HOKKAIDO UNIVERSITY

Title	論理関数の共有二分決定グラフによる表現とその効率的処理手法
Author(s)	湊, 真一; 石浦, 菜岐佐; 矢島, 脩三
Citation	情報処理学会論文誌, 32(1), 77-85
Issue Date	1991-01-15
Doc URL	http://hdl.handle.net/2115/47491
Rights	ここに掲載した著作物の利用に関する注意 本著作物の著作権は情報処理学会に帰属します。本著作物は著作権者である情報処理学会の許可のもとに掲載するものです。ご利用に当たっては「著作権法」ならびに「情報処理学会倫理綱領」に従うことをお願いいたします。
Type	article
File Information	63_joh32_77.pdf

Instructions for use

Hokkaido University Collection of Scholarly and Academic Papers : HUSCAP

論理関数の共有二分決定グラフによる表現とその効率的処理手法†

湊 真一†† 石浦 菜岐佐†† 矢島 脩三††

論理関数を効率よく表現し，論理演算を高速に行うことは，論理設計支援システムを実現する上で重要な問題である．Akers や Bryant が提案した二分決定グラフによる表現法は，入力変数の順序を固定することにより論理関数を一意に表すことができ，また，多くの実用的な関数を比較的コンパクトに表現できる．さらに，複数の関数を同時に効率良く表現できるように改良したものを共有二分決定グラフと呼ぶ．本論文では，共有二分決定グラフを用いた論理関数の処理手法について述べ，その効率化手法として，否定エッジ，入力反転エッジ，変数シフトエッジの3種類の属性エッジを用いて記憶量や処理時間を削減する方法を提案する．共有二分決定グラフにおいては，入力変数の順序づけはグラフの大きさに著しい影響を与えるため，この問題について考察し，効率のよい順序づけを行う動的重みづけ法を提案する．さらに，不完全指定論理関数（Don't care を含む関数）について考察し，D変数を導入した表現法を提案する．最後に，本論文で提案した手法の効果を調べるため，実用的な論理回路を共有二分決定グラフで表現する実験を行い，本論文の手法が有効であることを示す．

1. はじめに

論理関数（以下，単に関数とも呼ぶ）を効率よく表現し，高速に論理演算を行うことは，記号シミュレーションや，論理照合，テスト生成などの論理設計支援システムを実現する上での重要な問題である．特に，計算機内部での論理関数の表現法は重要である．論理演算の技術は，論理設計のほかにも，組合せ問題や人工知能等，計算機科学全般にわたる応用が期待される．

Akers[1] が提案し，Bryant[2] が発展させた二分決定グラフによる論理関数の表現法は，コンパクトであり，入力変数の順序を固定することにより関数を一意に表すことができるという特長を持っており，有望な方法の1つと考えられる．

さらに，複数の関数を同時に効率的に表現できるように改良した共有二分決定グラフのアイデアが提案されている[3]．共有二分決定グラフは，関数の一致判定が即座に行えるという重要な特長を持つ．本論文では，共有二分決定グラフを扱う技術を発展させ，実用化するための研究を行った．

以下の章では，まず，共有二分決定グラフの性質と，その処理手法について述べる．つぎに，共有二分決定グラフの効率化手法として，3種類の属性エッジや，真理値表を用いる方法を提案する．さらに，入力変数の順序づけがグラフの大きさに与える影響を考察し，効率のよい順序を求める方法を提案する．また，実用上しばしば必要とされる不完全指定論理関数（Don't care を含む関数）の表現方法について考察する．最後に，実際の論理回路を扱う実験の結果を示し，本論文の手法の評価を行う．

2. 共有二分決定グラフとその実現手法

2.1 二分決定グラフ[1),2)]

二分決定グラフは**図1**に示すようなグラフによる論理関数の表現法である．これは，シャノンの展開を再帰的に繰り返すことによって得られる二分木のグラフから，冗長なノードの削除と，同形なサブグラフの共有を行ったものであり，入力変数の順序を固定すれば，論理関数に対してグラフの形が一意に定まる．

二分決定グラフは以下のような特長を持つ．

（1） 関数の一致判定がグラフの同形判定で行える．

（2） n入力論理関数を表すグラフのノード数は，最悪の場合，$O(2^n/n)$ となるが，実用的な多くの関数は比較的少ないノード数で表現可能である[5]．

（3） 関数同士の論理演算が，グラフのノード数にほぼ比例する時間で効率よく行える．

2.2 共有二分決定グラフ[3),4)]

二分決定グラフを改良して，複数の関数を表すグラフの間においてもサブグラフの共有を行ったものを，共有二分決定グラフ（**図2**）と呼ぶ．共有二分決定グラフは，以下の点で二分決定グラフより優れている．

† Shared Binary Decision Diagrams for Efficient Boolean Function Manipulation by SHIN-ICHI MINATO, NAGISA ISHIURA and SHUZO YAJIMA (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 京都大学工学部情報工学教室

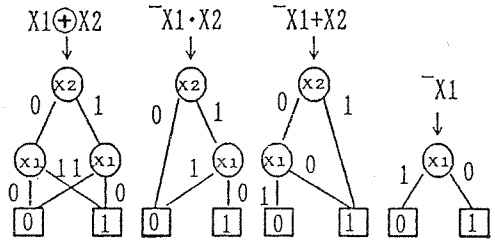


図 1 二分決定グラフ
Fig. 1 BDD's.

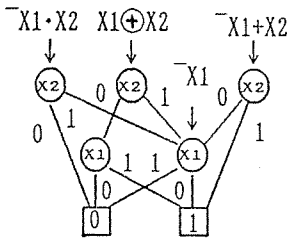


図 2 共有二分決定グラフ
Fig. 2 SBDD.

(1) 同じ関数を表すグラフを重複して生成することはないので，関数の一致判定がポインタの一致判定だけで行える．

(2) 論理演算の際に，高速な等価性判定を利用して，$f+f=f$，$f\oplus f=0$，$f+\bar{f}=1$ などの規則を適用することができる．

(3) $f+0=f$，$f\cdot f=f$ などの演算の際にグラフをコピーする必要がない．

(4) 複数の関数を，少ない記憶量で同時に表せる．

2.3 共有二分決定グラフの応用

共有二分決定グラフの性質を利用することにより，次のような様々な応用が考えられる．

(1) 論理照合[8),12)]

2つの論理回路の出力関数を共有二分決定グラフで表すことができれば，即座に一致判定が行える．

(2) テスト生成[6)]

正常回路と故障回路のそれぞれの出力関数を共有二分決定グラフで表すことができれば，それが異なる値を持つような入力値を求めることは容易である．また，正常回路と故障回路の関数はよく似ているので，グラフが共有されやすく，記憶効率がよい．

(3) 記号シミュレーション[13)]

共有二分決定グラフを用いれば，入力値のすべての組合せに対する論理シミュレーションを，多くの場合，実用的な速度と記憶量で行うことができる．従来の，不定値Xを用いる3値シミュレーションでは，結果が不正確になることがあるが，記号シミュレーションでは，正確なシミュレーションが可能である．

そのほかにも多くの応用が考えられる．いずれの場合も，論理関数を共有二分決定グラフで表すことができれば，非常に強力な処理を行うことができる．問題は，関数のグラフを実用的な記憶量で表せるかどうかにかかっている．そこで本論文では，より大規模な回路を扱えるようにするため，以下に示すような効率的な実現手法を提案する．

2.4 共有二分決定グラフの記憶管理

共有二分決定グラフでは，共有可能なサブグラフは必ず共有しなければならない．そこで，本論文で実現している処理プログラムでは，演算中に新しいノードを生成する際に，等価なノードがすでに存在していれば，そのノードを共有することにより，新しいノードを作らないようにしている．等価なノードが存在するかどうかの判定は，すべてのノードをハッシュテーブルに登録，参照することにより行う．

共有二分決定グラフは，複数の関数を効率よく表せるが，演算の途中結果のように，すでに不要になった関数のグラフは削除して，記憶領域の再利用を図りたい．このとき，不要な関数のサブグラフが，他の必要な関数のサブグラフと共有されている場合には，その部分は削除しないようにしなければならない．

本論文で実現したプログラムでは，各ノードごとに，そのノードを参照しているエッジの本数を記録するカウンタを用いることにより，ノードの必要性を判定している．すなわち，ある関数のグラフが不要になったときには，そのグラフの根のノードのカウンタの値を1減らし，これが0になったとき初めてそのノードの削除を行う．その際に，そのノードのサブグラフも不要になるので，同様にカウンタの値を減じる処理を行う．

このようにして削除可能となったグラフでも，すぐあとに再び必要となる場合もあるので，記憶に余裕があれば削除せずに温存しておく．そして，記憶量が限界に達したときや，正確なノード数を数えたいときに，ガベジコレクションを行って，不要なノードをまとめて削除するようにして高速化を図っている．

3. 属性エッジを用いた効率化手法

共有二分決定グラフを効率よく実現する手法として，属性エッジの使用を提案する．これは，グラフのエッジに以下に示すような機能を表す属性を与えることにより，グラフのノード数や処理時間を削減する方法である．

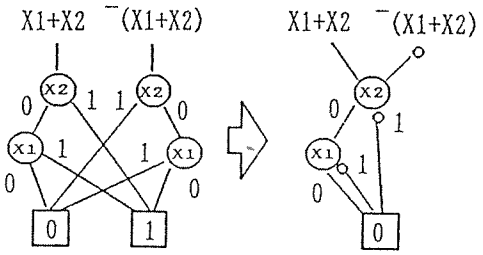


図 3　否定エッジ
Fig. 3　Output inverters.

3.1　否定エッジ[4]

エッジに否定演算を表す属性を与えるもので，グラフをたどる際に，否定エッジを通った回数だけ論理値を反転させる．これは，Akers[1] の inverter や，Madre[7] の typed edge と同じものである．これにより，互いに否定の関係にあるサブグラフを共有することができ（**図 3**），記憶量が最大 50% 削減される．また従来では否定演算の際にグラフをすべてたどる必要があったが，否定エッジを用いればグラフの根を指すエッジの属性を反転するだけでよいので，処理速度が向上する．

否定エッジを無制限に使うと，同じ関数を何通りにも表せるので，グラフの一意性が失われてしまう．そこで，次のような制限を加えることにより，一意性を保っている．

- 各ノードの “0” エッジには否定エッジを使わない．
- グラフの葉の値には，0だけを用いる．

この制限は，Madre[7] と基本的に同じものである．

3.2　入力反転エッジ

入力反転エッジは我々が新たに提案する属性エッジで，次のノードで “0” エッジと “1” エッジを交換することを表す（**図 4**）．すなわち，入力変数の値を反転させることを意味する．これによりノード数が最大 50% 削減される．特に，**図 5** の例では否定エッジはほとんど効果がないのに対して，入力反転エッジの効果は大きい．

否定エッジと同様に，入力反転エッジを無制限に使用するとグラフの一意性が失われてしまうので，何らかの制限を加える必要がある．

直観的には，図 4 において関数 F と G のどちらに入力反転エッジを使用するかを決定する規則を定めればよい．すなわち，すべての論理関数に，ある全順序関係を定めておき，

(“0” エッジの関数) < (“1” エッジの関数)

という制限を満たすように入力反転エッジを用いれば，一意性を保つことができる．本論文で実現したプログラムでは，各エッジは 1 ワードの整数で表現されているので，その値の大小によって順序関係を定めている．この場合，グラフの形はいつも同じになるとは限らないが，1 つのグラフの中では，関数は一意に表されている．

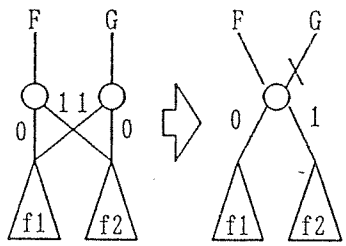


図 4　入力反転エッジ
Fig. 4　Input inverters.

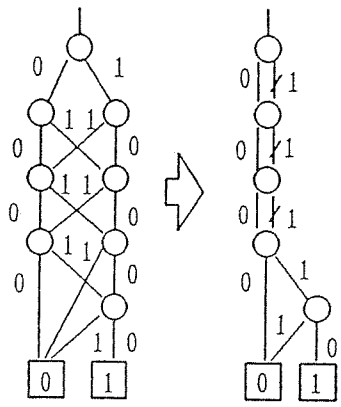


図 5　入力反転エッジが有効な例
Fig. 5　An example where the input inverters are effective.

3.3　変数シフトエッジ

図 6 のように，変数の番号がずれているだけで，構造が全く同じであるような 2 つのグラフに対しては，その番号の相違だけを記録しておけば，共有することができる．変数シフトエッジは変数番号に作用する属性をエッジに与えるもので，そのエッジの指すサブグラフの中の全入力変数の番号に，ある数を加える（変数番号をシフトする）という意味を持つ．

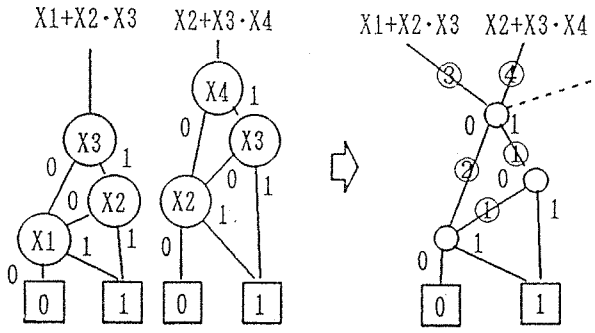


図 6　変数シフトエッジ
Fig. 6　Variable shifters.

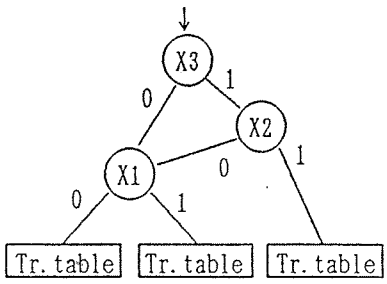


図 7 真理値表の併用
Fig. 7 Combination with truth tables.

変数シフトエッジを用いると，図6のように，従来より多くのサブグラフを共有することができ，グラフのノード数を削減できる．変数シフトエッジが有効な例としては，同じようなパタンが並んでいるデータ系の回路を扱う場合や，記号の系列を扱う場合等が考えられる．

変数シフトエッジを用いる場合，各ノードに入力変数を記録する必要はなく，そのかわりに各エッジに入力変数の間の相対的な関係が記録される．グラフの一意性を保つため，次のような規則にしたがって，変数シフトエッジを使用する．

(1) グラフの葉を指しているエッジには，変数シフトエッジは使用しない．

(2) グラフの根を指すエッジでは，変数シフトエッジによって，根のノードの変数番号を表す．

(3) その他のエッジでは，変数シフトエッジによって，起点ノードと終点ノードの変数番号の差を表す．

3.4 真理値表[4]

計算機上でグラフのエッジを表すのに，ポインタとして1ワードを必要とする．しかし，入力変数の少ない関数ならば，1ワードの真理値表で表現できる．そこで，グラフの葉に近い4変数関数のサブグラフは，真理値表にすることにより，ノード数を削減することができる（**図7**）．論理演算も，計算機のビット列論理演算を用いて高速に行える．否定エッジと組み合わせて使用する場合には，グラフの一意性を保証するため，すべての入力が0のときに，値が0となるような真理値表だけを用いるように制限を加える必要がある．

4. 入力変数の順序づけ

共有二分決定グラフを用いて論理回路を扱う場合，回路の各信号線の論理関数を共有二分決定グラフで表す必要がある．二分決定グラフは，入力変数の順序によってグラフの形が異なり，時には大きさが著しく変化することが知られている．共有二分決定グラフの処理時間はグラフの大きさにほぼ比例するので，入力変数の順序づけは重要である．そこで，回路の結線情報から関数の性質を予想し，最適に近い順序を求める手法が研究されている[8]．本稿では，入力変数の順序による影響を考察し，新しい順序づけの手法を提案する．

4.1 順序づけの影響

グラフの大きさに影響する要素として，次のような点があげられる．

(1) 局所計算性のある入力の組は，なるべく近い順序にする．

図8のような AND-OR 2段回路の例では，同じ AND ゲートの入力をそれぞれ隣接させるような順序と，互いに遠ざけるような順序とを比較すると，グラフのノード数は，入力数に関して指数の違いが生じる．つまり，局所計算性のある入力を近づけると，多くのサブグラフが共有できるという傾向がある．

(2) 出力を制御する力の強い入力は上位に配置する．

例えば，$\log k$ 本の制御入力と k 本のデータ入力を持つデータセレクタの出力関数を表すことを考える．

制御入力を上位に並べた場合，制御入力の値によって n 通りに枝わかれしたあと，それぞれのサブグラフはデータ入力の1変数関数となるのでノード数は $O(k)$ である．

これに対してデータ入力を上位に並べると，データ入力の値の 2^k 通りの組合せに対して，サブグラフはすべて異なる関数となるため共有できず，$O(2^k)$ 個のノードが必要になる．

この例のように，各入力の隣接関係に変化がなくても，ノード数に指数の違いが現れる場合がある．これは，制御入力の値を先に決定すると，多くのデータ入力を無視できるという性質があるためと考えられる．

これら2つの性質を満たすような順序づけを行えばよいのであるが，実際には，(1)の性質と(2)の性質

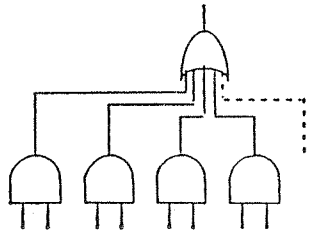


図 8 AND-OR 2段回路
Fig. 8 AND-OR 2-level circuit.

が同時に現れて，互いに相反する順序づけを要求する場合があり，これを両立させて最適な順序を求めるのは難しい．また，どんな順序でもそれほど変化の見られない回路もあり，その場合には，順序づけを工夫してもあまり効果が得られない．

以上の考察では1出力関数について述べたが，共有二分決定グラフで多出力関数を表す場合，複数の関数の間での共有についても考える必要がある．一般に，外部出力に近い（段数が浅い）入力を上位に置いた方が良い結果が得られることが多い．

また，複数の関数の間で相反する順序づけが要求される場合もあり，どの関数の都合を優先させるについても考慮する必要がある．

4.2 動的重みづけ法

回路の結線情報から変数の順序づけを行う手法として，外部出力から入力に向かって深さ優先探索を行い，先にたどり着いた外部入力から順に上位の変数を割り当てていく方法が提案され，比較的良い結果が得られている[6),8),14)]．

これに対して本論文では，動的重みづけ法による新しい順序づけの手法を提案する．以下に，その考え方を示す．

まず，前節(2)の性質を満たすため，制御性の高い入力を見つける．これは，

- ファンアウト数が多い（出力に至るパスが多い）．
- 出力までの段数が少ない．
- 出力へ至るパス上のゲートのファンイン数が少ない

という傾向を持つ．この傾向を評価するために，つぎに示すような簡単な重みづけの方法を用いる．

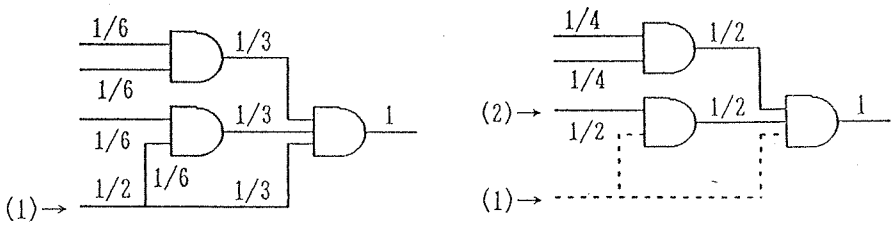
(1)　出力線の重みを1とする．

(2)　出力から入力に向かって重みを伝達させる．2入力以上のゲートでは，出力線の重みをファンイン数で均等に分けて入力線に伝える．

(3)　ファンアウトがある信号線で，複数のゲートから重みが伝わるときにはそれらを加える．

以上の手順で重みづけをした例を**図9**(a)に示す．この重みが最大となった入力が最も出力を制御しやすいとみなし，その入力に最上位の変数を割り当てる．

次に，最上位の変数を決定した後，その下のサブグラフを小さくすることを考える．このとき，すでに決定された入力は0か1に固定されてしまっていると考えれば，その近くの信号線は，制御性が増大しているはずである．そこで，すでに決定された信号線を切断したと仮定して，改めて重みづけを行う．このように動的に重みづけを行うことにより，前節(1)の局所計算性も反映することができる．その様子を図9(b)に示す．以下，これを動的重みづけ法と呼ぶ．



(a)　第1回目の重みづけ
(a)　First assignment.

(b)　第2回目の重みづけ
(b)　Second assignment.

図9　動的重みづけ法
Fig. 9　Dynamic weight assignment method.

多出力回路の関数を1つのグラフで表す場合には，どの出力から重みづけを始めるかという問題がある．各出力が，相異なる順序づけを要求する場合があり，総合的に考えることは難しい．ここでは，グラフが最も複雑になりそうな出力を優先して順序づけを行っている．複雑さの目安としては，その出力線に関係している回路の段数を用いた．これは比較的容易に算出できる．なお，多入力ゲートは2入力のゲートに変換して段数を求めている．

多くの場合，複雑な関数ほど多くの段数を必要とする．また，加算器や ALU 等のように，ある出力が他の出力の計算の途中結果となっている回路では最終結果の出力を優先すべきである．このような場合にも回路の段数は有効な目安となる．

順序づけに必要な時間は，入力数n，素子数をmとして，$O(n \cdot m)$ 程度であるが，順序づけによってnの指数の違いが見られるので，それでも十分有効である．

5. 不完全指定論理関数の表現法

実際の CAD システムで論理関数を扱う場合，無駄な計算を省いたり，近似解を求めたりする方法として，DまたはX (Don't care，不定値，未定値などと呼ばれる）という値を加えた3値出力の論理関数（不完全指定論理関数）を用いることがある[9)]．このような関数を二分決定グラフで表現する1つの方法として，グラフの葉に，0，1のほかに＊(Don't care) を

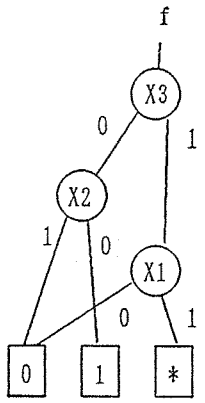


図 10 3値二分決定グラフ
Fig. 10 Ternary valued BDD.

加えて3種類の葉を用いる方法が考えられる (**図 10**). これを3値二分決定グラフと呼ぶことにする. この方法は, 直観的に理解しやすいが, 通常の二分決定グラフの処理プログラムでは扱えず, 属性エッジ等の効率化手法をそのまま適用できないという問題点もある. 本章では, 不完全指定関数を効率よく扱う方法について考察する.

5.1 2ビット符号化二分決定グラフ

不完全指定関数を表現するもう1つの方法として, 3値を2ビットで符号化し, 2個の2値の関数の組で1つの不完全指定関数を表し, それぞれを通常の二分決定グラフで表現する方法がある. これを2ビット符号化法と呼ぶことにする. 以後, 2つの関数の組を $[f_0, f_1]$ と表す. 符号の割り当て方によって効率が左右されるが, 以下に示す符号割り当てが有効である.

0 : [0, 0]
1 : [1, 1]
* : [0, 1]

f_0 は*を0とみなした場合, f_1 は*を1とみなした場合を表している. 例えば, 関数がどのような入力パタンに対しても1または*を返す (恒真とみなせる) 場合は, f_1 のサブグラフが恒真となるのですぐに判定できるという利点がある.

この符号化における論理演算は次のように定義される.

$$[\overline{f_0, f_1}] = [\overline{f_1}, \overline{f_0}]$$
$$[f_0, f_1] \cdot [g_0, g_1] = [f_0 \cdot g_0, f_1 \cdot g_1]$$
$$[f_0, f_1] + [g_0, g_1] = [f_0 + g_0, f_1 + g_1]$$

2ビット符号化法はすでに文献10) で用いられているが, 符号の割り当ては本稿といささか異なる. 本稿の符号割り当ては, 次節で述べるD変数を導入したときに, 美しい性質を持つ.

5.2 D 変数の導入

2ビット符号化法で表された2つのグラフを, 特殊なD変数によりまとめて表現する方法を提案する. D変数を用いると, **図 11** のように, 前節の2ビット符号化法によって得られた2つのグラフをまとめて1つのグラフで表すことができる. D変数のノードでは, "0" エッジが f_0 を, "1" エッジが f_1 を指している. このグラフは, 以下に述べるような性質を持つ.

関数がどのような入力パタンに対しても0または1を返す (*を持たない) 場合は, f_0 と f_1 は等価なグラフになるので完全に共有され, D変数のノードは除去される. その結果, 通常の2値の二分決定グラフと完全に一致する. また, 関数がどのような入力パタンに対しても*を返す (恒*関数) 場合は, D変数のノード1個だけのグラフとなる.

グラフ全体を2値の関数とみなして, これを非冗長積和形で表すと, D変数を含まない積項は, on set を表し, D変数を含む積項は, don't care set を表すという性質がある.

図11ではD変数を最上位に置いているが, これを最下位に移動させると, **図 12** のようなグラフに変換される. このグラフで, D変数のノードを経由するパスは, $f_0=0$, $f_1=1$, すなわち * (don't care) となる

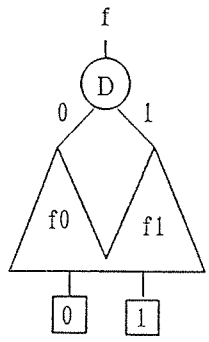


図 11 D 変 数
Fig. 11 D-variable.

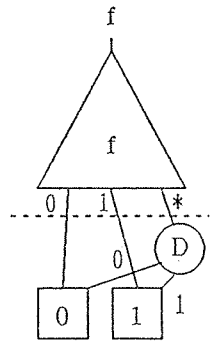


図 12 D変数を最下位に置いた例
Fig. 12 D-variable in the lowest order.

入力パタンを表している．また，D変数を経由しないパスは，D変数の値にかかわらず0または1であることを示す．このことから，図12の点線より上の部分は，3値二分決定グラフと完全に同形なグラフとなっていることがわかる．つまり，3値二分決定グラフや2ビット符号化法は，D変数を用いた表現方法の特殊な場合とみなすことができる．

D変数の順序づけの問題は，4章での議論をもとに考えると，D変数が出力に大きな影響を与える（すなわち制御する力が強い）ことが多いので，一般には上位に配置する2ビット符号化法が有利と言うことができる．

5.3 多値論理への拡張

これまで，3値の不完全指定関数について考察してきたが，4値以上のn値関数も符号化すれば，$\log n$個の二分決定グラフの組で表すことができる．逆にn出力の2値関数は，2^n種類の葉を持つ二分決定グラフで表せる．どちらが有利かは，これまでの議論と同様に，変数の順序づけの問題に置き換えて評価することができる．

6. 実験と評価

本稿で述べた種々の手法を用いて，共有二分決定グラフの処理プログラムを実現した．使用機種はSUN 3/60（主記憶24 MB，SunOS 4.0），プログラムは約700行のC言語で記述した．1ノードあたりの記憶量は約21バイトで，最大50万個のノードを扱うことができる．この実験では効率化手法として，否定エッジ，入力反転エッジ，変数シフトエッジを実現している．入力変換の順序づけは，動的重みづけ法によって自動的に行うことができる．

このプログラムの性能を評価するために，組合せ回路（一般に多出力）の内部信号線も含めた全信号線の論理関数を同時に，共有二分決定グラフで表す実験を行った．全信号線を同時に表せれば，論理照合やテスト生成等の処理が非常に効率よく行える．ただし，外部出力の関数だけを表す場合は，不要なグラフを解放することにより，この実験よりも少ないノード数で表現できる．各出力を逐次的に処理すればさらに少なくてすむ．また，正確なノード数を数えるために，最後にガベジコレクションを行っているが，これを省けば処理時間は短くなる．

実験結果を**表1**に示す．実験に用いた回路は，dec 8は8ビットデータセレクタ，enc 8は8ビットプライオリティエンコーダ，add 8，add 16は加算器，mult 4，mult 8は乗算器である．残りの回路は，ISCAS '85[11]のベンチマーク回路から選んだものである．

表中の処理時間は，回路データを読み込む時間と，入力変数の順序づけの時間，およびグラフを生成する時間の合計である．多くの実用的な回路を，コンパクトに短時間で表現できている．また，処理時間はグラフのサイズに，ほぼ比例していることが観測できる．共有二分決定グラフのサイズが実現可能な範囲であれ

表1 実験結果
Table 1 Experimental results.

Circuit	Circuit size			#Node	CPU (sec)
	In.	Out.	Nets		
dec 8	12	2	29	40	0.3
enc 8	9	4	31	33	0.3
add 8	18	9	65	49	0.4
add 16	33	17	129	97	0.7
mult 4	8	8	97	330	0.5
mult 8	16	16	418	46594	18.3
c 432	36	7	203	89338	34.1
c 499	41	32	275	36862	21.5
c 880	60	26	464	30548	11.5
c 1355	41	32	619	119201	51.4
c 1908	33	25	938	39373	22.5
c 5315	178	123	2608	40306	29.8

表2 属性エッジの効果
Table 2 Effect of the attributed edges.

Circuit	(A)		(B)		(C)		(D)	
	#Node	CPU (sec)	#Node	CPU	#Node	CPU	#Node	CPU
dec 8	78	0.3	51	0.3	51	0.4	40	0.3
enc 8	56	0.3	48	0.3	48	0.3	33	0.3
add 8	119	0.4	81	0.4	81	0.4	49	0.4
add 16	239	0.7	161	0.6	161	0.7	97	0.6
mult 4	524	0.5	417	0.5	400	0.4	330	0.5
mult 8	66161	24.8	52750	19.1	50504	19.8	46594	18.3
c 432	131299	55.5	104066	36.5	103998	36.8	89338	34.1
c 499	69217	22.9	65671	21.3	36986	21.8	36862	21.5
c 880	54019	17.5	31378	10.8	30903	11.1	30548	11.5
c 1355	212196	89.9	208324	49.3	119465	52.8	119201	51.4
c 1908	72537	33.0	60850	21.6	39533	22.3	39373	22.5
c 5315	60346	31.3	48353	29.2	41542	28.6	40306	29.8

(A) Using nothing, (B) (A)+output inverters, (C) (B)+input inverters, (D) (C)+variable shifters.

ば，十分効率よく処理を行える．

次に，属性エッジの効果を調べるために，3つの属性エッジを順に適用して効果を調べる実験を行った（**表2**）．

(A)は属性エッジを全く使わなかった場合の結果である．(B)は否定エッジだけを使用した場合，(C)は否定エッジと入力反転エッジを使用した場合，(D)はさらに変数シフトエッジを加えて，すべての属性エッジを使用した場合の結果である．

これらの実験結果から，属性エッジによってノード数が 30%-50% 削減される．処理速度については，否定エッジによって向上しており，他の属性エッジによっても低下することはなく，有効である．各属性エッジは，回路によって効果に偏りが見られるが，3種類の属性エッジを組み合わせることによって，多くの場合に効果をあげていることがわかる．

表3 真理値表の効果

Table 3 Effect of the truth tables.

Circuit	(Not using)		(Using)	
	#Node	CPU	#Node	CPU
dec 8	51	0.4	47	0.3
enc 8	48	0.3	38	0.3
add 8	81	0.4	66	0.4
add 16	161	0.7	146	0.6
mult 4	400	0.4	269	0.5
mult 8	50504	19.8	49780	16.7
c 432	103998	36.8	103954	35.5
c 499	36986	21.8	36934	22.4
c 880	30903	11.1	30899	11.1
c 1355	119465	52.8	119412	52.1
c 1908	39533	22.3	39517	22.4
c 5315	41542	28.6	41538	29.4

表4 入力変数の順序づけの影響

Table 4 Effect of the order of the input variables.

Orderings of the Inputs

Circuit	(A)		(B)		(C)		(D)	
	#Node	CPU	#Node	CPU	#Node	CPU	#Node	CPU
dec 8	40	0.3	41	0.3	390	0.4	57	0.4
enc 8	33	0.3	31	0.3	30	0.3	37	0.4
add 8	49	0.4	120	0.4	452	0.4	1183	0.6
add 16	97	0.7	248	0.5	1700	0.9	94814	24.1
mult 4	330	0.5	358	0.4	304	0.5	394	0.5
mult 8	46594	18.3	38187	14.5	31026	14.0	77517	26.1
c 432	89338	34.1	11348	7.4	6205	5.6	479711	278.6
c 499	36862	21.5	68816	39.1	32577	21.0	112815	78.0
c 880	30548	11.5	(>500 k)		(>500 k)		(>500 k)	
c 1355	119201	51.4	246937	102.9	103301	46.9	373974	179.0
c 1908	39373	22.5	47990	22.7	65895	63.3	91082	47.4
c 5315	40306	29.8	105200	32.5	(>500 k)		(>500 k)	

(A) Using dynamic weight assignment method, (B) In the original order, (C) In the original order (reverse), (D) In a random order.

表3は真理値表の効果を調べた実験の結果である．下位4変数のグラフが削減されるので，関数の入力数が多いときはほとんど効果が認められないが，入力数が少ない場合には有効である．

表4は入力変数の順序づけの実験結果である．(A)は本論文で述べた動的重みづけ法によるもの，(B)は回路のデータそのままの順序，(C)は(B)の逆順，(D)は乱数で作った順序である．これらの結果をみると，乱数順では全く良い結果が得られず，回路のデータそのままの順序では一部良い例があるが，一般には良い結果は得られない．これに対して動的重みづけ法では多くの場合良い結果を与えている．

入力変数の順序づけについては，文献14)に回路の深さ優先探索に基づく方法が提案されている．本論文とは実験方法が異なる（本論文では全信号線の論理関数を同時に表現しているのに対し，14)では外部出力の論理関数を逐次求めている）ため，単純には比較できないが，本論文の手法はこのベンチマーク回路に関しては14)の結果に及んでいない（記憶量の制約から，一部の回路でグラフを生成できなかった）．ただし，いずれの手法も発見的手法に属するものであり，その有効性は回路の論理関数や構造に大きく依存して変わり得る．有効な変数の順序づけの手法を開発することは今後の重要な研究課題であると考える．

7. おわりに

本論文では，共有二分決定グラフの性質について述べ，属性エッジを用いた効率化手法を提案した．また，入力変数の順序について考察し，動的重みづけ法を提案した．さらに，不完全指定論理関数の表現法について考察した．

共有二分決定グラフは，論理関数の表現方法として優れた性質を持つ．共有二分決定グラフにより，多くの実用的な論理関数をコンパクトに表現でき，高速に論理演算を行うことができる．今後の論理設計関係の研究や，そ

の他，広く計算機科学一般への応用も期待される．

謝辞　御討論頂いた本学平石裕実助教授，高木直史博士ほか，矢島研究室の諸氏に感謝いたします．

参考文献

1) Akers, S. B.: Binary Decision Diagrams, *IEEE Trans. Comput.*, Vol. C-27, No. 6, pp. 509-516 (1978).
2) Bryant, R. E.: Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Trans. Comput.*, Vol. C-35, No. 8, pp. 677-691 (1986).
3) 北嶋，高木，矢島：論理関数のグラフ表現を用いた記号シミュレーション，信学技報，VLD 87-113, pp. 47-52 (1988).
4) 湊，石浦，矢島：共有二分決定図の処理手法について，第 38 回情報処理学会全国大会論文集，5 S-9 (Mar. 1989).
5) Ishiura, N., Tohdo, T. and Yajima, S.: A Class of Logic Functions Expressible by a Polynomial-Size Binary Decision Diagram, 第 39 回情報処理学会全国大会論文集，7 W-2 (Oct. 1989).
6) 井置，石浦，矢島：共有二分決定図を用いた組合せ回路のテスト生成，第 38 回情報処理学会全国大会論文集，2 S-5 (Mar. 1989).
7) Madre, J. C., Coudert, O., Billon, J. P. and Berthet, C.: Formal Verification and Diagnosis of Digital Circuits Using a Propositional Theorem Prover, *Proc. IFIP Work. Conf. CAD Systems Using AI Techniques*, pp. 107-114 (1989).
8) 藤田，藤沢，川戸：2分決定グラフを用いた論理照合アルゴリズムの評価と改良，情報処理学会研究会報告，88-DA-43-2 (Jul. 1988).
9) 松永，藤田：2分決定グラフを利用したトランスダクション法の改良，第 39 回情報処理学会全国大会論文集，4 X-2 (Oct. 1989).
10) Cho, K. and Bryant, R. E.: Test Pattern Generation for Sequential MOS Circuits by Symbolic Fault Simulation, *26 th ACM/IEEE DAC*, pp. 418-423 (June 1989).
11) Brglez, F. and Fujiwara, H.: A Neutral Netlist of 10 Combinational Circuits, Special Session on ATPG and Fault Simulation, *Proc. 1985 IEEE International Symposium Circuit and Systems*, Kyoto, Japan (June 1985).
12) Minato, S., Ishiura, N. and Yajima, S.: Fast Tautology Checking Using Shared Binary Decision Diagram—Benchmark Results—, *Proc. IFIP International Workshop on Applied Formal Methods for Correct VLSI Design*, pp. 580-584 (Nov. 1989).
13) 湊，石浦，矢島：共有二分決定グラフを用いた記号シミュレーション，1989 年信学全大，SA-7-5 (1989).
14) 藤田，藤沢，松永，角田：2分決定グラフのための変数順決定アルゴリズムとその評価，情報処理学会論文誌，Vol. 31, No. 4, pp. 532-541 (1990).

(平成2年 3 月 19 日受付)
(平成2年 10 月 9 日採録)

湊　真一（正会員）

昭和 40 年生．昭和 63 年京都大学工学部情報工学科卒業．平成 2 年同大学院修士課程修了．同年日本電信電話(株)入社．現在，同社 LSI 研究所にて論理設計システムの研究に従事．在学中は，論理回路の記号シミュレーション，および論理関数の処理手法の研究に従事．

石浦菜岐佐（正会員）

昭和 36 年生．昭和 59 年京都大学工学部情報工学科卒業．昭和 61 年同大学院修士課程修了．昭和 62 年 1 月より京都大学工学部助手．論理回路の CAD/DA の研究に従事．電子情報通信学会会員．

矢島　脩三（正会員）

昭和 8 年生．昭和 31 年京都大学工学部電気工学科卒業．同大学院博士課程修了．工学博士．昭和 36 年より京都大学工学部に勤務，昭和 46 年情報工学科教授．昭和 35 年京都大学第一号計算機 KDC-1 を設計稼動，以来，計算機，論理設計，オートマトン等の研究教育に従事．著書は「電子計算機の機能と構造」(岩波，57 年) 等．本学会元常務理事，元会誌編集委員 (地方)，元 JIP 編集委員．電子情報通信学会元評議員およびオートマトンと言語研専元委員長，North-Holland 出版 元 IPL 編集委員，IEEE Senior Member.