5K−1

不確定入力に対する論理関数の評価アルゴリズムの比較*

池田由佳　井口幸洋†
明治大学理工学部情報科学科 ‡

1　はじめに

あるn変数2値の論理関数fに対して，入力変数のうち幾つかは0または1に確定しているが残りの幾つかは不確定である場合がある．図1のような順序回路の論理シュミレーションが典型的な例である．論理関数がBDD（二分決定グラフ）[1][2]で与えられている場合，不確定な入力に対する論理関数の値を評価する深さ優先の探索アルゴリズムは，筆者らによって既に報告されている[3]．本稿では，論理関数の評価に幅優先探索アルゴリズムを用いた場合の評価方法と探索量について，深さ優先探索アルゴリズムとの比較検討を行う．またBDDをハードウェア化することで，不確定入力に対する論理関数の評価をハードウェアで行う方法についても簡単に述べる．

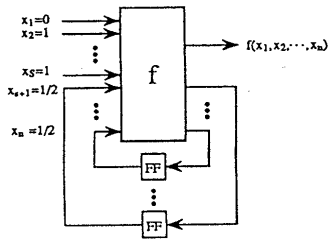


図1: 順序回路

2　不確定入力に対する評価

$B=0,1, V=0,1/2,1$ とする．但し1/2は0か1か不確定であることを表す真理値である．入力が0か1に確定している時、論理関数fは$f: B^n \rightarrow B$である．入力Iのうち幾つかの変数が不確定，即ち$I \in V^n$ ならば，Iの1/2を0か1で置換して得られる確定入力の集合をI^*とする．

［例1］ $I=(1/2,1,1/2,0)$ ならば，
$I^*=\{(0,1,0,0),(0,1,1,0),(1,1,0,0),(1,1,1,0)\}$ となる．

$f(I^*)=\{f(I')|I' \in I^*\}$ とすると，不確定入力Iに対する3値論理関数$\tilde{f}(I)$は以下のように定義される．

［定義1］ [4] [5]

$\tilde{f}(I)=0 \Leftrightarrow f(I^*)=\{0\}$
$\tilde{f}(I)=1 \Leftrightarrow f(I^*)=\{1\}$
$\tilde{f}(I)=1/2 \Leftrightarrow f(I^*)=\{0,1\}$

3　幅優先探索に基づいた不確定入力に対する論理関数の評価アルゴリズム

図2 (a)に5変数論理関数fが二分決定グラフで表された例を示す．(a)に例えば入力$I=(0,1/2,1,0,1/2)$を与え深さ優先探索で評価すると(b)のようになり，幅優先探索では(c)のようになる．すると,(b)で探索する枝は5本のみであるのに対し(c)では7本を探索する．

幅優先探索では，必要な枝だけを選択しグラフを一段ずつ下がって探索を行う．そのアルゴリズムを次に示す[6]．但し一度訪れたノードに再び訪れた場合はそこで探索を打ち切る．また，確定した入力すべてを探索しても出力値に到達しない場合，その時点で出力は1/2となる．これらの点は深さ優先探索アルゴリズムの場合と同様である．

```
top：二分決定グラフのルートノードを指すポインタ;
const：確定入力のノードの集合;
last：最も最近の出力値（初期値は未決定）;
node：現在見ているノードを指すポインタ;
{
 inqueue(top);  /*top を queue へ */
 while(queue != NULL){
   if (const == φ)  /* 確定入力がない */
     return 1/2;
   node = queue の先頭ノード;
   queue から今渡したノードを削除;
   if(node == 出力値){
     if(last == 未決定 || last == node の出力値)
       last = node の出力値;
     else if(last != 未決定)
       last = 1/2;
   }
   else if(node == 初めて来たノード){
     if(その node への入力が 1 または 0)
       const=const − { node };
     else{
       inqueue(0 側のノード); /*0 側のノードを queue へ */
       inqueue(1 側のノード); /*1 側のノードを queue へ */
     }
   }
 }
 return last;
}
```

4　各探索法の長所と短所

深さ優先探索は，一つの枝を選択することによって進める所まで行き，進めなくなったら戻って別の枝を選択して進んで行く．これに対し幅優先探索は，最初のノードを訪問した後このノードから到達可能なノードを順番に訪問する．これが終わると，さらにそれらのノードから到達可能なノードを訪問し，以下同様に繰り返していく．

どちらの場合も，一つのノードや一本の枝に対する処理は同じである．ノードの数は二分グラフによって決定されているので，問題となるのは探索する枝の本数である．但し出力値が0または1となる場合は，どちらの探索法を用いても探索する枝は同数である．したがって両者間の差が生じるのは出力値が1/2の時のみである．

グラフのルートノード近くのノードへの入力が1/2であった場合，幅優先探索では同レベルのノードへの探索は同時に行われるため，不確定入力によるノードの再訪問を繰り返さずに済むが、深さ優先探索では出力値まで探索を進めてからバックトラックを行うために，ノードの再訪問を繰り返しやすい．

---

*Comparison of Evaluation Algorithms of Switching Functions for an Unknown Input

†Yuka IKEDA,Yukihiro IGUCHI
‡Meiji University

不確定入力が多ければ出力値も 1/2 になる可能性が高い. この時枝分かれが増えるため，一見幅優先探索によってノードの再訪問を削減した方が良いように思われる．だが例 2 のように出力値のみを子として持つノードへの入力が不確定であれば，深さ優先探索を用いた方が効率的な評価を期待できることがわかる. また BDD による評価は, 未探索のノードが幾つ残っていようと出力値に 0 と 1 が得られた時点で $f = 1/2$ が決定し, 探索を打ち切ることができるという利点がある. したがって再訪問を削減することは幅優先探索の長所ではあるが，逆にそれが不必要な枝を探索してしまうという欠点も生じさせている.

さらに, 幅優先探索では探索すべきノードを待ち行列で記憶しているために，枝分かれが増えればそれだけ多くのメモリが必要となるという問題もある．一般に深さ優先探索を行った方がより効率的な探索ができる可能性が高い.

図 1 の論理回路を考える．順序回路 f の初期状態が 0 か 1 か不明であれば，フリップ・フロップからの出力も不明である．よってそれらの入力は不確定としなければならない．このようにどの入力が不確定になるかあらかじめわかっていれば、深さ優先探索の場合はその変数をルートノード近くに配値することで, 探索量が減ることが期待できる.BDD は変数順によって大きくその大きさが変わることが知られている. グラフの大きさを考慮しつつ,BDD の変数順を決定すると, より高速に実行できると考えられる.

5 BDD のハードウェア化

コンピュータ上のソフトウェアで論理関数から BDD を構成し, ソフトウェアで不確定入力に対する論理関数の評価を行う方法をここまでは扱ってきた. 以下では, ある論理関数 f を表す回路 C_f が与えられその回路の外部入力に不確定入力が与えられた時, 定義 1 に従って出力値を出す回路 C_f を構成する方法を述べる.

BDD の 1 つのノード i に着目すると,0 側につながる下位のグラフは i 番目の入力変数 x_i を 0 に,1 側は 1 に制限した部分関数を表す BDD となっている, 一方すべての入力変数が 0 または 1 に確定した入力 I が加わると, その出力値 $f(I)$ は図 3 (a) のように順に各変数の値に従って 0 側もしくは 1 側に進み,0 か 1 の葉に到達し出力値が求まることとなる.

見方を変えると, 各ノードは各変数によって切替えられるスイッチで, 葉の値 0 または 1 がルートノードまで伝搬されて出力値が求まると考えても良い. これは 2 入力マルチプレクサの機能に他ならない. 図 3 (b) のように各ノードをマルチプレクサに置き換えると,(a) の関数を実現するハードウェアが得られる. 不確定入力が加えられた時定義 1 に従い出力を求める回路を得るには, 詳細は省略するが, マルチプレクサを表 1 のような機能を持つ多値版のマルチプレクサに置換し C_f を得られる. 但し,2 値の論理関数で実現する場合には, マルチプレクサの入出力は 2 線式にし, 信号割り当ても工夫する等の必要がある.

6 まとめ

二分決定グラフを用いた不確定入力に対する論理関数の評価について，幅優先探索アルゴリズムの場合と深さ優先探索アルゴリズムの場合とを比較検討し，深さ優先探索の方が有効であることを述べた．また, ハードウェア化についても簡単に述べた.

今後の課題として，実際の大規模な回路にこれらの方法を適用し，上記の点について検討を行いたい.

参考文献

[1] S.B.Akers:Binary Decision Diagrams,IEEE Trans.Comput.,Vol.C-27,No.6,pp.509-516(1978).

[2] R.E.Bryant:Graph Based Algorithms for Boolean Function Manipulation,IEEE Trans.Comput.,Vol.C-35,No.8,pp.6 691(1986).

[3] 井口, 向殿: 二分決定グラフを用いた不確定入力に対する論理関数の評価アルゴリズム, 情報処理学会第 44 回全国大会 (1992).

[4] 向殿: 不確定入力に対する論理関数の評価法について, 情報処理学会設計自動化研究会夏期シンポジウム (1981).

[5] 井口, 向殿: 不確定入力に対する論理関数の評価の拡張と比較, 情報処理学会設計自動化研究会資料 21-3(1984).

[6] 石畑: アルゴリズムとデータ構造, 岩波講座ソフトウェア科学 3(1989).

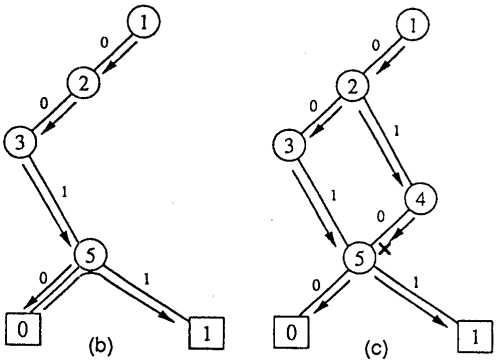
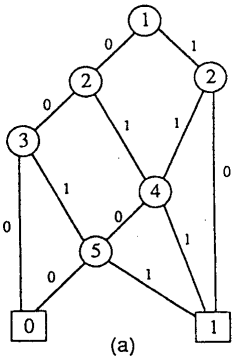


図 2: BDD 表現による論理関数 f

D0	D1	SEL	Y
0	X	0	0
1/2	X	0	1/2
1	X	0	1
X	0	1	0
X	1/2	1	1/2
X	1	1	1
0	0	1/2	0
1	1	1/2	1
otherwise		1/2	1/2

表 1:3 値 MUX の機能



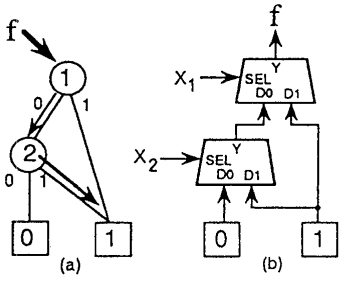


図 3: BDD の MUX 表現