テスト生成の静的学習における間接含意の習得法について

2L－1

野上 貴一郎† 市原 英行‡ 梶原 誠司†‡ 樹下 行三‡
†九州工業大学情報工学部電子情報工学科
‡大阪大学大学院工学研究科応用物理学専攻

1 はじめに

テスト生成アルゴリズムのSOCRATES［1］は、冗長故障判定を含めた処理能力の高さから、組合せ回路のテスト生成で一般によく用いられている。SOCRATESの処理能力が高い理由の一つとして、含意操作において間接含意を用いることが挙げられる。間接含意は前処理の静的学習で各々の信号線に対する含意操作で求めるが、得られる間接含意の数は静的学習においてどの信号線から処理するかにより異なる。

本研究では、信号線番号を決定するラベリングアルゴリズムの違いにより、得られる間接含意数がどのように変化するかを調べ、更にテスト生成におけるバックトラック回数にどれくらい影響するかを検証する。

2 間接含意

2.1 間接含意の習得

テスト生成において、論理回路内のある信号線の値が決まることにより一意的に決まる他の信号値を求める操作を含意操作という。含意操作に用いる信号線の含意関係は、直接含意と間接含意の２つに分類できる。直接含意とは、ゲートの入出力関係だけから求められる含意であり、例えば、図1における "A=1 ⇒ D=1" または "A=1 ⇒ F=1" などである。一方、間接含意とは、ゲートの入出力関係から直接得ることはできない含意関係である。例えば、図1の直接含意 "A=1 ⇒ F=1" に対して、この命題の対偶から "F=0 ⇒ A=0" という含意が得られる。このような含意を間接含意という。図2に示すようにF=0のとき入力線D,Eの値は一意的に決まらないが、この間接含意を用いれば、信号線Aの値を0に決定できる。

テスト生成の含意操作において、値が決まる信号線の数が多ければ探索空間を削減することができるため、間接含意を利用することがテスト生成を効率化する。

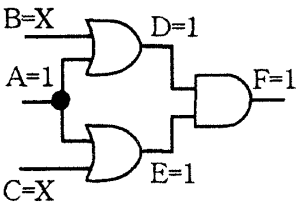

図1 直接含意
A=1 ⇒ F=1

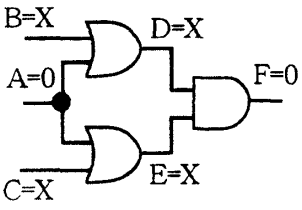

図2 間接含意
F=0 ⇒ A=0

2.2 含意操作の順序と間接含意数

間接含意は、直接含意に基づく含意操作から求められるが、この含意操作においても、それまでに得られている間接含意を利用できる。例えば、図３において F=0 に対して含意操作を行う場合を考える。もし、この含意操作の前に A=1 に対する含意操作を実行し、間接含意 "F=0 ⇒ A=0" が 得られていたとするならば、F=0 に対する含意操作により "F=0 ⇒ G=0"が得られ、この対偶から図４に示すような "G=1 ⇒ F=1" の間接含意を得ることが出来る。逆に F=0 に対して含意操作の前に "F=0 ⇒ A=0" が得られていなかったとすると、この含意操作で値を決定できる信号線はなく、"G=1 ⇒ F=1" の間接含意を見つけることは出来ない。このように、得られる間接含意の習得数は、静的学習においてどの信号線から含意操作を実行するかにより異なることになる。

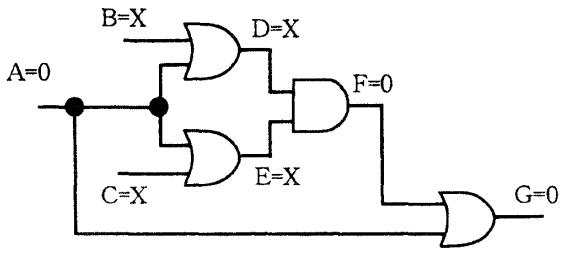

図3 F=0 に対する含意操作

A method for obtaining indirect implications in static learning
† Kyushu Institute of Technology, Dept. of Computer Science and Electronics
‡ Osaka University, Dept. of Applied Physics
† Kiichirou Nogami, ‡ Hideyuki Ichihara,
†‡ Seiji Kajihara, ‡ Kozo Kinoshita

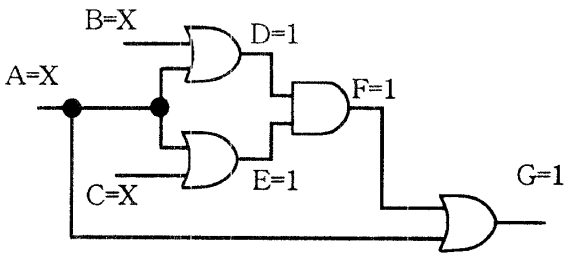

図4　間接含意G=1 ⇒ F=1

3　実験結果

3.1　ラベリング手法

本研究では、静的学習において含意操作を行う信号線の順序を決める信号線の番号付け(ラベリング)のアルゴリズムを４種類用意し、どのラベリング法が間接含意を多く求めることができるか実験する。

４種類の内の２種類のラベリングアルゴリズムは、幅優先順と深さ優先順に基づくものである。なお、どちらの方法も、外部入力から外部出力に向かって信号線番号を割り当て、どの信号線の番号もその信号線の入力信号線の番号より大きくなるように行った。例えば図4の回路に対して、幅優先順では、(B → A → C) → (D → E) → F → G となる。また、深さ優先順では (B → A) → D → C → E → F → G となる。ここで、括弧内の信号線の順序に関しては任意に選択する。

残りの２種類のラベリングアルゴリズムは、上述の幅優先順と深さ優先順に基づいて決定した信号線の番号を逆にしたものである。このため、どの信号線の番号もその信号線の入力信号線の番号より小さくなる。

3.2　結果と考察

前述のアルゴリズムについて、ISCAS'85のベンチマーク回路に対し、間接含意の習得数とそれらを用いたときのテスト生成におけるバックトラック回数の違いを調べる実験を行った。実験結果を表1と表2に示すが、表中の各欄は、

[lab1] 外部入力側から幅優先順にキュー構造を用いたラベリング

[lab1R] [lab1]の逆順にラベリング

[lab2] 外部入力側から深さ優先順にスタック構造を用いてラベリング

[lab2R] [lab2]の逆順にラベリング

を用いて静的学習を実行した結果、得られた間接含意数を示す。ただし、[1]の学習基準に従って保存すべき含意を決定したので、一部直接含意を含んでいる。幅優先と深さ優先のどちらにおいても、入力側から番号付けした結果の方が間接含意の習得数が多いことがわかった。これは、出力側の信号線の値が決まることにより入力側の信号線の値が決まるような間接含意が多いことを意味する。また、入力側から番号付けした結果を幅優先と深さ優先のアルゴリズムで比べた場合、幅優先の方が間接含意の習得数が多いことがわかった。

テスト生成におけるバックトラック回数に関しては、ほとんどの回路で差を確認できなかったが、c2670 では、得られた間接含意の習得数が多いほどバックトラックの回数が減ることが確認できた。

4　まとめ

本研究では、静的学習において含意操作を行う順序を変えることで得られる間接含意数が変化することを述べた。また実験から外部入力側から幅優先に基づいた信号線のラベリング手法が間接含意を多く得るのに有効であることがわかった。

参考文献

[1] M. Schulz, E. Trischler, and T. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," IEEE Trans. on CAD., pp. 126-137, Jan. 1988.

表1　静的学習で得られた含意数

回路	lab1	lab1R	lab2	lab2R
c432	142	133	142	133
c499	424	424	424	424
c880	291	287	291	287
c1355	1072	1072	1072	1072
c1908	1613	1094	1470	1130
c2670	1777	1397	1556	1434
c3540	5937	5210	5859	5258
c5315	3598	2038	2560	2122
c6288	2171	1702	2142	1703
c7552	11574	5004	8615	5372

表2　バックトラック数

回路	lab1	lab1R	lab2	lab2R
c2670	328	671	410	410
c7552	1200	1206	1200	1200