数理解析研究所講究録
第2019巻 2017年 124-142

不等式制約をもつ論理式に対する包括的グレブナー基底系を利用した限量記号消去の出力の簡単化

Formula Simplification for Real Quantifier Elimination by Comprehensive Gröbner Systems with Inequality Constraints

岩根秀直
(株) 富士通研究所/国立情報学研究所 *
Hidenao Iwane
Fujitsu Laboratories Ltd/National Institute of Informatics

深作亮也
東京理科大学 †
Ryoya Fukasaku
Tokyo University of Science

佐藤洋祐
東京理科大学 ‡
Yosuke Sato
Tokyo University of Science

1 はじめに

限量記号消去法（Quantifier Elimination: QE）[12, 5, 16] は限量記号がついた一階述語論理式を入力として，それと等価で限量記号のない論理式を出力するアルゴリズムである．

QE は多くの応用がある重要なアルゴリズムで，効率化のための様々な研究がなされている．しかし，汎用 QE は，最悪の計算量の下限が限量記号の交代の数に対し，二重指数であること [7] が示されており，現在知られている最も効率のいい手法である Cylindrical Algebraic Decomposition (CAD) [6] でも，多くとも 5 変数程度までの問題しか解けない．そのため，入力に制限を加えることで効率化を実現した専用 QE が提案されている．例えば，線形や 2 次など束縛変数の次数に制限を加えた一階述語論理に対する Virtual Substitution [13, 14], Sign Definite Condition (SDC) と呼ばれる多項式の正定性を求める問題に対して，Sturm-Habicht 列を用いた QE [2, 10] などがある．包括的グレブナー基底系（Comprehensive Gröbner System）を利用した QE (以下，CGS-QE) [15, 9] もその一つで（複数の）等式制約を持つ場合に効率的に動作する．

本稿では，不等式制約をもつ論理式に対する CGS-QE の出力の公式を論理関数処理による手法 [10] で簡単化する方法について述べる．QE は，自由変数がない閉論理式の場合を除いて，出力の論理式に対して

*iwane@jp.fujitsu.com
†fukasaku@rs.tus.ac.jp
‡ysato@rs.kagu.tus.ac.jp

別の操作（実行可能領域の描画）を行うための前処理として用いるものであり，簡単な論理式が生成されることが期待される．また，CGS-QE を含めて再帰的な QE アルゴリズムも多いため，QE 計算の時間を削減するためにも論理式の簡単化は重要である．

2 CGS-QE

本稿では 以下のような等式制約をもつ一階述語論理式で，等式制約に関するイデアル $\langle f_i \rangle$ が束縛変数 $\bar{X} := x_1, \ldots, x_n$ に関して零次元である場合の CGS-QE の出力の公式を考える．

$$\varphi \equiv \exists \bar{X} \left((\wedge_i f_i = 0) \wedge (\wedge_{j=1}^{\ell} h_j > 0) \right) \tag{1}$$

イデアルが束縛変数に関して零次元であるので，以下の多変数実根個数計算手法 [11] が適用できる．

定理 1
I を $\mathbb{R}[\bar{X}]$ 上の零次元イデアル，$V_{\mathbb{R}}(I)$ を I の $\mathbb{R}$ における多様体とする．$f \in \mathbb{R}[\bar{X}]$ に対して，$\mathbb{R}[\bar{X}]/I$ を $\mathbb{R}$ 上の線形空間とみなし，$t_1, \ldots, t_k$ をその基底とする．任意の $g \in \mathbb{R}[\bar{X}]$ に対して，$\mathbb{R}[\bar{X}]/I$ から $\mathbb{R}[\bar{X}]/I$ への線形写像 $\theta_{g,i,j}(f) = g t_i t_j f$ とする．$q_{g,i,j}$ を $\theta_{g,i,j}$ のトレース，M_g^I を (i,j) 成分を $q_{g,i,j}$ とする対称行列，$\chi_g^I(x)$ をその特性多項式，$\sigma(M_g^I)$ を M_g^I の符号数とする．このとき，以下が成立する．

$$\sigma(M_1^I) = \#(V_{\mathbb{R}}(I))$$

今，M_g^I は実対称行列であるため，その特性多項式は実根のみをもつ．したがって，デカルトの符号律を利用すれば，正と負の実根の数をそれぞれ正確に数え上げることが出来る．この事実を利用して，CGS-QE では $\sigma(M_1^I) \neq 0$ となる係数の符号条件を与えることにより，QE が実現される．以下，$\sigma(M_g^I)$ と $\sigma(\chi_g^I(x))$ を同一視する．

不等式制約を持つ場合には，特性多項式が可約であることが示されている [9]．

定理 2
I を $\mathbb{R}[\bar{X}]$ 上の零次元イデアル，$h_1, \ldots, h_\ell$ を $\mathbb{R}[\bar{X}]$ 上の多項式，$J = I + \langle Z_1^2 - h_1, \ldots, Z_\ell^2 - h_\ell \rangle$ を $\mathbb{R}[\bar{X}, \mathbb{Z}]$ 上のイデアルとする．k を $\mathbb{R}[\bar{X}]/I$ の次元，$\{t_1, \ldots, t_k\} \subset T(\bar{X})$ を $\mathbb{R}[\bar{X}]/I$ を $\mathbb{R}[\bar{X}]$ 上の線形空間とみなしたときの基底とするとき，$\{t_1 Z_1^{e_1} Z_2^{e_2} \cdots Z_\ell^{e_\ell}, \ldots, t_k Z_1^{e_1} Z_2^{e_2} \cdots Z_\ell^{e_\ell} | (e_1, \ldots, e_\ell) \in \{0,1\}^\ell\}$ はベクトル空間 $\mathbb{R}[\bar{X}, \mathbb{Z}]/J$ の基底をなす．χ_g^J を $g \in \mathbb{R}[\bar{X}]$ に対して，上記の $\mathbb{R}[\bar{X}, \mathbb{Z}]/J$ の基底から導入される特性多項式とする．χ_g^I を $g \in \mathbb{R}[\bar{X}]$ に対して，上記の $\mathbb{R}[\bar{X}]/I$ の基底から導入される特性多項式とする．このとき，非零の定数 c を用いて，以下が成立する．

$$\chi_g^J(2^\ell x) = c \prod_{(e_1, \ldots, e_\ell) \in \{0,1\}^\ell} \chi_{g h_1^{e_1} \cdots h_\ell^{e_\ell}}^I(x)$$

不等式制約がある場合に，可約なことを利用すると，因数分解しない場合の公式（$k = d, \ell = 0$）に代入した結果よりも非常に簡単になる．以下は，$k = 2, \ell = 2$ の場合で，特性多項式が

$$(x^2 + a_1 x + b_1)(x^2 + a_2 x + b_2)(x^2 + a_3 x + b_3)(x^2 + a_4 x + b_4)$$

とおいた場合の人手での簡単化結果 [8] である．

$(a_0 \neq 0 \wedge b_0 = 0 \wedge a_1 = 0 \wedge a_2 = 0 \wedge a_3 = 0) \vee (a_1 < 0 \wedge b_1 = 0 \wedge a_2 = 0 \wedge a_3 = 0) \vee (b_0 > 0 \wedge b_1 = 0 \wedge a_2 = 0 \wedge a_3 = 0) \vee (a_1 \leq 0 \wedge a_2 < 0 \wedge b_2 = 0 \wedge a_3 = 0) \vee (b_0 > 0 \wedge a_1 = 0 \wedge b_2 = 0 \wedge a_3 = 0) \vee (a_1 \leq 0 \wedge a_2 \leq 0 \wedge a_3 < 0 \wedge b_3 = 0) \vee (b_1 = 0 \wedge a_2 \leq 0 \wedge a_3 \leq 0 \wedge b_3 > 0) \vee (b_0 > 0 \wedge a_1 \leq 0 \wedge a_2 \leq 0 \wedge b_3 = 0) \vee (a_1 \leq 0 \wedge b_1 > 0 \wedge a_2 \leq 0 \wedge b_3 =$

$0)\vee(a_1\leq 0\wedge a_2\leq 0\wedge b_2>0\wedge b_3\leq 0)\vee(b_0>0\wedge b_1\leq 0\wedge a_2\leq 0\wedge b_2>0)\vee(b_0>0\wedge a_1\leq 0\wedge b_1>0\wedge b_2\leq 0)\vee(a_0\neq 0\wedge b_0=0\wedge b_1<0\wedge a_2=0\wedge a_3=0)\vee(a_0\neq 0\wedge b_0=0\wedge a_1=0\wedge b_2<0\wedge a_3=0)\vee(b_0>0\wedge a_1\leq 0\wedge a_3<0\wedge b_3\geq 0)\vee(a_1\leq 0\wedge b_1>0\wedge a_3<0\wedge b_3=0)\vee(b_0>0\wedge a_2\leq 0\wedge a_3<0\wedge b_3\geq 0)\vee(b_1<0\wedge a_2\leq 0\wedge a_3<0\wedge b_3\geq 0)\vee(a_2\leq 0\wedge b_2>0\wedge a_3<0\wedge b_3=0)\vee(a_1\leq 0\wedge b_2<0\wedge a_3<0\wedge b_3\geq 0)\vee(a_0\neq 0\wedge b_0=0\wedge b_1\neq 0\wedge b_2\neq 0\wedge b_3\neq 0)\vee(a_0=0\wedge a_1\neq 0\wedge b_1=0\wedge b_2\neq 0\wedge b_3\neq 0)\vee(b_0\neq 0\wedge a_1\neq 0\wedge b_1=0\wedge b_2\neq 0\wedge b_3\neq 0)\vee(a_0=0\wedge b_1\neq 0\wedge a_2\neq 0\wedge b_2=0\wedge b_3\neq 0)\vee(b_0\neq 0\wedge b_1\neq 0\wedge a_2\neq 0\wedge b_2=0\wedge b_3\neq 0)\vee(a_1<0\wedge b_1\geq 0\wedge a_3\leq 0\wedge b_3>0)\vee(b_0>0\wedge b_1<0\wedge a_2\leq 0\wedge b_3\leq 0)\vee(b_1<0\wedge a_2\leq 0\wedge b_2>0\wedge b_3\leq 0)\vee(b_0>0\wedge a_1\leq 0\wedge b_2<0\wedge b_3\leq 0)\vee(a_1\leq 0\wedge b_1>0\wedge b_2<0\wedge b_3\leq 0)\vee(a_0\neq 0\wedge b_0\geq 0\wedge a_1=0\wedge a_2=0\wedge b_3<0)\vee(a_1<0\wedge b_1\geq 0\wedge a_2\leq 0\wedge b_3<0)\vee(a_0\neq 0\wedge b_0\geq 0\wedge b_1\leq 0\wedge a_2<0\wedge b_2\geq 0\wedge a_3\leq 0)\vee(a_0\neq 0\wedge b_0\geq 0\wedge a_1<0\wedge b_1\geq 0\wedge b_2\leq 0\wedge a_3\leq 0)\vee(a_0\neq 0\wedge b_0=0\wedge b_1\neq 0\wedge b_2\neq 0\wedge a_3=0)\vee(a_0=0\wedge a_1\neq 0\wedge b_1=0\wedge b_2\neq 0\wedge a_3=0)\vee(b_0\neq 0\wedge a_1\neq 0\wedge b_1=0\wedge b_2\neq 0\wedge a_3=0)\vee(a_0=0\wedge b_1\neq 0\wedge a_2\neq 0\wedge b_2=0\wedge a_3=0)\vee(b_0\neq 0\wedge b_1\neq 0\wedge a_2\neq 0\wedge b_2=0\wedge a_3=0)\vee(b_0>0\wedge b_1<0\wedge a_3<0\wedge b_3\geq 0)\vee(b_0>0\wedge b_2<0\wedge a_3<0\wedge b_3\geq 0)\vee(b_1<0\wedge b_2<0\wedge a_3<0\wedge b_3\geq 0)\vee(a_0\neq 0\wedge b_0=0\wedge b_1\neq 0\wedge a_2=0\wedge b_3\neq 0)\vee(b_0>0\wedge a_1\neq 0\wedge b_1=0\wedge a_2=0\wedge b_3\neq 0)\vee(a_0\neq 0\wedge b_0=0\wedge a_1=0\wedge b_2\neq 0\wedge b_3\neq 0)\vee(a_0=0\wedge a_1=0\wedge a_2\neq 0\wedge b_2=0\wedge b_3\neq 0)\vee(b_0\neq 0\wedge a_1=0\wedge a_2\neq 0\wedge b_2=0\wedge b_3\neq 0)\vee(a_0\neq 0\wedge b_0\geq 0\wedge b_1\leq 0\wedge b_2\leq 0\wedge a_3\leq 0\wedge b_3>0)\vee(b_0>0\wedge b_1>0\wedge b_2>0\wedge b_3\leq 0)\vee(b_0>0\wedge b_1<0\wedge b_2<0\wedge b_3\leq 0)\vee(a_0\neq 0\wedge b_0\geq 0\wedge a_1<0\wedge b_1\geq 0\wedge a_2<0\wedge b_2\geq 0)\vee(b_0\leq 0\wedge b_1>0\wedge a_2<0\wedge b_2\geq 0\wedge b_3>0)\vee(a_0\neq 0\wedge b_0\geq 0\wedge b_1\leq 0\wedge a_2<0\wedge b_2\geq 0\wedge b_3<0)\vee(a_0\neq 0\wedge b_0\geq 0\wedge a_1<0\wedge b_1\geq 0\wedge b_2\leq 0\wedge b_3<0)\vee(a_0\neq 0\wedge b_0=0\wedge a_1\neq 0\wedge b_1=0\wedge b_2\neq 0\wedge a_3\neq 0\wedge b_3=0)\vee(a_0\neq 0\wedge b_0=0\wedge b_1\neq 0\wedge a_2\neq 0\wedge b_2=0\wedge a_3\neq 0\wedge b_3=0)\vee(a_0=0\wedge a_1\neq 0\wedge b_1=0\wedge a_2\neq 0\wedge b_2=0\wedge a_3\neq 0\wedge b_3=0)\vee(b_0\neq 0\wedge a_1\neq 0\wedge b_1=0\wedge a_2\neq 0\wedge b_2=0\wedge a_3\neq 0\wedge b_3=0)\vee(a_0\neq 0\wedge b_0=0\wedge a_1\neq 0\wedge b_1=0\wedge a_2=0\wedge a_3\neq 0\wedge b_3=0)\vee(a_0\neq 0\wedge b_0=0\wedge a_1=0\wedge a_2\neq 0\wedge b_2=0\wedge a_3\neq 0\wedge b_3=0)$

一方，$k=8,\ell=0$ の公式に上記の特性多項式を代入すると，上記の式よりも冗長になる上，係数は $a_1,b_1,\ldots,a_4,b_4$ の 4 次式となり，後処理での計算量も大きくなる．

3 論理式の簡単化

本節では，例を用いて，本稿で扱う論理式の簡単化について述べる．通常，自由変数がない閉論理式の場合を除いて出力の論理式に対して，実行可能領域の描画などの別の操作を行うための前処理として QE は用いられる．そのため，QE の出力はできるかぎり簡単であるべきである．また，再帰的な QE アルゴリズムも多いため，QE としての計算量を削減するためにも論理式の簡単化は重要である．(QE の出力はアルゴリズムや実装方法によって大きく異なる．いくつかの QE ツールでの実行結果を付録 A に掲載している)

以下では，定理 1 の手順で得られる特性多項式 $\chi_g^I(x)$ を単に特性多項式と表記する．モニックであると仮定しても一般性を失わないので，特性多項式を $x^d+\sum_{i=0}^{d-1}C_ix^i$ とする．

$d=2$ (例えば，$k=2,\ell=0$) の場合を例に考える．前節で述べたように，表 1 （これを本稿では**真偽値表**とよぶ）のように C_1, C_0 の符号により，与えられた論理式 (1) の真偽が判定できる．例えば 1 行目は，$C_1<0\wedge C_0<0$ を表していて，このとき特性多項式の係数の符号列は $[+,-,-]$ となり，正の実根の数も負の実根の数も 1 である．従って，符号数は 0 となり，この条件のとき定理 1 より (1) は偽となる．表 1 の真の行を取り出せば，(1) と等価な CGS-QE の出力の論理式 (2) が構築できる．ここでは，$>,<,=$ の

表 1: 真偽値表：次数 2 の特性多項式の係数の符号と真偽値の関係

	C_1	C_0	符号数	真偽値
1	$-$	$-$	0	偽
2	0	$-$	0	偽
3	$+$	$-$	0	偽
4	$-$	0	1	真
5	0	0	0	偽
6	$+$	0	-1	真
7	$-$	$+$	2	真
8	0	$+$	0	偽
9	$+$	$+$	-2	真

みを用いていることに注意されたい.

$$\begin{aligned} &C_1 < 0 \wedge C_0 = 0 \vee \\ &C_1 > 0 \wedge C_0 = 0 \vee \\ &C_1 < 0 \wedge C_0 > 0 \vee \\ &C_1 > 0 \wedge C_0 > 0 \end{aligned} \tag{2}$$

ここで， $g = 0 \vee g > 0 \Leftrightarrow g \geq 0$ を利用すると，以下の等価な式が得られる.

$$\begin{aligned} &C_1 < 0 \wedge C_0 \geq 0 \vee \\ &C_1 > 0 \wedge C_0 \geq 0 \end{aligned} \tag{3}$$

さらに，$g < 0 \vee g > 0 \Leftrightarrow g \neq 0$ を利用すると，以下のような簡単な式が得られる.

$$C_1 \neq 0 \wedge C_0 \geq 0$$

出力の形式を選言標準形（$\vee_j \wedge_i C_i \rho_{i,j} 0$）の形式に制限すると，論理式の簡単化は表の真の表を取り出して得られた論理式たちに対して，上記の操作をどううまく組み合わせるかという問題になる．しかし，次数が大きな問題では，組合せが膨大で，手作業で上記の操作を行って簡単化するのは困難である．そこで，論理式の簡単化を組合せ最適化問題のひとつである集合被覆問題として扱い，計算機で解くことを考える.

定義 3

要素集合 $T = \{1, \ldots, |T|\}$，その部分集合 S，T の部分集合の族 V' が与えられたとき，すべての $i \in S$ に対してある $v \in V'$ が存在し，$i \in v$ となる条件を満たすとき，V' は S を**被覆**するという．T の部分集合族 $V = \{v_1, \ldots, v_n\}$ が与えられたとき，S を被覆するようにいくつかの集合を選び，選んだ集合に付けられた重みの総和を最小にする問題を**集合被覆問題**という．この問題は以下のような $0-1$ 整数最適化問題として定式化される.

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} w_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i \in S \\ & x_j \in \{0,1\} \quad (\forall j \in \{1, \ldots, n\}) \end{aligned}$$

- a_{ij}: $i \in v_j$ ならば 1, そうでなければ 0 となる定数
- w_j: 集合 v_j の重み（正の定数）
- x_j: 集合 v_j が集合被覆に含まれるなら 1, そうでなければ 0 となる変数

T を真偽値表の行番号の集合，S を真偽値表の真となる行番号の集合（論理式 φ を表現），V の要素を論理積のみで表現される偽の行を含まない論理式の行番号の集合，つまり，$\wedge_i C_i \rho_i 0$ ($\rho_i \in \{>, <, =, \geq, \leq, \neq, \top\}$) で表現されるもの（ただし，$C_i \top 0 \leftrightarrow \top$ とする）で，偽の行を含まないものとすると，最適化によって得られる最適解のうち $x_j = 1$ となるような v_j に対応する論理式の論理和を取れば，重みによって定められる簡単な論理式が得られる．すべての重みを $w_j = 1$ とすると，論理和の数を最小にする問題となる．

上記の例（2 次の場合）では，$T = \{1, 2, \ldots, 9\}$, $S = \{4, 6, 7, 9\}$ で，$C_1 \rho_1 0 \wedge C_0 \rho_0 0$ で表現されるもので，偽の領域を含まない以下の論理式が V の要素となる．

$$
\begin{aligned}
&C_1 < 0 \wedge C_0 = 0 \leftrightarrow \{4\} \\
&C_1 > 0 \wedge C_0 = 0 \leftrightarrow \{6\} \\
&C_1 < 0 \wedge C_0 > 0 \leftrightarrow \{7\} \\
&C_1 > 0 \wedge C_0 > 0 \leftrightarrow \{9\} \\
&C_1 \neq 0 \wedge C_0 = 0 \leftrightarrow \{4, 6\} \\
&C_1 \neq 0 \wedge C_0 > 0 \leftrightarrow \{7, 9\} \\
&C_1 < 0 \wedge C_0 \geq 0 \leftrightarrow \{4, 7\} \\
&C_1 > 0 \wedge C_0 \geq 0 \leftrightarrow \{6, 9\} \\
&C_1 \neq 0 \wedge C_0 \geq 0 \leftrightarrow \{4, 6, 7, 9\}
\end{aligned}
$$

例えば，$\{1, 9\}$ は，$C_1 < 0 \wedge C_0 < 0 \vee C_1 > 0 \wedge C_0 > 0$ であり，$C_1 \cdot C_0 > 0$ と表現できるが，本稿では被覆の候補とはしていない．

集合被覆問題は整数計画問題なので，整数計画ソルバーを用いて解くことができる．しかし，その計算量は NP 困難であることが知られており，規模の大きな問題の厳密解を得ることは困難である．

そのため，Cylindrical Algebraic Decomposition による QE における論理式の構築においては，効率的に計算するための近似手法 [4] が提案されている．Sign Definite Condition 専用 QE の出力の簡単化においては，次節で説明する論理関数処理を用いた方法 [10] が提案されている．

4 論理関数処理を用いた論理式の簡単化

本節では，論理関数処理を用いた論理式の簡単化 [10] について述べる．本手法は，有限個の多項式が与えられ、それら符号により真偽値が決定されるような論理式の簡単化に適用できる．つまり，真偽値表が得られているような状況で，簡単な論理式を構築したい場合に適用可能である．論理関数処理は特別な集合被覆問題のみを扱うため，汎用の整数計画ソルバーで解くよりも効率的に解ける．

4.1 論理代数とブール式の簡単化

ここでは論理代数と論理関数を定義する．

定義 4
論理代数は，論理値の集合 $B = \{0, 1\}$ に関する論理積 $(\odot)$，論理和 $(\oplus)$，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．

X	Y	$X \odot Y$
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	$X \oplus Y$
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 5
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．

定義 6
定義 4 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**ブール式** *(Boolean expression)* と呼ぶ．また，関数 $f : B^n \to B$ を**論理関数** *(logic function)* と呼ぶ．

定義 7
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: DC)* と呼ぶ．

例えば，$(X \oplus Y)'$ と $X' \oplus Y'$ は等価な論理関数であるように 1 つの論理関数は複数の等価なブール式により表現できる．本稿では，与えられたブール式に対して，等価でより積項数の少ないブール式を得ることをブール式の**簡単化**と呼ぶ．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して，ブール式をより簡単化できる場合がある．

論理関数を表すブール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 論理式とブール式

論理関数処理を利用して，論理式を簡単化することを考える．考えている問題において，表 1 に対応する真偽値表（$\tau : \{-1, 0, 1\}^d \mapsto \{$ 真, 偽,DC$\}$）が得られているとする．つまり，d 個の多項式 $\{C_i\}_{i=0}^{d-1}$ が与えられて，それら符号を決定すれば真偽値が決まるとする．

まず，論理式 (2) をブール式で記述することを考える．論理変数は 2 つの値をとり，符号は 3 つの値をとるので 2 つの論理変数 X_i, Y_i を用いて C_i の符号を表現する．例えば，$X_i' \odot Y_i'$ を零，$X_i \odot Y_i'$ を正，$X_i' \odot Y_i$ を負とすると，表 2 のような対応になる．X_i' は $X_i \odot Y_i \oplus X_i \odot Y_i'$ と等価なので，$C_i < 0 \vee C_i = 0$

表 2: ブール式と論理式の対応

PLA	11	10	1-	01	00	0-	-1	-0	--
ブール式	$X_i \odot Y_i$	$X_i \odot Y_i'$	X_i	$X_i' \odot Y_i$	$X_i' \odot Y_i'$	X_i'	Y_i	Y_i'	1
論理式	DC	$C_i > 0$	$C_i > 0$	$C_i < 0$	$C_i = 0$	$C_i \leq 0$	$C_i < 0$	$C_i \geq 0$	$\top$

を表現しており，つまり，$C_i \leq 0$ と対応する．論理変数 2 つでは，非等式 ($\neq$) が表現されないことに注意されたい．

上記の設定で，論理式 (2) はブール式で

$$X_1' \odot Y_1 \odot X_0' \odot Y_0' \oplus$$
$$X_1 \odot Y_1' \odot X_0' \odot Y_0' \oplus$$
$$X_1' \odot Y_1 \odot X_0 \odot Y_0' \oplus$$
$$X_1 \odot Y_1' \odot X_0 \odot Y_0'$$

と記述できる．例えば，$C_1 < 0 \wedge C_0 > 0$ は，表 2 から $X_1 = 0 \wedge Y_1 = 1 \wedge X_0 = 1 \wedge Y_0 = 0$ であり，上記のブール式に代入すると 1 が復帰される．このブール式を論理関数処理により簡単化（$Z \oplus Z' = 1$ を利用）すると，

$$X_1 \odot Y_0' \oplus$$
$$Y_1 \odot Y_0'$$

が得られ，表 2 から簡単化された以下の論理式（(3) と同一）が得られる．

$$C_1 > 0 \wedge C_0 \geq 0 \vee$$
$$C_1 < 0 \wedge C_0 \geq 0$$

4.3 ドントケア

ドントケアを用いるとブール式がより簡単化できる．本稿の問題設定では，$X_i \odot Y_i$ は入力として現れないのでドントケアとして扱うことができる．

その他に，真偽値表の各行 $(\wedge C_i \rho_i 0)$ を満たす自由変数が存在しない場合には，その論理式は偽になるので，それを論理和として加えても加えなくても結果に影響しない．別の言い方をすると，この論理式によって表現される集合は空なので，その真偽値を真としても偽としても結果に影響が無いため，ドントケアと扱うことができる．

例えば，表 1 においては，符号数が負になっている行があるが，定理 1 から，$\sigma(\chi_1^J)$ は $V_{\mathbb{R}}(J)$ の要素数を表しており，非負の値を取ることが保証される．つまり，符号数が負になる行の条件を満たす自由変数は存在しないことになる．また，$C_1 = 0 \wedge C_0 > 0$ の場合には，特性多項式は実根をもたないので，特性多項式が実対称行列から生成されるという仮定を満たさない．このように各行を満たす実数が存在しないものをドントケアとすると，表 3 が得られ，すべてのドントケアを真と扱えば，

$$C_0 \geq 0$$

表 3: 真偽値表：次数 2 の特性多項式の係数の符号と真偽値の関係 (DC あり)

	C_1	C_0	符号数	真偽値
1	−	−	0	偽
2	0	−	0	偽
3	+	−	0	偽
4	−	0	1	真
5	0	0	0	DC
6	+	0	−1	DC
7	−	+	2	真
8	0	+	0	DC
9	+	+	−2	DC

が得られる.

ドントケアとなる符号条件をみつけさえすれば，論理関数処理がドントケアを真または偽のどちらに扱うべきかは判断し，簡単なブール式を生成する.

図 2 に表 3 に対応する ESPRESSO [1] を利用して解いた実行例を示す．入力ファイルの 3 行目から 8 行目が表 3 を表している．最初の 4 列が X_1, Y_1, X_0, Y_0 の値を表していて（表 2 の PLA 行参照），最後の列に入力値に対応する論理式の真偽値を表している．真偽値は真の場合に 1, 偽の場合に 0, ドントケアの場合に 2 を記述する．偽の場合はその行を省略可能である．9 行目から 10 行目は，$X_i \odot Y_i$ をドントケアと扱うことを表している．このファイルを入力として実行すると出力として,「---0 1」の 1 行が得られ，表 2 から，等価な論理式 $C_0 \geq 0$ が得られる.

```
1  .i 4
2  .o 1
3  0100 1
4  0000 2
5  1000 2
6  0110 1
7  0010 2
8  1010 2
9  11-- 2
10 --11 2
11 .e
```

```
.i 4
.o 1
.p 1
---0 1
.e
```

図 2: 表 3 に対応する ESPRESSO コマンドの入力（左）と出力（右）

注意 1

集合被覆問題でも自然にドントケアを扱うことが出来る．表 *1* を用いた，*3* 節の場合に対して，S は表 *3* のうち真となる行の論理和なので，より小さな集合 $\{4, 7\}$ になる．被覆の候補 V は，表 *3* のうち，偽を含

まない論理積で表現される集合は以下になる.(ドントケアのみを含むものは冗長なので削除している)

$$
\begin{aligned}
C_1 < 0 \wedge C_0 = 0 &\leftrightarrow \{4\} \\
C_1 \leq 0 \wedge C_0 = 0 &\leftrightarrow \{4,5\} \\
C_1 \neq 0 \wedge C_0 = 0 &\leftrightarrow \{4,6\} \\
C_0 = 0 &\leftrightarrow \{4,5,6\} \\
C_1 < 0 \wedge C_0 > 0 &\leftrightarrow \{7\} \\
C_1 \leq 0 \wedge C_0 > 0 &\leftrightarrow \{7,8\} \\
C_1 \neq 0 \wedge C_0 > 0 &\leftrightarrow \{7,9\} \\
C_0 > 0 &\leftrightarrow \{7,8,9\} \\
C_1 < 0 \wedge C_0 \geq 0 &\leftrightarrow \{4,7\} \\
C_1 \leq 0 \wedge C_0 \geq 0 &\leftrightarrow \{4,5,7,8\} \\
C_1 \neq 0 \wedge C_0 \geq 0 &\leftrightarrow \{4,6,7,9\} \\
C_0 \geq 0 &\leftrightarrow \{4,5,6,7,8,9\}
\end{aligned}
$$

対応する論理式に含まれる論理式の数を考慮するように重みを設定すると,$C_0 \geq 0$ を求めることができる.集合被覆問題の立場からは,ドントケアを利用することで,被覆される S は集合として小さくなる上,被覆の候補集合 V の要素が多くなるため,より簡単な結果が得られることになる.

5 特性多項式の係数が満たす条件

本節では,特性多項式 $\chi_1^J(x)$ の係数の符号がみたすべき必要条件を考える.本節で示す必要条件をみたさない符号条件は,ドントケアとして扱い,簡単化に利用する.ここで,ドントケアとして扱える符号条件を多く見つけることが,論理式の簡単化につながる.ここでは,イデアル $\langle f_i \rangle$ の任意の元に対して,$h_j \neq 0$ であると仮定する.つまり,$h_j > 0$ と $h_j \geq 0$ が同一視できる状況であるとする.

定理 1 を利用して得られる必要条件を以下に示す.

- χ_1^J は実根のみをもつ.
- $\sigma(\chi_1^I) \geq 0, \sigma(\chi_1^J) \geq 0.$
- $\chi_1^I \neq x^k, \chi_1^J \neq x^d.$

実根のみをもつときの係数の符号の条件として,以下の定理が利用できる.

定理 8
次数 d の多項式 $g(x)$ の係数の符号変化の数を $N(g(x))$, $x = 0$ の重根度を u とする.$g(x)$ が実根のみをもつ場合には,以下が成立する.

$$N(g(x)) + N(g(-x)) + u = d$$

証明 デカルトの符号律から,$N(g(x))$ は正の実根の最大値,$N(g(-x))$ は負の実根の最大値である.g の 実根の個数は重複を込めて高々 $N(g(x)) + N(g(-x)) + u$ 個であり,この値は d 以下の値をとる.したがって,d より小さい場合にはかならず虚根を持つ. ∎

定理 2 から以下の係数の符号に関する必要条件が得られる.

- $\sigma(\chi_1^J)$ は 2^ℓ で割り切れる.
- $\ell = 1$ のとき, $|\sigma(\chi_{h_1}^I)| \leq \sigma(\chi_1^I)$.
- 任意の $(e_1, \ldots, e_\ell) \in \{0,1\}^\ell$ に対して $(\sigma(\chi_1^I) + \sigma(\chi^I_{h_1^{e_1} \cdots h_\ell^{e_\ell}}(x)))/2 \geq \sigma(\chi_1^J(x))/2^\ell$.
- $\ell = 2$ のとき, $i \in \{1,2\}$ に対して $\sigma(\chi_{h_i}^I) = \sigma(\chi_1^I)$ ならば, $\sigma(\chi_{h_{3-i}}^I) = \sigma(\chi^I_{h_1 h_2})$.

さらに, 以下のように部分問題を定義すると, 任意の部分問題は φ の必要条件であり, 上記の条件を満たす必要がある.

定義 9
M を添字集合 $\{1, \ldots, \ell\}$ の部分集合, S を M の直和分解とする. すなわち, S の任意の 2 つの要素は互いに素で, $M = \cup_{s \in S} s$ が成立する. このとき, 以下を φ の**部分問題**と呼ぶ.

$$\exists x_1 \cdots \exists x_n \left((\wedge_i f_i = 0) \wedge (\wedge_{s \in S} \prod_{j \in s} h_j > 0) \right)$$

これらの条件を満たさない入力をドントケアとして簡単化実験を行った.

6 計算機実験結果

表 4 に統計情報を示す. d が同じ値の場合でも因数分解可能な条件を用いることで, 非常に小さな論理式を構成できていることが確認できる. たとえば, $k = 8, \ell = 0$ の場合には, 10 個の論理和だったのに対して, $k = 2, \ell = 2$ の場合には, 97.82% の符号情報をドントケアとして扱えたこともあり, 論理和記号 3 個にまで簡単化できている. 以下が簡単化によって得られた式である. 2 節で紹介した手計算での簡単化結果にくらべて大きく改善できていることがわかる.

$$\begin{aligned} &a_2 < 0 \wedge b_2 > 0 \wedge b_4 < 0 \vee \\ &a_3 < 0 \wedge b_3 > 0 \wedge b_4 < 0 \vee \\ &b_3 < 0 \wedge a_4 < 0 \wedge b_4 > 0 \vee \\ &a_2 < 0 \wedge b_2 \geq 0 \wedge a_4 < 0 \end{aligned}$$

3^d 個の符号条件の列挙および評価が必要で, 最大で特性多項式の次数 $d = 16$ まで計算できているが, 扱える不等式の数としては高々 3 程度という状態である. 不等式が 4 個以上の場合は, 例えば, 不等式が 4 個 ($\ell = 4$) の場合には, $k = 2$ の場合でも $d = 32$ で $3^{32} \approx 2^{50}$ 行の真偽値表が必要なので, 実時間での計算の完了が期待できない状態である.

7 まとめ

論理関数処理による簡単化手法により, 不等式制約をもつ論理式に対する CGS-QE の出力の公式を構築する方法について述べた. 特性多項式の係数の符号が満たす必要条件を利用して, 従来よりも簡単化した結果が得られるようになった.

今後の課題としては, 特性多項式の係数の符号が満たす必要十分条件を見つけることでさらに簡単化させることがある. また, 現在の手法を直接適用できない d が大きな問題に対する公式構築が課題として残っている.

表.4: 簡単化結果: 「k」 列は等式制約を満たす実根の数 $(\#(V_{\mathbb{R}}(I)) = \sigma(\chi_1^I))$, 「$\ell$」 列は不等式制約の数, 「$d$」 列は χ_1^I の次数を表す.「'15」 列は χ_1^I が虚根を持たない条件のみを利用した結果 [9] である.「実根」 列は, 定理 1 のみから得られる必要条件をドントケアとして簡単化した結果である. '–' 記号の部分は実験を行っていない.「提案手法」 列は,「実根」 に加えて, 定理 2 から得られる必要条件をドントケアとして簡単化した結果である. それぞれ, 論理和記号 $\vee$ の数を表し, * 記号がついているものは近似解である.「提案手法」における空欄は $\ell = 0$ で因数分解できないため,「実根」 の場合と同じ結果になる.「真」 列と「偽」 列はそれぞれ, 提案手法において真および偽となる符号条件の数を表す.「DC」 列は ドントケアと扱った符号条件の全体からの割合を表す.

k	ℓ	$d = k2^\ell$	'15	実根	提案手法	真	偽	DC
2	0	2	0	0		2	3	0.4444
4	0	4	2	1		16	16	0.6049
2	1	4		3	1	5	11	0.8025
6	0	6	6	3		104	79	0.7490
3	1	6		7	3	41	37	0.8930
8	0	8	20	10		640	400	0.8415
4	1	8		14	6	209	256	0.9291
2	2	8		60	4	29	114	0.9782
10	0	10	*41	21		3858	2083	0.8993
5	1	10		22	8	1434	963	0.9594
12	0	12	*143	71		23024	11072	0.9358
6	1	12		*41	*24	7582	6735	0.9649
3	2	12		*255	14	1180	2267	0.9935
7	1	14		-	*33	49198	27267	0.9840
4	2	16		-	*60	37156	65589	0.9976
2	3	16		-	*7	569	8276	0.9998

参 考 文 献

[1] Espresso. `http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/`.

[2] Hirokazu Anai and Shinji Hara. Fixed-structure robust controller synthesis based on sign definite condition by a special quantifier elimination. In *Proceedings of American Control Conference, 2000*, Vol. 2, pp. 1312–1316, 2000.

[3] Robert King Brayton, Alberto L. Sangiovanni-Vincentelli, Curtis T. McMullen, and Gary D. Hachtel. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, Norwell, MA, USA, 1984.

[4] Christopher W. Brown. *Solution formula construction for truth invariant CAD's*. PhD thesis, University of Delaware Newark, 1999.

[5] Bob F. Caviness and Jeremy R. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition (Texts and Monographs in Symbolic Computation)*. Springer, 1998.

[6] George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition.

In *Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern*, Vol. 33 of *Lecture Notes in Computer Science*, pp. 134–183. Springer-Verlag, May 1975.

[7] James H. Davenport and Joos Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, Vol. 5, No. 1-2, pp. 29–35, 1988.

[8] Ryoya Fukasaku, Hidenao Iwane, and Yosuke Sato. Improving a CGS-QE algorithm. In Ilias S. Kotsireas, Siegfried M. Rump, and Chee K. Yap, editors, *Mathematical Aspects of Computer and Information Sciences - 6th International Conference, MACIS 2015, Berlin, Germany, November 11-13, 2015, Revised Selected Papers*, Vol. 9582 of *Lecture Notes in Computer Science*, pp. 231–235. Springer, 2015.

[9] Ryoya Fukasaku, Hidenao Iwane, and Yosuke Sato. Real quantifier elimination by computation of comprehensive Gröbner systems. In *Proceedings of the 40th International Symposium on Symbolic and Algebraic Computation*, ISSAC '15, pp. 173–180. ACM, July 2015.

[10] Hidenao Iwane, Hiroyuki Higuchi, and Hirokazu Anai. An effective implementation of a special quantifier elimination for a sign definite condition by logical formula simplification. In *Computer Algebra in Scientific Computing*, Vol. 8136, pp. 194–208. Springer, September 2013.

[11] P. Pedersen, Marie-Francoise Roy, and Aviva Szpirglas. Counting real zeros in the multivariate case. In Frédéric Eyssette and André Galligo, editors, *Computational Algebraic Geometry*, Vol. 109 of *Progress in Mathematics*, pp. 203–224. Birkhäuser Boston, 1993.

[12] Alfred Tarski. *A decision method for elementary algebra and geometry.* University of California Press, 1951.

[13] Volker Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, Vol. 5, No. 1-2, pp. 3–27, February 1988.

[14] Volker Weispfenning. Quantifier elimination for real algebra - the quadratic case and beyond. *Applicable Algebra in Engineering, Communication and Computing*, Vol. 8, No. 2, pp. 85–101, January 1997.

[15] Volker Weispfenning. *A New Approach to Quantifier Elimination for Real Algebra*, pp. 376–392. In Caviness and Johnson [5], April 1998.

[16] 穴井宏和, 横山和弘. 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, August 2011.

A QE ツールの出力の比較

本節で，複数のアルゴリズムおよび実装での実行例を示す．一部、出力結果にインデントを揃える修正を加えている．入力は以下を用いた．

$$\exists x(-1 \leq x \leq 3 \wedge k = x^3/3 - ax^2/2 \wedge a > 0)$$

同じ CAD を用いていても、Mathematica, QEPCAD, SyNRAC, REDLOG で出力が大きく異なる．専用アルゴリズムは出力が大きいことが確認できる．

以下は， Mathematica の結果である． Cylindrical Algebraic Decomposition (CAD) が動作した結果と推測される． Resolve コマンドでも同じ結果であった．

```
Mathematica 10.2.0 for Linux x86 (64-bit)
Copyright 1988-2015 Wolfram Research, Inc.

In[1]:= Reduce[Exists[x, -1<=x<=3 && k==x^3/3-a*x^2 && a>0], Reals]

                        -1 - 3 a
Out[1]= (0 < a <= 1 && -------- <= k <= 9 - 9 a) ||
                           3

                         3
                3    -4 a                      3
>     (1 < a <= - && ----- <= k <= 0) || (a > - && 9 - 9 a <= k <= 0)
                2      3                       2
```

以下は，SyNRAC の実行結果である．qe コマンドは Sign Definite Condition 専用の QE [2, 10] が動作した結果で，cad コマンドは CAD による QE の結果である．

```
    |\^/|     Maple 2015 (X86 64 LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2015
 \  MAPLE  /  All rights reserved. Maple is a trademark of
  <____ ____>  Waterloo Maple Inc.
       |       Type ? for help.

> with(SyNRAC):
> qe(Ex([x], And(-1<=x,x<=3,k=x^3/3-a*x^2/2,a>0))); # SDC
Or(
    And(k = 0,-a < 0),
    And(3*a+6*k+2 = 0,3*k < -1),
    And(-a < 0,-3*a-6*k < 2,9*a+2*k <= 18),
    And(-a < 0,-9*a-2*k <= -18,3*a+6*k < -2),
    And(-a < 0,-3*a-6*k < 2,-5*a+6*k <= 6,a^3+6*k = 0),
    And(-a < 0,-a-2*k <= 6,3*a+6*k < -2,a^3+6*k = 0),
    And(-a < 0,3*a+6*k < -2,5*a-6*k <= -6,a^3+6*k = 0),
    And(k <= 0,-a < 0,-3*a-6*k < 2,-5*a+6*k <= 6,-a^3-6*k <= 0),
    And(k <= 0,-a < 0,-a-2*k <= 6,3*a+6*k < -2,-a^3-6*k <= 0),
    And(k <= 0,-a < 0,3*a+6*k < -2,5*a-6*k <= -6,-a^3-6*k <= 0),
    And(-a < 0,-k <= 0,-3*a-6*k < 2,-5*a+6*k <= 6,a^3+6*k <= 0),
    And(-a < 0,-k <= 0,-a-2*k <= 6,3*a+6*k < -2,a^3+6*k <= 0),
    And(-a < 0,-k <= 0,3*a+6*k < -2,5*a-6*k <= -6,a^3+6*k <= 0))
> cad(Ex([x], And(-1<=x,x<=3,k=x^3/3-a*x^2/2,a>0))); # CAD
Or(
    And(-a^3-6*k <= 0,-2*k <= 9,k < 0),
    And(-3*a-6*k <= 2,3*k < -1),
    And(-3*k <= 1,-a < 0,k <= 0),
    And(-9*a-2*k <= -18,k <= 0),
    And(-k <= 0,9*a+2*k <= 18,-a < 0,k <= 9))
```

以下は，QEPCAD の実行結果である． Mathematica と同様 CAD が動作している．必要条件を利用しており，選言標準形になっていないことが特徴の一つである．(式構築のオプションで変更可能)

```
============================================================
                Quantifier Elimination
                        in
           Elementary Algebra and Geometry
                        by
      Partial Cylindrical Algebraic Decomposition

               Version B 1.53, 16 Jul 2009

                        by
                    Hoon Hong
                (hhong@math.ncsu.edu)

With contributions by: Christopher W. Brown, George E.
Collins, Mark J. Encarnacion, Jeremy R. Johnson
Werner Krandick, Richard Liska, Scott McCallum,
Nicolas Robidoux, and Stanly Steinberg
============================================================
Enter an informal description  between '[' and ']':
[Tsukuba 2010 Ri 1]
Enter a variable list:
(a,k,x)
Enter the number of free variables:
2
Enter a prenex formula:
(E x) [ -1 <= x /\ x <= 3 /\ 6 k = 2 x^3 - 3 a x^2 /\ a > 0 ].


============================================================

Before Normalization >
finish

An equivalent quantifier-free formula:

a > 0 /\ [
    [ k <= 0 /\ 2 k + 9 a - 18 >= 0 ] \/
    [ a - 3 < 0 /\ 6 k + a^3 >= 0 /\ k <= 0 ] \/
    [ k > 0 /\ 2 k + 9 a - 18 <= 0 ] \/
    [ k <= 0 /\ 6 k + 3 a + 2 >= 0 ] ]
```

以下は，REDLOG による QE の実行結果である．rlqe は CAD による QE の結果だと推測されるが，rlcad と少し結果が異なるものだった．rlhqe は CGS-QE による結果で，非常に冗長なものになっている．

```
You are running i686-unknown-rh5.9 but the nearest match
I can find was built for i686-unknown-rh5.8. I will try
it, but if there are problems you need to compile
a version for yourself using "./configure; make".
Reduce (Free CSL version), 06-Sep-12 ...

1: load_package redlog$ rlset ofsf$ off nat$

4: rlqe  ex([x], -1 <= x <= 3 and k = x^3/3 - a*x^2/2 and a > 0);   # CAD

a**3 + 6*k > 0 and 9*a + 2*k - 18 > 0 and 3*a + 6*k + 2 > 0 and a - 3 >= 0 and k
 = 0 or a**3 + 6*k > 0 and 9*a + 2*k - 18 > 0 and 3*a + 6*k + 2 >= 0 and a - 3 >
 0 and k < 0 or a**3 + 6*k > 0 and 9*a + 2*k - 18 >= 0 and 3*a + 6*k + 2 < 0 and
 a - 3 > 0 and k < 0 or a**3 + 6*k > 0 and 9*a + 2*k - 18 > 0 and 3*a + 6*k + 2
< 0 and a - 3 = 0 and k < 0 or a**3 + 6*k = 0 and 9*a + 2*k - 18 = 0 and 3*a + 6
*k + 2 < 0 and a - 3 = 0 and k < 0 or a**3 + 6*k > 0 and 9*a + 2*k - 18 > 0 and
3*a + 6*k + 2 > 0 and 3*a - 7 > 0 and a - 3 < 0 and k = 0 or a**3 + 6*k > 0 and
9*a + 2*k - 18 > 0 and 3*a + 6*k + 2 >= 0 and 3*a - 7 > 0 and a - 3 < 0 and k <
0 or a**3 + 6*k > 0 and 9*a + 2*k - 18 >= 0 and 3*a + 6*k + 2 < 0 and 3*a - 7 >
0 and a - 3 < 0 and k < 0 or a**3 + 6*k >= 0 and 9*a + 2*k - 18 < 0 and 3*a + 6*
k + 2 < 0 and 3*a - 7 > 0 and a - 3 < 0 and k < 0 or a**3 + 6*k > 0 and 9*a + 2*
k - 18 > 0 and 3*a + 6*k + 2 > 0 and 3*a - 7 = 0 and k <= 0 or a**3 + 6*k > 0
and 9*a + 2*k - 18 = 0 and 3*a + 6*k + 2 = 0 and 3*a - 7 = 0 and k < 0 or a**3 +
 6*k >= 0 and 9*a + 2*k - 18 < 0 and 3*a + 6*k + 2 < 0 and 3*a - 7 = 0 and k < 0
 or a**3 + 6*k > 0 and 9*a + 2*k - 18 > 0 and 3*a + 6*k + 2 > 0 and 3*a - 7 < 0
and a - 2 > 0 and k = 0 or a**3 + 6*k > 0 and 9*a + 2*k - 18 >= 0 and 3*a + 6*k
+ 2 > 0 and 3*a - 7 < 0 and a - 2 > 0 and k < 0 or a**3 + 6*k > 0 and 9*a + 2*k
- 18 < 0 and 3*a + 6*k + 2 >= 0 and 3*a - 7 < 0 and a - 2 > 0 and k < 0 or a**3
+ 6*k >= 0 and 9*a + 2*k - 18 < 0 and 3*a + 6*k + 2 < 0 and 3*a - 7 < 0 and a -
2 > 0 and k < 0 or a**3 + 6*k > 0 and 9*a + 2*k - 18 = 0 and 3*a + 6*k + 2 > 0
and a - 2 = 0 and k = 0 or a**3 + 6*k > 0 and 9*a + 2*k - 18 < 0 and 3*a + 6*k +
 2 > 0 and a - 2 = 0 and k < 0 or a**3 + 6*k = 0 and 9*a + 2*k - 18 < 0 and 3*a
+ 6*k + 2 = 0 and a - 2 = 0 and k < 0 or a**3 + 6*k > 0 and 9*a + 2*k - 18 = 0
and 3*a + 6*k + 2 > 0 and a - 2 < 0 and a > 0 and k > 0 or a**3 + 6*k > 0 and 9*
a + 2*k - 18 < 0 and 3*a + 6*k + 2 > 0 and a - 2 < 0 and a > 0 and k >= 0 or a**
3 + 6*k >= 0 and 9*a + 2*k - 18 < 0 and 3*a + 6*k + 2 > 0 and a - 2 < 0 and a >
0 and k < 0 or a**3 + 6*k < 0 and 9*a + 2*k - 18 < 0 and 3*a + 6*k + 2 >= 0 and
a - 2 < 0 and a > 0 and k < 0$
```

```
5: rlhqe  ex([x], -1 <= x <= 3 and k = x^3/3 - a*x^2/2 and a > 0);   # CGS-QE

3*a + 6*k + 2 = 0 and a > 0 or 9*a + 2*k - 18 = 0 and a > 0 or a > 0 and not(((
486*a**7 - 513*a**6*k - 1620*a**6 - 864*a**5*k**2 + 1683*a**5*k + 864*a**5 - 108
*a**4*k**3 + 2304*a**4*k**2 + 3498*a**4*k + 576*a**4 + 36*a**3*k**3 - 2664*a**3*
k**2 - 13280*a**3*k + 288*a**3 - 4536*a**2*k**3 + 6096*a**2*k**2 + 4112*a**2*k +
 576*a**2 - 576*a*k**4 + 13248*a*k**3 + 4336*a*k**2 + 5328*a*k + 288*k**4 - 7200
*k**3 - 19904*k**2 + 2016*k = 0 or a**3 + 6*k = 0 or 9*a + 2*k - 18 = 0 or 3*a +
 6*k + 2 = 0 or k = 0) and (39366*a**13 - 39366*a**12*k - 183708*a**12 - 78732*a
**11*k**2 + 172773*a**11*k + 209952*a**11 + 358668*a**10*k**2 + 400950*a**10*k +
 46656*a**10 - 8748*a**9*k**3 - 816480*a**9*k**2 - 2534976*a**9*k - 38880*a**9 -
 868968*a**8*k**3 + 1574640*a**8*k**2 + 2501280*a**8*k - 25920*a**8 + 3872448*a
**7*k**3 + 1406160*a**7*k**2 + 888624*a**7*k - 129600*a**7 - 116640*a**6*k**4 -
5692032*a**6*k**3 - 11076480*a**6*k**2 - 319968*a**6*k + 4608*a**6 - 2540160*a**
5*k**4 + 4325184*a**5*k**3 + 9095040*a**5*k**2 - 920256*a**5*k - 109056*a**5 +
11155968*a**4*k**4 + 2488320*a**4*k**3 + 3284736*a**4*k**2 - 676224*a**4*k -
64512*a**4 - 393984*a**3*k**5 - 13400064*a**3*k**4 - 15950592*a**3*k**3 + 916736
*a**3*k**2 + 570368*a**3*k - 33792*a**3 - 870912*a**2*k**5 + 3621888*a**2*k**4 +
 9695232*a**2*k**3 - 4042752*a**2*k**2 - 542720*a**2*k - 36864*a**2 + 4036608*a*
k**5 + 3280896*a*k**4 - 476160*a*k**3 - 45056*a*k**2 - 377856*a*k - 165888*k**6
- 2488320*k**5 - 3852288*k**4 + 7815168*k**3 + 2666496*k**2 - 92160*k = 0 and
19683*a**13 - 174960*a**12*k - 91854*a**12 - 364500*a**11*k**2 + 803358*a**11*k
+ 104976*a**11 - 34992*a**10*k**3 + 1761264*a**10*k**2 - 481140*a**10*k + 23328*
a**10 + 46656*a**9*k**4 + 27216*a**9*k**3 - 4529520*a**9*k**2 - 1995840*a**9*k -
 23328*a**9 - 209952*a**8*k**4 - 4280040*a**8*k**3 + 9637488*a**8*k**2 + 1881792
*a**8*k + 5184*a**8 + 7776*a**7*k**5 - 292896*a**7*k**4 + 20667744*a**7*k**3 -
6330528*a**7*k**2 + 944064*a**7*k - 76032*a**7 + 510624*a**6*k**5 + 737856*a**6*
k**4 - 35097408*a**6*k**3 - 13859712*a**6*k**2 - 18816*a**6*k - 41472*a**6 -
2398464*a**5*k**5 - 13878144*a**5*k**4 + 34029888*a**5*k**3 + 11286400*a**5*k**2
 - 285696*a**5*k - 20736*a**5 + 103680*a**4*k**6 + 929664*a**4*k**5 + 65926656*a
**4*k**4 - 14887680*a**4*k**3 + 7301376*a**4*k**2 - 344064*a**4*k - 13824*a**4 +
 1313280*a**3*k**6 + 2822400*a**3*k**5 - 90716160*a**3*k**4 - 32504320*a**3*k**3
 + 2468864*a**3*k**2 - 59904*a**3*k - 6912000*a**2*k**6 - 8580096*a**2*k**5 +
31365120*a**2*k**4 + 22428160*a**2*k**3 - 2156544*a**2*k**2 - 82944*a**2*k +
331776*a*k**7 + 8128512*a*k**6 + 35536896*a*k**5 + 10027008*a*k**4 + 13763584*a*
k**3 + 626688*a*k**2 - 497664*k**7 - 1824768*k**6 - 34394112*k**5 - 12072960*k**
4 + 10702848*k**3 + 1548288*k**2 = 0 and (59049*a**13 - 275562*a**12 + 314928*a
**11 + 682344*a**10*k + 69984*a**10 - 2939328*a**9*k - 46656*a**9 + 3079296*a**8
*k - 93312*a**8 + 2426112*a**7*k**2 + 622080*a**7*k - 238464*a**7 - 9144576*a**6
*k**2 - 248832*a**6*k + 449280*a**6 + 8335872*a**5*k**2 - 1126656*a**5*k -
470016*a**5 + 2612736*a**4*k**3 + 967680*a**4*k**2 - 1539072*a**4*k - 301056*a**
(以下略. 全部で 3212 行)
```

以下は RegularChains パッケージの結果である.

```
> libname := "PowerSeries.mla", "RegularChains.mla", libname:
> with(RegularChains[SemiAlgebraicSetTools]):
> QuantifierElimination(`&E`([x]), `&and`(-1<=x,x<=3, k=x^3/3-a*x^2/2, a>0));
`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(
`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(
`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(`&or`(
  `&and`(a-2 = 0,k = 0),            `&and`(a-2 = 0,a^3+6*k = 0)),
  `&and`(`&and`(a-2 = 0,k < 0),0 < a^3+6*k)),
  `&and`(a-3 = 0,k = 0)),           `&and`(a-3 = 0,a^3+6*k = 0)),
  `&and`(a-3 = 0,6*k+3*a+2 = 0)),
  `&and`(`&and`(a-3 = 0,k < 0),0 < 6*k+3*a+2)),
  `&and`(`&and`(a-3 = 0,0 < a^3+6*k),6*k+3*a < -2)),
  `&and`(0 < a-3,k = 0)),           `&and`(0 < a-3,6*k+3*a+2 = 0)),
  `&and`(0 < a-3,2*k+9*a-18 = 0)),
  `&and`(`&and`(0 < a-3,k < 0),0 < 6*k+3*a+2)),
  `&and`(`&and`(0 < a-3,6*k+3*a < -2),0 < 2*k+9*a-18)),
  `&and`(3*a-7 = 0,k = 0)),         `&and`(3*a-7 = 0,a^3+6*k = 0)),
  `&and`(3*a-7 = 0,6*k+3*a+2 = 0)),
  `&and`(`&and`(3*a-7 = 0,k < 0),0 < 6*k+3*a+2)),
  `&and`(`&and`(3*a-7 = 0,0 < a^3+6*k),6*k+3*a < -2)),
  `&and`(`&and`(a < 2,0 < a),k = 0)), `&and`(`&and`(a < 2,0 < a),a^3+6*k = 0)),
  `&and`(`&and`(a < 2,0 < a),6*k+3*a+2 = 0)),
  `&and`(`&and`(a < 2,0 < a),2*k+9*a-18 = 0)),
  `&and`(`&and`(`&and`(a < 2,0 < a),k < 0),0 < a^3+6*k)),
  `&and`(`&and`(`&and`(a < 2,0 < a),0 < k),2*k+9*a < 18)),
  `&and`(`&and`(`&and`(a < 2,0 < a),a^3+6*k < 0),0 < 6*k+3*a+2)),
  `&and`(`&and`(0 < a-2,3*a < 7),k = 0)),
  `&and`(`&and`(0 < a-2,3*a < 7),a^3+6*k = 0)),
  `&and`(`&and`(0 < a-2,3*a < 7),6*k+3*a+2 = 0)),
  `&and`(`&and`(0 < a-2,3*a < 7),2*k+9*a-18 = 0)),
  `&and`(`&and`(`&and`(0 < a-2,3*a < 7),k < 0),0 < 2*k+9*a-18)),
  `&and`(`&and`(`&and`(0 < a-2,3*a < 7),0 < a^3+6*k),6*k+3*a < -2)),
  `&and`(`&and`(`&and`(0 < a-2,3*a < 7),0 < 6*k+3*a+2),2*k+9*a < 18)),
  `&and`(`&and`(a < 3,0 < 3*a-7),k = 0)),
  `&and`(`&and`(a < 3,0 < 3*a-7),a^3+6*k = 0)),
  `&and`(`&and`(a < 3,0 < 3*a-7),6*k+3*a+2 = 0)),`&and`(`&and`(a < 3,0 < 3*a-7),
    2*k+9*a-18 = 0)),
  `&and`(`&and`(`&and`(a < 3,0 < 3*a-7),k < 0),0 < 6*k+3*a+2)),
  `&and`(`&and`(`&and`(a < 3,0 < 3*a-7),0 < a^3+ 6*k),2*k+9*a < 18)),
  `&and`(`&and`(`&and`(a < 3,0 < 3*a-7),6*k+3*a < -2),0 < 2*k+9*a-18))
```

以下は，本稿の簡単化結果を利用した CGS-QE による実行結果である．SyNRAC の SDC よりは冗長であるが，REDLOG の rlqe, rlhqe よりも簡単な結果が得られている．RegularChains に比べると論理和の数が少ないが，原子論理式のサイズが大きくなっている．(これは本稿の簡単化の影響ではなく，特性多項式の係数のサイズが大きいことが理由)

```
Or(
  And(3*a-7 = 0,2*k+3 = 0),
  And(9*a+2*k-18 = 0,k < 9,2*k+3 <> 0),
  And(9*a+2*k-18 = 0,k < 9,k^6-48*k^5+441*k^4+1674*k^3-4536*k^2+3402*k < 4374),
  And(3*a+6*k+2 = 0,k < -1,81*k^4+108*k^3-18*k^2+6*k < 2),
  And(3*a+6*k+2 = 0,2*k+3 <> 0,3*k < -1),
  And(9*a+2*k-18 = 0,-k < 0,k < 9,
    40*k^5-1200*k^4-8334*k^3-10800*k^2-7533*k <= 8748),
  And(3*a+6*k+2 = 0,-k < 1,3*k < -1,-81*k^4-108*k^3+18*k^2-6*k < -2),
  And(9*a+2*k-18 = 0,k < 0,2*k+3 <> 0,2*k+9 <> 0,-40*k^5+1200*k^4+8334*k^3+10800*
    k^2+7533*k <= -8748),
  And(-a < 0,3*a+6*k+2 <> 0,9*a+2*k-18 <> 0,27*a^5*k+60*a^4*k^2+12*a^3*k^3-36*a^4
    *k-104*a^3*k^2-36*a^3*k+162*a^2*k^2+360*a*k^3+72*k^4-216*a*k^2-624*k^3-216*
    k^2 <= 0,243*a^6-324*a^5-216*a^4+1296*a^3*k-144*a^3-1440*a^2*k-96*a^2-768*a*
    k+576*k^2-64*a-384*k < 192),
  And(-a < 0,3*a+6*k+2 <> 0,9*a+2*k-18 <> 0,27*a^5*k+60*a^4*k^2+12*a^3*k^3-36*
    a^4*k-104*a^3*k^2-36*a^3*k+162*a^2*k^2+360*a*k^3+72*k^4-216*a*k^2-624*k^3-
    216*k^2 <= 0,-81*a^6+108*a^5*k+108*a^4*k^2+108*a^5-108*a^4*k-72*a^3*k^2+72*
    a^4-792*a^3*k+384*a^2*k^2+576*a*k^3+48*a^3+608*a^2*k-480*a*k^2-576*k^3+48*
    a^2+544*a*k-1632*k^2+192*k < 0),
  And(k < 0,- a < 0,3*a+6*k+2 <> 0,9*a+2*k-18 <> 0,-a^3-6*k < 0,-27*a^5*k-60*a^4*
    k^2-12*a^3* k^3+36*a^4*k+104*a^3*k^2+36*a^3*k-162*a^2*k^2-360*a*k^3-72*k^4+
    216*a*k^2+624*k^3+216*k^2 < 0),
  And(-a < 0,-k < 0,3*a+6*k+2 <> 0,9*a+2*k-18 <> 0,a^3+6*k < 0,- 27*a^5*k-60*a^4*
    k^2-12*a^3*k^3+36*a^4*k+104*a^3*k^2+36*a^3*k-162*a^2*k^2-360*a*k^3-72*k^4+
    216*a*k^2+624*k^3+216*k^2 < 0))
```