２分決定グラフ（BDD）使用による主項生成

5B－4

陳　紀成*　伊藤　貴　凌　暁萍　後藤　公雄
神奈川工科大学情報工学科

１．はじめに

本報告は論理関数の全主項の生成に２分決定グラフ（BDD）を用いた方法を述べる．この方法では，積和形の論理関数をＢＤＤ展開し，得られたリテラル列にコンセンサス法を適用する，なお，アルゴリズムを実現するプログラムにはC言語を用い，ビット処理手法を使用している．

２．アルゴリズム

本アルゴリズムでは最小数ノードのＢＤＤ展開を行うよう配慮してある．このアルゴリズムをステップごとに述べる．なお，プログラム化した場合ビット演算処理が行われることを配慮し，関数および変数の表示はすべて１６進数を用いる．

［ステップ１］ＢＤＤトリーのルートに，与えられた積和形関数ｆの１６進表示を付与する．

［ステップ２］関数ｆの真理値表を作成し，展開変数x_iまたは$\bar{x}_i$の１６進数の値を求める．

［ステップ３］与えられた展開変数x_iに基づいて，BDD 展開を行う．この展開は終端０または終端１が得られるまで各ノードごとに行う．この場合，親ノードから左の子ノードへのパス上には$\bar{x}_i$の，また右の子ノードへのパス上にはx_iの１６進表示リテラルを付記する．

［ステップ４］BDD 展開で生じた冗長ノードの除去によりＢＤＤ縮約を行う．

［ステップ５］ここで各終端１から出発してパスをルートへ向かって遡行し，各パス上のリテラルを連接させてリテラル列を作り，これを主項候補とする．この方法を，今後，直進遡行法と呼ぶ．

［ステップ６］得られたすべてのリテラル列をコンセンサス法によって簡単化する．

本アルゴリズムを式（１）の例を用いて説明する．

$$f = x_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_3\bar{x}_4 \qquad (1)$$

ステップ１で式（１）の関数を最小項集合S_fで表すと，式（２）となる．

$$S_f = \sum(1, 2, 3, 6, 7, 9, 14, 15). \qquad (2)$$

これを，16 進表示で表す C2CE となる．

ステップ２では，これをそれぞれの展開変数を１６進表示すると，次の式（３）が求まる．

$x_1 \approx 1^1 \approx$ FF00H, $\bar{x}_1 \approx 1^0 \approx$ 00FFH, (3a)

$x_2 \approx 2^1 \approx$ F0F0H, $\bar{x}_2 \approx 2^0 \approx$ 0F0FH, (3b)

$x_3 \approx 3^1 \approx$ CCCCH, $\bar{x}_3 \approx 3^0 \approx$ 3333H, (3c)

$x_4 \approx 4^1 \approx$ AAAAH, $\bar{x}_4 \approx 4^0 \approx$ 5555H, (3d)

ステップ３では，ステップ１で与えた関数f の 16 進表示 C2CE を，予測法によって求めた最適展開変数順序x_3，x_2，x_4，x_1 にしたがって BDD 展開を行う［１］．　ステップ４はステップ３の結果を縮約している．この縮約によって得られた結果を図１に示す．

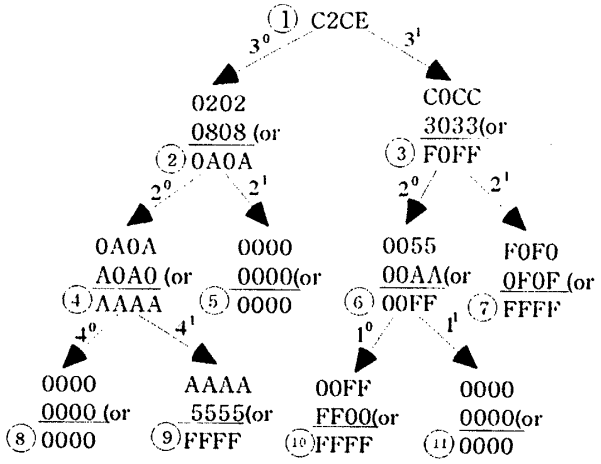

図１　BDD　展開

ステップ５では，直進遡行法によって求めたすべてのリテラル列の集合（$4\bar{2}\bar{3}$，$\bar{1}\bar{2}3$，23）から，コンセンサス法によって（$4\bar{2}\bar{3}$，23，$\bar{1}\bar{2}4$，$\bar{1}3$）を求める．この過程を図２（ａ）に示す．

Generation of Prime Implicants by Using BDD (Binary Decision Diagrams)
Department of Information and Computer Sciences, KANAGAWA Institute of Technology

$4\bar{2}\bar{3}$		
$\bar{1}\bar{2}3$	$\bar{1}\bar{2}4$	
23	−	$\bar{1}3$

(a) 1回目

$4\bar{2}\bar{3}$			
23	−		
$\bar{1}\bar{2}4$	−	$\bar{1}34$	
$\bar{1}3$	$\bar{1}\bar{2}4$	−	−

(b) 2回目

図2　コンセンサス法

上述したアルゴリズムのステップ5とステップ6を次のような方法で置き換えてみる．すなわち，各終端1からルート方向に遡行し，各パスのリテラルを連接してリテラル列を生成する．親ノードに達するごとに，他の子ノードから遡行して同様に生成されてきたリテラル列との間にコンセンサス法を適用する．この手続きをさらに上位のノードに到達するごとに繰り返す．このようにして主項生成する方法をレベル遡行法と呼ぶ．上述した事例についてこの模様を図3に示す．

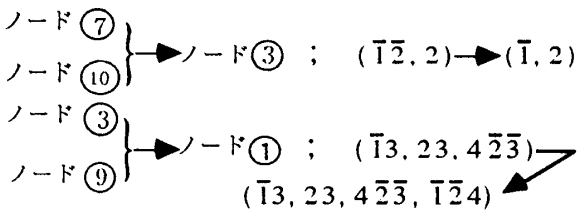

図3　レベル遡行法におけるコンセンサス

3．実験結果とその検討

本報告の BDD使用による主項生成アルゴリズムを用いてC言語プログラムを直進遡行法とレベル遡行法について作成した．SUN sparc station 5（メモリ 64 MB）上でこれを実行させ，主項の生成のCPU時間を測定した．この結果を図4に示す．

図4の縦軸は CPU時間を秒で表し，横軸は最小項の濃度を表す．この測定は各変数（8～15変数）をパラメータとし，最小項番号を，与えられた濃度について必要な数だけ乱数で発生させて入力した．これを各濃度について10回測定し，結果を平均した．図4より，次の事柄が判明した．

（1）低濃度，または低変数の論理関数については，レベル遡行法の　CPU時間より直進遡行法のCPU時間のほうが小さい．これはレベル遡行法にするときは，リテラル列の削除が行いにくくなるので，CPU時間を節約することができないためと思われる．

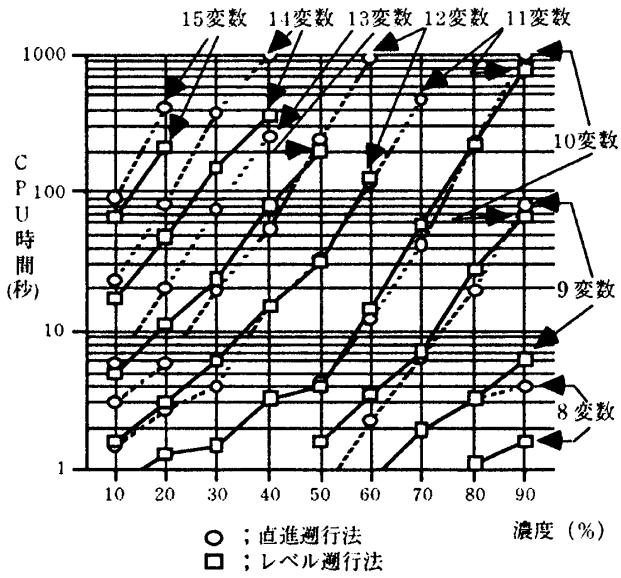

図4　直進遡行法とレベル遡行法のCPU時間比較

（2）一方，高濃度，または高変数の論理関数については，直進遡行法のCPU時間よりレベル遡行法の CPU時間のほうが小さい．これは，レベル遡行法のほうが膨大なリテラル列を削除でき，CPU時間を大きく節約できるからである．例えば10変数かつ90%濃度については，直進遡行法の CPU時間は 952.2 秒で，レベル遡行法の CPU 時間は66秒である．後者の時間は前者の時間のほぼ1／14となる．

（3）変数が高ければ高い程，また濃度が低ければ低い程，レベル遡行法の CPU時間は直進遡行法の時間より小さくなる．例えば，8変数では濃度90% から前者の CPU 時間は後者の CPU 時間より小さい．一方，11 変数以上の場合，濃度 10%からこの傾向が現れる．

4．むすび

本報告により，BDD を用いた全主項生成法としては，直進遡行法よりもレベル遡行法の方が高変数かつ高濃度の入力関数に対して優れていることが分かる．

5．参考文献

[1] Rendal E. Bryant, "Graph-Based Algorithms for Boolean Funtion Manipulation", IEEE Trans. Comput., Vol. C-35, PP.677-691, 1986.
[2] 後藤公雄：詳解・ディジタルＩＣ回路（上）ラジオ技術社（昭和５６年）．