# 論理回路簡単化問題に対する例題生成

2T−5

日野 健介　岩間 一雄
九州大学工学部

## 1. はじめに

我々はこれまでの研究[1]で，NAND素子に制限された論理回路に対し，1回の操作が多項式時間で実行できる基本操作の集合を提案し，その基本操作集合が任意の回路から任意の等価な回路への変換に対して完全であることを示した．任意の回路間の変換が可能であるといっても回路を簡単化する方向に利用することは明らかに容易ではない．しかし，回路を“複雑化”する方向なら，例えばある程度ランダムに規則を適用するだけでも目的を達成できるのではないかと考えられる．本稿では，そのような考え方のもとに，簡単な回路から複雑な回路を導くシステムの第1版を開発したので報告する．

本システムを論理合成アルゴリズム(多くの場合回路の簡単化アルゴリズム[2][3])の為のテスト例題生成に用いることを考えている．固定されたベンチマークテストは，その固定された例題集合に対してアルゴリズムをチューンアップするという不正が原理的に防止できないという欠点を有しており，その意味で上記のようなある意味でのランダム生成が重要になる．ただし，単にランダムにテスト問題(回路)を生成した場合，その問題に対する答(改良された回路)が出題者には判らないという問題が生じる．答(簡単な回路)から問題(複雑な回路)を作り上げるという考え方の重要性が理解していただけるであろう．

## 2. 変換ルール

例えば図1のような回路を我々は以下のような式の集合で表現する．

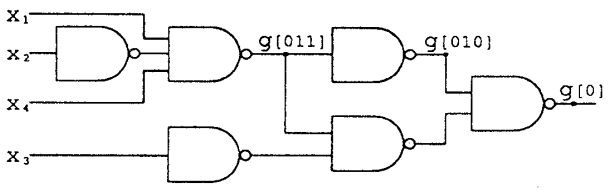


図 1: 4変数の NAND 回路

$$g[0] = (g[010], (g[011], (x_3)))$$
$$g[010] = (g[011])$$
$$g[011] = (x_1, (x_2), x_4)$$

回路をいくつかの部分回路に分けて各部分回路にラベルをつけておき，他の部分回路にラベルがあれば参照される．これによりファンアウト数が2以上の回路を表現することができる．ここで，$(\cdots)$はいわゆるNANDゲートを表す．

文献[1]で得られた完全な操作規則集合は以下の11個のルールから成る．

(1) $(1) \Longleftrightarrow 0 \qquad (0) \Longleftrightarrow 1$
(2) $(x,x) \Longleftrightarrow (x)$
(3) $(x,(x)) \Longleftrightarrow 1$
(4) $x,((y,z)) \Longleftrightarrow x,y,z$
(5) $x,y \Longleftrightarrow y,x$
(6) $(x,1) \Longleftrightarrow (x)$
(7) $((x)) \Longleftrightarrow x$
(8) $(x,(y,z)) \Longleftrightarrow (((x,(y)),(x,(z))))$
(9) $g[\ell] = f$ という部分式の定義が存在するとき，$g[\ell] \Longleftrightarrow f$
(10) ラベル $g[\ell]$ がいかなる部分式にも現れないとき，ラベル $g[\ell]$ の部分式の定義を，回路を表現する部分式の集合から除去できる．これを消去とよぶ．
(11) ラベル $g[\ell]$ の部分式の定義が存在しないとき，$g[\ell]$ = 部分式の形の定義を集合に付加できる．ただしそのサイズ(長さ)は，現在の式のサイズの多項式で制限される．これを生成とよぶ．

ここで，各 $x,y,z$ は任意の部分回路(定義は省略)にマッチする．$\Longleftrightarrow$ は左辺から右辺および右辺から左辺への両方向の操作が可能であることを意味する．以後例えば(3)の $\Longrightarrow$ を(3)で，$\Longleftarrow$ を(3')で表すことにする．例えば規則(4) $x,((y,z)) \Longrightarrow x,y,z$ を式

$$((x_1),((x_2,(g[01],((x_4,x_1))))),x_3)$$

に適用する場合，適用できる場所が2カ所存在し，次の2式のいずれかに変換される．

$$((x_1),x_2,(g[01],((x_4,x_1))),x_3)$$
$$((x_1),((x_2,(g[01],x_4,x_1))),x_3)$$

なお，本稿の範囲ではルール(9)以降は使用しない．すなわち木状(ファンアウト数が1)の回路のみを扱う．

---

A test-case generation system for the problem of simplifying switching circuits
Kensuke HINO, Kazuo IWAMA
Kyushu University

## 3. 例題生成システム

### 3.1 基本方針

(i)17種類のルールのうちの1つをランダムに選び，(ii) そのルールを現在の式に対して適用する．適用できる場所が2つ以上ある場合はいずれか1カ所をやはりランダムに選ぶ．(i),(ii) を規定回数繰り返す．

これが生成の基本的な手続きであるが，注意を要するルールが存在する．例えばルール (3') $1 \Longrightarrow (x,(x))$ の右辺の $x$ はどのような部分式でもよい．我々は $x$ として，現在の式のある部分式を任意に選択し，それをあてることにした．ランダムに部分式を生成する方法も有力であろうと思われる．また，このルール以外にも式を長くするルールがいくつかある (ルール (2'),(4'),(7'))．これらのルールをあまり頻繁に使うと，式そのものの複雑度を上げることなく長さのみをいたずらに増大させるので，注意が必要である．

### 3.2 データ構造

2節に述べたように，我々は回路を文字列で表現している．しかし，計算機上では木構造のデータとしてシステムを実現している．あるルールを適用しようとするとき，まず回路にルールの左辺 (右辺) にマッチする部分列があるかを調べる．このとき各 $x,y,z$ が部分式になっているかを調べる必要があるが，木構造では任意の節点を根とする部分木は必ず部分式になっている．また変換ルールも木の操作で容易に実現できる．各節点は次のような構造体である．

```
struct vertex{
   char  label[10];
   struct vertex  *par,*bros,*son;
}
```

### 3.3 ルールの適用

計算機上におけるルールの適用について，例えば (3') $1 \Longrightarrow (x,(x))$ は次のように実現されている．まず現在の回路に対してルールが適用できる場所を全て求める．この場合，ラベルが1であるような節点を指すポインタを求める．次にこの中からランダムに一つを選び，ポインタが指す部分木1を部分木 $(x,(x))$ に置き換える．$x$ としては回路の中から任意の節点を根とする部分木を選んでくる．以上が (3') の1回の適用となる．

## 4. 実験結果

現在のところルールの適用順に関してはいっさい制御してないので，各ルールの適用頻度のみ制御の対象となる．以下の例では初期回路は全て $((x1,x3),((x3),x4),((x2),x3))$ である．初めに全てのルールを等確率で50回適用した場合の生成例を図2に与える．

明らかに多重のカッコが頻繁に現われており，意味のある例題にはなっていない．そこで，2重カッコを取り去るルールの適用頻度を上げることが考えられる．また式の長さを無意味に増大させるような規則の適用を制限した．(8) は回路をより複雑にする性質があると考えられるため，この適用頻度も上げた．いろいろな実験の結果，具体的には以下のような頻度に到達した．(4) は他の30倍，(8),(8') は3倍の頻度で適用し，(2'),(3') については全体で2回に制限した．この頻度でルールを500回適用した場合の生成例を図3に与える．

(((((((x1,x3))),((((x3,(0)))),((((x4,x4))))))))),((((x2,((((((((((x1,x3))),((x3,1))),((((x1,x3))))),1,(((((x1,x3))),((((x3,1)))))),((((x4,x4,x4))))))),((((x1,x3))),((((x4,x4))))))),(((((((x1,1,x3))),((((x3,(0)))),((((x4,x4))))),(((((x4)))))))))),x3))))

図 2: 等確率で適用した場合

(((((((((x2,(x3,x1),((x3),x4)),((x4,(x3),1),(x1,x3),1),(0)),(x1)),((((x3),0),((x3,x1),((((x3),x4),x2),(x3),x4)),(0,(x1))),(x3))),x3),(((((((x2,(0),(x1,x3),((x3),x4,(0))),x4,(x3)),(x1),(((x1,(x2)),(x1,((x1,x3))),(x1,(((x3),x4,1)))),x3,1)),((1),(x1,x3)),(((x3),(x2,((x3),x4)),x4),(x1,x3))),((((x2,(x1,x3),((x3),1,x4,(0))),x4,(x3)),(x1),(x1,(x2,(x1,x3),((x3),x4)),1,x3)),x3)),(0)),(x4))),x2),((x1,(0),x3),((((x3),x4),x2),(x3),x4),(x3,1))))

図 3: 適用頻度を変えた場合

## 5. むすび

現時点ではプログラムの第1版が完成したばかりであり，これから様々な改良が必要と思われる．重要なものとして，(i) ルールのより適切な適用頻度を求める，(ii) 適用する場所を決める時，どこを優先的に選ぶか (例えば一番長い列を選ぶ等) を考える，(iii) ルールの適用順序もある程度制御する，(iv) 全てのルールを適用可能にする (ファンアウト数が2以上の回路も扱えるようにする)，などがある．実際にいくつかの簡単化アルゴリズムを本システムで生成した例題によってテストしてみて，この生成システムの問題点を探り，改良を進めていきたい．

## 参考文献


[1] 日野, 岩間, 澤田, "素子制限のある論理回路を等価変換するための基本操作集合について", 平成5年3月情報処理学会全国大会.

[2] R.K.Brayton, et al. "MIS: Multi-level Interactive Logic Optimization System", *Trans. on CAD*, Vol.CAD-6, pp.1062-1081, Nov. 1987.

[3] S.Muroga, et al. "The Transduction Method—Design of Logic Networks Based on Permissible Functions", *IEEE Trans. on Comput.*, Vol.38, No.10,October 1989.