アルゴリズム　30－20
（1992. 11. 21）

組合せ論理回路を等価変換するための基本操作集合について

日野 健介　岩間 一雄　澤田 直

九州大学工学部

論理設計は，基本的にはある回路をより良い回路に変換する技術に基づいているといってよい．またアルゴリズム評価のためのテスト例題生成においては，逆に，回路をより悪い回路に変換する必要がある．本稿では，このような論理回路の(等価)変換をより形式的に議論する．変換を1回の操作が多項式時間で実行できる基本操作の列で定義し，任意の回路から任意の等価な回路へ変換ができるためには，どのような基本操作の集合が必要かを議論する．本稿では，ファンインに制限のないAND素子とOR素子，及びNOT素子を使う組合せ回路のみを扱う．

キーワード：例題生成，等価変換，論理回路

On a complete set of basic operations to transform between equivalent switching circuits.

Kensuke Hino, Kazuo Iwama and Sunao Sawada

Department of Computer Science and Communication Engineering
Kyushu University

Fukuoka 812, Japan

What plays a basic role in designing switching circuits is a technique of transforming such circuits into better circuits. In contrast, an opposite approach, transforming circuits into worse ones, is important in generating test problems to evaluate the performance of the designing algorithms. In this paper, we discuss such transformation between equivalent switching circuits in the following way. The transformation is defined as a sequence of basic operations each of which can be carried out in polynomial time. We shall give a (finite) set of basic operations that is enough to transform each circuit into each equivalent one. The circuit in this paper is a combinational circuit that can use AND and OR gates of unlimited fan-in and NOT gates.

Keywords: Instance Generation, Equivalent Transformation, Switching Circuit

1. はじめに

論理回路の自動設計は近年その重要性を増々大きなものにしている．数多くの手法や考え方が提案されているが，その代表的なものは，まず何らかの方法で初期回路を設計し，段階的により性質の良い回路に変形していく [3][4] というものである．また，最近注目を集めているアルゴリズムの効率を実験的に評価するためのテスト例題の生成 [8] では，逆に性質の良い回路 (アルゴリズムが導き出すべき答) から悪い回路 (アルゴリズムの入力とする) に変形できる必要がある．いずれの場合も，論理回路の等価変換 (回路が実現する論理関数を変えることなく変形する) が鍵になる．我々は，"変換" を1回の操作が多項式時間で実行できる "基本操作" の列で定義し，任意の論理回路から任意の等価な論理回路へ変換ができるためには，どのような基本操作の集合が必要かを議論する．本稿では，論理回路としてはファンイン数に制限のない AND 素子と OR 素子，および NOT 素子が使える (フィードバックのない) 組合せ回路を対象とする．

本テーマは明らかに基本的問題であり，多くの結果が既に得られていると考えるのが普通である．しかし，調査の結果は必ずしもそうではない．たとえば，多くの教科書 ([1],[2] 等) には，論理式を変形するための公式 (例：ドモルガンの法則) が示されている．しかし，そのような公式のどのような集合がどのようなクラスの回路 (あるいは論理式) に対して完全であるか，あるいは1回の公式の適用が多項式時間で可能であるかどうかといった，本論文のポイントとなる事項に関しては全くといっていいほど説明がない．さらに，いわゆる公理系としては，命題論理に対する Kleene の公理系が有名である．しかし，それは論理体系の完全性に関してのものであり，ここでの等価な式 (回路) 間の変換という目的とはずれている．

上述のように，我々はこのような変換システムの応用の可能性として，方向の異なる2種類を考えている．ある回路からより良い回路 (例えば段数が少ないとか素子数が少ないとか) へ変換する場合について先に述べる．通常そのような変換は，綿密なヒューリスティクスによって行なわれることが多く，1回の変換で回路が必ず改良され，しかもその改良の程度がかなり大きくなることを保障している．しかしその反面，1回の変換に多くの時間がかかる．例えばそのような変換の代表例であるトランスダクション法 [4] は，1回の変換に指数時間を要している．しかし，最適化アルゴリズムやゲームアルゴリズムの最近の傾向を見ると，このような "1回の変換に時間をかけてかなり良いものにする" という基本方針は主流ではないように思われる．むしろ，"簡単な変換で現在の解の近傍を探索する" という手法がほとんどの場合の基本戦略になっている．さらにこの戦略は，実用的な多くの場合において驚くほどうまくいくことが実験等によって確かめられている．例えば巡回セールスマン問題 [9] 等の NP 完全問題が良い例になっている．別の例として最近我々を含む3つのグループによってほぼ同時に発見された CNF 論理式の充足解を求めるアルゴリズムを紹介する [5][6][7]．これは，最初に全ての論理変数に0と1をランダムに代入し，それが解になっているかどうか調べ，もし解になっていなければ，カルノー図上でその解の周りのセルでそれをおおっている節の数が解をおおっている節の数より少ない方へと進むことを基本戦略とする．これも非常に単純な考え方ではあるが，その効率は目をみはるものがあることがそれぞれの論文によって示されている．論理回路の変換の場合も，現在の回路の近傍を探索することによって (ステップ数はかかるかもしれないが) より良い回路に近付けることができるのではないだろうか．その場合，本稿での立場の変換システムが威力を発揮するのではないだろうかというのが我々の主張である．

アルゴリズムの効率を評価するためのテスト例題生成への応用については，より確実な期待を持っている．この場合，初期回路からランダムに変形していけば (多くの場合より悪い回路に変わっていくことが期待されるので) 目的が達成されよう．固定された例題集合による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．上述のような答から問題を作り上げるという戦略の重要性が理解していただけるであろう．

2. 回路の定義

2.1 基本方針

例えば，図1のような "回路" が与えられたとしよう．我々はこれを以下のような式の集合で表現する．

$$g[01] = \vee(g[010], \wedge(g[011], x_3))$$
$$g[010] = \langle g[011] \rangle$$
$$g[011] = \vee(x_1, x_2)$$

すなわち，回路をいくつかの部分回路に分けることが必要になる．ただし，ファンアウト数が1の木状回路の場

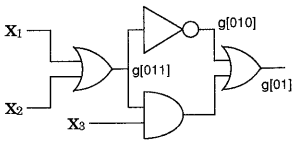

図 1:

合は，これら部分回路は必要なく，単一の式で表現することも可能である．以下に回路のより厳密な定義を与える．

2.2 部分式の定義

まず，**部分式**を以下のように $\{\mathbf{0}, \mathbf{1}, 0, 1, x, g, \vee, \wedge, (,), [,], \langle, \rangle\}$ 上の文字列として定義する．

(1) $\mathbf{0}$ 及び $\mathbf{1}$ は部分式

(2) $x[a]$ は部分式 (a は二進数)

(3) $g[\ell]$ は部分式 (ℓ は 0 と 1 の文字列)

(4) S が部分式ならば $\langle S\rangle$ は部分式

(5) $S_1, S_2, \cdots, S_m (m \geq 1)$ が部分式ならば，
$\vee(S_1, S_2, \cdots, S_m)$ は部分式

(6) $S_1, S_2, \cdots, S_m (m \geq 1)$ が部分式ならば，
$\wedge(S_1, S_2, \cdots, S_m)$ は部分式

ここで，$\mathbf{0}$ ($\mathbf{1}$) は，論理の偽 (真) を，0 と 1 は (2) と (3) の a と ℓ をそれぞれ表現するという意味で (特に必要というわけではないが) 区別して用いる．また $x[a]$ は入力変数を表している．(以下，簡単のため $x[a]$ を x_i(ただし i は a を十進数に読みかえたもの) と表記することがある)．$\langle S\rangle$ は，いわゆる S の否定を表し，$\vee(S_1, S_2, \cdots, S_m)$ と $\wedge(S_1, S_2, \cdots, S_m)$ は，それぞれ m 入力の OR ゲートと AND ゲートを意味している．また $g[\ell]$ をラベルとよぶ．ラベル $g[a_1], g[a_2]$ があるとき，それらの大小関係を次のように定義する．a_1, a_2 は 0 と 1 の文字列であるが，辞書順に a_1 が優先しているとき，$a_1 > a_2$ とする (ただし $0 > 1$ とする)．つまり最大のラベルは $g[0]$ である．

我々は，集合 $\{W_1, W_2, \cdots, W_m\}$ を**式**と呼ぶ．ここで，W_j は "$g[\ell]$ = 部分式" の形の文字列である．以下，この "部分式" をラベル $g[\ell]$ の部分式 (の定義) とよぶことがある．

2.3 回路の定義

上記で定義した式が次の条件を満たすとき，**回路**とよぶ．

(i) ラベル $g[0]$ の部分式が存在する．

(ii) ラベル $g[a_1]$ の部分式に $g[a_2]$ が現れるなら，$a_1 > a_2$ である

(iii) ある部分式の中に $g[\ell]$ が現れるとき，ラベル $g[\ell]$ の部分式の定義が必ず存在する．

(iv) 特定のラベル $g[\ell]$ の部分式が 2 つ以上存在しない．

(v) 式に現れる入力変数 x_i は，x_1 から始まり，$x_2, x_3, \cdots$ のように途切れることなく x_n まで続く．このとき n 変数の回路であるという．

(i) は，全体の回路を表現する部分式に対してラベル $g[0]$ を割り当てるという約束を表す．(ii) は，我々が扱う回路は組合せ回路であるので，回路にフィードバックを許さないための条件である．(iii) では，ゲートへの入力線のすべてに，必ず入力があることを保障している．また (iv) は，一つの入力線に対して複数の入力を許さないというものである．なお，式が回路を表しているとき，式の各部分式を**部分回路**とよぶこともある．

以上が本稿における回路の定義である．次節ではこれらの回路に対する基本操作について議論する．

3. 基本操作

本論文の目的は，回路 F を等価な回路 G に何回かの**基本操作**を適用することによって，変換することである．ここで，基本操作は，一部の例外を除き $g \Longrightarrow h$ の形で表され，有限個の**操作規則**の集合の中から 1 つを選んで，それを適用することによって行なわれる．例えば操作規則

$$\wedge(x, \wedge(y, z)) \Longrightarrow \wedge(x, y, z)$$

は，ある部分回路 f(を表す文字列) の中に，規則の左辺である $\wedge(x, \wedge(y, z))$(ただし，x, y, z は任意の部分回路でよい) が現れたら，その部分を規則の右辺で置き換えるような操作に対応している．この操作規則を，例えば，部分回路

$$\vee(\langle x_1\rangle, \vee(x_2, \vee(g[01], \langle\wedge(x_4, x_1)\rangle)), 0, x_3) \qquad (1)$$

に適用してみよう．$x = x_2$, $y = g[01]$, $z = \langle\wedge(x_4, x_1)\rangle$ と置くことによって，

$$\vee(\langle x_1\rangle, \vee(x_2, g[01], \langle\wedge(x_4, x_1)\rangle), 0, x_3) \qquad (2)$$

と変形される．こうして，部分回路 (1) を含む回路が 1 回の基本操作によって，(2) を含む回路に変換されたわけである．なお $x = \langle x_1\rangle$, $y = x_2$, $z = \vee(g[01], \langle\wedge(x_4, x_1)\rangle)$ と置くことによって (1) は，

$$\vee(\langle x_1\rangle, x_2, \vee(g[01], \langle\wedge(x_4, x_1)\rangle), 0, x_3)) \qquad (3)$$

と変形することも可能であることに注意されたい．

より形式的には，我々は基本操作を，関数

$$T(f_1, r, k) = f_2$$

で表す．ここで，f_1 は基本操作を行なおうとしている (元の) 回路，r は操作規則，f_2 は基本操作によって変形された (結果の) 回路である．一般に f_1 と r を指定しただけでは，上で述べたように様々な可能性があって f_2 が一意に定まらない．そこでパラメータ k(整数) を与える．これによって可能なすべての f_2 の中から適当な順序づけによる k 番目を指定する．

本論文で提案する操作規則集合 (公理系) は，以下のようなものである．

公理系

(1) $\langle \mathbf{1} \rangle \Longleftrightarrow \mathbf{0}$　　$\langle \mathbf{0} \rangle \Longleftrightarrow \mathbf{1}$

(2) $\vee(x) \Longleftrightarrow x$　　$\wedge(x) \Longleftrightarrow x$

(3) $\vee(x,x) \Longleftrightarrow x$　　$\wedge(x,x) \Longleftrightarrow x$

(4) $\vee(x,\langle x \rangle) \Longleftrightarrow \mathbf{1}$　　$\wedge(x,\langle x \rangle) \Longleftrightarrow \mathbf{0}$

(5) $\vee(x,\vee(y,z)) \Longleftrightarrow \vee(x,y,z)$
$\wedge(x,\wedge(y,z)) \Longleftrightarrow \wedge(x,y,z)$

(6) $x,y \Longleftrightarrow y,x$

(7) $\wedge(x,\mathbf{1}) \Longleftrightarrow x$

(8) $\langle\langle x \rangle\rangle \Longleftrightarrow x$

(9) $\langle \vee(x,y) \rangle \Longleftrightarrow \wedge(\langle x \rangle, \langle y \rangle)$

(10) $\wedge(x,\vee(y,z)) \Longleftrightarrow \vee(\wedge(x,y),\wedge(x,z))$

(11) $g[l] = f$ という部分式の定義が存在するとき，
$g[l] \Longleftrightarrow f$

(12) ラベル $g[l]$ がいかなる部分式にも現れないとき，ラベル $g[l]$ 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．

(13) ラベル $g[l]$ の部分式の定義が存在しないとき，$g[l] =$ 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．

上記の公理系において，各 x,y,z は，2.節の条件を満たす任意の部分回路にマッチする．また $\Longleftrightarrow$ は左辺から右辺へ及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．なお，消去と生成は互いに逆操作の関係にあることに注意されたい．

公理(2)は，1入力のOR,ANDゲートを取り除くものである．公理(3)は，2入力ゲートの2つの入力が同一である場合にやはりそのゲートを取り除くものである．このような同じ入力が3以上になる場合はどうであろうか．例えば $\vee(x,x,x,x)$ は公理(5)を適用して，$\vee(x,\vee(x,x),x)$ に変形できる．(公理(5)の両辺において，右の ')' が足りないようにみえるが，これでよい.) さらに公理(3)で $\vee(x,x,x)$ と変形でき，もう一度公理(5)と(3)を適用すると，$\vee(x,x)$ となり，さらに公理(3)で x まで簡単化できる．公理(7)において，$\wedge(x,\mathbf{1}) \Longleftrightarrow x$ の対遇は，$\vee(x,\mathbf{0}) \Longleftrightarrow x$ である．しかし，後者は公理(3)，公理(4)，公理(7)，公理(10)を適用して導くことができるので必要ない．公理(9)は，いわゆるドモルガンの法則である．ここでも，その対遇 $\langle\wedge(x,y)\rangle \Longleftrightarrow \vee(\langle x\rangle,\langle y\rangle)$ は次のような操作から導かれる．$\vee(\langle x\rangle,\langle y\rangle)$ から公理(8)によって $\langle\langle\vee(\langle x\rangle,\langle y\rangle)\rangle\rangle$ が導かれ，さらに公理(9)によって $\langle\vee(\langle\langle x\rangle\rangle,\langle\langle y\rangle\rangle)\rangle$ が導かれ，さらに公理(8)によって最終的に $\langle\wedge(x,y)\rangle$ が導かれる．公理(11)は，いわゆる代入とその逆の操作を表現している．ある部分式にラベル $g[l]$ が現れているとき，その $g[l]$ を $g[l]$ の定義によって置き換えること，またその逆として，ある部分式中の式(文字列)σ に対して，$g[l] = \sigma$ という定義が存在する場合は σ を $g[l]$ と置き換えることができる．$g[l] = \sigma$ の定義が存在しない場合，公理(13)の生成によってそれを作り出してから公理(11)を適用する点に注意されたい．

これら基本操作規則の組合せにより，例えば次のような操作規則を導くことができる．

公理(5)の一般化：

$\vee(T,\vee(S_1,S_2,\cdots,S_m) \Longleftrightarrow \vee(T,S_1,S_2,\cdots S_m$

左辺に公理(5)を適用すると，

$\vee(T,\vee(S_1,\vee(S_2,S_3),\cdots,S_m)$

さらに公理(5)を繰り返し適用していくと，

$\vee(T,\vee(S_1,\vee(\vee(\cdots\vee(S_2,S_3),S_4),\cdots),S_{m-1}),S_m))$

さらに公理(5)により，

$\vee(T,S_1,\vee(\vee(\cdots\vee(S_2,S_3),S_4),\cdots),S_{m-1}),S_m)$

公理(6)によって，

$\vee(S_1,\vee(\vee(\cdots\vee(S_2,S_3),S_4),\cdots),S_{m-1}),S_m),T$

再び公理(5)により，

$\vee(S_1,\vee(\vee(\cdots\vee(S_2,S_3),S_4),\cdots),S_{m-1}),S_m,T$

さらに公理(5)を繰り返し適用していくと，

$\vee(S_1,\vee(S_2,S_3),S_4,\cdots,S_m,T$

公理(5)によって，

$\vee(S_1,S_2,\cdots,S_m,T$

最後に公理(6)によって，右辺となる．

公理(9)の一般化：

$\langle\vee(S_1,S_2\cdots,S_m)\rangle \Longleftrightarrow \wedge(\langle S_1\rangle,\langle S_2\rangle,\cdots,\langle S_m\rangle)$

左辺に公理(5)の一般化を適用して，

$\langle\vee(S_1,\vee(S_2\cdots,S_m))\rangle$

公理(9)より，

$\wedge(\langle S_1\rangle,\langle\vee(S_2,\cdots,S_m)\rangle)$

$\langle\vee(S_2,\cdots,S_m)\rangle$ の部分に，さらに公理(5)の一般化と公理(9)を繰り返し適用して，右辺となる．公理(9)の対遇の一般化

$\langle\wedge(S_1,S_2\cdots,S_m)\rangle \Longleftrightarrow \vee(\langle S_1\rangle,\langle S_2\rangle,\cdots,\langle S_m\rangle)$

も，公理から容易に導くことができる．

公理(10)の一般化：

$\wedge(T,\vee(S_1,S_2\cdots,S_m)) \Longleftrightarrow$
$\vee(\wedge(T,S_1),\wedge(T,S_2),\cdots,\wedge(T,S_m))$

は，次のようにして導かれる．左辺に公理(5)の一般化を適用して，

$\wedge(T,\vee(S_1,\vee(S_2,\cdots,S_m))$

公理(10)により，

$\vee(\wedge(T,S_1),\wedge(T,\vee(S_2,\cdots,S_m)))$

さらに公理(10)を繰り返し適用して，

$\vee(\wedge(T,S_1),\vee(\wedge(T,S_2),\vee(\wedge,\cdots,\vee(\wedge(T,S_m)))\cdots)$

再び，公理(5)の一般化を繰り返し適用して，右辺となる．

次に基本操作に要する時間についての定理を示す．

[定理 1]　1回の基本操作は，多項式時間で実行できる．

(証明)　例えば，回路 f_1 に操作規則 $\wedge(x,\vee(y,z)) \Longrightarrow \vee(\wedge(x,y),\wedge(x,z))$ を適用する場合に，考慮する点は，部分列 $g = \wedge(x,\vee(y,z))$ としての全ての可能性，g の中で x,y,z としての全ての可能性，x,y,z のそれぞれが式かどうかの判定の3点であるが，それぞれが多項式時間で実行できることは容易に判ろう．よってこの操作は多項式時間で実行できる．公理(1)～(10)に関しては全く同様である．公理(11)も問題ない．公理(13)については，生成できる部分式の長さに制限を設けている点に注目されたい．　□

4. 基本操作規則集合の完全性

3.節で与えられた公理系によって定義される基本操作規則集合によって，任意の等価な回路 f_1 と f_2 の間で変換が可能であることを示す．ここで変換が可能であるとは，f_1 から f_2 へ変換する基本操作の列が存在することを意味する．(そのような列は一般に一意ではない．例えば，短い列をどのようにして得るかは難しい問題である．) 基本方針は3.節の基本操作規則集合によって論理式を「標準形」に変換できることを示すことである．標準形は特定の論理関数に対しては唯一つしか存在しないように決める．回路 f_1 と f_2 が等価であるとする．このとき共に等しい標準形 p へ f_1 からは基本変換列 σ_1 で，f_2 からは基本変換列 σ_2 で変換できるとする(定理2)．ここで，3.節の規則が両方向になっていることに注意されたい．つまり σ_2 を逆方向に適用することにより p から f_2 へ変換できる．よって f_1 から f_2 への変換が可能であることが示せる．

4.1 標準形

標準形は特定の論理関数に対しては唯一つしか存在しないようにするため，以下のように定義する．

n 変数の回路の標準形は $\vee(p_1,p_2,\cdots,p_j,p_{j+1},\cdots,p_m)$ (特別の場合として $\mathbf{0}$ を含む) の形である．ここで，各 p_j は $\wedge(X_1,X_2,\cdots,X_n)$ の形をしており，X_i は変数 x_i またはその否定 $\langle x_i\rangle$ である．任意の $j \geq 1$ に対し，p_j と p_{j+1} の間には以下の関係がなければならない．$N(p_i)$ を $X_1,\cdots,X_n$ の各 X_i に対しそれが x_i なら1，$\langle x_i\rangle$ なら0に置き換えて得られる (n ケタの前の方に0が存在してもよい) 二進数とする．このとき $N(p_j) < N(p_{j+1})$ である．

4.2 主定理

以下に，本稿の主定理を示す．

[定理 2]　f_1 が回路ならば f_1 を標準形に変形する基本操作の列が存在する．

(証明)　回路 f_1 を標準形に変換するアルゴリズムが存在する(4.3節)．このアルゴリズムは，回路 f_1 を文字列 f として入力すると，標準形を生成して終了する．ここで適用する操作は，基本操作規則またはその組合せにより導かれる操作である．　□

4.3 標準形への変形

任意の回路を標準形に変形する方法を以下に示す．

Step1：公理(11)を用いて代入の操作を行ない，回路を本状にする．ラベル $g[0]$ の部分式に，もし $g[a_1]$ があればラベル $g[a_1]$ に定義されている部分式と置き換える．置き換えた部分式に $g[a_2]$ があれば同様の操作を行なう．これを繰り返して，$g[l]$ がなくなれば，公理(12)の操作を行ない Step2 へ．

Step2：$\langle x_i\rangle$ 以外の否定 $\langle\ \rangle$ を取り除く．

(2-1)　公理(9)の一般化

$$\langle\vee(S_1,\cdots,S_m)\rangle \Longrightarrow \wedge(\langle S_1\rangle,\cdots,\langle S_m\rangle)$$
$$\langle\wedge(S_1,\cdots,S_m)\rangle \Longrightarrow \vee(\langle S_1\rangle,\cdots,\langle S_m\rangle)$$

を適用できる部分列があれば，すべてに繰り返し適用する．できなくなれば(2-2)へ．

(2-2)　$\langle\langle S\rangle\rangle \Longrightarrow S$

を適用できる部分列があれば，すべてに繰り返し適用する．できなくなれば，(2-3)へ．

(2-3)　$\langle\mathbf{0}\rangle \Longrightarrow \mathbf{1}$　　$\langle\mathbf{1}\rangle \Longrightarrow \mathbf{0}$

を適用できる部分列があれば，すべてに繰り返し適用する．できなくなれば，Step3へ．

Step3：ANDからAND，ORからORへの接続をなくす．

$$\vee(S_1) \Longrightarrow S_1 \qquad \wedge(S_1) \Longrightarrow S_1$$
$$\vee(Y_1,\cdots Y_l,\vee(S_1,\cdots S_m),Z_1,\cdots,Z_n) \Longrightarrow \vee(Y_1,\cdots Y_l,S_1,\cdots S_m,Z_1,\cdots,Z_n)$$
$$\wedge(Y_1,\cdots Y_l,\wedge(S_1,\cdots S_m),Z_1,\cdots,Z_n) \Longrightarrow \wedge(Y_1,\cdots Y_l,S_1,\cdots S_m,Z_1,\cdots,Z_n)$$

(ただし，$Y_1,\cdots,Y_l,S_1,\cdots,S_m,Z_1,\cdots,Z_n$ は部分式)

を適用できる部分列があれば，すべてに繰り返し適用する．できなくなれば，Step4へ．(第3の操作は，次のように導かれる．左辺に公理(6)を繰り返し適用し，

$$\vee(Y_1,\vee(S_1,\cdots S_m),Y_2,\cdots Y_l,Z_1,\cdots,Z_n)$$

公理(5)の一般化により，

$$\vee(Y_1,S_1,\cdots,S_m,Y_2,\cdots,Y_m,Z_1,\cdots,Z_n)$$

公理(6)により，右辺となる．第4の操作も同様に導かれる．)

Step4：回路をOR-ANDの2段にする．

$$\wedge(Y_1,\cdots Y_l,\vee(S_1,\cdots S_m),Z_1,\cdots,Z_n) \Longrightarrow \vee(\wedge(Y_1,\cdots,Y_l,S_1,Z_1,\cdots,Z_n),\wedge(Y_1,\cdots,Y_l,S_2,Z_1,\cdots,Z_n),\cdots\wedge(Y_1,\cdots,Y_l,S_m,Z_1,\cdots,Z_n))$$

適用できる部分列すべてに繰り返し適用する．できなくなれば Step5 へ．(この操作は，次のように導かれる．左辺に公理(6)を適用して，

$$\wedge(\vee(S_1,\cdots S_m),Y_1,\cdots Y_l,Z_1,\cdots,Z_n,)$$

公理(5)の一般化と公理(6)により，

$$\wedge(\wedge(Y_1,\cdots Y_l,Z_1,\cdots,Z_n),\vee(S_1,\cdots S_m))$$

公理(10)の一般化により，

$\vee(\wedge(\wedge(Y_1,\cdots Y_l,Z_1,\cdots,Z_n),S_1),$
$\cdots,\wedge(\wedge(Y_1,\cdots Y_l,Z_1,\cdots,Z_n),S_m))$
公理(6)と公理(5)の一般化により，
$\vee(\wedge(Y_1,\cdots Y_l,Z_1,\cdots,Z_n,S_1),$
$\cdots,\wedge(Y_1,\cdots Y_l,Z_1,\cdots,Z_n,S_m))$
公理(6)を適用して右辺となる．

Step5：**0**と**1**を取り除く．

(5-1)　ラベル$g[0]$の部分式が**1**ならば，部分式を$\vee(x_1,\langle x_1\rangle)$と置き換えてStep6へ．その他は(5-2)へ．

(5-2)　ラベル$g[0]$の部分式は$\vee(q_1,\cdots,q_m)$の形である．(q_jはANDゲートである)，q_jが**1**ならば$\vee(q_1,\cdots,q_m)$を$\vee(x_1,\langle x_1\rangle)$と置き換えてStep6へ．その他は(5-3)へ．(この操作は公理の(4),(6),(7)から導かれる．)

(5-3)　q_jがX_i（X_iはリテラル）ならば，$\wedge(X_i)$と置き換える．すべてのq_jについて行なったら(5-4)へ．

(5-4)　q_jに**0**またはx_iと$\langle x_i\rangle$があればq_jを**0**と置き換える．すべてのq_jについて行なったら(5-5)へ．

(5-5)　q_jに**1**があれば，

$$\mathbf{1},S \Longrightarrow S \qquad S,\mathbf{1} \Longrightarrow S$$

をq_jの中の適用できる部分列に適用する．すべてのq_jについて行なったらStep6へ．(この操作は，1つのANDゲートのみに適用するので，上記のような記述でよい．より厳密には，公理の(3)と(6)および公理(5)の一般化で導かれる．)

Step6：標準形にするための最終的な操作を行なう．（ラベル$g[0]$の）部分式は$\vee(q_1,\cdots,q_m)$と表せる．さらにq_jは$\wedge(p_j)$と表せる．

(6-1)　文字列(p_j)にリテラル$X_1,\cdots,X_n$のうち含まれないリテラルX_iがあれば，$\wedge(p_j)$を$\wedge(p_j,x_i),\wedge(p_j,\langle x_i\rangle)$と置き換えて(6-2)へ．(この操作は，公理の(4),(7),(10)から導かれる．)

(6-2)　部分式を新たに$\vee(q_1,\cdots,q_m)$とおいて，すべてのq_jにリテラル$X_1,\cdots,X_n$がすべて含まれていれば，(6-3)へ．その他は(6-1)へ．

(6-3)　q_j内のリテラルを添字iの順に並べ換える．すべてのq_jについて行なったら，

$$S,S \Longrightarrow S$$

を適用できる部分列に適用する．(この操作については，(5-5)と同様である．)できなくなれば(6-4)へ．

(6-4)　q_jを$N(p_j)$（ただし，$N(p_j)$は$X_1,\cdots,X_n$の各X_iに対しそれがx_iなら1，$\langle x_i\rangle$なら0に置き換えて得られる二進数）の順に並べ換えて，$S,S \Longrightarrow S$を適用できる部分列に適用する．できなくなれば，終了．

上記のアルゴリズムで用いる操作の中には基本操作規則にはないものも含んでいる．これらは，アルゴリズムの記述を容易にするためのものであって，基本操作規則から必ず導くことができるのは各所で述べたとおりである．

5.　おわりに

本稿では，"回路"を部分式の集合として定義し，1回の操作が多項式時間で実行できる基本操作規則の集合を提案し，その基本操作規則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．今後は，制限つきの回路（例えば，NAND素子のみ，NOR素子のみ，ファンイン制限 等）に対する定義や操作規則を導き，本研究の目的の一つであるアルゴリズム評価のためのテスト例題生成への利用について考えていく．

参考文献

[1] 室賀，笹尾，"論理設計とスイッチング理論"，共立出版，(1979).

[2] Zvi Kohavi, "Switching and finite automata theory", McGraw-Hill, Inc. (1978).

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

[4] 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.

[5] 宮崎，岩間，"CNF論理式の充足解に対する改良されたランダム探索法"，平成4年度九州支部連合大会（平成4年10月）．

[6] E.Koutsoupias and C.Papadimitrious, "On the greedy algorithm for satisfiability", Inform. Process. Lett. 43, pp.53-55(1992).

[7] B.Selman, H.Levesque and D.Mitchell, "A new method for solving hard satisfiability problems", Proc. Tenth. National Conf. on AI, pp.440-446(1992).

[8] K.Iwama, H.Abeta and E.Miyano, "Random Generations of Satisfiable and Unsatisfiable CNF Predicates", Proc. IFIP 12th World Computer Congress, pp.322-328(1992)

[9] E.Lawler, et al. "The traveling salesman problem", John Wileq and Sona, (1985).

盛光印刷所