

大規模有向グラフの効率的記憶法

5B-2

- アニエーリング法を利用したラベル付け -

赤嶺 暁子 脇田 竜次 内平直志

(株)東芝 システム・ソフトウェア技術研究所

1. はじめに

並行プログラムの解析や検証において、並行プログラムの挙動を、インターリーブ意味論に基づきグローバルな状態グラフで表現することが多い。例えば時相論理を用いた論理的検証では、検証項目を時相論理式 f で記述し、プログラムから生成された状態グラフ M が f のモデルになっているか否かを調べる。ところで、扱う並行プログラムが大規模になるに従い、生成される状態グラフのサイズは非常に大きくなり、例えば、我々が通常扱う並行プログラムでは、数万から数十万状態となってしまう。そこで必要となるのが、状態グラフ等の大規模有向グラフを、効率的に記憶する方法である。

さて、有向グラフの効率的記憶法の一つとして、2分決定グラフ(BDD: Binary Decision Diagram) [1] を利用したグラフ表現方法がClarkeらにより提案されている [2]。この方法では、有向グラフの各ノードに識別のためのラベル付けが行なわれるが、ラベル付けの方法によっては、BDD表現による効果が十分に出ないことがある。そこで、我々はBDD表現に効果的なラベル付け法を提案する。

2. BDDを用いた有向グラフ記憶法

(1) BDD

BDD(Binary Decision Diagram)は、ブール関数を、2分枝(ノードから出るエッジ数が2つ)のグラフで表現する。具体例を挙げると、ブール関数 $f_1(a, b, c, d) = a * b + c * d$ は、図1のBDDで表現される。これを $BDD_{(f_1)}$ と記す。ノード①は、変数 a を表し、ノード②からの0-エッジは、変数 a に0を割当て、1-エッジは変数 a に1を割当ててことを表している。また、ノード③、④は、 f_1 の真理値を表す。例えば、 $f_1(0, 1, 1, 1) = 1$ であるが、図1においてノード a, b, c, d に、それぞれ0, 1, 1, 1を割り当てるエッジをたどるとノード④に到達する事ができる。

BDD表現の利点は以下の二つである。

a. リダクション効果

BDD上の冗長な部分をリダクション操作により削除することで、ブール関数を非常にコンパクトに表現することができる。つまり、任意の関数 f に対し、BDD上の変数順を固定すれば、唯一最小のBDD表現(Reduced Graph)が得られることが、Bryantにより証明されている。図2は f_1 を表現するリダクション前のBDD、図1は、 f_1 の Reduced Graphである。

b. 演算

BDD上で、ブール関数のオペレータ $+, *, -$ の演算を直接行なうことができる。 op をオペレータとすると、 $BDD_{(f_1)}$ と $BDD_{(f_2)}$ に op を施したBDDを、 $APPLY(op, [BDD_{(f_1)}, BDD_{(f_2)}])$ で表すことにする。

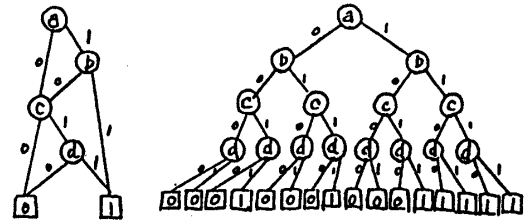


図1

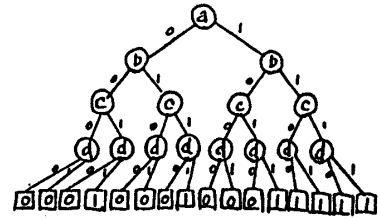


図2

(2) BDDを用いた有向グラフ記憶法

有向グラフの具体例を用いて、BDDによる有向グラフ記憶法を示す。

a. 有向グラフ

ノード数 $n=5$ 、エッジ数 $=8$ の図3の有向グラフを考える。ここで、 $s_0, \dots, s_4$ はノード名を表す。

b. ラベル付け

n 個のノードを、 m 個 ($2^m \geq n$ なる最小の m) の原子論理式の集合、 $P = \{p_0, \dots, p_{m-1}\}$ を使って区別する。即ち、 P の異なる全ての部分集合 (2^m 個) を各ノードにラベルとして割り付ける。

例えば、図3の有向グラフの場合、 $m=3$ 、 $P = \{p_0, p_1, p_2\}$ で、図4のようなラベル付けが可能である。

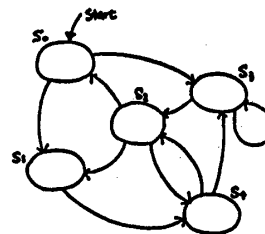


図3

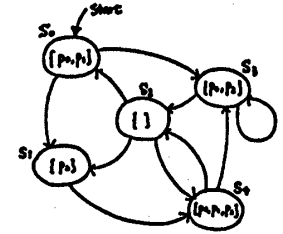


図4

c. 有向グラフのBDD表現

ラベル付けされた有向グラフのBDD表現法を示す。

ノードのBDD表現

ノード s のラベル $L_s \subseteq P$ の要素と、補集合 $\overline{L_s}$ の要素に $-$ 符号を付けたものの積で表されるブール関数 f_s を考える。この f_s を表現するBDDを、ノード s のBDD表現とする。例えば、図4のノード s_0 には、 $f_{s_0} = p_0 * p_1 * p_2$ が与えられ、 s_0 は図5のBDDで表現される。

エッジのBDD表現

ノード s から t へのエッジ e_{st} を考える。 s, t がそれぞれブール関数 $f_s(p_0, \dots, p_{m-1}), f_t(p_0, \dots, p_{m-1})$ で表されるとする。まず、エッジの始点、終点を区別するために $f_t(p_0, \dots, p_{m-1})$ を $f_t(p'_0, \dots, p'_{m-1})$ に書き換える。 $f_s(p_0, \dots, p_{m-1})$ のBDD表現を $BDDf_s$ 、 $f_t(p'_0, \dots, p'_{m-1})$ のBDD表現を $BDDf'_t$ とする。この時、 e_{st} のBDD表現は、 $APPLY(*, [BDDf_s, BDDf'_t])$ で得られるBDDである。例えば、図4のノード s_0 からノード s_1 へのエッジは、図6のBDDで表現される。

The Practical Method to Represent
a Large Directed Graph

A. Akamine, R. Wakizono, N. Uchihiro

TOSHIBA Corporation

有向グラフのBDD表現

個々のエッジ $e_0, \dots, e_{n-1}$ の BDD 表現 $BDD(e_0), \dots, BDD(e_{n-1})$ に対し、全てのエッジの論理和、即ち $G = \text{APPLY}(+, [BDD(e_0), \dots, BDD(e_{n-1})])$ で得られる G が、全エッジ $E = \{e_0, \dots, e_{n-1}\}$ をそれぞれで表現する BDD である。以上のように、全エッジを BDD 表現することにより、有向グラフを BDD 表現することができる。

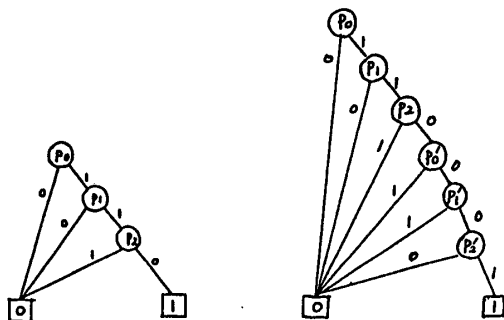


图 5

6

3. ラベル付けに対する考察

同一の有向グラフに対し、複数のラベル付けと変数順を与えて、BDD表現したところ、ラベル付けと変数順の与え方が、ノード数、即ちリダクション効果に大きな影響を与えることが明らかとなった。つまり、リダクション効果を十分に引き出すような、ラベル付けと変数順を考える必要がある。そこで、以下では効果的なラベル付け法を提案する。

(1) 基本的アイデア

$P = \{p_0, \dots, p_{m-1}\}$ に対し $\{p_0, \dots, p_{m-1}\}$ をベクトルとみなす。例えば $P = \{p_0, p_1, p_2, p_3\}$ の時、 $\{p_0, p_2\}$, $\{p_2, p_3\}$ がラベル付けされたノード s , t は、それぞれベクトル $v_s = (1, 0, 1, 0)$, $v_t = (0, 0, 1, 1)$ で表される。この時、ノード s から t へのエッジ $e_{s,t}$ に対し、差分ベクトル $v_{s,t} = v_t - v_s = (-1, 0, 0, 1)$ を、 $e_{s,t}$ の向きと呼ぶことにする。

さて、リダクションの効果は、有向グラフのエッジの向きに偏りがあるほど出易くなる。つまり、一つの有向グラフで、同じ向きのエッジが多いほどリダクション効果は大きくなる。何故なら、ある向きのエッジが全く無ければ、その部分がリダクションの対象になるからである。そこで、ラベル付けに規則性を持たせ、偏りのあるラベル付けを行ないたい。偏り具合を表す基準には色々なものが考えられるが、ここでは、規則的にラベル付けされたハイパー・キューブへのマッチング度で、ラベル付けの偏り具合（分散度）を表現する。

(2) n-キューブを利用したラベル付け法

直観的には、有向グラフを n -キューブ上に、なるべく多くのエッジが重なるように写して、各ノードに、対応するキューブ上のノードのラベルをラベル付けする。

Step 1

ノード数 = n の有向グラフ G に対し、 $2^m \geq n$ なる最小の m を考え、 m -キューブ Q_m を用意する。

Step 2

Gの各状態を、 Q_m の各ノードに重複せずに割り当てていく。割り当方は、全部で $2^m! / (2^m - n)!$ 通りある。

Step 3

各割当 w_i に対し分散度 $\sigma(i)$ を求める。
 $(i = 1, \dots, 2^m / (2^m - n)!))$

但し、

全エッジ $E = \{e_0, \dots, e_{n-1}\}$ のとき、
各エッジ $e_0, \dots, e_{n-1}$ に対し、
エッジの向きを $v_{\bullet 0}, \dots, v_{\bullet n-1}$ とし

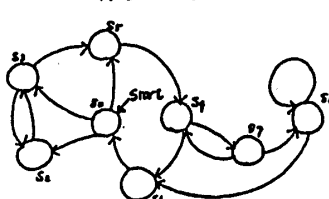
$$\sigma(i) = \sum_{j=0}^{n-1} (|v_{ij}| - 1)$$

と定義する。

分散度 $\sigma(x)$ が最小となる割当 w_x を求める。

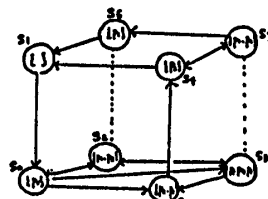
Step 4

Step 3 で求めた割当 w_x に基付き、各状態に割り当てられた Q_m のノードのラベルをラベル付けとする。例として、この手順に従い、図 7-1 の状態グラフ M ($n=8, m=3$) にラベル付けを行うと、図 7-2 が得られる。



状態グラフ M

图 7-1



ラベル付けされたM

图 7-2

(3) アニール法による近似解法

上述のラベル付け法の、Step 2, Step 3で、全ての割当を試行するには、莫大な計算量を要する。そこで、分散度 σ を評価関数として、アニーリング法により近似割当を求めることにする。

3. 評価実験

提案したラベル付け法の有効性を評価するために行なった評価実験について述べる。

(1) 実験方法

状態グラフに対し、アニーリング法を用いて、提案したラベル付け法によるラベル付けを行ない、実際に BDD 表現する。また、同一の状態グラフに対し、ランダムなラベル付けを行い、BDD 表現し、両 BDD 表現の使用ノード数の比較を行う。

(2) 実験結果

状態数10、エッジ数22の状態グラフM1、
状態数55、エッジ数90の状態グラフM2について実験した結果を下表に示す。

	BDDノード数	
	M1	M2
ラベル付け		
ラベル付け法による		
ラベル付け	37	170
ランダムなラベル付け		
1	42	197
2	43	207
3	45	193
4	38	200
5	42	197

4. まとめ

評価実験より、提案したラベル付け法では、ランダムなラベル付けに比べて、効率の良いラベル付けが得られることが示された。今後は、分散度の定義を改良することにより、更に効率のよいラベル付け法を検討していく。

〈参考文献〉

- [1]Randel E.Bryant : Graph-Based Algorithms for Boolean function Manipulation,IEEE transaction on Computer,vol.c-35,No.8
- [2]E.M.Clarke et.al : Sequential Circuit Verification Using Symbolic Model Checking,Private Memo .1989.