はトートロジィの判定を使って項を拡大するのに比べ，非常に効率的である

† 富山大学工学部電子情報工学科
Department of Electronics and Computer Science, Faculty of Engineering, Toyama University
†† 富山大学工学部化学生物工学科
Department of Chemical and Biochemical Engineering, Faculty of Engineering, Toyama University

といわれている．しかし，否定項の数は4章でも示すように，入力変数の数 $n=10$（四値）でも，発生関数によってはすぐ何万という項数が生成するので，記憶容量の大きな負担となる．たとえ，記憶容量の面で実行可能であっても，このような数の否定項を対象とした拡大操作では，とても実用時間内に解を得ることはできなくなる．このような否定項の個数の増大に対する方策として，最近，A. A. Malik らによって reduced offsets 法が発表された[11]．この方法では，項 C の拡大操作に関数 f の否定項，つまり，offset を直接使わずに，reduced offsets を使う．reduced offsets とは，項 C を拡大するにあたり，項 C と offset に含まれる項でへだてられている項は，たとえ onset（肯定項の集合：関数値がこれらの項で 1 をとる），あるいは，don't care set（未定義項：関数値がこれらの項で1となるか0となるか，まだ定まっていない）の項であろうとも，これらの項は拡大に寄与しないので，offset に含めてしまって，offset を構成する各項の体積を大きくして項数を減らし，記憶容量ならびに計算時間の軽減を図ろうとするものである．しかし，reduced offsets といえども，関数 f の全体否定の項数を減らしているだけで，全体否定を使っていることには変わりはない．

ここに提案するわれわれの MINI-LN は，部分マップ法の延長上から考案したもので，原理的には MINI の手法を継承するが，ある項の拡大操作にあたっては，あらかじめ見当づけた範囲に含まれる否定項のみ用いる手法である．すなわち，まず拡大したい項の拡大可能領域を周辺の肯定項および don't care 項より求める．次に，その限定領域内だけの否定項（これを局所否定：Local Negation と呼ぶ）を生成し，これらだけを使って拡大操作を実行する*．

われわれの方法では，項の拡大ごとに局所否定を求める必要も生じてくるが，対象とする否定項数が少ないので計算時間ならびに記憶容量の大幅な節減をもたらす．また，MINI-LN では，必須主項だけでなく擬似必須主項の検出も行っているため，MINI や MINI-II より解の項数は少ない．また，reduced offsets 法は二値入力しか扱えないが，本方法では多値入力を扱うことを想定している．

2章では，ここで使用する基礎的用語を説明し，3章で簡単化の基本手続きとアルゴリズムを例題により詳述する．4章ではそれを FORTRAN でプログラム化し，四値のランダム関数（$n=4$〜7）と各種関数（算術関数3例，および多変数関数 $n=10$, 25, 50）で行った実験結果を示し，MINI および MINI-II と性能評価の比較を行う．

2. 基礎的用語

（1） 多値入力二値出力関数

関数 $f(X_1, X_2, \cdots, X_n)$: $P_1\times P_2\times\cdots\times P_n\to B^m$, $P_i=\{0, 1, \cdots, p_i-1\}$, $B=\{0, 1, \mathrm{dc}^*\}$ を，（一般化した）n 変数多値入力二値 m 出力関数という．

ここで，$m=1$ の場合が単一出力関数であり，$m>1$ の場合が多出力関数となる．m 出力関数 f は m 個の単一出力関数 f_0, $f_1, \cdots, f_{m-1}$ の出力を $(f_0, f_1, \cdots, f_{m-1})$ と並べて B^m のベクトルとしたものである．

ここで，$(n+1)$ 変数多値入力二値単一出力関数 $G(X_1, X_2, \cdots, X_n, Z)$: $P_1\times P_2\times\cdots\times P_n\times\{0, 1, \cdots, m-1\}\to B$ を考える．ただし，$X_i\in P_i$ $(i=1, 2, \cdots, n)$, $Z\in\{0, 1, \cdots, m-1\}$．

$G(X_1, X_2, \cdots, X_n, j)$ で上記の n 変数多値入力二値単一出力関数 f_j $(j=0, 1, \cdots, m-1)$ を表すものとすれば，n 変数多値入力二値 m 出力関数 f は $n+1$ 変数多値入力二値単一出力関数 G として，取り扱うことができる[2]．したがって，以下 $m=1$ の場合を，多値入力二値出力関数と記す．

今，不完全記述関数 f に対して，f と同じ定義域と値域をもつ完全記述関数 f^1, f^0 および f^{dc} を考え，これらは $f=1$, 0 および未定義となる領域で，それぞれ $f^1=1$, $f^0=1$ および $f^{dc}=1$ とする．

X_i を入力変数とし，$S_i\subseteq P_i$ とすると，リテラル $X_i^{S_i}=1$ $(X_i\in S_i$ のとき$)$, $X_i^{S_i}=0$ $(X_i\notin S_i$ のとき$)$ である．リテラルの積項 $C=X_1^{S_1}\cdot X_2^{S_2}\cdots X_n^{S_n}$ を項という．また，項の体積を $\prod_{i=1}^{n}|S_i|$ とし，体積が1の項を最小項という．いくつかの項の和は一つの関数 g $(=C_1\vee C_2\vee\cdots\vee C_q)$ を表し，最小項の和に分解できる．関数 g_1 の（和を形成する）各項に含まれるどの最小項も関数 g_2 の（和を形成する）項のどれかの最小項になっているとき，g_1 は g_2 に含まれるといい，$g_1\leqq g_2$ と表す．関数 g の項 C が g の他の項に含まれなければ，項 C を関数 g の主項という．関数 g は和を形成する項の集合 $\{C_1, C_2, \cdots, C_q\}$ とも考えられ，

* A5 アルゴリズム[2]でも，拡大可能領域 CM_j（後出）と類似の拡大可能領域 C_m を使うが，その差異については3章で述べる．

* dc (don't care) は関数を与えた時点では0とも1ともみられるが，簡単化の過程で自動的に0か1に定まる．

場合によっては関数を項の集合として与えることもある．

ここで，上記の f^1 と f^{dc} を用いて次章で使う F と D は，簡単化アルゴリズムの出発時にはそれぞれ $f^1=\bigvee_{C\in F} C$，$f^{dc}=\bigvee_{C\in D} C$ である．

本論文では，f^1 の最小項をすべて被覆し，かつ場合によっては f^{dc} の一部の最小項をも被覆する項の和形式で表される関数 F_c の項数を最小にすることを主題とし，これを簡単化と呼ぼう．

項 C をキューブともいい，次のようにビット表現ができる．

$$C=b_1^0b_1^1\cdots b_1^{p_1-1}\text{-}\cdots\text{-}b_i^0\cdots b_i^k\cdots b_i^{p_i-1}\text{-}\cdots\text{-}b_n^0b_n^1\cdots b_n^{p_n-1}, \quad (1)$$

ここで，-で区切られた部分は左より $X_1^{S_1}, \cdots, X_i^{S_i}, \cdots, X_n^{S_n}$ の各リテラルに対応し，$k\in S_i$ なら $b_i^k=1$，$k\notin S_i$ なら，$b_i^k=0$ とする．また，式(1)を便宜上，

$C=\mu_1\mu_2\cdots\mu_i\cdots\mu_n$

と表したとき*，μ_i は $\text{-}b_i^0b_i^1\cdots b_i^{p_n-1}\text{-}$ と対応しており part i と呼ぶ．

[例題 2.1] 2変数四値入力二値出力関数 f_1 を示す．

$$f_1=X_1^{(1,2)}\cdot X_2^{(0)}\vee X_1^{(2,3)}\cdot X_2^{(3)}\vee X_1^{(0)}\cdot X_2^{(0,1)}\vee X_1^{(1)}\cdot X_2^{(3)},$$

ここで，$f_1: P_1\times P_2\to B$，$P_1=P_2=\{0,1,2,3\}$，$B=\{0,1,dc\}$ である．f_1 の各項はビット表現でそれぞれ C_1=0110-1000，C_2=0011-0001，C_3=1000-1100，C_4=0100-0001 と表すことができる．**図1**は，関数 f_1 をマップ表現したもので，項 C_1 と C_2 は f^1 の最小項の和からなり，それぞれ C_1=0100-1000∨0010-1000，C_2=0010-0001∨0001-0001 と表され，また，項 C_3 と C_4 は f^{dc} の最小項によって，それぞれ C_3=1000-1000∨1000-0100，C_4=0100-0001 と表される．それ以外の所は f^0 の領域(図1では無印)である．したがって，本例で与えた項で表示すれば，$F=\{C_1,C_2\}$，$D=\{C_3,C_4\}$ である．

(2) 項の最小（最大）番号の最小項

任意の項 C_j において，各 part の最左端（最右端）の1のみ残し，他の桁をすべて0と置いてできる最小項 $C_{j,mi}(C_{j,ma})$ は，C_j に含まれる最小項のうち，それらを表すビット表現による2進数の大きさで順序

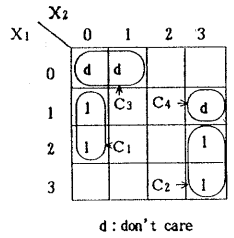


図1 関数 f_1
Fig. 1 Function f_1.

付けた（ただし，左側より右側に桁が上がるとして）最小（最大）番号のものである．例えば，上の C_1=0110-1000 の場合，$C_{1,mi}$=0100-1000，$C_{1,ma}$=0010-1000 である．

(3) 隣接する項，共通部分をもつ項

二つの項

$C_1=\mu_1\mu_2\cdots\mu_i\cdots\mu_n$,

$C_2=\pi_1\pi_2\cdots\pi_i\cdots\pi_n$ (2)

について，ビットごとの論理積 $C_1\cap C_2$ で，ただ1カ所の part i で $\mu_i\cap\pi_i=\phi$ (part i の全ビットが0)，その他の part k (ただし，$k=1,2,\cdots,n, k\neq i$) で $\mu_k\cap\pi_k\neq\phi$ の C_1 と C_2 は part i で隣接している項といい $A_i(C_1,C_2)$ と表す．また，すべての part i で $\mu_i\cap\pi_i\neq\phi$ なら，C_1 は C_2 は共通部分をもち $CO(C_1,C_2)$ と表す．さらに，項集合 V の項で，C_1 と part i で隣接する項の集合を $L_{Ai}(C_1,V)$ と表し，また，C_1 と共通部分をもつ項の集合を $Lco(C_1,V)$ と表記する．

(4) Super cube[2)]

項集合 $Q=\bigvee_{j=1}^{t} C_j$,

ただし，各項 $C_j=\mu_{j1}\mu_{j2}\cdots\mu_{ji}\cdots\mu_{jn}$, について，各 part i でのビットごとの論理和演算を，

$$\text{ORpart}_i\ Q=\begin{cases}\phi & (Q=\phi)\\ \bigcup_{j=1}^{t}\mu_{ji} & (Q\neq\phi),\end{cases}$$

と定義する．このとき，各 part i が $E_i=\text{ORpart}_i\ Q$ $(i=1,2,\cdots,n)$ からなる項 $E_1\cdots E_i\cdots E_n$ を項集合 Q のスーパーキューブといって，次のように表現する．

Super cube $Q=E_1\cdots E_i\cdots E_n$.

(5) 併合

式(2)で，$\pi_i\subseteq\mu_i$ ($\pi_i\cap\mu_i=\pi_i$ が成り立つこと，ただし，∩はビットごとの論理積演算とする) $(i=1,2,\cdots,n)$ のとき，C_2 は C_1 に包含されるといい，C_2 を除去できる．また，ただ一つの i について $\mu_i\neq\pi_i$，i 以外のすべての k で $\mu_k=\pi_k$ (ただし，$k=1,2,\cdots,n$,

* 項 C の表現として，本論文では $X_1^{S_1}\cdot X_2^{S_2}\cdots X_n^{S_n}$ または $\mu_1\mu_2\cdots\mu_n$ を両用する．

$k\neq i$) のとき，C_1 と C_2 は距離1であるといい，C_3=Super cube $\{C_1, C_2\}$ という一つの項 C_3 に併合できる．

（6） **ディスジョイント・シャープ演算[6]；⊛**

二つの項

$$C_j=\mu_1\mu_2\cdots\mu_i\cdots\mu_n,$$
$$C_e=\pi_1\pi_2\cdots\pi_i\cdots\pi_n$$

に対して，$C_j \circledast C_e=\bigvee_{i=1}^{n} C_i$ とする．ただし，

$$C_i=(\mu_1\cap\pi_1)\cdots(\mu_{i-1}\cap\pi_{i-1})(\mu_i\cap\bar{\pi}_i)\mu_{i+1}\cdots\mu_n$$

である．また，二つの項集合

$$Q_1=\bigvee_{j=1}^{a} x_j,\quad Q_2=\bigvee_{e=1}^{b} y_e,$$

に対して，

$$Q_1\circledast Q_2=((\cdots((Q_1\circledast y_1)\circledast y_2)\cdots)\circledast y_{b-1})\circledast y_b,$$
$$Q_1\circledast y_1=\bigvee_{x_j\in Q_1}(x_j\circledast y_1)$$

とする．

（7） **整形[6]**

二つの項を

$$C_1=\mu_1\mu_2\cdots\mu_i\cdots\mu_k\cdots\mu_n,$$
$$C_2=\pi_1\pi_2\cdots\pi_i\cdots\pi_k\cdots\pi_n$$

として，2カ所の part i, k だけで異なり，$\pi_i\not\subseteq\mu_i$ かつ $\pi_k\subseteq\mu_k$ ならば，C_1 と C_2 の二項はそれぞれ

$$C_1'=\mu_1\mu_2\cdots\mu_i\cdots(\mu_k\cap\bar{\pi}_k)\cdots\mu_n,$$
$$C_2'=\pi_1\pi_2\cdots(\mu_i\cup\pi_i)\cdots\pi_k\cdots\pi_n$$

と整形される．

（8） **コンセンサス**

式(2)の二つの項 C_1 と C_2 とのコンセンサスは，

$$\mathrm{CONS}(C_1, C_2)=\bigvee_{i=1}^{n}(\mu_1\cap\pi_1)(\mu_2\cap\pi_2)\cdots(\mu_i\cup\pi_i)\cdots(\mu_n\cap\pi_n)$$

と定義される．

（9） **真理値表濃度**

マップの全最小項数 $\prod_{i=1}^{n}|P_i|$ に対する関数値1をとる最小項の総数 $|f^{-1}(1)|$ の割合を真理値表濃度 d という．

3. 簡単化アルゴリズム

3.1 基本手続き

簡単化アルゴリズムの中で使用する基本手続き［拡大］，［必須主項の検出］について説明する．

［拡大］ ある項 C_j を周辺に拡げて主項にまで拡大する手順は次のとおりである．

（1） 項 C_j の最小番号の最小項 $C_{j,mi}$ と最大番号の最小項 $C_{j,ma}$ を求める．

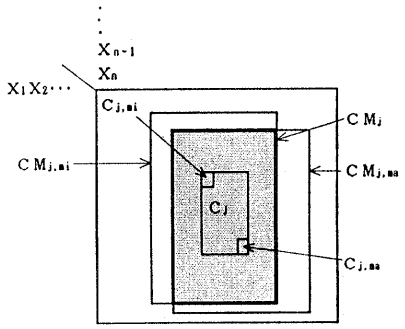


図 2 項 C_j とその拡大可能領域 CM_j
Fig. 2 Term C_j and its expandable domain CM_j.

（2） まず，$C_{j,mi}$ の拡大可能領域 $CM_{j,mi}$ を求める．

$$CM_{j,mi}=\text{Super cube }(Lco(C_{j,mi}, F\vee D)\vee C_A),\qquad(3)$$

ここで，$C_A=X_1^{T_1}\cdots X_i^{T_i}\cdots X_n^{T_n}$, $T_i=\mathrm{ORpart}_i\{C_{j,mi}, L_{Ai}(C_{j,mi}, F\vee D)\}$ である．同様にして，$C_{j,ma}$ の拡大可能領域 $CM_{j,ma}$ も求める．そして，C_j の拡大可能領域 CM_j を次のように定義する．

$$CM_j=CM_{j,mi}\cap CM_{j,ma}.\qquad(4)$$

このような拡大可能領域を用いる理由を**図2**の模式図で説明しよう．項 C_j が拡大できるためには，少なくとも，C_j の左上端点の最小項 $C_{j,mi}$ および右下端点の最小項 $C_{j,ma}$ が周辺の項と併合して拡大されねばならない．そこでまず，最小番号の最小項 $C_{j,mi}$ の拡大可能領域 $CM_{j,mi}$ を関数 F の項と D の項で $C_{j,mi}$ と共通部分のある項ならびに隣接する項，すなわち，$C_{j,mi}$ の周辺の項から推定する．$CM_{j,mi}$ には C_j 自身も含まれるので，たとえ，周辺に拡大できなくとも，C_j の大きさは確保している．同様に最大番号の最小項 $C_{j,ma}$ から $CM_{j,ma}$ を推定し，これらの共通領域 CM_j は少なくとも拡大不可能な場合でも C_j を含み，C_j が拡大可能な場合には，できるだけ小さく見積もった拡大可能領域の一つと考えられる．

（3） 限定領域 CM_j は一つの項でもある．次の項集合 G を考える．$G=\{CM_j\cap C_e | C_e\in F\vee D\}$，$G$ は $F\vee D$ の項集合のうち，CM_j の領域に限定した部分のみからなる項集合である．さて，ここで G は CM_j に含まれる最小項の領域でのみ定義された関数と考え，この領域での G の否定 $\bar{G}_L$ を，次の手続きで求まるとしよう．

COMP (CM_j,G,$\bar{G}_L$, r),

ただし，r は $\bar{G}_L$ の項数である．$\bar{G}_L$ はある領域に限定した否定なので局所否定と呼ぼう．

この手続きで，$CM_j=C_u$ (universal cube：すべての part が1のみ 1111-…-1111 である項，ただし，- は Part の区切りを示す)，$G=F\vee D$ で求めた $\bar{G}_L$ は，一般に関数 f の全体否定といわれるものである．われわれは手続き COMP として，笹尾の関数 f の全体否定を導出する高速アルゴリズム[10]を任意の限定領域での局所否定が導出できるよう修正して使っている．例えば，図1の関数で，$CM_j=C_u$，$G=\{C_1, C_2, C_3, C_4\}$ として，COMP (C_u, G, $\bar{G}_L$, r) を実行すれば $\bar{G}_L$ として3個 ($=r$) の否定項 {0111-0110, 0001-1000, 1000-0011} が求まるし，CM_j として項 C_1 の拡大可能領域 CM_1=1110-1001 を，G として CM_1 内の項集合 {0110-1000, 0010-0001, 1000-1000, 0100-0001} を与えて，COMP (CM_1, G, $\bar{G}_L$, r) を実行すれば，$\bar{G}_L$ として1個の否定項 {1000-0001} が得られる．このように自由に設定した領域内の否定項を簡単に求めることができる．ここで，もし $r=0$ の場合，すなわち，限定領域 CM_j 内に否定項が存在しない場合は，CM_j が主項である．さらに，拡大前の項 C_j が最小項ならば，CM_j は（後述する）必須主項*である．

(4) 項 C_j の各 part i ($i=1,2,\cdots,n$) が CM_j 内の項集合 ($\in F$) のうち W 個の項の part i を包含するとき，その個数 W を part i の重みとした数値 W_i を順次求める．

(5) 項 C_j をこの重み W_i の昇順に，一 part ずつ拡大してゆき主項を得る．ただし，C_j と CM_j を

$$C_j=\mu_1\mu_2\cdots\mu_i\cdots\mu_n,$$
$$CM_j=\pi_1\pi_2\cdots\pi_i\cdots\pi_n$$

とすれば，拡大は $\pi_i-\mu_i\neq\phi$ である part i でだけ実行する．ここで，$-$は μ_i の各ビットを 0→1, 1→0 としたものを $\bar{\mu}_i$ とすれば，$\pi_i-\mu_i=\pi_i\cap\bar{\mu}_i$ のことである．一つの part i の拡大は，

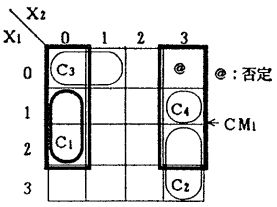
$$C_j=\mu_1\mu_2\cdots(\pi_i-z_i)\cdots\mu_n$$

となる．ただし，$L_{Ai}(C_j, \bar{G}_L)$ である否定項の集合を R と表せば，$z_i=\mathrm{OR}\,\mathrm{part}_i\ R$ とする．

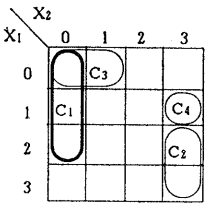
(6) 主項 PI_j を C_j として拡大終了．

[例題 3.1] 図1の関数例で項 C_1 を拡大する場合を説明する．項 C_1=0110-1000 なので，$C_{1,mi}$=0100-1000，$C_{1,ma}$=0010-1000 である．また，$C_{1,mi}$ の拡大

* 文献 9) の定理 (4) 参照.



(a) 拡大前
(a) Before expansion



(b) 拡大後
(b) After expansion

図 3 項 C_1 の拡大例
Fig. 3 Example of expansion of a term C_1.

可能領域 $CM_{1,mi}$ は $L_{CO}(C_{1,mi}, F\vee D)=C_1$, $L_{A1}(C_{1,mi}, F\vee D)=C_3$, $L_{A2}(C_{1,mi}, F\vee D)=C_4$ なので，$CM_{1,mi}$= Super cube (0110-1000∨1100-1001)=1110-1001 となる，同様にして，$CM_{1,ma}$ も求める（この例では $CM_{1,mi}$ と等しくなる)．そこで，項 C_1 の拡大可能領域 CM_1 は，

$$CM_1=CM_{1,mi}\cap CM_{1,ma}=1110\text{-}1001$$

となる(**図3**(a) 太線枠)．次に，この領域 CM_1 内に存在する項集合 G を求め，COMP (CM_1, G, $\bar{G}_L$, r) の実行により，先に述べた $\bar{G}_L=\{1000\text{-}0001\}$ が得られる（図3(a)で@表示)．CM_1 に完全に包含される F 項は C_1 だけなので，拡大する part 順は 1, 2 となる．

まず，part 1 で実行すると，

z_1=0000 ($R=L_{A1}(C_1, \bar{G}_L)=\phi$),

C_1=(1110-0000)-1000=1110-1000 となる．

次いで，part 2 で行うと，

z_2=0001 ($R=L_{A2}(C_1, \bar{G}_L)\neq\phi$),

C_1=1110-(1001-0001)=1110-1000 となる．

これが求まった主項である．図3(b)は主項 C_1 の様子を示す．

[必須主項の検出] 拡大した主項 C が必須主項かどうかを判定し，必須主項なら優先的に解に採用する．今，$F=C\vee G$ とする．ここで，C は主項，G は C 以外の F の項集合である．また，D_1 を D 項のうち C と共通部分を持つ項集合，D_2 を D 項のうち C

と隣接する項集合とする．C と G のすべての項とコンセンサスをとってできる項からなる集合を H_1 とする．すなわち，

$$H_1=\bigvee_{h_1\in G}\mathrm{CONS}\ (C,h_1), \qquad (5)$$

ただし，C に含まれる項は除外する．

また，さらに，C と D_2 のすべての項とのコンセンサスをとってできる項の全体のうち，H_1 の中の一項ともコンセンサスがとれる項の集合（ただし，C に含まれる項は除外して考える）を H_2 とする．すなわち，

$h_2=\mathrm{CONS}\ (C,d_2)$，ただし，$d_2\in D_2$ とし，もし，

$$\mathrm{CONS}_{h_1\in H_1}\ (h_2,h_1)\neq\phi \qquad (6)$$

なら，そのとき h_2 を H_2 に含める．

ここで，d_2 は D_2 のすべての項について繰り返す．

このとき，主項 C が必須主項と判定できるのは，

$$C\not\leq\{H_1\vee D_1\vee H_2\} \qquad (7)$$

が成立するときである．

（証明） 主項 C を $C=X_1^{S_1}X_2^{S_2}\cdots X_i^{S_i}\cdots X_n^{S_n}$ とし，$C\not\leq\{H_1\vee D_1\vee H_2\}$ であるとしよう．すると，$C\not\leq\{H_1\vee D_1\vee H_2\}$ なので，C 内に $v\cap f^1\neq\phi$ かつ $v\leqq C\cdot\overline{\{H_1\vee D_1\vee H_2\}}$ である最小項 $v=X_1^{a_1}X_2^{a_2}\cdots X_i^{a_i}\cdots X_n^{a_n}$ が存在する．ここで，$a_i\in S_i$ $(i=1,2,\cdots,n)$ である．

そこで，v を含む別の主項 C' が存在すると仮定しよう．$C'=X_1^{T_1}X_2^{T_2}\cdots X_i^{T_i}\cdots X_n^{T_n}$ とする．$C\cap C'\neq\phi$，$C\not\subseteq C'$ なので，ある k において $T_k-S_k\neq\phi$ であり，C' 内に最小項 $v'=X_1^{a_1}X_2^{a_2}\cdots X_{k-1}^{a_{k-1}}X_k^{b_k}X_{k+1}^{a_{k+1}}\cdots X_n^{a_n}$ $(b_k\in T_k-S_k)$ を考えることができる．$v'\notin C$，C と part k で隣接しているから，v' は $\{H_1\vee H_2\}$ のどれかの項に含まれる最小項である．したがって，その項を $q=X_1^{E_1}X_2^{E_2}\cdots X_k^{E_k}\cdots X_n^{E_n}$ と表す．ここで，$a_i\in E_i$ $(i=1,2,\cdots,n,i\neq k)$，$b_k\in E_k$ である．そこで，q と C とのコンセンサスを考えてみると，

$$h=\mathrm{CONS}\ (C,q)\supseteq X_1^{S_1\cap E_1}X_2^{S_2\cap E_2}\cdots X_k^{S_k\cup E_k}\cdots X_n^{S_n\cap E_n},$$

$a_i\in S_i\cap E_i$ $(i\neq k)$，かつ $a_k\in S_k\cup E_k$ なので，$v\in h$ である．これは $v\leqq\{H_1\vee D_1\vee H_2\}$ ということであり，はじめの $v\leqq C\cdot\overline{\{H_1\vee D_1\vee H_2\}}$ という仮定に反する．ゆえに，v は特異最小項であり，主項 C は必須主項である． （証明終り）

［例題 3.2］［例題 3.1］で求めた主項 C_1 が必須主項かどうかを判定してみる．C_1=1110-1000，C_2=0011-0001，C_3=1000-1100，C_4=0100-0001，$D_1=C_3$，$D_2=C_4$ である．まず H_1 を求める．$h_1=\mathrm{CONS}\ (C_1,C_2)$=0010-1001 を H_1 の項とする．次に H_2 を求める．D_2 の項は C_4 だけである．$h_2=\mathrm{CONS}\ (C_1,C_4)$=0100-1001，ここで生成された h_2 が，先に求めた h_1 とさらに結合するか調べる．そこで，h_2 と h_1 のコンセンサスを求めてみると，$h_3=\mathrm{CONS}\ (h_2,h_1)$=0110-1001 であり，$h_3\not\leq C_1$ なので，h_2 を H_2 に加える．すると，$H_1=h_1$，$D_1=C_3$ および $H_2=h_2$ より，$C_1\leqq\{H_1\vee D_1\vee H_2\}$ となり C_1 は必須主項ではない．しかし，C_2 が C_2' の位置にあった場合（**図 4** 参照）は，$h_1'=\mathrm{CONS}\ (C_1,C_2')$=0010-1100，$h_3=\mathrm{CONS}\ (h_2,h_1')$=0110-1000$\subseteq C_1$ となるので h_2 を H_2 に加えない．したがって，$H_1=h_1'$，$D_1=C_3$ および $H_2=\phi$ より，$C_1\not\leq\{H_1\vee D_1\vee H_2\}$ となり，主項 C_1 は必須主項と判定される．

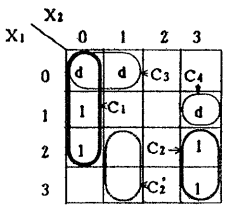


図 4 必須主項の検出例 (C_1)
Fig. 4 Detecting example of an essential prime implicant (C_1).

3.2 アルゴリズム

本方法のおおまかな手順は次のとおりである．ただし，［ ］は 3.1 節に従う．

1° F_C に解が，F_E に必須主項がそれぞれ求まるものとする．最初に $F_C=\phi$, $EPI=1$ としておく．また，関数として項集合 F および D（完全記述関数は $D=\phi$）を与える．

2° （併合）F で併合できるものは併合する．

3° （入力部拡大，出力部縮小）多出力関数の場合だけ行う．二項 C_1，$C_2(\in F)$ に異なる二 part i，k が存在し，このうち，k が出力部 ($k=n$) のとき，二項に整形を適用する．

#: F の項数を t_1，$F_E=\phi$ とした後，F の各項について次の 4°と 5°を行う．ただし，$EPI=0$ のときは 5°をとばす．

4° ［拡大］一つの項 C_j を拡大する．

5° ［必須主項の検出］主項 C_j が必須主項か否かを判定する．もし必須主項なら，$F_E=F_E\vee C_j$ とする．F の各項をひととおり調べた後，

6° $F_E=\phi$ なら $EPI=0$ とする．$F_E\neq\phi$ なら，5°で得られた F_E について，$F=F-F_E$，$F_C=F_C\vee F_E$，

$D=D\vee F_E$ とする．また，$F\neq\phi$ なら，F の項数を t_2 として 7° へ行き，$F=\phi$ なら 10° へ行く．

7° $t_2<t_1$ なら次の 8° へ行き，$t_2\geqq t_1$ なら $F_C=F_C\vee F$ として 10° へ行く．

8° （縮小）項 C_j が他の項と重複する部分を除去する．

$$C_j=\text{Super cube}\ (C_j\circledast((F-C_j)\vee D)),$$

ここで，$C_j=\phi$ となる場合は項 C_j が除去される．

9° （整形）F 内の二項 C_1, C_2 が変形できるかすべての組み合わせで調べる．再び 4° の前 ♯ へ戻る．

10° （検証）得られた解 F_C が，1° の F と D を使い $F_C\subseteq F\vee D$ かどうかを確認する．

（アルゴリズム終り）

基本手続き［拡大］および基本操作（併合），（縮小），（整形）を行う前にそれぞれ前処理として項 C_j を並べかえることがある．本方法では，この並べかえ操作を，それぞれ MINI の ORDF，各項を2進数とみて昇順にする方法および修正 ORDF[6] に従った．通常，ステップ 4°〜9° の繰り返し計算のあと検証に至る．その際，繰り返し回数1回目で必須主項が検出されれば，以後，ステップ 5° は擬似必須主項*を検出することになる．ただし，必須主項が検出されなければ（6° で $EPI=0$ とする），以後，擬似必須主項は検出されず，MINI のように，単に項の縮小，整形，拡大を繰り返すだけで，解の項数の最小化を図ることになる．

［例題 3.3］ 図1の2変数四値入力二値出力関数例に本アルゴリズムを適用してみる．$F=\{C_1, C_2\}$，$D=\{C_3, C_4\}$ である．$F_C=F_E=\phi$ とする．必須主項の検出を行うため $EPI=1$ としておく．併合はできない（2°）．$t_1=2$．項 C_1 の拡大で，主項 C_1=1110-1000 が得られる（4°，［例題 3.1］参照）が，必須主項ではない（5°，［例題 3.2］参照）．次に，項 C_2 の拡大より，主項 C_2=0111-0001 が得られる．H_1=0110-1001，$D_1=C_4$ および $H_2=\phi$ から，$C_2\not\leqq\{H_1\vee D_1\vee H_2\}$ であり必須主項として検出される．ひととおり項を拡大し，必須主項を検出した結果，$F_E=C_2$ が得られたので，$F=F-C_2$，$F_C=F_C\vee C_2$，$D=D\vee C_2$ とする（6°）．ここで，F に残っているのは項 C_1 だけ（$t_2=1$）で，$t_2<t_1$ である（7°）．主項 C_1 は重複している C_3 により縮小（8°）されて元へ戻る．C_1 1個だけなので整形（9°）は行わない．4° の前へ戻って再度 C_1 を拡大すると，前と同じく C_1=1110-1000 が得られる．今度は C_2 が D に変わった後なので，$H_1=H_2=\phi$ および $D_1=C_3$ で $C_1\not\leqq\{H_1\vee D_1\vee H_2\}$ となり，擬似必須主項として検出される．$F=\phi$ なので検証（10°）して終る．解として，F_C=1110-1000∨0111-0001 が求まる．

3.3 他の方法との比較

ここで，MINI-LN と次章で計算結果を比較する他の方法との差異を明らかにしておく．

MINI では，項の縮小，整形，拡大を繰り返すだけで，解の項数の最小化を図っている．MINI-Ⅱ では，これらの操作のほかに，さらに必須主項の検出も行って解の項数を減らすように改良している．われわれの MINI-LN では，その上，擬似必須主項の検出も行って，解の項数の一層の最小化を心がけている．

MINI や MINI-Ⅱ では，項の拡大に関数 f の（全体）否定 $\bar{f}$ を使うが，われわれの MINI-LN では項の拡大に局所否定を使って，必要記憶量の削減を図っている．

一方で関数の否定を使わないという立場で A5 アルゴリズムが提案されているが，つぶさに検討すると，われわれのいう局所否定が使われている．

われわれの拡大可能領域 CM_j は項 C_j の最小番号の最小項の拡大可能領域 $CM_{j,mi}$（式(3)）と，最大番号の拡大可能領域 $CM_{j,ma}$ の共通領域から求めている．A5 の拡大可能領域 C_m は，式(3)の最小項 $C_{j,mi}$ を項 C_j と置き換えたもので，C_j そのものから拡大可能領域を求めるといってよい．$CM_j\subseteq C_m$ がいえて，本方法の方がより狭い範囲に設定できる．

本方法では一つの項 C_j の拡大にあたり，拡大可能領域 CM_j に限定した $f=F\vee D$ の否定を高速なプログラム COMP でそれらの項を利用して1回求めるだけで C_j の拡大が可能となる．

A5 では C_m をさらに細分した領域から拡大しはじめる．これを2変数の模式**図5**で示す．

まず，C_m を拡大したい項，1変数（例えば）X_1 成分を除いて C_j にまで縮小する．この部分領域を C_{m1} としよう（図5(a)網掛け部分）．$C_{m1}\circledast(F\vee D)$ の項を使って，C_j を X_1 方向に拡大する．仮に C_j が図5(b)のように拡大されたとする．今度は C_m を X_2 を除いて，C_j にまで縮小する．これを図5(c)のように C_{m2} とする．$C_{m2}\circledast(F\vee D)$ の項を使って，C_j を拡大する．すなわち，A5 アルゴリズムでは一つの項の拡大にあたり，最悪な場合には変数の数だけ⊛演算

* 関数 F において必須主項を検出して，D (don't care) の項に変えた後，あらためて必須主項となる項を擬似必須主項という．

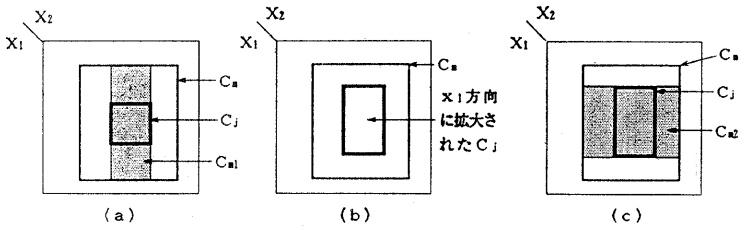


図 5 A5 の拡大を表す模式図
Fig. 5 A schematic diagram for explanation of expansion in the A5 algorithm.

で否定を求めなければならない．

また，真理値表濃度の大きい関数では，C_m がマップ全体に一致することも生じてくるので，㊀演算で否定を求めることは非常に能率が悪い[2]．

4. 実験結果

3.2 節で述べた本方法のほかに，MINI，MINI-Ⅱ（MINI の改良版）および A5 アルゴリズム*をそれぞれ FORTRAN 言語でプログラム化し，IBM 3081-KX4（富山大学情報処理センター）で実行した．ただし，Hong らの MINI では，全体否定の導出にディスジョイント・シャープ演算㊀を使用しているが，これよりも高速に否定が導出できる MINI-Ⅱ で使われた木形アルゴリズムを，われわれは MINI にも組み込んで使った．

表 1 は，入力変数の数 $n=4\sim7$ の四値入力二値出力関数において，真理値表濃度 $d=0.2, 0.5, 0.8$ で各 5 個ずつ発生したランダム関数による解の項数および計算時間（秒）の平均値であり，したがって，平均値の項では，計 15 個の関数による解の項数および計算時間の総平均値を示している．ただし，$n=7$ の各数値は 2 個ずつの関数による平均値である．$n=7$ のとき，MINI および MINI-Ⅱ では関数の否定項数が多くなり，ユーザ割当て領域 8 MB のわれわれの計算機では，配列の最大サイズが 20,000 程度しかとれずオーバフローとなって結果が得られなかった（表 1，—表示）．一方，MINI-LN でも，$n=7$, $d=0.8$ のとき，局所否定を求める領域が全体否定を求める領域（マップ全体）と一致するようになり，記憶容量が不足して計算不能となった．

解の項数では，MINI-LN が表のすべてにわたって，MINI および MINI-Ⅱ よりも少なく，とりわけ，真理値表濃度 0.8 で顕著である一方，平均計算時間では，MINI-LN が MINI および MINI-Ⅱ より大きい．しかし，このことを真理値表濃度 d ごとに見てみると，$d=0.2$ では MINI-LN が最も高速である．また，$d=0.5$ では，n が 4, 5, 6 と増えると，MINI-LN の計算時間の方が他の二方法より小さくなっていく．$d=0.8$ では他の方法より 3，4 倍かかるようになり，これが総平均計算時間を大きくしている．その理由としては，真理値表濃度が高くなると擬似必須主項の検出に関係する項が増えるので，その手数が増えることがまず考えられる．それと本方法で，ある項 C_j を拡大する場合，周辺に共通部分をもつ項や隣接する項が多数存在することから，拡大可能領域が大きく設定される傾向にある．場合によっては C_u（マップ全体）に近い大きさの領域の否定が幾度も導出されて極端に時間がかかるのではないかと考えられる．なお，本プログラムでは，局所否定を求める領域が全体否定を求める領域（マップ全体）と一度一致したら，以後それを利用して，局所否定をいちいち求めないように改善している．

しかしながら，MINI-LN が $d\leqq0.5$ で高速であるということは，次の点で価値がある．

（1） 実用上，有用な関数は真理値表濃度が小さい．

（2） 真理値表濃度が大きい関数は，その否定を求めれば 0.5 以下となり，この否定の簡単化を求め，あと否定（NOT）ゲートを一つ加えれば元の関数が得られる．

さらに，表 1 に参考までに A5 アルゴリズムの計算結果も載せてあるが，MINI-LN より解の項数が多くなり，計算時間でも劣っていることがわかる．

表 2 は，選出した算術関数[1]および多変数関数の実行例である．ただし，算術関数（先の 3 例）は四値で実行するために，入力変数を二変数ずつ対にして一変数に割り当てた．また，MLP_4 だけは同位桁を同一変数に割り当てた．これらの関数では，3 方法にあまり差異はみられないが，MINI-LN の解の項数が若干少ない．

後の 3 例は，多変数関数（$n=10, 25, 50$）での比較結果である（関数発生法は付録参照）．多変数関数になると，関数の否定項数も増えてきて，FUN 1 で

* A5 アルゴリズムの結果は，参考のため一部の表でのみ示す．

表 1 乱数で発生した関数での比較結果（四値）
Table 1 Minimization results of randomly generated functions with 4-valued inputs.

入力数 n	真理値表濃度 d	MINI 解の項数	MINI Time（秒）	MINI-II 解の項数	MINI-II Time（秒）	MINI-LN 解の項数	MINI-LN Time（秒）	A5 解の項数	A5 Time（秒）
4	0.2	26.2	0.2	26.0	0.2	26.0(26.0)	0.1(0.1)	26.4	0.5
	0.5	35.8	0.4	35.2	0.5	35.0(35.0)	0.8(0.5)	36.8	3.5
	0.8	26.0	0.3	27.2	0.5	25.6(25.6)	1.2(0.4)	27.4	5.9
	平均値	29.3	0.3	29.4	0.4	28.8(28.8)	0.7(0.3)	30.2	3.3
5	0.2	100.0	4.4	99.2	4.2	99.4(99.4)	1.6(1.6)	101.2	17.5
	0.5	127.2	10.0	127.4	13.2	126.8(126.8)	12.8(8.8)	139.2	85.9
	0.8	96.6	8.8	96.4	10.0	91.2(91.2)	33.6(8.6)	103.6	147.7
	平均値	107.9	7.7	107.6	9.1	105.8(105.8)	16.0(6.3)	114.6	83.7
6	0.2	366.4	110.9	365.4	106.2	364.4(364.4)	33.2(34.1)	380.6	654.9
	0.5	474.0	304.2	478.2	319.6	465.8(465.8)	259.6(195.9)	529.0	3412.6
	0.8	345.6	213.6	347.4	224.7	327.4(327.4)	850.5(155.3)	377.2	3952.7
	平均値	395.3	209.5	397.0	216.8	385.8(385.8)	381.1(128.4)	428.9	2673.4
7	0.2	—*	—	—	—	1364.5(—)	848.4(—)		
	0.5	—	—	—	—	1754.0(—)	5017.9(—)		
	0.8	—	—	—	—	— —	— —		
	平均値								

n=4, 5, 6 は d=0.2, 0.5, 0.8 で各5個ずつ発生した関数の平均値.
n=7 は2個ずつの関数の平均値. ( ) 内は全体否定で実行した結果.
*否定項数（配列上限 20,000, 実行領域 8 MB）が over flow で不可.

19,501 個, FUN 2 で 9,715 個および FUN 3 で 13,048 個生成する. したがって, これらを拡大に使用する MINI や MINI-II には次第に負担となってくる. それに対して, 本方法はマップの限定領域だけで拡大操作を行うため, 真理値表濃度が低くなるにつれて, 一層有効になるものと考えられる.

表 3 は, 局所否定を本方法に用いることで, 全体否定を使う MINI などに比べ, どのような記憶量の節約が可能かを4値6変数関数で調べてみたものである.

確かに真理値表濃度 d が小さいところで, 局所否定を求める最大領域の体積はマップ全体の体積の数%であり, d=0.55 で約 30% になっている（表中②の欄）. また, 局所否定の項数も全体否定の項数に比べ, d の小さいところで数 %, d=0.55 で約 30% になっている.

また, われわれは本方法において, 局所否定のかわりに全体否定を使用した場合のデータも載せておいた（表1の MINI-LN 中（ ）で表示）.

全体否定を使っても, 解の項数は当然変わらないが, 計算時間の方は, MINI, MINI-II および A5 アルゴリズムのどれよりも短縮されている.

ここで, ついでに手数について述べておく.

MINI-LN では, 拡大から整形まで一回手順が実行されると, その手数はごくおおまかに考えると $q_F q$ に比例する. ただし, q_F は今考えている時点での F の項数, q は $F \vee D$ の項の数 $q_F + q_D$ にほぼ等しく（はじめは $q_F = q$, $q_D = 0$）, 最初に F に与えた関数の項の数である.

そこで, q 個の項からなる関数が r 個ずつ必須主項あるいは擬似必須主項として検出されたり, あるいは冗長な項のため取り除かれて減って

表 2 各種関数での比較結果（四値）
Table 2 Minimization results of various functions with 4-valued inputs.

関数名	入力数	出力数	発生項数	MINI 解の項数	MINI Time（秒）	MINI-II 解の項数	MINI-II Time（秒）	MINI-LN 解の項数	MINI-LN Time（秒）	備考
MLP_4	8	8	225	91	5	93	6	92	6	
SQR_8	8	16	255	160	14	161	27	158	37	
SYM_{12}	12	1	3,498	97	32	98	33	94	38	
FUN 1	10	1	300	298	4,668	300	6,111	298	23	0.15
FUN 2	25	1	250	245	4,279	245	4,906	245	759	0.49
FUN 3	50	1	200	196	15,170	196	16,406	196	328	0.33

上位3例の入力数は二変数ずつ一変数に割り当てた. ただし, MLP_4 は同位桁を同一変数に割り当てた. 部分濃度, FUN 1 : (d_1, d_2, d_3, d_4)=(0.5, 0.2, 0.2, 0.1), FUN 2 : (d_1, d_2, d_3, d_4)=(0.4, 0.3, 0.2, 0.1), FUN 3 : (d_1, d_2, d_3, d_4)=(0.45, 0.2, 0.35, 0.0) で実行. 備考は真理値表濃度を表す.

表 3 局所否定と全体否定の比較（四値6変数関数）
Table 3 Comparison of local negation and whole negation in the case of 4-valued 6-variable functions.

真理値表濃度 d	① CM_{max}	② ①/C_u	③CM_{max}内の否定項の最大数	④ 全体否定 $\bar{f}$ の項数	⑤ ③/④	MINI 解の項数	MINI T(秒)	MINI-LN 解の項数	MINI-LN T(秒)
0.05	7.6	0.002	2.4	386.4	0.006(0.0)	153.2	10.3	153.2	2.1
0.15	37.6	0.009	10.4	724.0	0.01 (0.0)	315.2	71.8	313.6	15.5
0.25	115.2	0.03	30.0	858.8	0.03 (0.0)	405.6	154.1	405.2	58.8
0.35	295.2	0.07	85.2	901.8	0.09 (0.0)	454.6	193.2	454.4	116.0
0.45	706.8	0.17	183.2	876.8	0.20 (0.0)	479.0	241.6	472.8	182.1
0.55	1231.2	0.30	296.2	829.8	0.36 (0.0)	471.0	268.8	461.4	272.3
0.65	1958.4	0.48	408.4	742.4	0.55 (0.0)	443.8	269.9	427.6	579.2
0.75	2764.8	0.68	474.0	607.6	0.78 (0.0)	389.2	235.1	364.8	1088.9
0.85	4096.0	1.00	418.2	418.2	1.00 (12.4)	296.4	151.8	270.2	320.9
0.95	4096.0	1.00	172.8	172.8	1.00 (242.6)	154.6	64.8	138.4	156.3

①：局所否定を求めた最大領域 CM_{max} の体積，②：CM_{max} とマップ全体 C_u の体積比率，③：CM_{max} 内の否定項の最大数，④：全体否定 $\bar{f}$ の項数，⑤：③と④の比．T：計算時間(秒)．数値は各真理値表濃度 d で5個ずつ発生した関数の平均値を示す．⑤の（ ）内は局所否定領域がマップ全体と一致した回数．マップ全体の領域と局所否定を求める最大領域は各変数がもつ1のビットの数を掛け合わせた体積で表し，マップ全体は 1111-1111-1111-1111-1111-1111 なので，4×4×4×4×4×4=4096 であり，局所否定の領域が 0001-0011-1000-1100-0111-1111 とすれば，1×2×1×2×3×4=48 となる．

いき，αq 個の解（ただし，$\alpha \leqq 1$）が F_c に求まったとすると，その手数は $\frac{(2-\alpha)q}{2}\cdot\left(\frac{\alpha q}{r}+1\right)q$ に比例する．解が求まるプロセスとして，

（a） q 個の項を F に与えて，一度の MINI-LN の拡大から整形までの手順の適用で，q 個の必須主項が得られるときが，手数が最も少なくて q^2 に比例し，

（b） MINI-LN の拡大から整形までの手順が適用されるごとに，F の q 個の項が1つずつ必須主項あるいは擬似必須主項と判定されて F_c に加わって，q 回 MINI-LN が適用されて，F_c に q 個の項からなる解が求まるときが，手数が最も多く $q^3/2$ に比例する．

その他の解が求まるプロセスの手数はこの間に入る．表1の場合，関数として，はじめに与えた項の数 q は，変数の数 n が 4, 5, 6 のとき，それぞれ（併合後の）平均値で 38.2, 153.1, 587.7 と1変数増えるごとに，約4倍* ずつ増加する．したがって，上のことから MINI-LN では1変数増えるごとに手数（計算時間）は，最小16倍から最大32倍の間の倍数ずつ増えていくといえる．MINI, MINI-Ⅱ や A5 アルゴリズムでもほぼこの傾向はあてはまる．

* n 変数 p 値入力の最小項の総数は p^n なので，真理値表濃度が一定の関数の項数も p^n に比例し，1変数増えれば p 倍になる．

5. おわりに

本論文では，多値入力二値出力関数を表現する論理式簡単化の一手法を提案し，基本的考え方が類似している MINI, MINI-Ⅱ と共にプログラムを作成し，四値入力関数で計算機実験を行った．その結果，本方法は，

（1） 真理値表濃度が0.5以下の関数（入力変数の数 $n=4\sim7$）では計算時間，解の項数の両面で優れている．この程度の真理値表濃度では局所否定の項数は，全体否定の項数に比べ数％から約30％にすぎない．したがって，$n=7$ で，MINI, MINI-Ⅱ では否定項が多すぎて解が求まらない場合でも，われわれの方法では求まり，局所否定を使った効果が確認できた．

（2） 多変数関数（$n=10, 25, 50$）においても，高速に解が求まるが，本論文では，その3例を示した．

（3） 解の項数は一番少なかった．これは必須主項，擬似必須主項をもつ関数の場合はもとより，もたない関数（表2中 SYM_{12}）の場合にも同じことがいえる．

10 変数を越えると，体積の大きい項（最小項を多く含む項）を多数含む関数では，どの方法でも非常に多くの時間と記憶容量を要するようになる．このための解決法として，多少，解の項数を犠牲にしてでも，分割手法を取り入れることを検討したい．

参考文献

1) 笹尾 勤：PLAの作り方・使い方，日刊工業新聞社（1986）.
2) 石川啓二，笹尾 勤，寺田浩詔：論理式簡単化アルゴリズム：A 5，電子通信学会論文誌，Vol. J 66-D, No. 1, pp. 41-48 (1983).
3) 宮腰 隆，松田秀雄：2値論理関数簡単化の一手法—部分マップ法について—，電子情報通信学会論文誌 D, Vol. J 71-D, No. 11, pp. 2259-2265 (1988).
4) Tirumalai, P. and Butler, J. T.: Analysis of Minimization Algorithms for Multiple-Valued Programmable Logic Arrays, *18th ISMVL*, pp. 226-236 (1988).
5) Dueck, G. W. and Miller, D. M.: Directed Search Minimization of Multiple-Valued Functions, *18th ISMVL*, pp. 218-225 (1988).
6) Hong, S. J., Cain, R. G. and Ostapko, D. L.: MINI: A Heuristic Approach for Logic Minimization, *IBM J. Res. Dev.*, Vol. 18, No. 5, pp. 443-458 (1974).
7) Sasao, T.: Input Variable Assignment and Output Phase Optimization of PLA's, *IEEE Trans. on Comput.*, Vol. C-33, No. 10, pp. 879-894 (1984).
8) Rudell, R. L. and Sangiovanni-Vincentelli, A. L. M.: ESPRESSO-MV: Algorithms for Multiple-Valued Logic Minimization, *CICC-85*, pp. 230-234 (1985).
9) Kuo, Y. S.: Generating Essential Primes for a Boolean Function with Multiple-Valued Inputs, *IEEE Trans. on Comput.*, Vol. C-36, No. 3, pp. 356-359 (1987).
10) Sasao, T.: An Algorithm to Derive the Complement of a Binary Function with Multiple-Valued Inputs, *IEEE Trans. on Comput.*, Vol. C-34, No. 2, pp. 131-140 (1985).
11) Malik, A. A., Brayton, R. K., Newton, A. R. and Sangiovanni-Vincentelli, A.: Reduced Offsets for Minimization of Binary-Valued Functions, *IEEE Trans. on Computer-Aided Design*, Vol. 10, No. 4, pp. 413-426 (1991).
12) 松田秀雄，宮腰 隆：論理式を分離加法形式で表現する一手法，情報処理学会論文誌，Vol. 33, No. 4, pp. 560-569 (1992).

付録 多変数関数の発生法

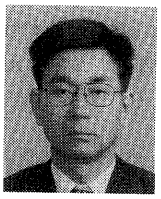
多値入力二値出力関数を手軽く発生させるために，われわれは次のようなプログラム[12]を考えた．例えば，四値入力の場合で説明するなら，各成分（各 part）に1が1個現れる，2個現れる，3個現れる，4個現れる（各成分のどの位置に1が現れるかは全くランダム）確率を d_1, d_2, d_3, d_4 として，成分濃度といおう．この成分濃度のとり方によって，項の大きさが指定でき，それを C_t 個発生させる．例えば，d_1, d_2, d_3, d_4 を 1, 0, 0, 0 とおけば最小項を発生し，0, 0, 0, 1 とすれば C_u が生成され，成分濃度を変えて種々の項が生成できる．FUN 1 は，$n=10$, $C_t=300$, $(d_1, d_2, d_3, d_4)=(0.5, 0.2, 0.2, 0.1)$ で発生した関数の一例である．

しかし，多変数関数（10 変数以上）になると，このプログラムで指定発生させた関数を直接用いたのでは，成分濃度が大きい（d_1, d_2 より d_3, d_4 を大きい値にする）と，時間がかかりすぎたり，否定項が多すぎたりして，MINI や MINI-II では求まらなかったり，逆に，成分濃度が小さい（d_1, d_2 を d_3, d_4 より大きい値にする）と，簡単化しても初めに与えた関数と同じであったりして，各方法の性能比較を行うのに適した関数が見出せなかった．

そこで，FUN 2 は，まず，$n=8$, $C_t=240$, $(d_1, d_2, d_3, d_4)=(0.4, 0.3, 0.2, 0.1)$ を発生する．次いで，これら $n=8$ の各項を $n=25$ まで拡張（各項に part 9 から part n まで 1111-1111-……-1111 を追加）する．さらに，$n=25$ で同一成分濃度により 10 個の項を追加発生し，項数 250 個とした関数である．

FUN 3 は，FUN 2 と同様，$n=8$, $C_t=190$, $(d_1, d_2, d_3, d_4)=(0.45, 0.2, 0.35, 0.0)$ で発生した各項を $n=50$ まで拡張した後，$n=50$ で同一成分濃度により 10 個の項を追加発生し，項数 200 個とした関数である．なお，表2中，備考で示した真理値表濃度は関数を分離加法形[12]に直して求めたものである．

（平成4年12月14日受付）
（平成5年9月8日採録）



宮腰 隆

昭和22年生．昭和45年富山大学工学部電気工学科卒業．昭和47年同大学院工学研究科修士課程修了．同年同大学工学部電気工学科助手．現在，同電子情報工学科助手．主として論理回路，多値論理に関する研究に従事．電子情報通信学会会員．

松田 秀雄（正会員）

昭和34年富山大学工学部電気工学科卒業．昭和46年大阪大学大学院工学研究科博士課程修了．同年4月より富山大学工学部電気工学科（現在，電子情報工学科）に勤務，現在に至る．基礎情報担当．助教授．工学博士．主として論理回路の自動設計の研究に従事．生体電気工学にも興味をもつ．電子情報通信学会会員．

畠山 豊正

昭和39年富山大学文理学部理学科卒業．昭和41年金沢大学大学院修士課程修了．同年金沢工業大学工学部助手．昭和43年富山大学工学部助手．平成2年同教授．工学博士．現在，生体の内的制御機構および細胞の電気生理的現象と電気工学的応用に関する研究に従事．電子情報通信学会，日本生物工学会，計測自動制御学会各会員．