数理解析研究所講究録
第1843巻 2013年 66-79

Sign Definite Condition 専用 Quantifier Elimination における論理式の簡単化

岩根秀直
(株) 富士通研究所*
HIDENAO IWANE
FUJITSU LABORATORIES LTD

樋口博之
(株) 富士通研究所†
HROYUKI HIGUCHI
FUJITSU LABORATORIES LTD

穴井宏和
(株) 富士通研究所/九州大学‡
HIROKAZU ANAI
FUJITSU LABORATORIES LTD/KYUSHU UNIVERSITY

1 はじめに

限量記号消去アルゴリズム (quantifier elimination (QE) algorithm) とは，与えられた形式的理論 (formal theory) について「限量記号付きの式 (一階述語論理式)」を入力とし「等価で限量記号無しの式」を出力するアルゴリズムのことである．QE は工学や産業上の問題などの多くの応用があり重要なアルゴリズムである．しかし，QE の計算量の下限が限量記号がついた変数の数に対して二重指数であることが示されており，本質的に規模の大きな問題を解くことができない．そのため，限量記号がついた変数が線形の場合 [5] など制限された入力の一階述語論理式に対する専用アルゴリズムの研究がすすめられている．

多項式 $f(x) \in \mathbb{Q}[x]$ に対して，$\forall x\ (x \geq 0 \to f(x) > 0)$ を sign definite condition (SDC) と呼ぶ．制御系設計の様々な条件が SDC で記述することができる [2] ため，SDC に対する QE は実用上重要であり，SDC に対して高速に計算可能な手法が提案されている [6]．提案された手法は，SDC が「主係数が正の多項式が $x \geq 0$ において実根を持たないこと」と等価なことを利用して，Sturm-Habicht 列 [4] を用いた実根の数え上げにより QE を実現している．具体的には，あらかじめオフラインで，次数毎に多項式の実根の数が 0 となる論理式を Sturm-Habicht 列の符号の組み合わせから構築しておく．これにより，入力が与えられたあとのオンライン計算では Sturm-Habicht 列の計算と代入計算のみを行うことになり，高速化が実現される．

本稿では，SDC の出力となる論理式の簡単化の手法について述べる．SDC 専用 QE においてオフラインで構築する論理式の簡単化は，オンラインでの代入計算や，後処理での実行可能領域の描画などの高速化につながる．論理式の簡単化のために，最初に，SDC を満たす多項式の Sturm-Habicht 列が満たす条件を示し，Sturm-Habicht 列による実根の数え上げによって得られる符号列を削減することで，論理式を簡

*iwane@jp.fujitsu.com
†h-higuchi@jp.fujitsu.com
‡anai@jp.fujitsu.com

単化する．次に，既存の論理関数処理による方法を適用することで $g>0 \wedge g=0 \leftrightarrow g \geq 0$ による論理式の簡単化を行う．計算機実験結果により，本稿で提案する手法の効果を示す．

本稿の構成は以下の通りである．まず，2 章において Sturm-Habicht 列と SDC 専用 QE アルゴリズムについて紹介する．3 章では，SDC を満たす多項式の Sturm-Habicht 列においてあらわれない符号列を示し，4 章では，論理関数処理の紹介とそれを用いた論理式の簡単化手法を述べる．5 章で，実験結果により本稿で述べる手法による論理式簡単化の効果を示す．最後に 6 章で本稿のまとめと今後の課題を述べる．

2 Sign Definite Condition 専用 Quantifier Elimination

本章では，sign definite condition の定義と，[6] で提案されている専用の QE アルゴリズムについて述べる．

2.1 Sign Definite Condition と Sturm-Habicht 列による実根の数え上げ

最初に sign definite condition (SDC) を定義する．

定義 1
多項式 $f(x) \in \mathbb{Q}[x]$ に対する以下の条件を **sign definite condition** *(SDC)* という．

$$\forall x \ (x \geq 0 \rightarrow f(x) > 0)$$

制御系設計の様々な条件が SDC で記述できるため，SDC は重要な問題のクラスであり，専用の QE アルゴリズムが提案されている [6, pp. 208–211]．提案された専用アルゴリズムは SDC が「主係数が正の多項式 $f(x)$ が $x \geq 0$ において実根を持たないこと」と等価なことを利用し，Sturm-Habicht 列 [4] を用いた実根の数え上げにより QE を実現している．

次に専用の QE アルゴリズムで利用される Sturm-Habicht 列について紹介する．

定義 2
n 次の多項式 $f(x) \in \mathbb{Q}[x]$ に対して，以下の多項式列 $\mathsf{SH}(f) := \{\mathsf{SH}_n(f), \ldots, \mathsf{SH}_0(f)\}$ を **Sturm-Habicht** 列とよぶ．

$$\begin{aligned} \mathsf{SH}_n(f) &= f, \\ \mathsf{SH}_{n-1}(f) &= \frac{df}{dx}, \\ \mathsf{SH}_j(f) &= \delta_{n-j} \ \mathrm{Sres}_j(f, \frac{df}{dx}) \qquad (j = 0, \ldots, n-2). \end{aligned}$$

ここで，$\mathrm{Sres}_j(f,g)$ は f と g の j 次部分終結式 *[6, p. 129]* で，$\delta_j = (-1)^{j(j+1)/2}$ である．

定義 3
実数の有限列 $A = \{a_m, \ldots, a_0\}$ における符号変化の数 $V(A)$ は，以下の規則に従い数える．

- 次の符号列を *1* と数える： $\{-,+\}, \{+,-\}\{-,0,+\}, \{+,0,-\}, \{-,0,0,+\}, \{+,0,0,-\}$
- 次の符号列を *2* と数える： $\{+,0,0,+\}, \{-,0,0,-\}$

さらに，実数係数の有限個の多項式列 $S(x) = \{S_n(x), S_{n-1}(x), \ldots, S_0(x)\}$ と実数 α に対して，$V_\alpha(S)$ で $\{S_n(\alpha), S_{n-1}(\alpha), \ldots, S_0(\alpha)\}$ の符号変化の数 $V(\{S_n(\alpha), S_{n-1}(\alpha), \ldots, S_0(\alpha)\})$ を表す．

次の定理により Sturm-Habicht 列を用いて与えられた区間における多項式の実根の数を求めることができる.

定理 4
$f \in \mathbb{Q}[x]$, $a, b \in \mathbb{R}$ $(a < b)$ に対し $f(a)f(b) \neq 0$ とし, $H(f)$ を $\mathsf{SH}(f)$ から恒等的に 0 になる多項式を取り除いたものとする.
このとき, $V_a(H(f)) - V_b(H(f))$ は区間 $[a, b]$ における $f(x)$ の実根の数に一致する.

定理 4 を用いれば, 区間 $[0, \infty)$ における実根の数を求められる. $x = 0$ と $x = \infty$ における $\mathsf{SH}_k(f)$ の符号を表すため, 以下では次の記法を用いる.

記法 1
$\mathsf{SH}_k(f)$ の $x = \infty$ における符号を s_k, $\mathsf{SH}_k(f)$ の $x = 0$ における符号を c_k と表記する.
符号とは, 正, 負または 0 のことで, それぞれ, $+, -, 0$ で表す. 本稿では, $+$ を 1, $-$ を -1, 0 を 0 と扱い, 同一視する. 例えば, t が 0 または $+$ の場合には $t \geq 0$ と表記する.

注意 1
$\mathsf{SH}_k(f) = a_{k,k}x^k + a_{k,k-1}x^{k-1} + \cdots + a_{k,0}$ とするとき, 以下が成立する.

$$
\begin{aligned}
s_k = 0 \quad &\leftrightarrow \quad a_{k,i} = a_{k,k-1} = \cdots = a_{k,0} = 0 \\
s_k > 0 \quad &\leftrightarrow \quad (a_{k,k} > 0) \vee (a_{k,k} = 0 \wedge a_{k,k-1} > 0) \vee \cdots \vee (a_{k,k} = a_{k,k-1} = \cdots = a_{k,1} = 0 \wedge a_{k,0} > 0)
\end{aligned}
$$

したがって, $x = \infty$ で $\mathsf{SH}_k(f)$ が 0 になるとき, $\mathsf{SH}_k(f)$ は恒等的に 0 である. また, c_k は $a_{k,0}$ の符号に一致する.

例 1
$f(x) = 25x^5 + 25x^4 + 10x^3 + 2x^2 + 25x + 1$ とする.

$$
\begin{array}{lllll}
\mathsf{SH}_5(f) & = & f(x) & = & 25x^5 + 25x^4 + 10x^3 + 2x^2 + 25x + 1, \\
\mathsf{SH}_4(f) & = & \frac{df}{dx}(x) & = & 125x^4 + 100x^3 + 30x^2 + 4x + 25, \\
\mathsf{SH}_3(f) & = & -\mathrm{Sres}_3(f, \frac{df}{dx}) & = & -310000x, \\
\mathsf{SH}_2(f) & = & -\mathrm{Sres}_2(f, \frac{df}{dx}) & = & 0, \\
\mathsf{SH}_1(f) & = & \mathrm{Sres}_1(f, \frac{df}{dx}) & = & 1906624000000x, \\
\mathsf{SH}_0(f) & = & \mathrm{Sres}_0(f, \frac{df}{dx}) & = & 945685504000000
\end{array}
$$

であるので,

$$
\begin{aligned}
H(f) &= \{\mathsf{SH}_5(f), \mathsf{SH}_4(f), \mathsf{SH}_3(f), \mathsf{SH}_1(f), \mathsf{SH}_0(f)\} \\
\{s_5, s_4, s_3, s_1, s_0\} &= \{+, +, -, +, +\} \\
\{c_5, c_4, c_3, c_1, c_0\} &= \{+, +, 0, 0, +\}
\end{aligned}
$$

となり, $V_\infty(H(f)) = V_0(H(f)) = 2$ なので, $f(x)$ の $x \geq 0$ における実根の数は 0 である. $f(x)$ の係数の符号変化の数は 0 なので, デカルトの符号律からも $f(x)$ が $x \geq 0$ で実根を持たないことが確認できる.

注意 2
n 次の多項式に対して, $s_n = s_{n-1}$, $s_0 = c_0$ となることに注意する.

定義 5
$\deg(\mathsf{SH}_k(f)) = k$ のとき, $\mathsf{SH}_k(f)$ は正則であるという.

Sturm-Habicht 列に対して以下の Sturm-Habicht Structure Theorem [4] が成立する.

定理 6

f を次数 n (≥ 2) の多項式とする．$\mathsf{SH}_{k+1}(f)$ が正則なすべての k に対して，$\deg(\mathsf{SH}_k(f)) = r \leq k$ とするとき，以下が成立する．

(A) $r < k-1$ のとき，$\mathsf{SH}_{k-1}(f) = \cdots = \mathsf{SH}_{r+1}(f) = 0$,

(B) $r < k$ のとき，$\mathrm{lc}(\mathsf{SH}_{k+1}(f))^{k-r}\mathsf{SH}_r(f) = \delta_{k-r}\mathrm{lc}(\mathsf{SH}_k(f))^{k-r}\mathsf{SH}_k$,

(C) $r < k$ のとき，$\mathrm{lc}(\mathsf{SH}_{k+1}(f))^{k-r+2}\mathsf{SH}_{r-1}(f) = \delta_{k-r+2}\mathrm{Prem}(\mathsf{SH}_{k+1}(f), \mathsf{SH}_k(f))$.

ここで，$\mathrm{lc}(g)$ は g の主係数で，$\mathrm{Prem}(g,h)$ は，以下で定義される擬剰余を表す．

$$\mathrm{Prem}(g,h) = \mathrm{remainder}(\mathrm{lc}(h)^{\deg(g)-\deg(h)+1}g, h)$$

index	$k+1$	k	$k-1$	$\cdots$	$r+1$	r	$r-1$
SH_k	SH_{k+1}	SH_k	0	$\cdots$	0	SH_r	SH_{r-1}
deg	$k+1$	r	$-\infty$	$\cdots$	$-\infty$	r	

2.2 SDC 専用 QE の実装

本節では，穴井らにより提案された SDC 専用の QE アルゴリズムの実装方法 [6] について述べる．

Algorithm 1 は，SDC 専用の QE アルゴリズムの実装方法の概要である．最初に，オフラインで次数毎に係数をパラメタとする多項式に対して，$x \geq 0$ で実根を持たない符号条件をあらかじめ求めておく．f が入力された後（オンライン）の計算は Sturm-Habicht 列を求めて代入する部分だけとなるので，高速な QE 計算が実現される．実際，5 次の問題は汎用の QE アルゴリズムである Cylindrical Algebraic Decomposition では 1 時間たっても計算が停止しないが，提案手法では 1 秒に満たない時間で計算できる．

オンラインの計算を高速化するためには，オフラインでの φ_n の表現を簡単化することが必要である．また，φ_n の表現の簡単化は出力される論理式の簡単化につながり，実行可能領域の描画や真偽値を判定する場合など後処理の高速化にもつながる．次章以降では，オフラインで計算する φ_n の簡単化について考える．

Algorithm 1 qe_{sdc}

Input: 次数 n の多項式 $f(x) \in \mathbb{Q}[x]$

Output: $f(x)$ が SDC を満たす条件

$\varphi_n \leftarrow V_0(H(f)) - V_\infty(H(f)) = 0$ となる SH の符号条件 (オフラインで計算，定理 4)

$\mathsf{SH}(f) \leftarrow f$ の Sturm-Habicht 列

φ_n に $\mathsf{SH}(f)$ を代入する．

例 2

2 次の多項式 $f(x) = x^2 + bx + c$ の場合を考える．$V_0(H(f)) - V_\infty(H(f)) = 0$ となる符号条件は表 1 のようになる．各行が $f(x)$ が $x \geq 0$ で実根を持たないときの s_k, c_k の符号を表している．ここで，注意 2 より，$s_2 = s_1 = +$ であり，$s_0 = c_0$ であり，$f(0) > 0$ より，$c > 0$ なので $c_2 = +$ となることに注意する．

表 1: φ_2

s_2	s_1	s_0	c_2	c_1	c_0
+	+	-	+	-	-
+	+	-	+	0	-
+	+	-	+	+	-
+	+	0	+	0	0
+	+	0	+	+	0
+	+	+	+	+	+

$f(x)$ の *Sturm-Habicht* 列は $\mathsf{SH}(f) = \{x^2 + bx + c, 2x + b, b^2 - 4c\}$ なので，以下が φ_2 の表現となる．

$$\begin{array}{ll} (b^2 - 4c < 0 \wedge c > 0 \wedge b < 0) & \vee \\ (b^2 - 4c < 0 \wedge c > 0 \wedge b = 0) & \vee \\ (b^2 - 4c < 0 \wedge c > 0 \wedge b > 0) & \vee \\ (b^2 - 4c = 0 \wedge c > 0 \wedge b = 0) & \vee \\ (b^2 - 4c = 0 \wedge c > 0 \wedge b > 0) & \vee \\ (b^2 - 4c > 0 \wedge c > 0 \wedge b > 0) & \end{array} \tag{1}$$

式 (1) は $g > 0 \wedge g = 0 \leftrightarrow g \geq 0$ を利用することで，以下のように簡単化することができる．

$$\begin{array}{ll} (b^2 - 4c < 0 \wedge c > 0) & \vee \\ (b^2 - 4c = 0 \wedge c > 0 \wedge b \geq 0) & \vee \\ (b^2 - 4c > 0 \wedge c > 0 \wedge b > 0) & \end{array} \tag{2}$$

上記の条件をオフラインで構築しておき，具体的な問題に対して b, c に値を代入するだけで *QE* が実現される．このように φ_2 をあらかじめ簡単化しておくことでオンラインでの代入回数を削減できる．

3 SDC の必要条件

例 2 で得られた式 (2) について考えると，明らかに $b^2 - 4c = 0 \wedge c > 0 \wedge b = 0$ を満たす実数 b, c は存在しない．この結果を利用すると，式 (2) は以下のようにさらに簡単化することができる．

$$\begin{array}{ll} (b^2 - 4c < 0 \wedge c > 0) & \vee \\ (b^2 - 4c \geq 0 \wedge c > 0 \wedge b > 0) & \end{array}$$

このように，Sturm-Habicht 列にはあらわれない符号列が存在し，それらを削減することで論理式の簡単化が実現できる．本章では SDC を満たす多項式 の Sturm-Habicht 列が満たす条件を示す．

次の定理は，多項式が SDC を満たすときの必要条件を与える．定理 4 および記法 1 の記法を使用している．

定理 7

f を n 次の多項式，u を $s_k \neq 0$ となる最小の非負整数 k とする．f が *SDC* を満足するとき，以下に示す条件を満たす．

$V_0(H(f)) - V_\infty(H(f)) = 0,$

$s_n > 0, s_{n-1} > 0, c_n > 0, s_0 = c_0,$

$s_k = 0 \to c_k = 0, \ (\forall k \in \{0, \ldots, n-2\}),$

$c_u \neq 0,$

$c_{n-1} = 0 \to c_{n-2} < 0,$

$\{c_k, c_{k+1}, c_{k+2}\} \neq \{+, 0, +\}, \{-, 0, -\}, \ (\forall k \in \mathcal{N} = \{u, \ldots, n-2\}),$

$c_{k-1} \neq 0, c_k = c_{k+1} = 0, c_{k+2} \neq 0, s_k \neq 0, s_{k+1} \neq 0 \to s_{k+1}s_{k-1} < 0, \ (\forall k \in \mathcal{N}),$

$c_k = \cdots = c_{k+m} = 0 \to s_{k+1} = \cdots = s_{k+m-1} = 0 \ (\forall k \in \mathcal{N}, m > 1),$

$s_{k+2} = 0, s_{k+1} \neq 0 \to s_k \neq 0, \ (\forall k \in \mathcal{N}),$

$s_{k-1} \neq 0, s_k = \cdots = s_{k+m} = 0, s_{k+m+1} \neq 0 \to$

$(s^m_{k+m+2}s_{k-1} = \delta_{m+2}s^{m+1}_{k+m+1}, c^m_{k+m+2}c_{k-1} = \delta_{m+2}s^m_{k+m+1}c_{k+m+1}), \ (\forall k \in \mathcal{N}, m \geq 0).$

定理の証明は以下の補題で与える.

補題 8

次数 n の多項式 $f(x) \in \mathbb{Q}[x]$ に対して，$s_n = s_{n-1} = c_n = +$ は SDC の必要条件である.

証明 $f = p_n x^n + p_{n-1}x^{n-1} + \cdots + p_0$ とする．SDC を満たすとき，$f(0) = p_0 > 0$ なので，$c_n = +$. また，十分大きな x に対して $f(x) > 0$ を満たすには，$p_n > 0$ が必要なので，$s_n = +$ となる．注意 2 より $s_{n-1} = s_n = +$ が成立する.

∎

以下では，補題 8 より，主係数および定数項は正と仮定する．したがって，$s_n = c_n = +$ とする．このとき，f はモニックと仮定しても一般性を失わないので，$f = x^n + p_{n-1}x^{n-1} + \cdots + p_0 \ (p_0 > 0)$ とする．簡単のため，$\mathsf{SH}_k(f)$ を SH_k と表記する.

補題 9

任意の $k = 1, \ldots, n$ に対して，以下が成立する.

$$s_k = 0 \to c_k = 0.$$

証明 注意 1 より，$s_k = 0$ のとき， $\mathsf{SH}_k = 0$ なので，$c_k = 0$. ∎

補題 10

$c_u \neq 0.$

証明 u の定義から SH_u は $\gcd(f, f')$ の定数倍になる．$p_0 > 0$ から，$f(0) \neq 0$ なので，$\mathsf{SH}_u(0) \neq 0$. したがって，$c_u \neq 0$. ∎

補題 11

$c_{n-1} = 0$ ならば $c_{n-2} = -$.

証明 $\mathsf{SH}_{n-1}(0) = p_1, \mathsf{SH}_{n-2}(0) = p_1p_{n-1} - n^2p_0$ で，今 $p_0 > 0$ なので，

$$c_{n-1} = 0 \leftrightarrow p_1 = 0 \to \mathsf{SH}_{n-2}(0) = -n^2p_0 < 0$$

定理 4 で定義される $H_k(f)$ に対して，$\{+,0,+\},\{-,0,-\}$ の符号列があらわれないことが [4] で示されているが，c_k に対してはさらに強い制限がある．

補題 12

$k=1,\ldots,n-1$ に対して $\{c_{k+1},c_k,c_{k-1}\}\neq\{+,0,+\},\{-,0,-\}$.

証明 $c_k=0$ となるのは，$s_k=0$ の場合と $s_k\neq 0$ の場合がある．

(1) $\deg(\mathsf{SH}_k)=r,\deg(\mathsf{SH}_{k+1})=k+1,\mathsf{SH}_k(0)=c_k=0$ のとき，

(a) $r=k$ の場合，定理 6 (C) から，

$$\mathrm{lc}(\mathsf{SH}_{k+1})^2\mathsf{SH}_{k-1}=-\mathrm{Prem}(\mathsf{SH}_{k+1},\mathsf{SH}_k).$$

よって，

$$c_{k-1}=-\mathrm{lc}(\mathsf{SH}_{k+1})^{-2}c_{k+1}.$$

$f(0)\neq 0$ なので，SH_k と SH_{k+1} が同時に x を共通因子に持つことはないので，$c_{k+1}\neq 0$ となる．したがって，$\{c_{k+1},c_k,c_{k-1}\}\neq\{+,0,+\},\{-,0,-\}$.

index	$k+1$	k	$k-1$
SH_i	SH_{k+1}	SH_k	SH_{k-1}
deg	$k+1$	k	
c_i	$\neq 0$	0	

(b) $r<k$ の場合，定理 6 (A)，(C) より，2 回以上 0 が連続してあらわれる．

index	$k+1$	k	$k-1$	$\cdots$	$r+1$	r
SH_i	SH_{k+1}	SH_k	0	$\cdots$	0	SH_r
deg	$k+1$	r	$-\infty$	$\cdots$	$-\infty$	r
c_i	$\neq 0$	0	0	$\cdots$	0	0

(2) $\deg(\mathsf{SH}_{k+1})=r<k+1,\deg(\mathsf{SH}_{k+2})=k+2,\mathsf{SH}_k(0)=c_k=0$ のとき，

(a) $r<k-1$ 場合，定理 6 (A) より 0 が 2 回以上連続してあらわれる．

index	$k+2$	$k+1$	k	$k-1$	$\cdots$	$r+1$	r
SH_i	SH_{k+2}	SH_{k+1}	0	0	$\cdots$	0	SH_r
deg	$k+2$	r	$-\infty$	$-\infty$	$\cdots$	$-\infty$	r
c_i	$\neq 0$	c_{k+1}	0	0	$\cdots$	0	

(b) $r=k-1$ 場合，定理 6 (B) より

$$\mathrm{lc}(\mathsf{SH}_{k+2})^2\mathsf{SH}_{k-1}=-\mathrm{lc}(\mathsf{SH}_{k+1})^2\mathsf{SH}_{k+1}$$

なので，$\mathsf{SH}_{k-1}\mathsf{SH}_{k+1}\leq 0$ となる．

index	$k+2$	$k+1$	k	$k-1$
SH_i	SH_{k+2}	SH_{k+1}	0	SH_{k-1}
deg	$k+2$	$k-1$	$-\infty$	$k-1$
c_i	$\neq 0$	c_{k+1}	0	$-c_{k+1}$

(c) $r=k$ 場合，定理 6 (B) より $c_{k+1}=0$ となり 0 が 2 回あらわれる．

index	$k+2$	$k+1$	k	$k-1$
SH_i	SH_{k+2}	SH_{k+1}	SH_k	SH_{k-1}
deg	$k+2$	k	k	
c_i	$\neq 0$	0	0	c_{k-1}

補題 13
任意の $k=u+1,\ldots,n=2$ に対して，

$$c_{k-1}\neq 0, c_k=c_{k+1}=0, c_{k+2}\neq 0, s_k\neq 0, s_{k+1}\neq 0 \rightarrow s_{k-1}=-s_{k+1}.$$

証明 補題 9 より $c_{k+2}\neq 0$ なので $\mathsf{SH}_{k+2}\neq 0$．SH_{k+2} が正則でないと仮定すると，定理 6 (A) より SH_{k+1} は正則となる．さらに，定理 6 (B) より SH_{k+1} は SH_{k+2} の定数倍となり，$c_{k+2}\neq 0$ かつ $c_{k+1}=0$ に矛盾する．したがって，SH_{k+2} は正則．

SH_{k+1} が正則であるとすると，定理 6 (C) より，

$$\mathrm{lc}(\mathsf{SH}_{k+2})^2\mathsf{SH}_k=\delta_2\mathrm{Prem}(\mathsf{SH}_{k+2},\mathsf{SH}_{k+1})$$

SH_k と SH_{k+1} が x を共通因子を持ち，$f(0)\neq 0$ に矛盾する．したがって，SH_{k+1} は正則ではない．また定理 6 (A) よりその次数は k でなければならない．

定理 6 (B) より，$s_{k-1}=-s_{k+1}$ が得られる．

i	$k+2$	$k+1$	k	$k-1$
s_i	$\neq 0$	$\neq 0$	$\neq 0$	
c_i	$\neq 0$	0	0	$\neq 0$

補題 14
任意の $k=u+1,\ldots,n-2, m=2,\ldots,n-k-1$ に対して，

$$c_{k+m+1}\neq 0, c_{k+m}=\cdots=c_k=0 \rightarrow s_{k+m-1}=\cdots=s_{k+1}=0$$

証明 $m=2$ の場合を示す．

(i) $s_{k+2}\neq 0$ のとき，

i	$k+3$	$k+2$	$k+1$	k
s_i	$\neq 0$	$\neq 0$	?	
c_i	$\neq 0$	0	0	0

$s_{k+1}\neq 0$ と仮定する．補題 13 の証明より，SH_{k+2} は正則ではなく，SH_{k+3}, SH_{k+1} は正則となる．定理 6 (C) より，実数 t を用いて以下が成立する．

$$\mathsf{SH}_k=t\mathrm{Prem}(\mathsf{SH}_{k+3},\mathsf{SH}_{k+2}).$$

今，$c_{k+2}=c_k=0$ より，x が共通因子となり，$f(0)\neq 0$ に矛盾する．

(ii) $s_{k+2}=0$ のとき,

i	$k+3$	$k+2$	$k+1$	k
s_i	$\neq 0$	0	?	
c_i	$\neq 0$	0	0	0

$s_{k+1}\neq 0$ と仮定すると，定理 6 (B) より，SH_{k+1} は SH_{k+3} の定数倍なので $c_{k+3}\neq 0$, $c_{k+1}=0$ に矛盾する.

(i), (ii) より $s_{k+1}=0$ となる．$m>2$ の場合も同様. ∎

補題 15
任意の $k=u+1,\ldots,n-3$ に対して,

$$s_{k+2}=0, s_{k+1}\neq 0 \rightarrow s_k\neq 0.$$

証明 定理 6 (B) より，SH_{k+1} は正則．定理 6 (C) から，SH_k はある SH_r $(r>k+1)$ と SH_{k+1} の擬剰余により得られる．今，$k>u$ なので， $\mathsf{SH}_k\neq 0$ となる. ∎

補題 16
$k=u+1,\ldots,n-3$ に対して，$s_{k+m+1}\neq 0$, $s_{k+m}=\cdots=s_k=0$, $s_{k-1}\neq 0$ ならば，以下が成立する.

$$s^m_{k+m+2}s_{k-1} = \delta_{m+2}s^{m+1}_{k+m+1},$$
$$c^m_{k+m+2}c_{k-1} = \delta_{m+2}s^m_{k+m+1}c_{k+m+1}$$

i	$k+m+1$	$k+m$	$\cdots$	k	$k-1$
s_i	$\neq 0$	0	$\cdots$	0	$\neq 0$
c_i	?	0	$\cdots$	0	?

証明 定理 6 (B) より得られる.

∎

以上の補題により定理は示された.

4 論理式の簡単化

本章では，論理関数処理を用いた論理式の簡単化について述べる.

4.1 論理代数と論理関数の簡単化

本節では論理代数と論理関数を定義する.

定義 17
論理代数は，論理値の集合 $B=\{0,1\}$ に関する論理積 $(\cdot)$，論理和 $(+)$，論理否定 $(')$ の 3 つの演算からなる代数系として定義される．ここで論理積，論理和，論理否定は図 1 のように定義される.

定義 18
論理変数およびその否定のことをリテラル *(literal)*，1 個のリテラル，または複数個の互いに異なる変数のリテラルの論理積を積項 *(product term)*，1 個の積項，または複数の異なる積項の論理和を積和形 *(sum-of-products form* または *disjunctive form)* と呼ぶ.

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

x	x'
0	1
1	0

図 1: 論理演算子（論理積・論理和・論理否定）

定義 19
定義 *17* における論理演算子と括弧，及び任意の個数の論理変数と論理定数 *(0* または *1)* を組み合わせて計算手順を表した式を論理式と呼ぶ．また，関数 $f : B^n \to B$ を論理関数 *(logic function)* と呼ぶ．

定義 20
n 変数論理関数の入力値の *0*, *1* の組み合わせは 2^n 通りある．通常の論理関数は，すべての入力の組み合わせに対する出力が定義されており，そのような論理関数を完全指定論理関数 *(completely specified logic function)* と呼ぶ．それに対して，一部の入力値に対しては，出力が未定義な論理関数を不完全指定論理関数 *(incompletely specified logic function)* と呼び，出力が未定義な入力値をドントケア *(don't care)* と呼ぶ．

例えば，$(x+y)'$ と $x'+y'$ は等価な論理式であるように 1 つの論理関数は複数の等価な論理式により表現することができる．与えられた論理式に対して，冗長な部分があればこれを取り除き，より等価で簡単な論理式を得ることは，論理式の簡単化と呼ばれる．本稿では特に積項数の最小化を簡単化の目的とする．

例 3
以下の真理値表で定義される不完全指定論理関数 F を考える．出力値が未定義な場合には F 列に d を設定している．

x	y	z	F
0	0	0	1
0	0	1	d
0	1	0	0
0	1	1	0
1	0	0	d
1	0	1	1
1	1	0	1
1	1	1	1

不完全指定論理関数では，ドントケアを都合の良い値に解釈することにより，論理式を簡単化できる場合がある．上記の例では，d を *1* と解釈することにより，$F = x + z$ と簡単化することができる．

論理関数を表す論理式を簡単化することは回路素子の数や配線の本数が少なくなる設計につながるため実用上重要である．そのため，二分決定グラフ（Binary Decision Diagram：BDD）を用いた厳密解法やヒューリスティクスを用いた近似解法 ESPRESSO [3] など多くの研究がなされている．

4.2 論理関数処理を用いた φ_n の簡単化

論理式簡単化手法を利用して，Algorithm 1 の φ_n を簡単化することを考える．

まず，論理値は 2 値で，本稿で扱う多項式の符号は 3 値なので 2 つの変数を用いて s_k, c_k の符号を表現する．ここでは，x, y を用いて，$x'y'$ を 0，xy' を正，$x'y$ を負と表現し，φ_n を論理変数を用いた論理式として記述する．その結果に対して論理関数処理による簡単化手法の適用で，より簡単な φ_n の表現を得る．

例 4
式 (1) を論理関数で表現することを考える．3 つの多項式 b^2-4c, c, b の符号をそれぞれ x_1y_1, x_2y_2, x_3y_3 により表現すると，式 (1) は以下のような論理式で記述できる．

$$x_1'y_1x_2y_2'x_3'y_3 + x_1'y_1x_2y_2'x_3y_3 + x_1'y_1x_2y_2'x_3y_3' + x_1y_1x_2y_2'x_3y_3 + x_1y_1x_2y_2'x_3y_3' + x_1y_1'x_2y_2'x_3y_3'$$

4.1 節で述べたように，ドントケアの設定で論理式がより簡単化できることが期待される．SDC 専用の QE においては以下のようにドントケアとして設定できる．

1. 多項式の符号を x, y の 2 変数で表現することを上で述べた．符号は 3 値使用し，論理変数 2 変数で表すことができるのは 4 値であり，xy は現在使用していないため，ドントケアとして設定できる．
2. また，3 節の補題を満たさない符号列は，それを満たす実数が存在しないということなので，真と偽，どちらの値として扱っても結果に影響がないためドントケアとして設定できる．
3. 同様に，$V_0(H(f)) < V_\infty(H(f))$ とはならないので，この場合もドントケアとして設定する．

5 実験結果

論理関数処理による論理式簡単化に対する近似手法 ESPRESSO [1] を用いて φ_n の簡単化を行った結果を表 2 に示す．計算は Intel(R) Core(TM) 2 Duo CPU 1.6 GHz, 2.0 Gbyte メモリ上で行った．「次数」は入力となる多項式の次数，「変数の数」は ESPRESSO に与える論理変数の数，「KK」 は SyNRAC の以前の実装での積項の数，「DC」 は 4.2 節の補題 1 と補題 9 のみをドントケアで設定して ESPRESSO の近似解法を用いて簡単化した積項の数，「ESP 近似」 は ESPRESSO の近似解法を用いて簡単化した積項の数，「ESP 厳密」 は ESPRESSO の厳密解法を用いて簡単化した積項の数，「積項数」は定理 6 を満たす符号列の数，「近似時間」は 「ESP 近似」での実行時間（秒）を表す．「厳密時間」は 「ESP 厳密」での実行時間（秒）を表す．

次数 n の場合には Sturm-Habicht 列は $n+1$ 個からなるが，補題 8 より，$s_n = s_{n-1} = c_n = +$ であり，$s_0 = c_0$ となるので，符号条件を求めるのに考慮すべき対象は $2(n+1)-4 = 2n-2$ 個である．したがって論理変数の数としては $4n-4$ となる．実験結果から ESPRESSO を用いた簡単化により以前の実装「KK」に比べて論理和の数を大幅に削減できていることが確認できる．「DC」列と「ESP 近似」列の比較によりドントケアの設定で，より簡単化ができていることが確認できる．ESPRESSO の厳密解法では 7 次以上の結果が得られていないが，6 次以下の結果により近似解法で厳密解に近い結果が得られていることが確認できる．

図 2 は，3 次の場合の ESPRESSO コマンドに対する Programmable Logic Array (PLA) 形式で記述された入出力ファイルである．入力ファイルの 5 行目以降で符号列に対する出力値を定義し，出力ファイルの 6 行目以降で簡単化された論理式を表す．ハイフンは対応する論理変数が 0 でも 1 でも良いことを表し，行末の 1, 2 はそれぞれ，入力値が真であることとドントケアであることを表す．各変数は x_0y_0, x_1y_1, x_2y_2, x_3y_3 がそれぞれ，s_1, s_0, c_2, c_1 を表す．例えば 19 行目は，$s_1 = -, s_0 = 0, c_2 = +, c_1 = +$ を表し，このとき，$V_0(\{+,+,+\}) = 0 < 1 = V_\infty(\{+,+,-\})$ なのでドントケアとして設定している．

19 行目，および 27 行目は $V_0(H(f)) < V_\infty(H(f))$ であること，30 行目から 33 行目は xy が未使用であることからドントケアを設定し，28 行目，および 36 行目から 37 行目は補題 12，34 行目から 35 行目

表 2: 計算実験結果

次数	変数の数	KK	DC	ESP 近似	ESP 厳密	積項数	近似時間	厳密時間
2	4	5	6	2	2	5	0.11	0.09
3	8	17	32	4	4	22	0.15	0.10
4	12	64	176	10	10	103	0.17	0.17
5	16	302	998	19	19	494	0.31	1.47
6	20	1229	5781	59	58	2397	1.71	74.36
7	24	5238	33938	124	-	11772	40.95	>350h
8	28	20468	201105	362	-	58369	1582.31	>350h

は補題 9，38 行目から 39 行目は補題 11，40 行目から 41 行目は補題 10 を満たさないためドントケアとして設定している．

ESPRESSO コマンドにより，22 個の真の符号列が，4 個の積項に簡単化できており，この結果は，以下を表す．

$$(s_1 < 0 \wedge s_0 > 0) \vee (s_1 < 0 \wedge c_1 < 0) \vee (s_0 < 0 \wedge c_1 < 0) \vee (c_2 \geq 0 \wedge c_1 \geq 0)$$

$f(x) = x^3 + ax^2 + bx + c$ とすると，Sturm-Habicht 列は $\mathsf{SH}_3 = f(x)$, $\mathsf{SH}_2 = f'(x) = 3x^2 + 2ax + b$, $\mathsf{SH}_1 = (2a^2 - 6b)x + ab - 9c$, $\mathsf{SH}_0 = -4b^3 + a^2b^2 - 4a^3c + 18bac - 27c^2$ なので，

$$\begin{aligned}\forall x(x \geq 0 \rightarrow f(x) > 0) \leftrightarrow^{QE} \quad c > 0 \wedge \quad & ((2a^2 - 6b < 0 \vee 2a^2 - 6b = 0 \wedge ab - 9c < 0) \wedge \mathsf{SH}_0 > 0 \vee \\ & (2a^2 - 6b < 0 \vee 2a^2 - 6b = 0 \wedge ab - 9c < 0) \wedge ab - 9c < 0 \vee \\ & \mathsf{SH}_0 < 0 \wedge ab - 9c < 0 \vee \\ & b \geq 0 \wedge ab - 9c \geq 0)\end{aligned}$$

となる．

SyNRAC の以前の実装では積項数が 17 個だったのに対し，提案手法では積項数が 4 個と簡単化が実現できているが，さらに簡単化する余地があることがわかっている．例えば，3 次の結果における最初の積項から得られる条件

$$c > 0 \wedge ((2a^2 - 6b < 0 \vee 2a^2 - 6b = 0 \wedge ab - 9c < 0) \wedge \mathsf{SH}_0 > 0)$$

を満たす実数は存在しないので，さらに簡単化できる．実際，Cylindrical Algebraic Decomposition を用いた QE で 3 次の問題を解くと，以下のような結果が得られる．

$$c > 0 \wedge (b \geq 0 \wedge a \geq 0 \vee \mathsf{SH}_0 < 0)$$

3 節で示した SDC を満たす多項式に対する Sturm-Habicht 列に対する条件は，s_k または c_k が 0 になる部分に着目しており，0 を含まない場合に不要な符号列を削減することは今後の課題である．

6 まとめ

実用上重要なクラスである SDC に対する専用の QE の高速化のために論理式簡単化手法を利用し出力となる論理式の簡単化を行った．簡単化を実現するために，Sturm-Habicht 列が満たす条件を示し，簡単化には論理簡単化手法 ESPRESSO を用いた．このときドントケアを設定することによりより簡単な結果が得られることが確認できた．

今後の課題としては，以下のことが挙げられる．

- 符号が 0 を含まない部分でのあらわれない符号列を求めてさらに簡単な論理式を得る
- 7 次以上で得られているのは近似解であるため，二分決定グラフ (BDD) を用いた厳密解法による簡単化との比較.

参 考 文 献

[1] Espresso: Logic minimization software. `http://www.ecs.umass.edu/ece/labs/vlsicad/ece667/links/espresso.html`.

[2] H. Anai and S. Hara. A parameter space approach to fixed-order robust controller synthesis by quantifier elimination. *International Journal of Control*, 79(11):1321–1330, 2006.

[3] R. K. Brayton, A. L. Sangiovanni-Vincentelli, C. T. McMullen, and G. D. Hachtel. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, Norwell, MA, USA, 1984.

[4] L. González-Vega, T. Recio, H. Lombardi, and M.-F. Roy. *Sturm-Habicht sequences determinants and real roots of univariate polynomials*, pp. 300–316. Texts and Monographs in Symbolic Computation. Springer, 1998.

[5] R. Loos and V. Weispfenning. Applying linear quantifier elimination. *The Computer Journal*, 36(5):450–462, 1993.

[6] 穴井, 横山. QE の計算アルゴリズムとその応用 – 数式処理による最適化. 東京大学出版会, 8 2011.

```
 1 .lib x0 y0 x1 y1 x2 y2 x3 y3
 2 .i 8
 3 .o 1
 4 .ob f0
 5 01010101 1
 6 01010001 1
 7 01011001 1
 8 01011000 1
 9 01011010 1
10 00011000 1
11 10010101 1
12 10010001 1
13 10011001 1
14 10011000 1
15 10011010 1
16 01000101 1
17 01000001 1
18 01001001 1
19 01001010 2
20 00001000 1
21 10001010 1
22 01100101 1
23 01100001 1
24 01101001 1
25 01100100 1
26 01100110 1
27 01101010 2
28 00100100 2
29 10101010 1
30 11------ 2
31 --11---- 2
32 ----11-- 2
33 ------11 2
34 00----1- 2
35 00-----1 2
36 --101000 2
37 --010100 2
38 ----0010 2
39 ----0000 2
40 1000--00 2
41 0100--00 2
42 .e
```

```
.lib x0 y0 x1 y1 x2 y2 x3 y3
.i 8
.o 1
.ob f0
.p 4
-11----- 1
-1-----1 1
---1---1 1
-----0-0 1
.e
```

図 2: 3 次の問題に対する ESPRESSO コマンドの入力ファイル（左）と出力ファイル（右）