5S-9

共有二分決定図を用いた論理関数の処理手法について

湊 真一　石浦菜岐佐　矢島 脩三
(京都大学工学部)

1.はじめに

Bryantが提案した二分決定図[1]による論理関数のグラフ表現は、コンパクトであり、変数の順序を固定すれば関数を一意に表すことができるため、記号シミュレーションや、論理照合、テスト生成などの処理に対して非常に有効である。さらに、複数の関数を同時に効率的に表現するための共有二分決定図が、北嶋他により提案されている[2]。本稿では、共有二分決定図の処理を効率良く行う手法として、演算結果テーブルの効率化、真理値表との組合せ、否定エッジの使用等について述べる。

2.共有二分決定図

二分決定図は図1に示すような、論理関数のグラフによる表現法で、冗長なノードの削除と、同型なサブグラフの共有により、既約な(reduced)グラフが得られ、変数の順序を固定すれば関数を一意に表すことができる。変数の多い関数に対しても比較的コンパクトな表現が得られるため、論理検証や、テスト生成への応用など多くの研究が行われている[3]。

これに対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なったものを、共有二分決定図（図2）と呼ぶ[2]。共有二分決定図では、同じグラフを重複して生成しないので、関数の等価性判定がノードを指すポインタの一致判定だけで行える。またグラフの共有により、無駄な記憶量を省くことができる。さらに関数同士の演算の際に、新たにノードを生成する数が少なくなり高速である。

3.共有二分決定図の処理手法

3.1 論理演算

グラフ同士の二項演算では、変数を1と0に場合分けしたときのそれぞれのサブグラフ同士について、再帰的に論理演算を施す処理を繰り返し、どちらかのグラフが葉になったところで演算結果のノードが作られて行く。[1]では、二項演算の手続きが全て終わってから、reductionと呼ばれる手続きによって冗長なノードを除去しているが、二項演算の途中で一時的にノード数が増大するという欠点がある。これに対し共有二分決定図では、演算の途中でノードが作られるたびに、同じノードがすでに存在しているかを調べ、共有を行なって冗長なノードを省いているので記憶効率がよく、最後にreductionを行う必要もない。　冗長ノードの判定は、全てのノードをハッシュテーブルに登録しておくことにより高速に行える。

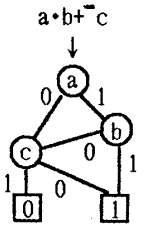


図1　二分決定図

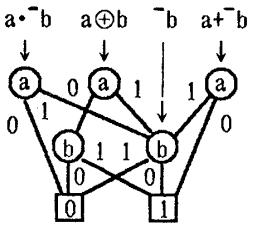
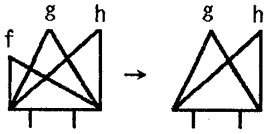


図2　共有二分決定図

3.2 ノードの解放

全ての関数のグラフを保持する必要がない場合は、不要なノードを解放して再利用を行うことにより、記憶量を節約することができる。グラフを解放する場合、他の関数と共有している部分のノードは残さなければならない（図3）。そこで、各ノードにカウンタを設け（共有カウンタと呼ぶ）そのノードを指しているエッジの本数を記録しておくことにより、ノードの必要性を判定する。ノードを解放する際には、そのノードの共有カウンタの値を1減らし、これが0になるまでは解放を行わない。そして実際にノードが解放されたら、そのサブグラフについても同様な処理を繰り返す。この手法によって、解放部分のノード数に比例する時間で、グラフを解放することができる。



(関数fのグラフを解放)

図3　ノードの解放

3.3 演算結果テーブル

論理演算をサブグラフに対して再帰的に適用する場合、サブグラフが共有されている部分では、同じ演算が重複して行われる可能性がある。そこで、これまでに演算を行なったノードの組とその結果のノードを記録しておく演算結果テーブルを用意することにより、無駄な演算を省く高速化手法が提案されている[1]。演算結果テーブルはハッシュテーブルとして実現されるが、過去の全ての演算を記録すると記憶量が大きくなり過ぎるため、サイズを限定して最近の演算だけを記憶することになる。このサイズが不十分だと、必要な情報が忘れられて無駄な演算が多くなり、急激に処理速度が低下する。

考察の結果、冗長な演算を省くためには全ての演算を記録する必要はなく、複数のエッジで共有されているノードに関する演算だけを記録すれば十分であることがわかった。本稿では、共有カウンタの値を調べることにより、無駄なノードの登録・参照を省いて、演算結果テーブルのサイズを小さく抑えている。

4.二分決定図の効率化手法

4.1 真理値表との組合せ

二分決定図を計算機上で実現する場合、リスト構造を表現

On Boolean Function Manipulation
Using Shared Binary Decision Diagram
Shin-ichi MINATO, Nagisa ISHIURA, Shuzo YAJIMA
Kyoto University

するために、ポインタとして1ワードが使用される。しかし、変数の個数が少なければ、真理値表を表すビット列として1ワードに詰め込み、ビット列演算で高速に処理できる。そこで図4のように、二分決定図の下位4変数に当たる部分のポインタを真理値表に置き換える。（1ワード32ビットで最大5変数まで表せるが、ポインタと真理値表の判別フラグも必要である。）これによって、グラフの下位4段が削除されたことになり、記憶量の削減と処理速度の向上が期待できる。

4.2 否定エッジの使用

共有二分決定図で関数の否定演算を行うと、関数のグラフの葉(0,1)を交換した対称なグラフが生成される。この関数の間ではサブグラフの共有が全く行えず、元の関数と同数のノードが新たに生成されて記憶効率が悪い。

本稿では、否定演算の効率化手法として否定エッジの使用を提案する。これは図5のようにグラフのエッジで否定を表す方法である。これによって、否定演算は定数時間で行うことができ、新たなノードの生成が不要になる。否定エッジを用いて表されたグラフから関数の値を求めるには、根から葉に至るまでに通った否定エッジの数の偶奇によって結果の(0,1)を反転すればよい。またグラフの間の演算についても、途中でエッジの属性を考慮する必要はあるが、従来の演算とほぼ同様に行うことができる。

無制限に否定エッジを使用すると、グラフの一意性が失われて、グラフの共有が行えなくなる。これを避けるために、次の制限を設ける。

- グラフの葉は必ず0とする。
- 各ノードにおける変数X_i=1を表すエッジおよびグラフの根を指すエッジのみに否定エッジの使用を限定する。

この制限によりグラフの一意性が保たれるので、従来通りサブグラフの共有を行うことができる。前節で述べた真理値表との組合せを行う場合は、真理値表の最下位ビットが0となるようにする。

否定エッジの使用により否定演算の手間が省け、ノード数が削減される。実際の回路によく現れるNANDやNOR、XORゲートでは、1回演算するたびに否定演算が必要であり、否定エッジの効果は大きい。

5.実験と評価

以上で述べた手法により、共有二分決定図のグラフの生成、演算等の処理を行う実験プログラム（C言語で約500行）を作成した。実験にはSUN3/60を使用した。

このプログラムで、ISCASのベンチマーク回路の全信号線の論理関数を、共有二分決定図で表す実験を行なった[3]。二分決定図では、入力変数の順序によって効率が大きく変化するが、ここでは与えられたネットリストに記述されている順番通りとした。またノード数の上限は128k個とした。演算結果テーブルの効率化などを行なった結果、使用記憶量は1ノード当たり約20バイトに抑えられている。

実験の結果、表1に示された回路について、全信号線の関数を同時に表現することができた。計算時間も数十秒以内と非常に高速である。

さらに、真理値表との組合せや否定エッジの使用の効果についても評価を行なった。真理値表については、ベンチマーク回路の入力数が多いので、4変数の削減による効果は少なかったが、10入力程度で複雑な回路を扱う場合には有効であると考えられる。一方、否定エッジに関しては、ノード数や計算時間が10～40%減少しており、有効であることがわかった。

6.おわりに

共有二分決定図の処理手法について述べた。我々は、この結果をテスト生成などに利用して成果をあげつつある [4]。不要な信号線のグラフを解放したり、変数の順序づけを工夫することにより、さらに大きな回路も扱うことができる。変数の順序づけの問題については、テスト生成で用いられるObservability measureを用いてよい結果が得られている[4]。共有二分決定図は多くの関数を同時に効率よく表すことができ、その用途は広いと考えられる。

謝辞

本研究に多大な協力を頂いた井置一哉氏を始め、矢島研究室の諸氏に感謝いたします。

参考文献

[1] R. E. Bryant: "Graph-Based Algorithms for Boolean Function Manipulation", IEEE Trans. Comput. C 35-8, pp. 677-691 (1986).

[2] 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", 情処研報, 87-DA-40, pp.111-116, (Dec.1987).

[3] 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", 情処研報, 88-DA-43-2 (Jul,1988).

[4] 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", 本大会予稿集, (Mar.1989).

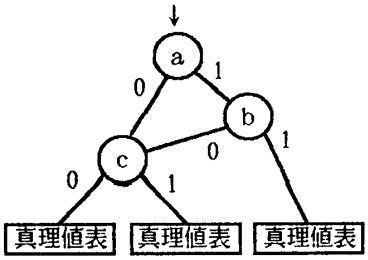


図4　真理値表との組合せ

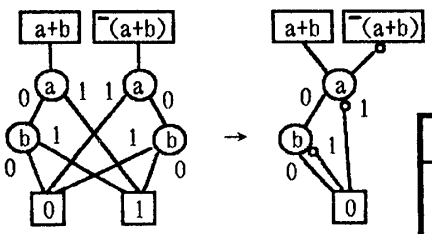


図5　否定エッジの使用

表1　実験結果（SUN3/60 8MByte）

				（全て使用）		（真理値表なし）		（否定エッジなし）	
回路	入力	出力	ﾈｯﾄ数	ﾉｰﾄﾞ数	CPU(秒)	ﾉｰﾄﾞ数	CPU(秒)	ﾉｰﾄﾞ数	CPU(秒)
c432	36	7	203	11483	5.8	11512	5.8	14155	7.9
c499	41	32	275	128489	35.3	128637	35.6	>128k	—
c1908	33	25	938	71400	34.5	71680	35.3	80747	53.4
c5315	178	123	2608	127924	36.1	127928	37.8	>128k	—