論理関数のプライムインプリカントを計算機で導出する一手法について

松田秀雄・宮腰　隆

緒　言

論理設計を行なう場合，与えられた関数のプライムインプリカント（以後PIと略記）を求める必要がある。これを計算機で求めるには従来から，Quine-McCluskey 法がよく用いられてきたが，昨年，筆者らはカルノー図による方法に立脚した関数変換法を提案し，他の方式との比較検討を行なった。今回は，これを更に効率的に改善し，計算時間を½程度にまで短縮できる縮小カルノー図法について，原理，プログラム化，及び計算結果について述べる。

縮少カルノー図法はカルノー図を次々と小さくしていって，繰返し的にPIを見出す方法なので，まさしく，計算機向きアルゴリズムであるといえる。

1．理　論

1.1　許容キューブ

論理関数Fが図1のようにカルノー図で与えられているとする。但し，ここで○印のセルでtrue，無印のセルでfalseを示す。又，各セルの数字は座標ベクトル(x_1, x_2, x_3, x_4)で1の成分の少ない順に，もし同一の1の数を持つ場合には2進数字とみなして小さい順に並べた順位番号を表わす(表1参照)。

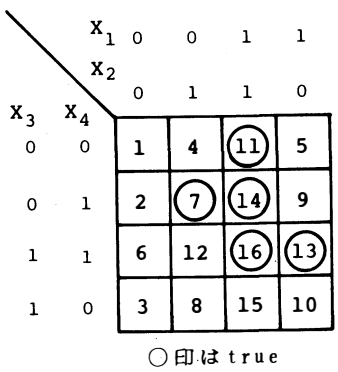

図1　関数Fの例

$x_1\bar{x}_2x_4$ のように変数の積項をキューブというが，4変数の場合，81個（一般にn変数で3^n）個のキューブがある。このうち，とくに否定形の変数を含まない変数の積項で表わされるキューブを許容キューブという。4変数の場合，許容キューブは x_1, x_2, x_3, x_4, x_1x_2, x_2x_3, ……, $x_1x_2x_3x_4$ の15個と $f=1$，つまりカルノー図全体の計16（一般に n 変数で2^n）個あり，いくつかの例を図2に示す。上述の番号付けのもとでは，許容キューブに共通な性質としてセル16（座標(1, 1, 1, 1)のセル）を含んでいることがいえる。又，許容キューブはセル番号で表わせ，この番号が，更にキューブの大きさの順位をも示しているようにできる。例えば許容キューブx_4は図3のように {2, 6, 7, 9, 12, 13, 14, 16} の各セルからなるが，このうち最小のセル番号2に注目して“セル16に中心をもつ，セル2を通るキューブ”といい，記号P_{16}(2)と表わす。又，許容キューブx_1x_2は {11, 14, 15, 16} のセルからなるが，最小のセル11に注目して，“セル16に中心をもつ，セル11を通るキューブといえ，記号 P_{16}(11)で示す。このように約束するとカルノー図全体，すなわち2^n個のセルからなる許容キューブは記号 P_{16}(1)で表わされて最大，セル16だけからなるキューブ $x_1x_2x_3x_4$ が記号P_{16}(16)となり，最小の許容キューブで，他の許容キュ

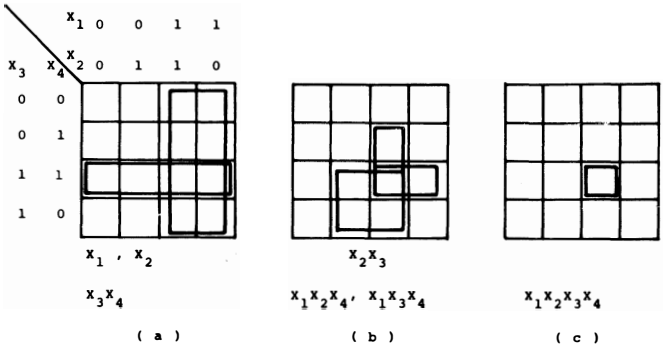

図２　許容キューブの例

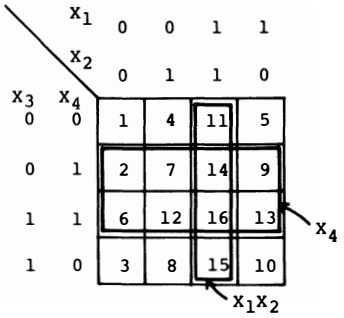

図３　セル番号と許容キューブとの関係

ーブはこれらの間の大きさで，セル番号に応じて，順次等しいか，あるいは小さくなる。

1.2 原　　理

図１の関数Ｆは2^n次元ブールベクトル

F ＝(1, 2, 3, 4, 5, 6, ⑦, 8, 9, ⑩, ⑪, 12, ⑬, ⑭, 15, ⑯)

としても表現できる。あるキューブq_iも2^n次元ベクトルで表わしたとき，q_iとＦとの間に

（関数F）⋂（キューブq_i）＝（キューブq_i）　　（１）

が成り立つなら，Fのインプリカントである。ここで記号⋂はFとq_iの各成分ごとの論理積演算をとるもので，式(１)を関数Fとキューブq_iとの論理積と呼ぼう。q_iがインプリカントで他のどのPIにも含まれなければＦのPIであると判定できる。このための操作をプライムインプリカントの包含関係の照合という意味で“PIの照合”と略記する。

さて，以上の準備のもとで縮小カルノー図法の原理を図１の関数Fの例で説明する。

まず，許容キューブを大きいものから$P_{16}(1)$，$P_{16}(2)$，……と順次発生させ関数Ｆとの論理積をとる。式(１)が成り立てばPIの照合を行なう。本例では$P_{16}(13)$，すなわち，$x_1\,x_3\,x_4$｛セル⑬，⑯｝と$P_{16}(14)$，すなわち，$x_1\,x_2\,x_4$｛⑭，⑯｝との２つのPIが見つかる。許容キューブでないPI，$x_2\,\bar{x}_3\,x_4$｛⑦，⑭｝と$x_1\,x_2\,\bar{x}_3$｛⑪，⑭｝については次のようにある縮小図の許容キューブとして求める。

セル番号14の座標は（1，1，0.1）である。ここで０の成分x_3に注目して，カルノー図上$x_3=0$の部分を考える。図４の太線内はこれを示し，もしx_3の成分を無視すると，x_1，x_2，x_4の３変数だけで１つのカルノー図を構成する（図４点線内）。これを縮小カルノー図（略して，縮小図）14という。この図での許容キューブは１，x_1，x_2，x_4，$x_1\ x_2$，……，$x_1\ x_2\ x_4$でいずれもセル14（座標（1，1，1））を含む。上記同様，許容キューブが含む最小のセル番号と中心セル14に注目して，これらの許容キューブはそれぞれP_{14}(１)，P_{14}(５)，P_{14}(４)，P_{14}(11)，……，P_{14}(14)と表わす。

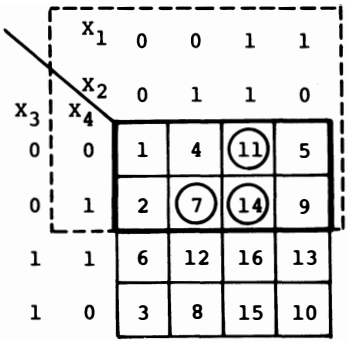

図４　縮小カルノー図の例

これらの縮小図14の許容キューブを用いて，この図でのPIを見出すには式(１)にならって，

（F_{14}）⋂（許容キューブ$P_{14}(j)$）＝（許容キューブ$P_{14}(j)$）　　（２）

を順次，j＝1，2，4，5，7，9，11，14ととって調べればよい。ここでF_{14}は関数Fの縮小図14への縮小で，この図だけで定義されていて，ここではFと同じ真理値をとる関数である。いまの例ではP_{14}

(7)とP_{14}(11)，すなわち，$x_2\,x_4$｛⑦，⑭｝と$x_1\,x_2$｛⑪，⑭｝とがこの図でのPIとなる。もし，PIがセルの集合として表わされているなら，これら，｛7，14｝，｛11，14｝はすでに得られている｛13，16及び｛14，16｝と直接PIの照合ができ，新しいPIとしてつけ加えられる。又PIが積項$x_2\,x_4$，$x_1\,x_2$で表わされているなら，これに縮小図14を作る上で省略された変数$\bar{x}_3$を付加して，$x_2\,\bar{x}_3\,x_4$，$x_1\,x_2\,\bar{x}_3$になおしてから，すでに得られているPIと比較されねばならない。図1の例ではただ1つの縮小図ですべてのPIが求まったが，一般には，このように簡単でなく，縮小図14にならって，各セルiでの縮小図が必要となる。縮小図iの許容キューブを$P_i(j)$と表わし，この図への関数Fの縮小をF_iとすれば，論理積

（関数F_i）⋂（許容キューブ$P_i(j)$）＝（許容キューブ$P_i(j)$）　　(3)

の成否を調べれば，縮小図iのインプリカントが求まる。

いま，セル番号iを16，15，…，2，1とかえないがら，各縮小図iごとに大きい許容キューブから論理積式(4)をとっていけばPIとなるすべてのキューブが見つかることがわかる。なお，縮小図16はもとのカルノー図そのものになることを一言注意しておく。

1.3 縮小カルノー図法の諸性質

n変数の場合でも手続きは上例と同様，次のように行なわれるものとする。すなわち，縮小図は大きいセル番号のものから順次求め，各縮小図ごとに行なわれる式(3)の論理積は大きな許容キューブから順序立ててなされる。

縮小カルノー図法の論理積数を求めるために次の2つの補題を上げる。

〔補題1〕　すべての縮小図の許容キューブの総数は3^nである。

（証明）　セルiの座標ベクトル（x_1，x_2，…，x_n）で，0をとる成分の数をk個とする。縮小図iは（$n-k$）変数のカルノー図であるから許容キューブの数は$2^{(n-k)}$個あり，このようなセルは${}_nC_k$個ある。kは0，1，2，…，nとかわり得るので，縮小図の許容キューブの総数は

$$\sum_{k=0}^{n} {}_nC_k\ 2^{n-k} = 3^n \qquad (4)$$

となる。

〔補題2〕　すべての縮小図の許容キューブの集合Pからキューブ全体の集合Qへの1対1写像が存在する。

（証明）　縮小図iの許容キューブはもとのカルノー図のキューブでもある。この対応を写像$f: P \to Q$とおく。一方，あるキューブを$\bar{x}_{i1}\,\bar{x}_{i2}\,\bar{x}_{i3}\cdots\bar{x}_{ik}\,x_{ip1}\,x_{ip2}\cdots x_{ipl}$とする。このとき，$x_{i1}=x_{i2}=\cdots=x_{ik}=0$，その他の成分を1としたベクトル座標のセルを$i$とするなら，このキューブに縮小図$i$の許容キューブ$x_{ip1}\,x_{ip2}\cdots x_{ipl}$を対応させる写像$g: Q \to P$とおく。$f$と$g$は互に逆写像で一意的である。故に1対1写像である。

全キューブと関数との論理積をとるなら，すべてのPIが求まることは明らかである。〔補題2〕によって，これは全縮小図の許容キューブと関数との論理積をとることと同じである。よって，次の定理が成り立つ。

〔定理〕　縮小カルノー図法ですべてのPIを見出すに必要な論理積数は高々3^nである。

ところで縮小カルノー図法ではいくつかの性質があって，論理積数を3^nよりかなり小さくできる。例えば，(i)セルiが無印のセルなら，縮小図iについて考える必要はない。又，(ii)セルjが無印のセルなら，セルjを通る許容キューブ$P_i(j)$と関数F_iとの論理積は不用である，などである。これらの諸性質を使って，次にアルゴリズムを書き表わす。

2．アルゴリズムとプログラム化

2.1　アルゴリズム

i と j はセル番号で最初 $i=2^n$，つまり最大のセル番号としておく。PITABLEはその時点までにすでに得られているPIの表で，初め空 ϕ としておく。

1．F のセル i が○印のセルなら縮小図 i を発生し，$F_i'=F_i$，$j=1$ として手順2へいく。無印なら手順6へいく。

2．F_i' のセル j が無印のセルなら，手順5へいく。○印のセルなら，

3．次式を調べる。

$$F_i \cap P_i(j)=P_i(j)$$

成り立てば，$P_i(j)$ に属する F_i' の○印のセルを無印にかえて手順4へいく。成り立たなければ，F_i' のセル j を無印にして手順5へいく。

4．$P_i(j)$　p となる p がPITABLEにあるかどうか調べる。あれば手順5へいく。なければ，$P_i(j)$ をPITABLEにつけ加える。

5．F_i' の全部のセルが無印にかわったら，7へいく。そうでなければ次へいく。

6．$j<i$　ならば，$j=j+1$　として手順2へいく。$j\geqq i$ ならば次へいく。

7．$i>1$　ならば，$i=i-1$　として手順1へいく。$i\leqq 1$ ならば　STOP。

なお，縮小図 i ではセル番号は1から i までとびとびの値をとるが，ここでは便宜上，連続した整数値をとるように表わしている。

このアルゴリズムでは，関数Fがただ1個のセルだけで○印をもつなら，論理積数は1回だけですむ。又，全部のセルが○印なら，縮小図ごとに論理積数を1回づつ，計 2^n 回でよい。極端な例とはいえ，3^n よりはるかに小さい。その他の関数については，後節で他の方法と比較しながら，論理積数の減少の様子を示す。

2.2　プログラム化

表1は2進ベクトルを1の成分の少ない順に並べたものであるが，これらのベクトルによって，各縮少図，及び，それらの許容キューブが簡単に得られ，更にこのキューブの発生過程が関数と論理積をとる操作もかねているのでプログラム化に非常に好都合であることを以下に示そう。

まず，縮小図16はカルノー図全体に一致するので，表1の全セルからなる。縮小図15はセル15の座標が(1，1，1，0)から，x_4 が0となるセルを表1で，セル15から上に向って探し出せば得られる。すなわち，{1，3，4，5，8，10，11，15} のセルの集合がこれにあたる。一般に縮小図 i はセル i の座標 $(x_{1i}, x_{2i}, x_{3i}, x_{4i})$ で0となる成分がやはり0となっているセルを表1でセル i から上向きに探し出せば得られる。

次に，縮小図 i のセル j を通る許容キューブについて考えよう。例えば，$P_{16}(1)$ は表1のすべてのセルからなるものを発生する。$P_{16}(2)$ はセル2の座標が(0，0，0，1)であるから，$x_4=1$ となっているセルを表1でセル2よりFにたどって探し出す。

表1　ベクトルの並べ方

セル番号	2進ベクトル x_1	x_2	x_3	x_4
1	0	0	0	0
2	0	0	0	1
3	0	0	1	0
4	0	1	0	0
5	1	0	0	0
6	0	0	1	1
⑦	0	1	0	1
8	0	1	1	0
9	1	0	0	1
10	1	0	1	0
⑪	1	1	0	0
12	0	1	1	1
⑬	1	0	1	1
⑭	1	1	0	1
15	1	1	1	0
⑯	1	1	1	1

すなわち，{2，6，7，9，11，13，14，16} のセルの集合がこれにあたる。更に$P_{16}(j)$については，セルjの座標で1をとる成分がやはり1となっているセルを表1で，セルjから下にたどって探し出せば求まる。一般に縮小図iのセルjを通る許容キューブ$P_i(j)$も，いまの操作で表1の全セルを考えた代わりに，縮小図iに含まれるセルだけについて考える以外は全く同じようにして求まる。例えば，$P_{14}(2)$はセル2の座標が(0，0，0，1)であるから$x_4=1$となっているセルを表1で，セル2から下にたどって，縮小図14に含まれるセル {1，2，4，5，7，9，11，14} の中から選び出せばよい。すなわち，{2，7，9，14} が$P_{14}(2)$のセルとなる。

表1のセル番号の欄は真理値表でもあり，図1の関数Fの例が示してある。いま許容キューブ$P_i(j)$を発生し，それが無印のセルを1個でも含めば論理積は不成立で捨てられる。又，$P_i(j)$が○印のセルのみからなればインプリカントで，“PIの照合”を行なって，他のPIに含まれなければPI として採用される。

本例では$P_{16}(7)=$ {⑦，12，⑭，⑯}，$P_{16}(11)=$ {⑪，⑭，15，⑯} は捨てられ，$P_{16}(13)=$ {⑬，⑯}，$P_{16}(14)=$ {⑭，⑯}，$P_{16}(16)=$ {⑯} はインプリカントである。このうち，$P_{16}(16)$は他に含まれるので除かれ，$P_{16}(13)$，$P_{16}(14)$がPIとして採用される。更に，$P_{14}(7)=$ {⑦，⑭}，$P_{14}(11)=$ {⑪，⑭} が○印のみからなるので，インプリカントであり，かつPIであることがわかる。

3．計　算　結　果

3.1　比較する他の方式

プライムインプリカントを計算機で求めるには従来から，Quine-McCluskey 法（本節では単にMcCluskey 法と略記）がよく知られている。これはT（true）をとるセルのみからなる最小のキューブからはじめて，2セル，4セル，……と次第に大きなキューブに結合していけるかどうかを見る方法で，単位操作は2つの3進n次元ベクトルで表わされたキューブの間の処理となる。ここではこの単位操作のことを単に論理積ということにする。

関数変換法はまず，関数Fと許容キューブとの論理積（式(1)参照）をとって，PI を見出す。但し，ここでいう許容キューブは狭義のもので本論文の縮小図16の許容キューブ$P_{16}(j)$に相当する。許容キューブ以外のキューブでPIとなるものは，各セルごとに定義される変換T_iを関数にほどこした後，やはり許容キューブと式(1)の論理積をとって見出す方法である。この方法で，関数変換のために要する手数を省略できるよう工夫し効率を高めたのが縮小カルノー図法である。関数変換法と縮小カルノー図法とは原理的に同じなので，論理積数は変わらない。又，関数の形あるいは変数の数による計算時間の影響は両者ともほぼ同じなので，主として，以下の考察での比較は縮小カルノー図法とMcCluskey 法との間で行なう。

関数F_{3k}を変数の数nが$n=3k$($k=1$，2，3，……) で，それを構成するすべてのキューブがk個の肯定，否定，任意変数からなる論理積項で表わされるものとしよう。F_{3k}は$n(=3k)$変数関数のうちで最も多数のPIをもち，$n=9$のとき，1680個にもなる。表2はF_{3k}について，各方式での計算結果を表で示したもので，縮小図法が非常にすぐれており，9変数で約35%関数変換法の計算時間を短縮している。

さて，上ではF_{3k}によって，縮小図法の優位性を示したのであるが，実は各方式とも計算時間は関数の形に依存するもので，図5はこれを示すための1例である。$n=9$の場合，カルノ図を左右に両分すると，256個のセルからなるキューブが2つできる。ついで上下に2分すると，128個のセルのキューブが4個できる。これを又左右に両分……，上下に両分……していくと次第に小さくなり，最後に1個のセルのみからなる最小のキューブになる。ここで，各分割ごとに等しい数の false のキュー

表２　肯定，否定，任意変数がｋ個ずつからなる積項で表わされる関数（F_{3A}）

		論理積数	計算時間 M　S　MS	PI数
n ＝ 3	縮小カルノー図法	9	31	6
	関数変換法	9	—	6
	McCluskey 法	9	42	6
n ＝ 6	縮小カルノー図法	165	847	90
	関数変換法	165	1 － 393	90
	McCluskey 法	4200	2 － 487	90
n ＝ 9	縮小カルノー図法	3528	1 － 29 － 774	1680
	関数変換法	3528	2 － 17 － 97	1680
	McCluskey 法	2259684	15 － 27 － 249	1680

ブと true のキューブにわけ，それぞれカルノー図上交互に配置するようにすると，９つの関数ができる。図５の横軸はキューブの大きさで，これらの関数を区別するよう目盛ったもので，たて軸はそれらの計算時間を示す。これからわかるように小さなキューブの関数なら，McCluskey法の方が早くPIが求まる。しかし，キューブが大きくなるにしたがって，計算時間が大きくなり，結局はほぼ一定の特性を示す縮小図法に比べ不利となる。

次に計算時間が関数の形の影響を受けることを示す第２の例として，図６を上げる。これは全体のセルの中で true のあらわれる確率をいろいろ変えて，各方式の計算時間を対比させたものである。それぞれの点は横軸の目盛の確率でtrueのセルが表われるよう乱数を使って発生させた８変数の関数５個の平均計算時間である。いずれの方式もtrue のセルの割合いと共に計算時間も増加するが，その増加の傾向はMcCluskey法で著しい。この場合も，McCluskey 法の有利な場合と，縮小図法の有利な場合のそれぞれの true のセルの割合い範囲がある。

このように，関数のPIとなるキューブの大きさ，Tのセルの割合いで各方式の計算時間が異なってくるので，多数の関数の平均値でもって計算時間を算出するのが望ましい。表３はこれを示したもので，４～７変数については無作為に選んだ 100個の関数の平均値，８変数は50個

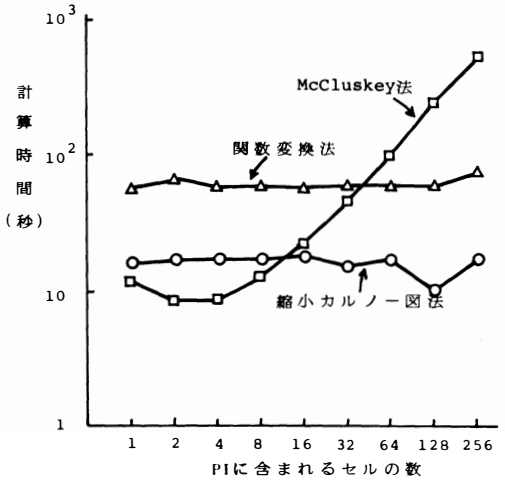

図５　プライムインプリカントの大きさと計算時間との関係（ｎ＝９，true のセルの割合い0.5）

の関数の平均値，9変数は20個の関数の平均値である。McCluskey 法より関数変換法が有利であり，又，関数変換法より縮小図法の方が更に½程度の計算時間で求まることがわかる。

縮小図法（関数変換法も同じだが）の論理積数の上限は 3^n であるが，実際にはこの数よりかなり小さいことを示すために図7をあげる。これは1.3で述べた性質(i)，及び(ii)，それにアルゴリズムのステップ5の処理の効果によるもので，非常に効率化に役

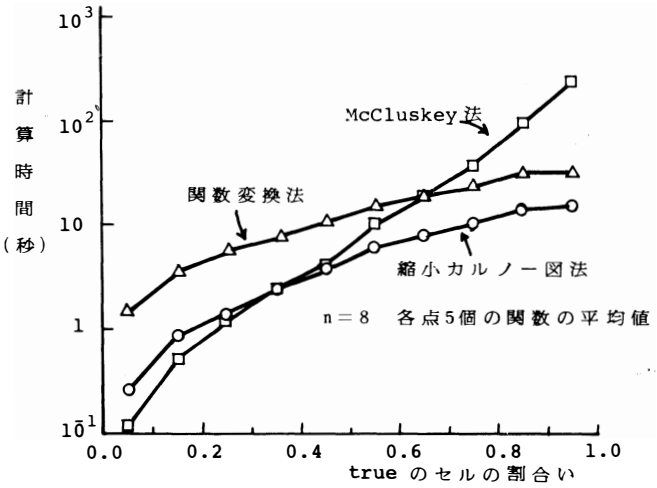

図6　True のセルの割合いと計算時間

表3　平均計算時間の比較

変数	McCluskey 法			関数変換法			縮小カルノー図法		
	M	S	MS	M	S	MS	M	S	MS
4			35			46			28
5			166			185			93
6			879			765			364
7		6	159		3	287		1	525
8		41	767		15	26		6	847
9	4	45	490	1	12	331		32	7

立っている。これに対し，McCluskey 法では論理積数は 3^n より大きく，変数の数 n の増加と共に膨大な数になっていく。

結　　言

縮小カルノー図法は関数変換法で関数変換のための操作を省略できるよう工夫したもので，½程度の計算時間で，論理関数のすべてのプライムインプリカントを自動的に求めることができる。この方法は従来からもっとも標準的な方法として知られてきたQuine-McCluskey 法にくらべても一段と有利である。表1の順に並べた2進ベクトルを用いると，縮小図と許容キューブが計算機上簡単に発生でき，プログラム化が容易である。又，占有記憶量もQuine-McCluskey 法のようにPI表以外に中間段階のキューブのための表を必要としないので少くてすむ。

アルゴリズムは計算機向きに述べたが，もちろん手計算にも利用でき，数枚のカルノー図を用意することによって，すべてのプライムインプリカントを規則的に得ることが可能である。

なお，使用計算機は本学計算センター（FACOM230－45S）のものである。データをとる際ご協力下かったセンターの皆様に感謝申し上げる。

参　考　文　献

1） E. J. McCluskey, Jr.：Bell Syst. Tech. J., **35**, 1417,（1956）
2） M. Karnaugh：AIEE Trans. Commun. Electron., **72**, 593,（1953）
3） 宮腰，松田，野末：富山大学工学部紀要，**30**，8，（1979）
4） 宮腰，松田：信学技報，**AL 78**，17，（1979）

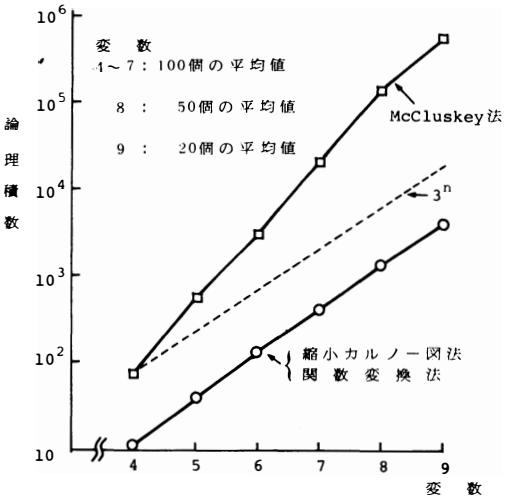

図7　変数の数と論理積数との関係

1979年1月　電子通信学会オートマトントと言語研究会（大阪大学）で一部発表

A Computer Algorithm for Generating All the Prime Implicants of Logical Functions

Hideo MATSUDA and Takashi MIYAGOSHI

In this paper we propose a Karnaugh submap method which determines prime implicants of a logical function by the computer. Reducing a Karnaugh map to smaller one's repeatedly, we obtain prime implicants by logical product of the logical function and the permissible cube of each submap.

This algorithm is very efficient because the number of logical product is reduced by a few properties. This algorithm finds out all the prime implicants of a given function in less computing time than a half of the function transformation method which has already been reported in the previous number of this bulletin.

（1979年10月31日受理）