近接関数を用いた多出力論理関数の最小化について

7W-3

安岡孝一　高木直史　矢島脩三
京都大学工学部

1．はじめに

論理関数最小化問題、すなわち論理関数の最小積和形を求める問題は、論理設計における基本的問題として古くから研究されている。また、その拡張である多出力論理関数最小化問題は、ＰＬＡの最小化問題とあいまって、論理設計における重要な問題の１つとなっている。しかしQuine-McCluskey法に代表される従来の論理関数最小化アルゴリズムは、最小化を全素項の組み合わせ問題として解いているため、莫大な計算量が必要となる。

本稿では、近接関数を用いた多出力論理関数の最小化アルゴリズムを提案する。本稿のアルゴリズムは近接関数を用いることにより、全素項の生成をおこなうことなく、必須項、選択項を直接生成できる。また近接関数は、解の項数の下界の算出にも有効であることから、分枝限定法における枝刈りにも効果がある。したがって、従来のアルゴリズムに較べて、高速であると考えられる。

2．準備

2.1.論理関数

本稿ではｎ入力m出力の不完全指定多出力論理関数を扱う。そのような論理関数ｆに対し、ｆの第ｉ入力に０を入力することで得られる関数を$f(x_i=0)$、１を入力することで得られる関数を$f(x_i=1)$と表す（$1 \leq i \leq n$）。また、ｆの第ｊ出力の関数をf_jと表す（$1 \leq j \leq m$）。

全ての入力に依存する積項で、ただ１つの出力にしか現れないものを、最小項という。本稿では積項の表現として、多出力リテラル表現[1]を用いる。

2.2.近接関数

［定義］最小項ｅのｎ入力m出力論理関数ｆにおける近接関数とは、ｅを包含する全ての内項の論理和である。□

例として、図１の４入力３出力論理関数ψにおける最小項$y_2x_1x_2\bar{x}_3x_4$の近接関数γを、図２に示す。

近接関数に対して、以下の定理が成り立つ。

［定理１］ある未被覆最小項ｅの近接関数をｇとおくとき、ｇに含まれる未被覆最小項を全て包含しうる内項があるならば、その内項は必須項である。□

［定理２］未被覆最小項のある部分集合をＥとおき、Ｅの要素である最小項の近接関数の論理和が、全ての未被覆最小項を含んでいるとする。このとき、Ｅのいかなる真部分集合Ｄについても、Ｄの要素である最小項の近接関数の論理和が未被覆最小項を全ては含みえないならば、被覆を構成する内項はあと少なくとも|Ｅ|個必要である。□

3．近接関数を用いた論理関数最小化アルゴリズム

近接関数を用いた論理関数最小化アルゴリズムは、一般的な最小化アルゴリズムと同様、以下のようになっている。

step1．ｆの最小項が全て被覆に包含されるまで、step2からstep3までを繰り返す。

step2．ｆの必須項を１つ取り出す。なければ、ｆのある未被覆最小項について、それを包含する選択項全てについて分枝する。

step3.step2で取り出された内項を被覆に加え、その内項が包含する未被覆最小項を全てDON'T CAREとみなす。

生成された被覆のうち、項数が最小のものが解である。

なお、必須項および選択項の生成において近接関数を用いるため、アルゴリズムの前段階で近接関数を生成する必要がある。

3.1.近接関数生成アルゴリズム

ｎ入力m出力論理関数ｆにおける最小項ｅの近接関数ｇを生成するアルゴリズムを、以下に示す。

step1．ｆのDON'T CAREと１の部分を全て１とみなした関数をｇとおく。

step2．ｉ:=１tonについて、順にstep3をおこなう。

step3．ｅのリテラル表現がx_iを含んでいるならば、$g:=g \wedge g(x_i=1)$とする。さもなくば、$g:=g \wedge g(x_i=0)$とする。

step4．ｅが関係している出力をｋとおき、ｊ:=１tomについて、$g_j:=g_j \wedge g_k$をおこなう。

例として、図２の近接関数γの生成過程を、図３に示す。

3.2.近接関数を用いた必須項生成アルゴリズム

近接関数を用いて必須項を生成するアルゴリズムを、以下に示す。なお、このアルゴリズムの正当性は［定理１］に拠る。

step1.全ての未被覆最小項ｅについて、step2からstep3までを繰り返す。

step2．ｅの近接関数に含まれる未被覆最小項を、全て包含する最小の積項をｃとおく。

step3．ｃが内項ならば必須項とみなし、アルゴリズムを終了する。

3.3.近接関数を用いた選択項生成アルゴリズム

未被覆最小項ｅの近接関数を用いて、ｅを包含する選択

Exact Minimization of Multiple-Output Logic Function Using Neighfunction

Koichi YASUOKA, Naofumi TAKAGI, and Shuzo YAJIMA
Faculty of Engineering, Kyoto University

項の集合Pを生成するアルゴリズムを、以下に示す。

step1. P:=φ、E:={eの近接関数に含まれる値が1もしくはDON'T CAREの最小項}とおき、Eが空集合となるまでstep2からstep4までを繰り返す。

step2. Eの要素を1つ取り出しdとおく。D:={Eの要素でdと同じ入力の値の組み合わせに依存するもの}とおく。

step3. E:=E−Dとおき、D∪{e}の要素を全て包含する積項をcとおく。

step4. cに包含される未被覆最小項を、全て包含する最小の積項をpとおき、P:=P∪{p}とおく。

step5. Pの要素p全てについて、step6を繰り返す。

step6. {pに包含される未被覆最小項}⊂{qに包含される未被覆最小項}を満たすPの要素qがあれば、P:=P−{p}とおく。

4. 実現と評価

近接関数を用いた多出力論理関数最小化プログラムを、UNIX上のC言語を用いて作成した。ただし、3.のアルゴリズム中で述べられている分枝については、深さ優先探索とバックトラッキングによって実現している。なお、[定理2]によって現在探索中の解の項数の下界を算出し、それ以前に見つかっている解の項数を上回った場合には、枝刈りをおこなっている。また、3.3のアルゴリズムにおいて最小項eを選ぶ際に、近接関数の比較的小さいものを選ぶようにしており、これによって、分枝数を抑えている。

DAC86ベンチマーク[2]の関数に対して、sun-3/60上で最小化した結果を、表1に示す。表中、−となっているところは、計算時間が4時間を超えたため計測を打ち切ったもの、×となっているところは、記憶容量の制限（8MB）を超えたため実行できなかったものである。なお、9symとsao1については、計測を打ち切った時点で見つかっていた解の項数を、表中に括弧つきで示してある。

5. おわりに

近接関数を用いた多出力論理関数最小化アルゴリズムと、その実現について述べた。このアルゴリズムは近接関数を用いることにより、必須項や選択項の生成、および、解の項数の下界の算出を容易におこなうことができるため、非常に高速であると考えられる。

謝辞

御討論いただいた矢島研究室の諸氏に感謝します。

参考文献

[1]G.C.Vandling: The Simplification of Multiple-Output Switching Networks Composed of Unilateral Devices, IRE Transactions on Electronic Computers, Vol.EC-9, No.12 (Dec.1960), pp.477-486.

[2]A.J.de Geus: Logic Synthesis and Optimization Benchmarks for the 1986 Design Automation Conference, ACM/IEEE Proceedings of 23rd Design Automation Conference (Jun.1986), pp.78.

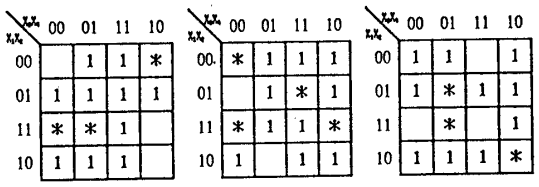

図1　4入力3出力論理関数ψ

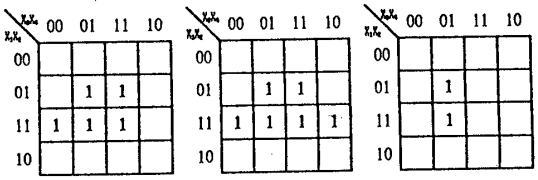

図2　$y_2x_1x_2\bar{x}_3x_4$のψにおける近接関数γ

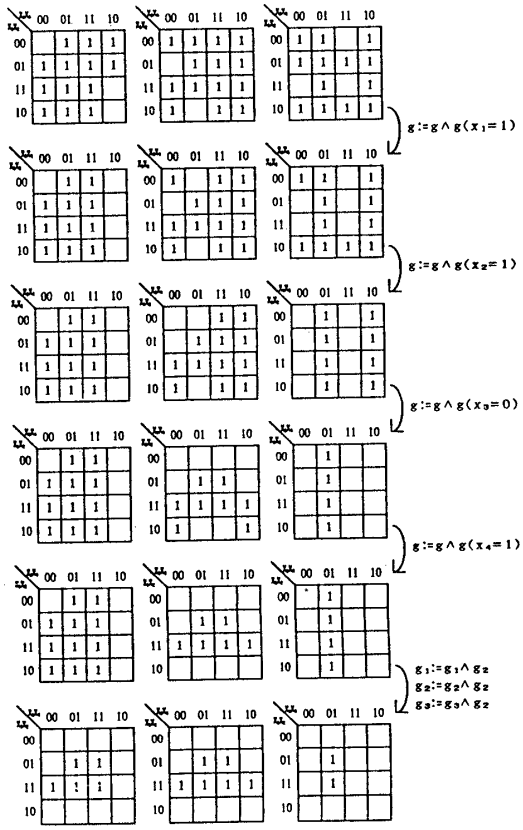

図3　近接関数の生成

関数名	入力数	出力数	CPU時間(秒)	項数
5xp1	7	10	7995.8	63
9sym	9	1	−	(88)
alupla	25	5	×	×
bw	5	28	0.6	22
duke2	22	29	×	×
f2	4	4	0.1	8
rd53	5	3	0.1	31
rd73	7	3	0.8	127
sao1	8	3	−	(155)
sao2	10	4	0.4	58
vg2	25	8	×	×

表1　評価結果