数理解析研究所講究録
第2054巻 2017年 77-95

論理式簡単化アルゴリズム White-Box の拡張と実装

國廣 尭之 *
Takayuki Kunihiro
筑波大学大学院数理物質科学研究科
Graduate School of Pure and Applied Sciences, University of Tsukuba

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

和田 優未 ‡
Yumi Wada
筑波大学大学院数理物質科学研究科
Graduate School of Pure and Applied Sciences, University of Tsukuba

照井 章 §
Akira Terui
筑波大学数理物質系
Faculty of Pure and Applied Sciences, University of Tsukuba

1 はじめに

本稿では, 冗長な論理式の簡単化による実閉体上の限量子消去 (Quantifier Elimination; QE) ([1], [5]) 計算の効率化について論ずる.

QE の問題点として計算量の大きさが挙げられる. QE の計算量は変数の個数に対して二重指数的であることが知られており, 効率的な QE 計算のための研究が数多く行われている. 仮想置換法 (Virtual Substituition) [10] に代表される, 特殊な形の一階述語論理式に対して有効な QE アルゴリズムの研究もその一例である. その一方で, QE アルゴリズムに入力する論理式をあらかじめ簡単化することでより効率的な QE 計算を実現しようとする研究も存在する. QE の計算量は, 入力する一階述語論理式に含まれる多項式や変数の個数に依存することから, 計算過程で扱う論理式と等価で, かつ, 多項式や変数の個数がより小さな論理式に変換してから計算を行うことで, QE 計算の効率化を図ろうという考え方である.

しかしながら, 論理式の簡単化に関する研究は多くない ([3], [12]). 本研究では C. W. Brown, A. Strzeboński により提案されている論理式簡単化アルゴリズム White-Box [4] の拡張と実装を行う.

以下, 「(論理式) 簡単化」とは, 「論理式に含まれる多項式の個数を減らすこと」を意味し, 簡単化した論理式はも

* hirophirop@gmail.com
† iwane@jp.fujitsu.com
‡ wada.yumi.ww@alumni.tsukuba.ac.jp
§ terui@math.tsukuba.ac.jp

との論理式と等価であるものとする. また, 「White-Box」とは, 「論理式簡単化アルゴリズム White-Box」を意味するものとする.

White-Box は「原子論理式の論理積」のみに適用可能としている. そこで, 我々はまず, White-Box を「任意の論理式」へ適用可能となるように拡張した. 次に, 拡張したアルゴリズムを数式処理システム Maple [11] に実装し, 簡単化の効果を確かめる 2 種類の実験を行った. 1 つ目は, QE 計算の前処理として簡単化を行うことにより, QE 計算をどれだけ効率化できたかを確かめる実験, もう 1 つは Redlog [7], QEPCAD [9], SyNRAC [8] の 3 つの QE ツールに実装されている簡単化関数に対し簡単化の効果を比較する実験である.

以下, 第 2 章では Brown らによるオリジナルの White-Box の簡単化の考え方を紹介する. 第 3 章では我々による White-Box の拡張を示し, 第 4 章では拡張した White-Box の簡単化実験の結果と考察を示す.

2 White-Box

本章では, Brown and Strzeboński [4] により提案されている論理式簡単化アルゴリズム **White-Box** について記す. なお, 本章では Brown らによる「オリジナルの」White-Box について記し, 筆者による拡張については次章で述べる.

2.1 準備

White-Box の詳細について記す前に, 記法, 語を定義する.

定義 1 (原子論理式) 整数係数多項式 p に対し, 論理式 $\varphi(p)$ が **原子論理式** であるとは,

$$\varphi(p) \in \{p = 0, p \neq 0, p < 0, p \leq 0, p > 0, p \geq 0, \text{true}, \text{false}\}$$

であることをいう. ここで, true, false とはそれぞれ, $0 = 0, 0 \neq 0$ を表す. □

定義 1 における true, false は, それぞれ「常に真」, 「常に偽」であることを表す論理式である.

定義 2 (集合 OP) 集合 OP を以下の 8 つの写像の集合とする. なお各写像は $\mathbb{R}$ を定義域, { true, false } を値域とする.

1. $NOOP(x) :=$ false.
2. $LTOP(x) :=$ true if $x < 0$, else false.
3. $LEOP(x) :=$ true if $x \leq 0$, else false.
4. $GTOP(x) :=$ true if $x > 0$, else false.
5. $GEOP(x) :=$ true if $x \geq 0$, else false.
6. $EQOP(x) :=$ true if $x = 0$, else false.
7. $NEOP(x) :=$ true if $x \neq 0$, else false.
8. $ALOP(x) :=$ true. □

例えば, 原子論理式 $x^2 + 1 > 0$ は $GTOP(x^2 + 1)$ と表現する.

多項式集合 $Q \subset \mathbb{Z}[x_1, \ldots, x_n]$ と写像 $\alpha : Q \to OP$ により, 以下を満たす任意の点 x の半代数的集合が定義できる.

$$\bigwedge_{q \in Q} \alpha(q)(q(x)).$$

この表現で原子論理式の論理積の表現が可能になる. 例えば,

$$Q = \{x + y, x - y\}, \quad \alpha(q) = \begin{cases} LTOP & (q = x + y) \\ GEOP & (q = x - y) \end{cases}$$

により, $x + y < 0 \wedge x - y \geq 0$ が表現できる.

次に, OP の元の演算を以下の通り定義する.

定義 3 (OP の元の演算) OP の二項演算 $+, \cdot, \wedge : OP \times OP \to OP$ をそれぞれ以下のように定義する.

- $(\alpha + \beta)(z) :=$ true if $\exists x, y\,[z = x + y \wedge \alpha(x) \wedge \beta(y)]$, else false.
- $(\alpha\beta)(z) :=$ true if $\exists x, y\,[z = xy \wedge \alpha(x) \wedge \beta(y)]$, else false.
- $(\alpha \wedge \beta)(z) :=$ true if $[\alpha(z) \wedge \beta(z)]$, else false.

また, 単項演算 $SQ : OP \to OP$ を以下のように定義する.

- $SQ(\alpha)(z) :=$ true if $\exists x[z = x^2 \wedge \alpha(x)]$, else false. □

定義 4 (実数の符号と OP) sgn: $\mathbb{R} \to OP$ を以下のように定義する.

$$sgn(x) := \begin{cases} LTOP & (x < 0) \\ EQOP & (x = 0) \\ GTOP & (x > 0) \end{cases}$$

□

定義 5 ($PP(x_1, \ldots, x_n)$) $\mathbb{Z}[x_1, \ldots, x_n]$ 上の定数でない原始多項式[*2]で, 辞書式順序[*3]による先頭項係数が正である多項式からなる集合を $PP(x_1, \ldots x_n)$ と表す. □

定義 6 (強さ) $a \in OP, b \in OP$ に対し, $\forall x[a(x) \Rightarrow b(x)]$ であるとき, a は b より強いという. □

2.2 多項式の符号推論アルゴリズム

まず White-Box がどのように論理式を簡単化するか例を用いて紹介する.

例 1 論理式 $y^2 - x + 1 > 0 \wedge 2x - 1 < 0$ を簡単化する.

$$\begin{aligned} & y^2 - x + 1 > 0 \wedge 2x - 1 < 0 \\ \equiv\ & y^2 + \frac{1}{2} - \frac{2x-1}{2} > 0 \wedge 2x - 1 < 0 \\ \equiv\ & 2x - 1 < 0 \end{aligned}$$

$\phi_1 := y^2 - x + 1 > 0$, $\phi_2 := 2x - 1 < 0$ とする. 上では, ϕ_1 を 2 行目のように変形すると, 第 3 項 $-\frac{2x-1}{2}$ は ϕ_2 の仮定の下で常に正と推論できる. 第 1 項と第 2 項の和 $y^2 + \frac{1}{2}$ は正であるので, ϕ_2 を仮定すれば ϕ_1 が常に成り立ち, $\phi_1 \wedge \phi_2 \equiv \phi_2$ と簡単化できる. □

White-Box による簡単化の対象は「原子論理式の論理積」であり, その簡単化の核となっているのは, 1 つの原子論理式を仮定した状況で他の多項式の符号を推論することにある. 上の例では $2x - 1 < 0$ を仮定して $y^2 - x + 1$ の符号を正だと推論した.

このことを踏まえて本節では White-Box を構成するための部分的なアルゴリズムを順に紹介していく.

White-Box における多項式の符号推論は大きく分けて 2 つある.

第一の推論は, 各変数の符号の情報を仮定して多項式の符号を推論するアルゴリズム, PolynomialSign であり, Algorithm 2 に示す. PolynomialSign では, Algorithm 1 に示す単項式の符号を推論するアルゴリズム, MonomialSign を用いる.

第二の推論は, 与えられた 2 つの多項式 p, q に対し, q の符号を仮定して, p の符号を推論するアルゴリズム, DeduceSign であり, Algorithm 4 に示す. DeduceSign では, Algorithm 3 アルゴリズム, FindInterval を用いる.

[*2] 整数係数多項式で, 係数の最大公約数が 1 であるものを原始多項式と呼ぶ.

[*3] 辞書式順序の詳細は Cox, Little and O'Shea [6, 第 2 章, 定義 3] 等を参照. 以下, $x_1, \ldots, x_n$ 及び $a, b, \ldots, y, z$ の辞書式順序はそれぞれ

$$x_1 > x_2 > \cdots > x_n,\ a > b > \cdots > y > z$$

とする.

Algorithm 1 MonomialSign [4, Algorithm 2]

Input: $M = x_1^{e_1} \ldots x_n^{e_n}$: 変数のベキ積, $\alpha_1, \ldots, \alpha_n \in OP$

Output: $\beta \in OP$ s.t. $\alpha_1(x_1) \wedge \cdots \wedge \alpha_n(x_n) \Rightarrow \beta(M)$

1: $\beta \leftarrow GTOP$
2: **for** $1 \leq i \leq n$ **do**
3: **if** e_i: even **then** $\beta \leftarrow SQ(\alpha_i) \cdot \beta$ **else** $\beta \leftarrow \alpha_i \cdot \beta$ **end if**
4: **end for**
5: **return** β

Algorithm 2 PolynomialSign [4, Algorithm3]

Input: $p = a_1 M_1 + \cdots + a_k M_k \in \mathbb{Z}[x_1, \ldots, x_n]$ (M_i は変数 $x_1, \ldots, x_n$ 上のベキ積), $\alpha_1, \ldots, \alpha_n \in OP$

Output: $\beta \in OP$ s.t. $\alpha_1(x_1) \wedge \cdots \wedge \alpha_n(x_n) \Rightarrow \beta(p)$

1: $\beta \leftarrow EQOP$
2: **for** $1 \leq i \leq k$ **do**
3: $\beta \leftarrow sgn(a_i)$ MonomialSign$(M_i; \alpha_1, \ldots, \alpha_n) + \beta$
4: **end for**
5: **return** β

例 2 例 1 において, $p = y^2 - x + 1$, $q = 2x - 1$, $\alpha(x) = ALOP$, $\alpha(y) = ALOP$, $\beta = LTOP$ とし DeduceSign を適用する. すなわち, $2x - 1 < 0$ を仮定して $y^2 - x + 1$ の符号を推論する.

1. FindIntervals より $I_1 = \left\{\frac{1}{2}\right\}$, $I_2 = \emptyset$, *strict* = true が計算できる. (すなわち $p + \frac{1}{2}q > 0$ であることがわかる.)
2. $\beta = LTOP$ と $I_1 \cap \mathbb{R}_+ \neq \emptyset$ から, DeduceSign の 3, 4 行目が実行され, $\gamma = GTOP$ が計算できる.

以上より $2x - 1 < 0$ を仮定したとき, $y^2 - x + 1 > 0$ が推論された. □

1 個の多項式の符号を仮定して, 1 個の多項式の符号を推論する DeduceSign を, 仮定する多項式の符号を変えながら繰り返し使うことで, より多くの情報から 1 個の多項式の符号を推論するアルゴリズムが DeduceAll であり, Algorithm 5 に示す.

DeduceAll を用いて, 入力した多項式の符号と変数の符号を仮定して, 各変数の符号, 各多項式の符号を順に推論するアルゴリズムが DeduceAllSigns であり, Algorithm 6 に示す.

2.3 White-Box

本節ではこれまで紹介した部分アルゴリズムを用いて, Brown らが提案する White-Box を Algorithm 7 に示す. なお, White-Box の入出力である原始多項式の有限集合 $P \subseteq PP(x_1, \ldots, x_n)$ と, 写像 σ: $P \to OP$ は「原子論理式の論理積」 $\bigwedge_{p \in P} \sigma_p(p)$ を表す.

White-Box は大きく分けて 5 つの手順に分かれる.

1. 各多項式集合, 符号情報の初期化 (1–10 行)
2. 多項式の符号推論, 及び符号情報の更新. (12–15 行)
3. 手順 2 で新たな情報が推論された場合, 再度手順 2 を実行 (11, 16 行)
4. 多項式の符号情報のうち, 他の情報 (詳細は下記を参照) から推論出来るものを取り除く (17–31 行)
5. 多項式集合と符号情報から論理式を構成 (32–36 行)

上記の手順 4 は多項式の符号推論を用いて論理式の簡単化を行う本アルゴリズムの特に重要な手順である. 「他の情報から推論する」方法には大きく分けて以下の 3 種類がある.

Algorithm 3 FindIntervals [4, Algorithm 4]

Input: $p = a_1M_1 + \cdots + a_kM_k$, $q = b_1M_1 + \cdots + b_kM_k \in \mathbb{Z}[x_1, \ldots, x_n]$, $\alpha_1, \ldots, \alpha_n \in OP$

Output: $I_1, I_2 : \mathbb{R}$ 上の閉区間, *strict* $\in$ { true , false } s.t.

$$\alpha_1(x_1) \wedge \cdots \wedge \alpha_n(x_n) \Rightarrow \forall t \in I_1[p + tq \geq 0] \wedge \forall t \in I_2[p + tq \leq 0],$$

かつ, *strict* = true ならば

$$\alpha_1(x_1) \wedge \cdots \wedge \alpha_n(x_n) \Rightarrow \forall t \in \mathrm{int}(I_1)[p + tq > 0] \wedge \forall t \in \mathrm{int}(I_2)[p + tq < 0]$$

も成り立つ. 但し, int(I_1), int(I_2) はそれぞれ I_1, I_2 の内部である.

1: $I_1 \leftarrow \mathbb{R}$, $I_2 \leftarrow \mathbb{R}$, *strict* $\leftarrow$ false
2: **for** $1 \leq i \leq k$ **do**
3: $s \leftarrow$ MonomialSign($M_i; \alpha_1, \ldots, \alpha_n$)
4: **if** $s = NOOP$ **then** $I_1 \leftarrow \emptyset$ **end if**
5: **if** $s = LEOP \vee s = LTOP$ **then**
6: **if** $s = LTOP$ **then** *strict* $\leftarrow$ true **end if**
7: **if** $b_i = 0 \wedge a_i < 0$ **then** $I_2 \leftarrow \emptyset$
8: **else if** $b_i = 0 \wedge a_i > 0$ **then** $I_1 \leftarrow \emptyset$
9: **else if** $b_i < 0$ **then** $I_1 \leftarrow I_1 \cap [-\frac{a_i}{b_i}, \infty)$, $I_2 \leftarrow I_2 \cap (-\infty, -\frac{a_i}{b_i}]$
10: **else if** $b_i > 0$ **then** $I_1 \leftarrow I_1 \cap (-\infty, -\frac{a_i}{b_i}]$, $I_2 \leftarrow I_2 \cap [-\frac{a_i}{b_i}, \infty)$
11: **end if**
12: **else if** $s = GEOP \vee s = GTOP$ **then**
13: **if** $s = GTOP$ **then** *strict* $\leftarrow$ true **end if**
14: **if** $b_i = 0 \wedge a_i < 0$ **then** $I_1 \leftarrow \emptyset$
15: **else if** $b_i = 0 \wedge a_i > 0$ **then** $I_2 \leftarrow \emptyset$
16: **else if** $b_i < 0$ **then** $I_1 \leftarrow I_1 \cap (-\infty, -\frac{a_i}{b_i}]$, $I_2 \leftarrow I_2 \cap [-\frac{a_i}{b_i}, \infty)$
17: **else if** $b_i > 0$ **then** $I_1 \leftarrow I_1 \cap [-\frac{a_i}{b_i}, \infty)$, $I_2 \leftarrow I_2 \cap (-\infty, -\frac{a_i}{b_i}]$
18: **end if**
19: **else if** $s = NEOP \vee s = ALOP$ **then**
20: **if** $b_i = 0 \wedge a_i \neq 0$ **then** $I_1 \leftarrow \emptyset$, $I_2 \leftarrow \emptyset$, **else** $I_1 \leftarrow I_1 \cap \{-\frac{a_i}{b_i}\}$, $I_2 \leftarrow I_2 \cap \{-\frac{a_i}{b_i}\}$ **end if**
21: **end if**
22: **end for**
23: **return** I_1 , I_2 , *strict*

1. p が可約多項式 q の因子かつ, $q \neq 0$ ならば $p \neq 0$ は推論できる.
2. p の符号が PolynomialSign により推論できる.
3. p の符号が他の原子論理式の集合を仮定して DeduceAll により推論できる.

ある多項式の符号が, 上のいずれかにより推論出来た場合, この多項式を取り除くことで論理式を簡単化する. 以上がオリジナルの White-Box アルゴリズムである.

3 White-Box アルゴリズムの拡張

White-Box は「原子論理式の論理積」のみに適用可能であった. 以下では, White-Box を任意の論理式に適用できるよう拡張するために構成した 3 つのアルゴリズムを示す. また, 以下に提案するアルゴリズムと区別するため, オリ

Algorithm 4 DeduceSign [4, Algorithm 5]

Input: $p = a_1M_1 + \cdots + a_kM_k$, $q = b_1M_1 + \cdots + b_kM_k \in \mathbb{Z}[x_1, \ldots, x_n]$, $\alpha_1, \ldots, \alpha_n, \beta \in OP$

Output: $\gamma \in OP$ s.t. $\alpha_1(x_1) \wedge \cdots \wedge \alpha_n(x_n) \wedge \beta(q) \Rightarrow \gamma(p)$

1: $\gamma \leftarrow ALOP$, $(I_1, I_2, strict) \leftarrow$ FindIntervals$(p; q; \alpha_1, \ldots, \alpha_n)$
2: **if** $\beta = LTOP$ **then**
3: **if** $I_1 \cap \mathbb{R}_+ \neq \emptyset$ **then** $\gamma \leftarrow GTOP$ **else if** $0 \in I_1$ **then** $\gamma \leftarrow GEOP$ **end if**
4: **if** $I_2 \cap \mathbb{R}_- \neq \emptyset$ **then** $\gamma \leftarrow \gamma \wedge LTOP$ **else if** $0 \in I_2$ **then** $\gamma \leftarrow \gamma \wedge LEOP$ **end if**
5: **else if** $\beta = LEOP$ **then**
6: **if** *strict* and $int(I_1) \cap \mathbb{R}_+ \neq \emptyset$ **then** $\gamma \leftarrow GTOP$ **else if** $I_1 \cap (\mathbb{R}_+ \cup \{0\}) \neq \emptyset$ **then** $\gamma \leftarrow GEOP$ **end if**
7: **if** *strict* and $int(I_2) \cap \mathbb{R}_- \neq \emptyset$ **then** $\gamma \leftarrow \gamma \wedge LTOP$ **else if** $I_2 \cap (\mathbb{R}_- \cup \{0\}) \neq \emptyset$ **then** $\gamma \leftarrow \gamma \wedge LEOP$ **end if**
8: **else if** $\beta = GTOP$ **then**
9: **if** $I_1 \cap \mathbb{R}_- \neq \emptyset$ **then** $\gamma \leftarrow GTOP$ **else if** $0 \in I_1$ **then** $\gamma \leftarrow GEOP$ **end if**
10: **if** $I_2 \cap \mathbb{R}_+ \neq \emptyset$ **then** $\gamma \leftarrow \gamma \wedge LTOP$ **else if** $0 \in I_2$ **then** $\gamma \leftarrow \gamma \wedge LEOP$ **end if**
11: **else if** $\beta = GEOP$ **then**
12: **if** *strict* and $int(I_1) \cap \mathbb{R}_- \neq \emptyset$ **then** $\gamma \leftarrow GTOP$ **else if** $I_1 \cap (\mathbb{R}_- \cup \{0\}) \neq \emptyset$ **then** $\gamma \leftarrow GEOP$ **end if**
13: **if** *strict* and $int(I_2) \cap \mathbb{R}_+ \neq \emptyset$ **then** $\gamma \leftarrow \gamma \wedge LTOP$ **else if** $I_2 \cap (\mathbb{R}_+ \cup \{0\}) \neq \emptyset$ **then** $\gamma \leftarrow \gamma \wedge LEOP$ **end if**
14: **else if** $\beta = EQOP$ **then**
15: **if** *strict* and $int(I_1) \neq \emptyset$ **then** $\gamma \leftarrow GTOP$ **else if** $I_1 \neq \emptyset$ **then** $\gamma \leftarrow GEOP$ **end if**
16: **if** *strict* and $int(I_2) \neq \emptyset$ **then** $\gamma \leftarrow \gamma \wedge LTOP$ **else if** $I_2 \neq \emptyset$ **then** $\gamma \leftarrow \gamma \wedge LEOP$ **end if**
17: **end if**
18: **return** γ

Algorithm 5 DeduceAll [4, Algorithm 6]

Input: $p = a_1M_1 + \cdots + a_kM_k$, $Q \subseteq \mathbb{Z}[x_1, \ldots, x_n]$, $\alpha_1, \ldots, \alpha_n \in OP, \beta : Q \ni q \mapsto \beta_q \in OP$

Output: $\gamma \in OP$ s.t. $\alpha_1(x_1) \wedge \cdots \wedge \alpha_n(x_n) \wedge \bigwedge_{q \in Q} \beta_q(q) \Rightarrow \gamma(p)$

1: $\gamma \leftarrow ALOP$
2: **for all** $q \in Q$ **do**
3: **if** $\beta_q \neq NEOP \vee \beta_q \neq ALOP$ **then**
4: $\gamma \leftarrow \gamma \wedge$ DeduceSign$(p; q; \alpha_1, \ldots, \alpha_n; \beta_q)$
5: **if** $\gamma = NOOP$ **then return** $NOOP$ **end if**
6: **end if**
7: **end for**
8: **return** γ

ジナルの White-Box (Algorithm 7) を以下では AndSimplify と呼ぶ.

以降, 論理積は $\bigwedge_{\psi \in \Psi} \psi$ (Ψ: 原子論理式と論理和の集合) の形のみを考え, 論理和は $\bigvee_{\phi \in \Phi} \phi$ (Φ: 原子論理式と論理積の集合) の形のみを考える. これらの形を満たさない論理式に対しては, いずれかの形に当てはまるように式変形を施すものとする.

3.1 原子論理式の論理和への拡張

「原子論理式の論理和」に対して AndSimplify の考え方を適用可能にした拡張を考える. そのためには論理和を「論理積の否定」と考えて適用すればよい. ある原子論理式 (の論理積) の否定は, 別の原子論理式 (の論理和) と同値にな

Algorithm 6 DeduceAllSigns [4, Algorithm 7]

Input: $P, Q \subseteq \mathbb{Z}[x_1, \ldots, x_n], \alpha_1, \ldots, \alpha_n \in OP, \beta : P \ni p \mapsto \beta_p \in OP, \gamma : Q \ni q \mapsto \gamma_q \in OP$

Output: $\overline{\alpha}_1, \ldots, \overline{\alpha}_n \in OP, \overline{\beta} : P \ni p \mapsto \overline{\beta}_p \in OP$, *unsat* $\in$ {true, false}

s.t. if *unsat* = true, $F \Leftrightarrow$ false , else $F \Leftrightarrow \overline{\alpha}_1(x_1) \wedge \cdots \wedge \overline{\alpha}_n(x_n) \wedge \bigwedge_{p \in P} \overline{\beta}_p(p) \wedge \bigwedge_{q \in Q} \gamma_q(q)$

$(F \coloneqq \alpha_1(x_1) \wedge \cdots \wedge \alpha_n(x_n) \wedge \bigwedge_{p \in P} \beta_p(p) \wedge \bigwedge_{q \in Q} \gamma_q(q))$

1: *unsat* $\leftarrow$ false, $R \leftarrow P \cup Q$. **for** $1 \le i \le n$ **do** $\overline{\alpha}_i \leftarrow \alpha_i$ **end for**. **for all** $p \in P$ **do** $\overline{\beta}_p \leftarrow \beta_p$ **end for**.
2: **for all** $r \in R$ **do**
3: **if** $r \in P$ **then** $\delta_r \leftarrow \beta_r$ **else** $\delta_r \leftarrow \gamma_r$ **end if**
4: **end for**
5: **for all** $1 \le i \le n$ **do**
6: $\overline{\alpha}_i \leftarrow \overline{\alpha}_i \wedge$ DeduceAll$(x_i; R; \overline{\alpha}_1, \ldots, \overline{\alpha}_n; \delta)$
7: **if** $\overline{\alpha}_i = NOOP$ **then** *unsat* $\leftarrow$ true **end if**
8: **end for**
9: **for all** $p \in P$ **do**
10: $\overline{R} \leftarrow R \setminus \{p\}, \overline{\delta} \leftarrow \delta \mid \overline{R}, \overline{\beta}_p \leftarrow \overline{\beta}_p \wedge$ DeduceAll$(p; \overline{R}; \overline{\alpha}_1, \ldots, \overline{\alpha}_n; \delta)$
11: **if** $\overline{\beta}_p = NOOP$ **then** *unsat* $\leftarrow$ true **end if**
12: **end for**
13: **return** $\overline{\alpha}_1, \ldots, \overline{\alpha}_n, \overline{\beta}$, *unsat*

るので, 次式の要領で AndSimplify を適用することが出来る.

$$
\begin{aligned}
\textbf{Input} &= \bigvee_{p \in P} \varphi(p) \\
&\Leftrightarrow \neg \left(\bigwedge_{p \in P} \neg \varphi(p) \right) \\
&\Leftrightarrow \neg \left(\bigwedge_{p \in P} \neg \psi(p) \right) \quad \left(\text{但し, } \bigwedge_{p \in P} \neg \psi(p) \text{ は, AndSimplify} \left(\bigwedge_{p \in P} \neg \varphi(p) \right) \text{の出力論理式を表す} \right) \\
&\Leftrightarrow \bigvee_{p \in P} \psi(p) = \textbf{Output}.
\end{aligned}
\tag{1}
$$

以上の拡張を OrSimplify と呼ぶ. OrSimplify を明記するために OP の元の否定をとる単項演算を定義する.

定義 7 (OP の否定) 単項演算 $NEG : OP \to OP$ を $NEG(op)(x) \coloneqq$ true if $\neg(op(x))$, else false と定義する. 但し, $op \in OP$ とする. □

OrSimplify を Algorithm 8 に示す. なお, OrSimplify の入出力である原始多項式の有限集合 $P \subseteq PP(x_1, \ldots, x_n)$ と, 写像 σ: $P \to OP$ は「原子論理式の論理和」 $\bigvee_{p \in P} \sigma_p(p)$ を表す. ここで, *unsat* の値は AndSimplify からそのまま引き継いでいることに注意する. *unsat* = true の場合, AndSimplify では論理式全体が false であるので, その否定をとる OrSimplify では論理式全体が true となる.

例 3 論理式 $\psi \equiv y^2 - x + 1 \le 0 \vee 2x - 1 \ge 0$ を OrSimplify により簡単化する.

式変形 (1) の各行に対応するように簡単化の詳細を示すと,

$$
\begin{aligned}
\psi &\equiv y^2 - x + 1 \le 0 \vee 2x - 1 \ge 0 \\
&\Leftrightarrow \neg(y^2 - x + 1 > 0 \wedge 2x - 1 < 0) \\
&\Leftrightarrow \neg(2x - 1 < 0) \quad (\because \text{例 1}) \\
&\Leftrightarrow 2x - 1 \ge 0
\end{aligned}
$$

となり, 原子論理式の論理和 ψ が簡単化された. □

Algorithm 7 White-Box (AndSimplify) [4, Algorithm 8]

Input: 有限集合 $P \subseteq PP(x_1, \ldots, x_n)$, $\sigma : P \ni p \mapsto \sigma_p \in OP$

Output: 有限集合 $Q \subseteq PP(x_1, \ldots, x_n)$, $\tau : Q \ni q \mapsto \tau_q \in OP$, $unsat \in \{\text{true, false}\}$

s.t. if $unsat$ = true, $\bigwedge_{p \in P} \sigma_p(p) \Leftrightarrow$ false, else $\bigwedge_{p \in P} \sigma_p(p) \Leftrightarrow \bigwedge_{q \in Q} \tau_q(q)$

$Q \leftarrow P$, $\tau \leftarrow \sigma$, $unsat \leftarrow$ false, $P_1 \leftarrow P$ の元の既約因子すべての集合, $P_2 \leftarrow P_1 \setminus \{x_1, \ldots, x_n\}$, $P_3 \leftarrow P \setminus P_1$

for $1 \le i \le n$ **do**

  if $x_i \in P$ **then** $\alpha_i \leftarrow \sigma_{x_i}$ **else** $\alpha_i \leftarrow ALOP$ **end if**

end for

for all $p \in P_2$ **do**

  if $p \in P$ **then** $\beta_p \leftarrow \sigma_p$ **else** $\beta_p \leftarrow ALOP$ **end if**

  $\beta_p \leftarrow \beta_p \wedge \text{PolynomialSign}(p; \alpha_1, \ldots, \alpha_n)$

  if $\beta_p = NOOP$ **then** $unsat \leftarrow$ true, **return** $Q, \tau, unsat$ **end if**

end for

for all $p \in P_3$ **do** $\gamma_p \leftarrow \sigma_p$ **end for**

repeat

  $changed \leftarrow$ false

  $(\overline{\alpha}_1, \ldots, \overline{\alpha}_n), \overline{\beta}, unsat \leftarrow \text{DeduceAllSigns}(P_2; P_3; \alpha_1, \ldots, \alpha_n; \beta; \gamma)$

  if $unsat$ = true **then return** $Q, \tau, unsat$ **end if**

  if $(\overline{\alpha}_1, \ldots, \overline{\alpha}_n) \neq (\alpha_1, \ldots, \alpha_n) \vee \overline{\beta} \neq \beta$ **then** $(\alpha_1, \ldots, \alpha_n) \leftarrow (\overline{\alpha}_1, \ldots, \overline{\alpha}_n)$, $\beta \leftarrow \overline{\beta}$, $changed \leftarrow$ true **end if**

until $changed$ = false

for all $p \in P_2$ **do**

  if $\beta_p = NEOP \wedge p$ が $q \in P_3$ の因子 $\wedge\ \gamma_q \in \{LTOP, GTOP, NEOP\}$ **then** $\beta_p \leftarrow ALOP$ **end if**

  if $\beta_p \neq ALOP \wedge \beta_p = \text{PolynomialSign}(p; \alpha_1, \ldots, \alpha_n)$ **then** $\beta_p \leftarrow ALOP$ **end if**

  if $\beta_p \neq ALOP$ **then**

    $R \leftarrow (P_2 \setminus \{p\}) \cup P_3$, $\rho \leftarrow (\beta \mid P_2 \setminus \{p\}) \cup \gamma$

    if $\beta_p = \text{DeduceAll}(p; R; \alpha_1, \ldots, \alpha_n; \rho)$ **then** $\beta_p \leftarrow ALOP$ **end if**

  end if

end for

for all $p \in P_3$ **do**

  if $\gamma_p \neq ALOP \wedge\ \gamma_p = \text{PolynomialSign}(p; \alpha_1, \ldots, \alpha_n)$ **then** $\gamma_p \leftarrow ALOP$ **end if**

  if $\gamma_p \neq ALOP$ **then**

    $R \leftarrow P_2 \cup (P_3 \setminus \{p\})$, $\rho \leftarrow \beta \cup (\gamma \mid P_3 \setminus \{p\})$

    if $\gamma_p = \text{DeduceAll}(p; R; \alpha_1, \ldots, \alpha_n; \rho)$ **then** $\gamma_p \leftarrow ALOP$ **end if**

  end if

end for

$Q_1 \leftarrow \{x_i \mid \alpha_i \neq ALOP\}$, $Q_2 \leftarrow \{p \in P_2 \mid \beta_p \neq ALOP\}$, $Q_3 \leftarrow \{p \in P_3 \mid \gamma_p \neq ALOP\}$ $Q \leftarrow Q_1 \cup Q_2 \cup Q_3$

for all $x_i \in Q_1$ **do** $\tau_{x_i} \leftarrow \alpha_i$ **end for**

for all $p \in Q_2$ **do** $\tau_p \leftarrow \beta_p$ **end for**

for all $p \in Q_3$ **do** $\tau_p \leftarrow \gamma_p$ **end for**

return $Q, \tau, unsat$

Algorithm 8 OrSimplify

Input: 有限集合 $P \subseteq PP(x_1, \ldots, x_n)$, $\sigma : P \ni p \mapsto \sigma_p \in OP$

Output: 有限集合 $Q \subseteq PP(x_1, \ldots, x_n)$, $\tau : Q \ni q \mapsto \tau_q \in OP$, $unsat \in \{\text{true, false}\}$

s.t. if $unsat$ = true, $\bigvee_{p \in P} \sigma_p(p) \Leftrightarrow$ true, else $\bigvee_{p \in P} \sigma_p(p) \Leftrightarrow \bigvee_{q \in Q} \tau_q(q)$

1: **for all** $p \in P$ **do** $\overline{\sigma}_p \leftarrow NEG(\sigma_p)$ **end for**

2: $Q, \overline{\tau}, unsat \leftarrow$ AndSimplify$(P; \overline{\sigma})$

3: **for all** $q \in Q$ **do** $\tau_q \leftarrow NEG(\overline{\tau}_q)$ **end for**

4: **return** $Q, \tau, unsat$

3.2 論理積に論理和が入れ子になった論理式への拡張

次に「論理積に論理和が入れ子になった論理式」に対して AndSimplify の考え方を適用可能にした拡張を考える. 本拡張では, 論理積に論理和が入れ子になった論理式に対して常に存在する等価性のもとで AndSimplify の考え方を適用する. ($p_i, q_j \in \mathbb{Z}[x_1, \ldots, x_n]$ とし, $\varphi(p_i), \varphi(q_j)$ をそれぞれ原子論理式とする.)

$$\bigwedge_{j=1}^{t} \varphi(q_j) \wedge (\varphi(p_1) \vee \cdots \vee \varphi(p_s)) \tag{2}$$

$$\Leftrightarrow ((\bigwedge_{j=1}^{t} \varphi(q_j) \wedge \varphi(p_1)) \vee \cdots \vee (\bigwedge_{j=1}^{t} \varphi(q_j) \wedge \varphi(p_s))) \tag{3}$$

$$\Leftrightarrow ((\bigwedge_{j=1}^{t} \varphi(q_j) \wedge \varphi'(p_1)) \vee \cdots \vee (\bigwedge_{j=1}^{t} \varphi(q_j) \wedge \varphi'(p_s))) \tag{4}$$

$$\Leftrightarrow \bigwedge_{j=1}^{t} \varphi(q_j) \wedge (\varphi'(p_1) \vee \cdots \vee \varphi'(p_s)). \tag{5}$$

与えられた式 (2) に対し, 論理和の外の原子論理式の論理積 $\bigwedge_{j=1}^{t} \varphi(q_j)$ を論理和の中の各原子論理式 $\varphi(p_i)$ に分配する (式 (3)). 次に式 (3) で書き換えた $\bigwedge_{j=1}^{t} \varphi(q_j) \wedge \varphi(p_i)$ に対して DeduceAll を適用する. 原子論理式の論理積 $\bigwedge_{j=1}^{t} \varphi(q_j)$ を仮定して多項式 p_i の符号を推論し, 可能ならば簡単化を行う. 式 (4) における $\psi'(p_i)$ は $\psi(p_i)$ を簡単化した結果である. 最後に, $\bigwedge_{j=1}^{t} \varphi(q_j)$ を論理和の外に戻すことで, 論理和の中の各原子論理式 $\varphi(p_i)$ が簡単化された形に論理式を変形する (式 (5)).

以上の拡張を OrNestedSimplify と呼び, Algorithm 9 に示す. なお, OrNestedSimplify の入出力である原始多項式の有限集合 $P, Q \subseteq PP(x_1, \ldots, x_n)$ と, 写像 σ: $P \to OP$, τ: $Q \to OP$ は「論理積に論理和が入れ子になった論理式」$\bigwedge_{q \in Q} \tau_q(q) \wedge \bigvee_{p \in P} \sigma_p(p)$ を表し, 第 1, 2 引数 P, σ が簡単化される入れ子になった論理式を表し, 第 3, 4 引数 Q, τ が仮定とする入れ子の外の論理式を表す.

例 4 論理式 $\varphi \equiv 2x - 1 < 0 \wedge (y^2 - x + 1 \leq 0 \vee z + 1 > 0)$ を OrNestedSimplify により簡単化する.

式 (2) から式 (5) の手順に沿って簡単化の詳細を示すと,

$$\begin{aligned}
\varphi &\equiv 2x - 1 < 0 \wedge (y^2 - x + 1 \leq 0 \vee z + 1 > 0) \\
&\Leftrightarrow (2x - 1 < 0 \wedge y^2 - x + 1 \leq 0) \vee (2x - 1 < 0 \wedge z + 1 > 0) \\
&\Leftrightarrow (2x - 1 < 0 \wedge \text{false}) \vee (2x - 1 < 0 \wedge z + 1 > 0) \\
&\Leftrightarrow 2x - 1 < 0 \wedge (\text{false} \vee z + 1 > 0)
\end{aligned}$$

となる. 論理和に含まれた false は取り除いても等価であるので

$$2x - 1 < 0 \wedge z + 1 > 0$$

が出力となる. □

Algorithm 9 OrNestedSimplify

Input: 有限集合 $P \subseteq PP(x_1, \ldots, x_n)$, $\sigma : P \ni p \mapsto \sigma_p \in OP$, 有限集合 $Q \subseteq PP(x_1, \ldots, x_n)$,
$\tau : Q \ni q \mapsto \tau_q \in OP$ ($F \coloneqq \bigwedge_{q \in Q} \tau_q(q) \wedge \bigvee_{p \in P} \sigma_p(p)$ とする.)

Output: 有限集合 $R \subseteq PP(x_1, \ldots, x_n)$, $\overline{\sigma} : R \ni r \mapsto \overline{\sigma}_r \in OP$. $unsat \in \{$ true, false $\}$
s.t. if $unsat$ = true, $F \Leftrightarrow$ false, else $F \Leftrightarrow \bigwedge_{q \in Q} \tau_q(q) \wedge (R \neq \emptyset \rightarrow \bigvee_{r \in R} \overline{\sigma}_r(r))$

1: $R \leftarrow P, \overline{\sigma} \leftarrow \sigma, unsat \leftarrow$ false
2: **for** $1 \leq i \leq n$ **do**
3: **if** $x_i \in Q$ **then** $\alpha_i \leftarrow \tau_{x_i}$ **else** $\alpha_i \leftarrow ALOP$ **end if**
4: **end for**
5: **for all** $r \in R$ **do**
6: **for** $1 \leq i \leq n$ **do**
7: **if** $r = x_i$ **then** $\overline{\alpha}_i \leftarrow \overline{\sigma}_r$ **else** $\overline{\alpha}_i \leftarrow \alpha_i$ **end if**
8: **end for**
9: **if** DeduceAll($r; Q; \overline{\alpha}_1, \ldots, \overline{\alpha}_n; \tau$) が $\overline{\sigma}_r$ より強い **then** $R \leftarrow \emptyset$, $\overline{\sigma} \leftarrow \emptyset$, **return** $R, \overline{\sigma}, unsat$
10: **else** $\overline{\sigma}_r \leftarrow \overline{\sigma}_r \wedge$ DeduceAll($r; Q; \overline{\alpha}_1, \ldots, \overline{\alpha}_n; \tau$)
11: **end if**
12: **end for**
13: **if** all $\overline{\sigma}_r = NOOP$ **then** $unsat \leftarrow$ true, **return** $R, \overline{\sigma}, unsat$ **end if**
14: $R \leftarrow \{r \in R \mid \overline{\sigma}_r \neq NOOP\}$
15: **return** $R, \overline{\sigma}, unsat$

OrNestedSimplify の長所は, ある原子論理式を簡単化する際, AndSimplify, OrSimplify のみでは仮定できない原子論理式を仮定できる点である. 例 4 において, 仮に AndSimplify と OrSimplify のみを簡単化アルゴリズムとして用いた場合, 原子論理式 $y^2 - x + 1 \leq 0$ の簡単化は, 同じ論理和に含まれる原子論理式 $z + 1 > 0$ を仮定して, OrSimplify を用いて行う. この場合, $y^2 - x + 1 > 0$ は簡単化できない. ところが, OrNestedSimplify を導入すると, OrSimplify で仮定した $z + 1 > 0$ に加えて, 論理和の外の原子論理式 $2x - 1 < 0$ も仮定することが可能になる. すなわち, OrNestedSimplify を導入すると, より多くの情報を持って簡単化を行うことができ, 論理式の簡単化の可能性を更に広げることができる.

3.3 論理和に論理積が入れ子になった論理式への拡張

最後の拡張は「論理和に論理積が入れ子になった論理式」に対する拡張である. 本拡張の考え方は, OrSimplify と同様であり, 論理和を「論理積の否定」と考え, OrNestedSimplify を適用する.

$$
\begin{aligned}
\textbf{Input} &= \bigvee_{\phi \in \Phi} \phi \\
&\Leftrightarrow \neg \left(\bigwedge_{\phi \in \Phi} \neg \phi \right) \\
&\Leftrightarrow \neg \left(\bigwedge_{\psi \in \Psi} \neg \psi \right) \quad \left(\text{但し, } \bigwedge_{\psi \in \Psi} \neg \psi \text{は,OrNestedSimplify} \left(\bigwedge_{\phi \in \Phi} \neg \phi \right) \text{の出力論理式を表す} \right) \\
&\Leftrightarrow \bigvee_{\psi \in \Psi} \psi = \textbf{Output}.
\end{aligned}
$$

以下の Algorithm 10 に 本拡張 AndNestedSimplify を示す. なお, AndNestedSimplify の入出力である原始多項式の有限集合 $P, Q \subseteq PP(x_1, \ldots, x_n)$ と, 写像 σ: $P \rightarrow OP$, τ: $Q \rightarrow OP$ は「論理和に論理積が入れ子になった論理式」$\bigvee_{q \in Q} \tau_q(q) \vee \bigwedge_{p \in P} \sigma_p(p)$ を表す.

Algorithm 10 AndNestedSimplify

Input: 有限集合 $P \subseteq PP(x_1, \ldots, x_n)$, $\sigma : P \ni p \mapsto \sigma_p \in OP$, 有限集合 $Q \subseteq PP(x_1, \ldots, x_n)$, $\tau : Q \ni q \mapsto \tau_q \in OP$ ($F \coloneqq \bigvee_{q \in Q} \tau_q(q) \vee \bigwedge_{p \in P} \sigma_p(p)$ とする.)

Output: 有限集合 $R \subseteq PP(x_1, \ldots, x_n)$, $\overline{\sigma} : R \ni r \mapsto \overline{\sigma}_r \in OP$. $unsat \in \{$ true, false $\}$

s.t. if $unsat$ = true, $F \Leftrightarrow$ true, else $F \Leftrightarrow \bigvee_{q \in Q} \tau_q(q) \vee (R \neq \emptyset \rightarrow \bigwedge_{r \in R} \overline{\sigma}_r(r))$

1: **for all** $p \in P$ **do** $\overline{\sigma}_p \leftarrow NEG(\sigma_p)$ **end for**
2: **for all** $q \in Q$ **do** $\overline{\tau}_q \leftarrow NEG(\tau_q)$ **end for**
3: $R, \overline{\sigma}, unsat \leftarrow$ OrNestedSimplify$(P; \overline{\sigma}; Q; \overline{\tau})$
4: **for all** $r \in R$ **do** $\overline{\sigma}_r \leftarrow NEG(\overline{\sigma}_r)$ **end for**
5: **return** $R, \overline{\sigma}, unsat$

ここで, $unsat$ の値は OrNestedSimplify からそのまま引き継いでいることに注意する. $unsat$ = true の場合, OrNestedSimplify では論理式全体が false であるので, その否定をとる AndNestedSimplify では論理式全体が true となる.

3.4 拡張 White-Box

オリジナルの White-Box (AndSimplify) に, 3 つの拡張アルゴリズム (OrSimplify, OrNestedSimplify, AndNested-Simplify) を加え構成した, 拡張 White-Box アルゴリズム (ExtendedWhite-Box) を Algorithm 11 に示す.

Algorithm 11 では, 本章の冒頭でも述べたように, 入力とする論理式として, 論理和は $\bigvee_{\phi \in \Phi} \phi \vee \bigvee_{\psi \in \Psi} \psi$ (ϕ: 原子論理式, ψ: 論理積) の形のみを考え, 論理積は $\bigwedge_{\phi \in \Phi} \phi \wedge \bigwedge_{\psi \in \Psi} \psi$ (ϕ: 原子論理式, ψ: 論理和) の形のみを考える. そして, 原子論理式 ϕ や, 論理積 (論理和) ψ に含まれる原子論理式 χ の簡単化を行う. 論理積 (論理和) ψ に入れ子になった論理和 (論理積) ω に対する簡単化は, Algorithm 11 の再帰的適用 (5, 23 行目) により, 前もって行われており, Algorithm 11 中では ω' を用いて簡単化結果を表現している.

4 実験

拡張 White-Box を数式処理システム Maple を用いて実装を行い, その効果を検証するために以下の 2 つの実験を行った.

実験 1 (QE 計算時間の短縮) いくつかの論理式に対し SyNRAC による QE 計算を行い,

- QE 計算の前に拡張 White-Box を施さない場合, および
- QE 計算の前に拡張 White-Box を施した場合 (拡張 White-Box + QE)

の 2 つの計算時間の比較を行う.

実験 2 (他 QE ツールとの比較) 同一の論理式に対し,

- Redlog (rlsimpl),
- QEPCAD (slfq),
- SyNRAC (synsimpl),
- 拡張 White-Box

の計 4 つの簡単化関数を適用し, それぞれの簡単化結果を比較する.

本実験の実験環境は以下の通りである.

Algorithm 11 ExtendedWhite-Box

Input: F: 限量子を含まない論理式

Output: F': 限量子を含まない論理式 s.t. $F' \Leftrightarrow F$

if $F = \bigvee_{\phi\in\Phi}\phi \vee \bigvee_{\psi\in\Psi}\psi$ (ϕ: 原子論理式, ψ: 論理積) **then**
  /* 論理積 ψ は $\psi = \bigwedge_{\chi\in X}\chi \wedge \bigwedge_{\omega\in\Omega}\omega$ (χ: 原子論理式, ω: 論理和) を満たすもの. */
  $\Phi' \leftarrow \Phi, \Psi' \leftarrow \Psi$
  if $\Psi \neq \emptyset$ **then**
    $\bigvee_{\psi'\in\Psi'}\psi' \leftarrow$ ExtendedWhite-Box($\bigvee_{\psi\in\Psi}\psi$)
    /* 論理積 ψ' は $\psi' = \bigwedge_{\chi'\in X'}\chi' \wedge \bigwedge_{\omega'\in\Omega'}\omega'$ (χ': 原子論理式, ω': 論理和) で $\bigvee_{\psi'\in\Psi'}\psi' \Leftrightarrow \bigvee_{\psi\in\Psi}\psi$ を満たすもの. */
    if $\bigvee_{\psi'\in\Psi'}\psi'$: 原子論理式 **then** $\Phi' \leftarrow \Phi \cup \{\bigvee_{\psi'\in\Psi'}\psi'\}, \Psi' \leftarrow \emptyset$ **end if**
  end if
  for all $\psi' \in \Psi'$ **do**
    /* 有限集合 $P \subseteq PP(x_1,\ldots,x_n)$ と σ: $P \ni p \mapsto \sigma_p \in OP$ は $\bigvee_{p\in P}\sigma_p(p) = \bigvee_{\phi'\in\Phi'}\phi'$ を満たすもの. */
    /* 有限集合 $Q \subseteq PP(x_1,\ldots,x_n)$ と τ: $Q \ni q \mapsto \tau_q \in OP$ は $\bigwedge_{q\in Q}\tau_q(q) = \bigwedge_{\chi'\in X'}\chi'$ を満たすもの. */
    $\overline{Q}, \overline{\tau}, unsat \leftarrow$ AndNestedSimplify(Q; τ; P; σ)
    if $unsat$ = true $\wedge\ \Omega' = \emptyset$ **then** $\psi' \leftarrow$ true **else** $\psi' \leftarrow \bigwedge_{q\in\overline{Q}}\overline{\tau}_q(q) \wedge \bigwedge_{\omega'\in\Omega'}\omega'$ **end if**
  end for
  $\overline{P}, \overline{\sigma}, unsat \leftarrow$ OrSimplify(P; σ)
  if $unsat$ = true **then return** true **end if**
  $\Phi' \leftarrow \{\overline{\sigma}_p(p) \mid p \in \overline{P}, \overline{\sigma} : P \ni p \mapsto \overline{\sigma}_p \in OP\}$
  $F' \leftarrow \bigvee_{\phi'\in\Phi'}\phi' \vee \bigvee_{\psi'\in\Psi'}\psi'$
else if $F = \bigwedge_{\phi\in\Phi}\phi \wedge \bigwedge_{\psi\in\Psi}\psi$ (ϕ: 原子論理式, ψ: 論理和) **then**
  /* 論理和 ψ は $\psi = \bigvee_{\chi\in X}\chi \vee \bigvee_{\omega\in\Omega}\omega$ (χ: 原子論理式, ω: 論理積) を満たすもの. */
  $\Phi' \leftarrow \Phi, \Psi' \leftarrow \Psi$
  if $\Psi \neq \emptyset$ **then**
    $\bigwedge_{\psi'\in\Psi'}\psi' \leftarrow$ ExtendedWhite-Box($\bigwedge_{\psi\in\Psi}\psi$)
    /* 論理和 ψ' は $\psi' = \bigvee_{\chi'\in X'}\chi' \vee \bigvee_{\omega'\in\Omega'}\omega'$ (χ': 原子論理式, ω': 論理積) で $\bigwedge_{\psi'\in\Psi'}\psi' \Leftrightarrow \bigwedge_{\psi\in\Psi}\psi$ を満たすもの. */
    if $\bigwedge_{\psi'\in\Psi'}\psi'$: 原子論理式 **then** $\Phi' \leftarrow \Phi \cup \{\bigwedge_{\psi'\in\Psi'}\psi'\}, \Psi' \leftarrow \emptyset$ **end if**
  end if
  for all $\psi' \in \Psi'$ **do**
    /* 有限集合 $P \subseteq PP(x_1,\ldots,x_n)$ と σ: $P \ni p \mapsto \sigma_p \in OP$ は $\bigwedge_{p\in P}\sigma_p(p) = \bigwedge_{\phi'\in\Phi'}\phi'$ を満たすもの. */
    /* 有限集合 $Q \subseteq PP(x_1,\ldots,x_n)$ と τ: $Q \ni q \mapsto \tau_q \in OP$ は $\bigvee_{q\in Q}\tau_q(q) = \bigvee_{\chi'\in X'}\chi'$ を満たすもの. */
    $\overline{Q}, \overline{\tau}, unsat \leftarrow$ OrNestedSimplify(Q; τ; P; σ)
    if $unsat$ = true $\wedge\ \Omega' = \emptyset$ **then** $\psi' \leftarrow$ false **else** $\psi' \leftarrow \bigvee_{q\in\overline{Q}}\overline{\tau}_q(q) \vee \bigvee_{\omega'\in\Omega'}\omega'$ **end if**
  end for
  $\overline{P}, \overline{\sigma}, unsat \leftarrow$ AndSimplify(P; σ)
  if $unsat$ = true **then return** false **end if**
  $\Phi' \leftarrow \{\overline{\sigma}_p(p) \mid p \in \overline{P}, \overline{\sigma} : P \ni p \mapsto \overline{\sigma}_p \in OP\}$
  $F' \leftarrow \bigwedge_{\phi'\in\Phi'}\phi' \wedge \bigwedge_{\psi'\in\Psi'}\psi'$
end if
return F'

- ホスト OS
  CPU: Intel Core i5-4300U 1.90GHz
  RAM: 8GB
  OS: Windows 8.1 Pro
  仮想化ソフトウェア: VMware Player 7.1.2
- ゲスト OS
  RAM: 4GB
  OS: Linux 3.16.0 SMP

4.1 実験 1: QE 計算時間の短縮

今回は 207 個の論理式を QE の入力として,

- QE 計算の前に拡張 White-Box を施さない場合の QE 計算時間
- QE 計算の前に拡張 White-Box を施した場合の (拡張 White-Box + QE) 計算時間

の 2 つを比較した.

これら 207 個の論理式は CAD アルゴリズムによる QE 計算を行う前に, 効率化のため少しでも簡単化が必要な例題として集められたものである.

これらの論理式に対して, 上記の 2 つの計算時間を測定した. なお, 計算時間が 3600 秒 (1 時間) を超えた論理式はタイムアウトとして計算を終了させた.

207 個のうち 8 個の論理式 (付録 A 参照) に対して, QE 計算の効率化の点で特に効果が見受けられた. この 8 個の論理式に対する計算時間を比較した表が以下である.

論理式	T_1	T_2	$T_1 - T_2$	T_2/T_1
論理式 1	0.324	0.064	0.260	0.198
論理式 2	110.359	0.092	110.267	0.001
論理式 3	0.336	0.144	0.192	0.480
論理式 4	2.393	0.632	1.761	0.264
論理式 5	1.064	0.016	1.048	0.015
論理式 6	141.745	0.032	141.713	0.000
論理式 7	58.964	48.367	10.597	0.820
論理式 8	1.328	0.492	0.836	0.370

表 1 計算時間の比較 (秒)

実験結果を表 1 に示す. ここに, T_1, T_2 はそれぞれ QE 計算の前に拡張 White-Box を施さない場合の QE 計算時間 (秒), QE 計算の前に拡張 White-Box を施した場合の (拡張 White-Box + QE) 計算時間 (秒) を表す. 今回は計算時間に関して 2 つの比較を行った. 一方は計算時間の差, もう一方は計算時間の比である.

計算時間の差 $T_1 - T_2$ を見ると, 最も効果があった例, 論理式 6 では 2 分以上の効率化が実現されている. 付録 A から論理式が拡張 White-Box を施すことによりどれだけ簡単化されたかも参照することができるが, 非常に冗長な論理式であっても, 簡単化後に false という最も単純な形まで簡単化できる例もあり, このような例は拡張 White-Box が得意とする例ということができる.

もともと数秒程度で QE 計算が終わっていた例に関しても計算時間の比 T_2/T_1 を見ることで効率化の様子を探ることができる. 計算時間の差のみでは効果を見受けられない例でも, それらの比を見ることで計算時間が半分以下になった論理式を挙げることができる. (論理式 1, 3, 4, 5, 8)

4.2 実験 2: 他の簡単化手法との比較

次に, 上と同じ実験例に対し, 他の簡単化手法との比較結果を見る. 付録 A の論理式には各簡単化関数による簡単化結果も合わせて記載している.

拡張 White-Box による簡単化結果と他の QE ツール上の簡単化関数との簡単化結果を比較する. 論理式 1–8 は拡張 White-Box により QE 計算時間の効率化が行われた論理式であるので, 拡張 White-Box がある程度得意な形の論理式であることがわかっている. Redlog, SyNRAC と比較すると, 拡張 White-Box はより簡単化できている. しかし QEPCAD と比較すると 8 つすべての論理式において, QEPCAD は拡張 White-Box と同程度, またはより効果的な簡単化が行われている.

そこで拡張 White-Box で簡単化ができ, QEPCAD には簡単化が行えない例として論理式 9 (付録 A 参照) を挙げる. White-Box のアイデアに基づくと, 変数の符号情報を複数持つことができることに注意すれば, 変数の和の多項式の符号は正であることが推論できる. 一方, この論理式を QEPCAD により簡単化を行うと, 30 変数の論理式 9 の場合で 1 時間たっても計算が終了しなかった. QEPCAD の簡単化は, 計算時間が入力論理式の変数の個数に大きく依存する CAD アルゴリズムによる簡単化を行っている [2]. これが計算時間増大の原因であると考えられる. 論理式 9 のような数十変数の論理式に対する簡単化は QEPCAD の不得手な例ということがわかった.

5 結論

本稿では, 論理式簡単化アルゴリズム White-Box の拡張を提案し, その実装及び計算機実験の結果について述べた. 「原子論理式の論理積」のみに対して適用可能であった White-Box を「任意の論理式」に対して適用可能となるよう拡張し, 計算機実験により, その簡単化効果を検証したところ, 一部の例に対して QE 計算の効率化を実現した. さらに, 他の QE ツールの簡単化関数との比較実験において, Redlog, SyNRAC では簡単化できないが, 拡張 White-Box では簡単化できる論理式を発見できた. また, QEPCAD による簡単化が終了しなかった数十変数の論理式に対して, 拡張 White-Box では変数の個数の増加に大きな影響を受けずに簡単化を実行できることもわかった.

今後の課題として以下が挙げられる. 1 点目は, 拡張 White-Box を構成する 4 個のアルゴリズム (AndSimplify, OrSimplify, OrNestedSimplify, AndNestedSimplify) をどのような順序で論理式に適用すれば簡単化の効果が最も大きくなるかを検証することである. 2 点目は, 複数個の原子論理式を同時に仮定して, 1 個の原子論理式を推論するアルゴリズムを構築することで White-Box のさらなる拡張を行うことである. 例えば

$$x+1>0 \wedge y+1>0 \wedge x+y+2>0$$

において, $x+1>0, y+1>0$ から $x+y+2>0$ を推論する, といったものが挙げられる.

付録 A 実験データ: 論理式と各簡単化関数による簡単化結果

論理式 1

元の論理式

$$\exists x_1((x_2<0 \vee 2x_1^2-2x_1-1\neq 0) \wedge (x_2<0 \vee 2x_1^2+2x_1-1\neq 0) \wedge (-x_1^2-x_1\leq 1 \vee x_1^2-x_1\leq -1) \wedge (-x_1^2+x_1\leq 1 \vee x_1^2+x_1\leq -1) \wedge (2x_1^2-2x_1-1=0 \vee 2x_1^2+2x_1-1=0) \wedge -x_1<-2 \wedge x_1<4 \wedge -4x_1^2\leq -3 \wedge x_1^2-x_1+1\neq 0 \wedge x_1^2+x_1+1\neq 0 \wedge 2x_1^4-13x_1^2-1=0)$$

拡張 White-Box

$$\exists x_1(0<x_1-2 \wedge x_1<4 \wedge 0\leq 4x_1^2-3 \wedge 2x_1^4-13x_1^2-1=0 \wedge 2x_1^2-2x_1-1=0 \wedge (x_2<0 \vee 2x_1^2-2x_1-1\neq 0))$$

Redlog

$$\exists x_1(x_1-4<0 \wedge x_1-2>0 \wedge x_1^2-x_1+1>0 \wedge 4x_1^2-3\geq 0 \wedge 2x_1^4-13x_1^2-1=0 \wedge (x_2<0 \vee 2x_1^2-2x_1-1\neq 0) \wedge (x_2<0 \vee 2x_1^2+2x_1-1\neq 0) \wedge (2x_1^2-2x_1-1=0 \vee 2x_1^2+2x_1-1=0))$$

QEPCAD

false

SyNRAC

変化無し

論理式 2

元の論理式

$$\exists x_2((-x_2^2 \le -2 \vee x_2^2x_1^2 - x_2^2 \le -2) \wedge (-x_2x_1 < 2 \vee -2x_2^2x_1^2 + x_2^2 - 4x_2x_1 < 4) \wedge (-x_2x_1 < 2 \vee -x_2^4 - 8x_2^3x_1 - 8x_2^2x_1^2 - 8x_2^2 < -16) \wedge (x_2x_1 < 2 \vee 2x_2^2x_1^2 - x_2^2 - 4x_2x_1 < -4) \wedge (x_2x_1 < 2 \vee -x_2^4 + 8x_2^3x_1 - 8x_2^2x_1^2 - 8x_2^2 < -16) \wedge (-6x_2^2x_1^2 + 5x_2^2 \le 4 \vee 24x_2^4x_1^4 - 32x_2^4x_1^2 + 9x_2^4 + 40x_2^2x_1^2 - 24x_2^2 \le -16) \wedge (6x_2^2x_1^2 - 5x_2^2 \le -4 \vee -24x_2^4x_1^4 + 32x_2^4x_1^2 - 9x_2^4 - 40x_2^2x_1^2 + 24x_2^2 \le 16) \wedge (6x_2^2x_1^2 - 3x_2^2 + 4 \ne 0 \vee 9x_2^8 - 48x_2^6x_1^2 + 576x_2^4x_1^4 - 48x_2^6 - 384x_2^4x_1^2 + 160x_2^4 + 768x_2^2x_1^2 - 256x_2^2 + 256 \ne 0) \wedge (-2x_2^2x_1^2 + x_2^2 - 4x_2x_1 < 4 \vee x_2^4 + 8x_2^3x_1 + 8x_2^2x_1^2 + 8x_2^2 < 16) \wedge (2x_2^2x_1^2 - x_2^2 - 4x_2x_1 < -4 \vee x_2^4 - 8x_2^3x_1 + 8x_2^2x_1^2 + 8x_2^2 < 16) \wedge (-48x_2^4x_1^4 + 64x_2^4x_1^2 - 19x_2^4 - 56x_2^2x_1^2 + 32x_2^2 \le 16 \vee 96x_2^4x_1^6 - 176x_2^4x_1^4 + 90x_2^4x_1^2 + 208x_2^2x_1^4 - 9x_2^4 - 200x_2^2x_1^2 + 24x_2^2 + 96x_1^2 \le 16) \wedge (24x_2^4x_1^4 - 24x_2^4x_1^2 + 3x_2^4 + 40x_2^2x_1^2 - 8x_2^2 + 16 \ne 0 \vee 9x_2^8 - 48x_2^6x_1^2 + 576x_2^4x_1^4 - 48x_2^6 - 384x_2^4x_1^2 + 160x_2^4 + 768x_2^2x_1^2 - 256x_2^2 + 256 \ne 0) \wedge (48x_2^4x_1^4 - 64x_2^4x_1^2 + 19x_2^4 + 56x_2^2x_1^2 - 32x_2^2 \le -16 \vee -96x_2^4x_1^6 + 176x_2^4x_1^4 - 90x_2^4x_1^2 - 208x_2^2x_1^4 + 9x_2^4 + 200x_2^2x_1^2 - 24x_2^2 - 96x_1^2 \le -16) \wedge (-6x_2^2x_1^2 + 3x_2^2 < 4 \vee -24x_2^4x_1^4 + 24x_2^4x_1^2 - 3x_2^4 - 40x_2^2x_1^2 + 8x_2^2 < 16 \vee 9x_2^8 - 48x_2^6x_1^2 + 576x_2^4x_1^4 - 48x_2^6 - 384x_2^4x_1^2 + 160x_2^4 + 768x_2^2x_1^2 - 256x_2^2 + 256 \ne 0) \wedge (6x_2^2x_1^2 - 3x_2^2 < -4 \vee 24x_2^4x_1^4 - 24x_2^4x_1^2 + 3x_2^4 + 40x_2^2x_1^2 - 8x_2^2 < -16 \vee 9x_2^8 - 48x_2^6x_1^2 + 576x_2^4x_1^4 - 48x_2^6 - 384x_2^4x_1^2 + 160x_2^4 + 768x_2^2x_1^2 - 256x_2^2 + 256 \ne 0) \wedge x_1 < 0 \wedge -x_2 < 0 \wedge x_2^2 \le 2 \wedge -x_2^2x_1^2 + x_2^2 \le 2 \wedge 96x_2^8x_1^4 - 176x_2^8x_1^2 - 384x_2^6x_1^4 + 81x_2^8 + 816x_2^6x_1^2 + 576x_2^4x_1^4 - 432x_2^6 - 1408x_2^4x_1^2 + 864x_2^4 + 768x_2^2x_1^2 - 768x_2^2 + 256 = 0)$$

拡張 White-Box

false

Redlog

$$\exists x_2(x_2 > 0 \wedge x_2^2 - 2 \le 0 \wedge x_1 < 0 \wedge x_1^2x_2^2 - x_2^2 + 2 \ge 0 \wedge 96x_1^4x_2^8 - 384x_1^4x_2^6 + 576x_1^4x_2^4 - 176x_1^2x_2^8 + 816x_1^2x_2^6 - 1408x_1^2x_2^4 + 768x_1^2x_2^2 + 81x_2^8 - 432x_2^6 + 864x_2^4 - 768x_2^2 + 256 = 0 \wedge (x_2^2 - 2 = 0 \vee x_1^2x_2^2 - x_2^2 + 2 = 0) \wedge (x_1x_2 + 2 > 0 \vee 2x_1^2x_2^2 + 4x_1x_2 - x_2^2 + 4 > 0) \wedge (x_1x_2 + 2 > 0 \vee 8x_1^2x_2^2 + 8x_1x_2^3 + x_2^4 + 8x_2^2 - 16 > 0) \wedge (2x_1^2x_2^2 - 4x_1x_2 - x_2^2 + 4 < 0 \vee 8x_1^2x_2^2 - 8x_1x_2^3 + x_2^4 + 8x_2^2 - 16 < 0) \wedge (2x_1^2x_2^2 + 4x_1x_2 - x_2^2 + 4 > 0 \vee 8x_1^2x_2^2 + 8x_1x_2^3 + x_2^4 + 8x_2^2 - 16 < 0) \wedge (6x_1^2x_2^2 - 5x_2^2 + 4 \le 0 \vee 24x_1^4x_2^4 - 32x_1^2x_2^4 + 40x_1^2x_2^2 + 9x_2^4 - 24x_2^2 + 16 \ge 0) \wedge (6x_1^2x_2^2 - 5x_2^2 + 4 \ge 0 \vee 24x_1^4x_2^4 - 32x_1^2x_2^4 + 40x_1^2x_2^2 + 9x_2^4 - 24x_2^2 + 16 \le 0) \wedge (6x_1^2x_2^2 - 3x_2^2 + 4 \ne 0 \vee 576x_1^4x_2^4 - 48x_1^2x_2^6 - 384x_1^2x_2^4 + 768x_1^2x_2^2 + 9x_2^8 - 48x_2^6 + 160x_2^4 - 256x_2^2 + 256 \ne 0) \wedge (6x_1^2x_2^2 - 3x_2^2 + 4 < 0 \vee 24x_1^4x_2^4 - 24x_1^2x_2^4 + 40x_1^2x_2^2 + 3x_2^4 - 8x_2^2 + 16 < 0 \vee 576x_1^4x_2^4 - 48x_1^2x_2^6 - 384x_1^2x_2^4 + 768x_1^2x_2^2 + 9x_2^8 - 48x_2^6 + 160x_2^4 - 256x_2^2 + 256 \ne 0) \wedge (6x_1^2x_2^2 - 3x_2^2 + 4 > 0 \vee 24x_1^4x_2^4 - 24x_1^2x_2^4 + 40x_1^2x_2^2 + 3x_2^4 - 8x_2^2 + 16 > 0 \vee 576x_1^4x_2^4 - 48x_1^2x_2^6 - 384x_1^2x_2^4 + 768x_1^2x_2^2 + 9x_2^8 - 48x_2^6 + 160x_2^4 - 256x_2^2 + 256 \ne 0) \wedge (24x_1^4x_2^4 - 24x_1^2x_2^4 + 40x_1^2x_2^2 + 3x_2^4 - 8x_2^2 + 16 \ne 0 \vee 576x_1^4x_2^4 - 48x_1^2x_2^6 - 384x_1^2x_2^4 + 768x_1^2x_2^2 + 9x_2^8 - 48x_2^6 + 160x_2^4 - 256x_2^2 + 256 \ne 0) \wedge (48x_1^4x_2^4 - 64x_1^2x_2^4 + 56x_1^2x_2^2 + 19x_2^4 - 32x_2^2 + 16 \le 0 \vee 96x_1^6x_2^4 - 176x_1^4x_2^4 + 208x_1^4x_2^2 + 90x_1^2x_2^4 - 200x_1^2x_2^2 + 96x_1^2 - 9x_2^4 + 24x_2^2 - 16 \ge 0) \wedge (48x_1^4x_2^4 - 64x_1^2x_2^4 + 56x_1^2x_2^2 + 19x_2^4 - 32x_2^2 + 16 \ge 0 \vee 96x_1^6x_2^4 - 176x_1^4x_2^4 + 208x_1^4x_2^2 + 90x_1^2x_2^4 - 200x_1^2x_2^2 + 96x_1^2 - 9x_2^4 + 24x_2^2 - 16 \le 0))$$

QEPCAD

false

SyNRAC

$$\exists x_2((-x_2^2 \le -2 \vee x_2^2x_1^2 - x_2^2 \le -2) \wedge (-x_2x_1 < 2 \vee -2x_2^2x_1^2 + x_2^2 - 4x_2x_1 < 4) \wedge (-x_2x_1 < 2 \vee -x_2^4 - 8x_2^3x_1 - 8x_2^2x_1^2 - 8x_2^2 < -16) \wedge (x_2x_1 < 2 \vee 2x_2^2x_1^2 - x_2^2 - 4x_2x_1 < -4) \wedge (x_2x_1 < 2 \vee -x_2^4 + 8x_2^3x_1 - 8x_2^2x_1^2 - 8x_2^2 < -16) \wedge (9x_1^4 - 6x_1^2 + 1 \ne 0 \vee x_2^2 - 24x_1^2 + 4 \ne 0) \wedge (-6x_2^2x_1^2 + 5x_2^2 \le 4 \vee 24x_2^4x_1^4 - 32x_2^4x_1^2 + 9x_2^4 + 40x_2^2x_1^2 - 24x_2^2 \le -16) \wedge (6x_2^2x_1^2 - 5x_2^2 \le -4 \vee -24x_2^4x_1^4 + 32x_2^4x_1^2 - 9x_2^4 - 40x_2^2x_1^2 + 24x_2^2 \le 16) \wedge (-2x_2^2x_1^2 + x_2^2 - 4x_2x_1 < 4 \vee x_2^4 + 8x_2^3x_1 + 8x_2^2x_1^2 + 8x_2^2 < 16) \wedge (2x_2^2x_1^2 - x_2^2 - 4x_2x_1 < -4 \vee x_2^4 - 8x_2^3x_1 + 8x_2^2x_1^2 + 8x_2^2 < 16) \wedge (-48x_2^4x_1^4 + 64x_2^4x_1^2 - 19x_2^4 - 56x_2^2x_1^2 + 32x_2^2 \le 16 \vee 96x_2^4x_1^6 - 176x_2^4x_1^4 + 90x_2^4x_1^2 + 208x_2^2x_1^4 - 9x_2^4 - 200x_2^2x_1^2 + 24x_2^2 + 96x_1^2 \le 16) \wedge (48x_2^4x_1^4 - 64x_2^4x_1^2 + 19x_2^4 + 56x_2^2x_1^2 - 32x_2^2 \le$$

$$-16 \vee -96x_2^4x_1^6 + 176x_2^4x_1^4 - 90x_2^4x_1^2 - 208x_2^2x_1^4 + 9x_2^4 + 200x_2^2x_1^2 - 24x_2^2 - 96x_1^2 \leq -16) \wedge (-6x_2^2x_1^2 + 3x_2^2 < 4 \vee -24x_2^4x_1^4 + 24x_2^4x_1^2 - 3x_2^4 - 40x_2^2x_1^2 + 8x_2^2 < 16 \vee 9x_2^8 - 48x_2^6x_1^2 + 576x_2^4x_1^4 - 48x_2^6 - 384x_2^4x_1^2 + 160x_2^4 + 768x_2^2x_1^2 - 256x_2^2 + 256 \neq 0) \wedge (6x_2^2x_1^2 - 3x_2^2 < -4 \vee 24x_2^4x_1^4 - 24x_2^4x_1^2 + 3x_2^4 + 40x_2^2x_1^2 - 8x_2^2 < -16 \vee 9x_2^8 - 48x_2^6x_1^2 + 576x_2^4x_1^4 - 48x_2^6 - 384x_2^4x_1^2 + 160x_2^4 + 768x_2^2x_1^2 - 256x_2^2 + 256 \neq 0) \wedge (-64512x_1^6 + 237x_2^4 + 1528x_2^2x_1^2 + 52992x_1^4 - 632x_2^2 - 24192x_1^2 + 1264 \neq 0 \vee 1344x_2^2x_1^4 + 57x_2^4 - 1640x_2^2x_1^2 + 768x_1^4 - 152x_2^2 + 2688x_1^2 + 304 \neq 0 \vee 24x_2^4x_1^2 + 9x_2^4 - 296x_2^2x_1^2 + 768x_1^4 - 24x_2^2 + 256x_1^2 + 48 \neq 0 \vee 63x_2^6 - 267x_2^4 - 2504x_2^2x_1^2 + 9984x_1^4 + 600x_2^2 + 1792x_1^2 - 528 \neq 0) \wedge x_1 < 0 \wedge -x_2 < 0 \wedge x_2^2 \leq 2 \wedge -x_2^2x_1^2 + x_2^2 \leq 2 \wedge 96x_2^8x_1^4 - 176x_2^8x_1^2 - 384x_2^6x_1^4 + 81x_2^8 + 816x_2^6x_1^2 + 576x_2^4x_1^4 - 432x_2^6 - 1408x_2^4x_1^2 + 864x_2^4 + 768x_2^2x_1^2 - 76x_2^2 + 256 = 0)$$

論理式 3

元の論理式

$$\exists x_1 \exists x_2 (-x_2 < 0 \wedge x_1 < -1 \wedge x_1^2 + x_2^2 \leq 1 \wedge -x_1^2 - x_2^2 + x_1 \leq 0 \wedge -x_1^2 - x_2^2 + 2x_1 + 2x_2 \leq 1))$$

拡張 White-Box

$$\exists x_1 \exists x_2 (0 < x_2 \wedge x_1 < -1 \wedge x_1^2 + x_2^2 \leq 1 \wedge 0 \leq x_1^2 + x_2^2 - 2x_1 - 2x_2 + 1)$$

Redlog

$$\exists x_1 \exists x_2 (x_2 > 0 \wedge x_1 + 1 < 0 \wedge x_1^2 + x_2^2 - 1 \leq 0 \wedge x_1^2 - 2x_1 + x_2^2 - 2x_2 + 1 \geq 0)$$

QEPCAD

false

SyNRAC

変化無し

論理式 4

元の論理式

$$\exists x_2 (x_2 < 0 \wedge -x_1 + x_2 < 0 \wedge -2x_1x_2 + x_2^2 \leq -1 \wedge -x_1x_2 + x_2^2 < -1 \wedge x_1^2 - x_1x_2 + x_2^2 < -1 \wedge 2x_1^2 - 2x_1x_2 + 3x_2^2 \leq -3 \wedge 9x_1^8 - 36x_1^7x_2 + 66x_1^6x_2^2 - 72x_1^5x_2^3 + 75x_1^4x_2^4 - 72x_1^3x_2^5 + 66x_1^2x_2^6 - 36x_1x_2^7 + 9x_2^8 + 6x_1^6 - 24x_1^5x_2 + 54x_1^4x_2^2 - 96x_1^3x_2^3 + 138x_1^2x_2^4 - 108x_1x_2^5 + 36x_2^6 - 5x_1^4 - 24x_1^3x_2 + 78x_1^2x_2^2 - 108x_1x_2^3 + 54x_2^4 + 6x_1^2 - 36x_1x_2 + 36x_2^2 < -9))$$

拡張 White-Box

$$\exists x_2 (x_2 < 0 \wedge 0 < x_1 - x_2 \wedge 2x_1^2 - 2x_1x_2 + 3x_2^2 \leq -3 \wedge 9x_1^8 - 36x_1^7x_2 + 66x_1^6x_2^2 - 72x_1^5x_2^3 + 75x_1^4x_2^4 - 72x_1^3x_2^5 + 66x_1^2x_2^6 - 36x_1x_2^7 + 9x_2^8 + 6x_1^6 - 24x_1^5x_2 + 54x_1^4x_2^2 - 96x_1^3x_2^3 + 138x_1^2x_2^4 - 108x_1x_2^5 + 36x_2^6 - 5x_1^4 - 24x_1^3x_2 + 78x_1^2x_2^2 - 108x_1x_2^3 + 54x_2^4 + 6x_1^2 - 36x_1x_2 + 36x_2^2 < -9)$$

Redlog

変化無し

QEPCAD

false

SyNRAC

変化無し

論理式 5

元の論理式

$$\exists x_2 (x_2 < -2 \wedge x_2^2 - x_1 \leq 0 \wedge -4x_2^2 + 3x_1 - 2x_2 < 0 \wedge -x_2^2 + x_1 - 4x_2 < -8 \wedge 5x_2^2 - 4x_1 + 4x_2 \leq -4)$$

拡張 White-Box

false

Redlog

変化無し

QEPCAD

false

SyNRAC

変化無し

論理式 6

元の論理式

$$\exists x_4(-x_4 < 0 \wedge x_2^2x_3^2 + 2x_1x_2x_3 + x_2^2x_4 + x_1^2 + 2x_1x_4 \neq 0 \wedge x_2^2x_3^4 + 2x_1x_2x_3^3 + x_2^2x_3^2x_4 + x_1^2x_3^2 + 4x_1x_2x_3x_4 - 2x_1x_3^2x_4 + 2x_1^2x_4 < 0 \wedge x_2^2x_3^2 - 4x_2x_3^3 + 4x_3^4 + 2x_1x_2x_3 - 4x_1x_3^2 + x_2^2x_4 - 4x_2x_3x_4 + 4x_3^2x_4 + x_1^2 - 2x_1x_4 \neq 0 \wedge -x_2^2x_3^4 + 2x_2x_3^5 - x_3^6 - 2x_1x_2x_3^3 + 2x_1x_3^4 - x_2^2x_3^2x_4 + 2x_2x_3^3x_4 - x_3^4x_4 - x_1^2x_3^2 - 4x_1x_2x_3x_4 + 4x_1x_3^2x_4 - 2x_1^2x_4 < 0 \wedge x_2^2x_3^4 - 2x_2x_3^5 + 2x_3^6 + 2x_1x_2x_3^3 - 2x_1x_3^4 + x_2^2x_3^2x_4 - 2x_2x_3^3x_4 + 2x_3^4x_4 + x_1^2x_3^2 + 4x_1x_2x_3x_4 - 4x_1x_3^2x_4 + 2x_1^2x_4 < 0 \wedge -x_2^4x_3^4 + 4x_2^3x_3^5 - 8x_2^2x_3^6 + 8x_2x_3^7 - 4x_3^8 - 4x_1x_2^3x_3^3 + 12x_1x_2^2x_3^4 - 16x_1x_2x_3^5 + 8x_1x_3^6 - 2x_2^4x_3^2x_4 + 8x_2^3x_3^3x_4 - 16x_2^2x_3^4x_4 + 16x_2x_3^5x_4 - 8x_3^6x_4 - 6x_1^2x_2^2x_3^2 + 12x_1^2x_2x_3^3 - 8x_1^2x_3^4 - 4x_1x_2^3x_3x_4 + 12x_1x_2^2x_3^2x_4 - 24x_1x_2x_3^3x_4 + 16x_1x_3^4x_4 - x_2^4x_4^2 + 4x_2^3x_3x_4^2 - 8x_2^2x_3^2x_4^2 + 8x_2x_3^3x_4^2 - 4x_3^4x_4^2 - 4x_1^3x_2x_3 + 4x_1^3x_3^2 - 2x_1^2x_2^2x_4 + 4x_1^2x_2x_3x_4 - 8x_1^2x_3^2x_4 - 8x_1x_2x_3x_4^2 + 8x_1x_3^2x_4^2 - x_1^4 - 4x_1^2x_4^2 \leq 0)$$

拡張 White-Box

false

Redlog

変化無し

QEPCAD

false

SyNRAC

変化無し

論理式 7

元の論理式

$$\exists x_2((3x_2^2 + 7x_1 \neq 0 \vee 15x_2^2 - x_1 \neq 0) \wedge (15x_2^2 - x_1 \neq 0 \vee 33x_2^2 + 5x_1 \neq 0) \wedge (7x_2^2 - 16x_2 - 5x_1 < 0 \vee -25x_2^2 + 80x_2 + 15x_1 < 64) \wedge (7x_2^2 - 16x_2 - 5x_1 < 0 \vee 15x_2^3 - 24x_2^2 - 5x_2x_1 - 40x_1 < 0) \wedge (-5x_2^2 + 10x_2 + 3x_1 < 5 \vee -9x_2^6 + 9x_2^5 - 105x_2^4x_1 + 150x_2^3x_1 + 25x_2^2x_1^2 + 125x_2x_1^2 + 25x_1^3 < 0) \wedge (5x_2^2 - 10x_2 - 3x_1 < -5 \vee -198x_2^5 + 225x_2^4 - 300x_2^3x_1 + 750x_2^2x_1 + 250x_2x_1^2 + 125x_1^2 < 0) \wedge (25x_2^2 - 80x_2 - 15x_1 < -64 \vee 15x_2^3 - 24x_2^2 - 5x_2x_1 - 40x_1 < 0) \wedge (-198x_2^5 + 225x_2^4 - 300x_2^3x_1 + 750x_2^2x_1 + 250x_2x_1^2 + 125x_1^2 < 0 \vee -9x_2^6 + 9x_2^5 - 105x_2^4x_1 + 150x_2^3x_1 + 25x_2^2x_1^2 + 125x_2x_1^2 + 25x_1^3 < 0) \wedge (-25x_2^4 + 100x_2^3 + 30x_2^2x_1 - 180x_2^2 - 60x_2x_1 - 9x_1^2 + 160x_2 + 12x_1 < 64 \vee -108x_2^7 + 243x_2^6 - 156x_2^5x_1 - 216x_2^5 + 825x_2^4x_1 + 300x_2^3x_1^2 - 1200x_2^3x_1 - 375x_2^2x_1^2 - 100x_2x_1^3 - 600x_2x_1^2 - 125x_1^3 < 0) \wedge (25x_2^4 - 100x_2^3 - 30x_2^2x_1 + 180x_2^2 + 60x_2x_1 + 9x_1^2 - 160x_2 - 12x_1 < -64 \vee -135x_2^8 + 270x_2^7 - 1836x_2^6x_1 - 216x_2^6 + 5130x_2^5x_1 + 1350x_2^4x_1^2 - 5400x_2^4x_1 + 2250x_2^3x_1^2 + 500x_2^2x_1^3 - 9000x_2^2x_1^2 - 3250x_2x_1^3 - 375x_1^4 - 1000x_1^3 < 0) \wedge (-108x_2^7 + 243x_2^6 - 156x_2^5x_1 - 216x_2^5 + 825x_2^4x_1 + 300x_2^3x_1^2 - 1200x_2^3x_1 - 375x_2^2x_1^2 - 100x_2x_1^3 - 600x_2x_1^2 - 125x_1^3 < 0 \vee -135x_2^8 + 270x_2^7 - 1836x_2^6x_1 - 216x_2^6 + 5130x_2^5x_1 + 1350x_2^4x_1^2 - 5400x_2^4x_1 + 2250x_2^3x_1^2 + 500x_2^2x_1^3 - 9000x_2^2x_1^2 - 3250x_2x_1^3 - 375x_1^4 - 1000x_1^3 < 0) \wedge (-33x_2^2 - 5x_1 < 0 \vee 3x_2^2 + 7x_1 < 0 \vee 15x_2^2 - x_1 \neq 0) \wedge (-3x_2^2 - 7x_1 < 0 \vee 15x_2^2 - x_1 \neq 0 \vee 33x_2^2 + 5x_1 < 0) \wedge -x_2 < 0 \wedge -5x_2^2 + 3x_1 < 0 \wedge -3x_2^2 - 5x_1 < 0 \wedge x_2^2 - x_1 < 0 \wedge 3x_2^2 - 5x_1 < 0))$$

拡張 White-Box

$$\exists x_2(0 < x_2 \wedge 0 < 5x_2^2 - 3x_1 \wedge x_2^2 - x_1 < 0 \wedge (7x_2^2 - 16x_2 - 5x_1 < 0 \vee 0 < 25x_2^2 - 80x_2 - 15x_1 + 64) \wedge (7x_2^2 - 16x_2 - 5x_1 < 0 \vee 15x_2^3 - 24x_2^2 - 5x_2x_1 - 40x_1 < 0) \wedge (0 < 5x_2^2 - 10x_2 - 3x_1 + 5 \vee 0 < 9x_2^6 - 9x_2^5 + 105x_2^4x_1 - 150x_2^3x_1 - 25x_2^2x_1^2 - 125x_2x_1^2 - 25x_1^3) \wedge (5x_2^2 - 10x_2 - 3x_1 < -5 \vee 0 < 198x_2^5 - 225x_2^4 + 300x_2^3x_1 - 750x_2^2x_1 - 250x_2x_1^2 - 125x_1^2) \wedge (25x_2^2 - 80x_2 - 15x_1 < -64 \vee 15x_2^3 - 24x_2^2 - 5x_2x_1 - 40x_1 < 0) \wedge (0 < 198x_2^5 - 225x_2^4 + 300x_2^3x_1 - 750x_2^2x_1 - 250x_2x_1^2 - 125x_1^2 \vee 0 < 9x_2^6 - 9x_2^5 + 105x_2^4x_1 - 150x_2^3x_1 - 25x_2^2x_1^2 - 125x_2x_1^2 - 25x_1^3) \wedge (0 < 25x_2^4 - 100x_2^3 - 30x_2^2x_1 + 180x_2^2 + 60x_2x_1 + 9x_1^2 - 160x_2 -$$

$$12x_1+64\vee 0<108x_2^7-243x_2^6+156x_2^5x_1+216x_2^5-825x_2^4x_1-300x_2^3x_1^2+1200x_2^3x_1+375x_2^2x_1^2+100x_2x_1^3+600x_2x_1^2+125x_1^3)\wedge(25x_2^4-100x_2^3-30x_2^2x_1+180x_2^2+60x_2x_1+9x_1^2-160x_2-12x_1<-64\vee 0<135x_2^8-270x_2^7+1836x_2^6x_1+216x_2^6-5130x_2^5x_1-1350x_2^4x_1^2+5400x_2^4x_1-2250x_2^3x_1^2-500x_2^2x_1^3+9000x_2^2x_1^2+3250x_2x_1^3+375x_1^4+1000x_1^3)\wedge(0<108x_2^7-243x_2^6+156x_2^5x_1+216x_2^5-825x_2^4x_1-300x_2^3x_1^2+1200x_2^3x_1+375x_2^2x_1^2+100x_2x_1^3+600x_2x_1^2+125x_1^3\vee 0<135x_2^8-270x_2^7+1836x_2^6x_1+216x_2^6-5130x_2^5x_1-1350x_2^4x_1^2+5400x_2^4x_1-2250x_2^3x_1^2-500x_2^2x_1^3+9000x_2^2x_1^2+3250x_2x_1^3+375x_1^4+1000x_1^3))$$

Redlog

変化無し

QEPCAD

$$10x_2-13>0\wedge 3x_1-5x_2^2+10x_2-5<0\wedge 15x_1-25x_2+80x_2-64>0\wedge x_1-x_2^2>0$$

SyNRAC

$$\exists x_2((x_1\neq 0\vee x_2^2\neq 0)\wedge(x_1\neq 0\vee 15x_2^2-x_1\neq 0)\wedge(7x_2^2-16x_2-5x_1<0\vee -25x_2^2+80x_2+15x_1<64)\wedge(7x_2^2-16x_2-5x_1<0\vee 15x_2^3-24x_2^2-5x_2x_1-40x_1<0)\wedge(-5x_2^2+10x_2+3x_1<5\vee -9x_2^6+9x_2^5-105x_2^4x_1+150x_2^3x_1+25x_2^2x_1^2+125x_2x_1^2+25x_1^3<0)\wedge(5x_2^2-10x_2-3x_1<-5\vee -198x_2^5+225x_2^4-300x_2^3x_1+750x_2^2x_1+250x_2x_1^2+125x_1^2<0)\wedge(25x_2^2-80x_2-15x_1<-64\vee 15x_2^3-24x_2^2-5x_2x_1-40x_1<0)\wedge(-198x_2^5+225x_2^4-300x_2^3x_1+750x_2^2x_1+250x_2x_1^2+125x_1^2<0\vee -9x_2^6+9x_2^5-105x_2^4x_1+150x_2^3x_1+25x_2^2x_1^2+125x_2x_1^2+25x_1^3<0)\wedge(-25x_2^4+100x_2^3+30x_2^2x_1-180x_2^2-60x_2x_1-9x_1^2+160x_2+12x_1<64\vee -108x_2^7+243x_2^6-156x_2^5x_1-216x_2^5+825x_2^4x_1+300x_2^3x_1^2-1200x_2^3x_1-375x_2^2x_1^2-100x_2x_1^3-600x_2x_1^2-125x_1^3<0)\wedge(25x_2^4-100x_2^3-30x_2^2x_1+180x_2^2+60x_2x_1+9x_1^2-160x_2-12x_1<-64\vee -135x_2^8+270x_2^7-1836x_2^6x_1-216x_2^6+5130x_2^5x_1+1350x_2^4x_1^2-5400x_2^4x_1+2250x_2^3x_1^2+500x_2^2x_1^3-9000x_2^2x_1^2-3250x_2x_1^3-375x_1^4-1000x_1^3<0)\wedge(-108x_2^7+243x_2^6-156x_2^5x_1-216x_2^5+825x_2^4x_1+300x_2^3x_1^2-1200x_2^3x_1-375x_2^2x_1^2-100x_2x_1^3-600x_2x_1^2-125x_1^3<0\vee -135x_2^8+270x_2^7-1836x_2^6x_1-216x_2^6+5130x_2^5x_1+1350x_2^4x_1^2-5400x_2^4x_1+2250x_2^3x_1^2+500x_2^2x_1^3-9000x_2^2x_1^2-3250x_2x_1^3-375x_1^4-1000x_1^3<0)\wedge -x_2<0\wedge -5x_2^2+3x_1<0\wedge -3x_2^2-5x_1<0\wedge x_2^2-x_1<0\wedge 3x_2^2-5x_1<0)$$

論理式 8

元の論理式

$$\exists x_1(-x_1<0\wedge x_1\leq 1\wedge x_2^2+x_1<1\wedge 4x_2^3+x_2^2-4x_2+x_1<1\wedge 16x_2^4+8x_2^3-16x_2^2x_1-15x_2^2+16x_1^2-8x_2+x_1-1\neq 0)$$

拡張 White-Box

$$\exists x_1(0<x_1\wedge x_2^2+x_1<1\wedge 4x_2^3+x_2^2-4x_2+x_1<1\wedge 16x_2^4+8x_2^3-16x_2^2x_1-15x_2^2+16x_1^2-8x_2+x_1-1\neq 0)$$

Redlog

変化無し

QEPCAD

$$\exists x_1(0<x_1\wedge x_2^2+x_1<1\wedge 4x_2^3+x_2^2-4x_2+x_1<1\wedge 16x_2^4+8x_2^3-16x_2^2x_1-15x_2^2+16x_1^2-8x_2+x_1-1\neq 0)$$

SyNRAC

変化無し

論理式 9

元の論理式

$$x_1>0\wedge\cdots\wedge x_{30}>0\wedge x_1+\cdots+x_{30}>0$$

拡張 White-Box

$x_1>0\wedge\cdots\wedge x_{30}>0$ (実行時間 0.080 秒)

Redlog

$x_1>0\wedge\cdots\wedge x_{30}>0$ (実行時間 0.001 秒)

QEPCAD

time out (実行時間 3600 秒 以上)

SyNRAC

変化無し (実行時間 0.010 秒)

参考文献

[1] 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, 2011.

[2] C. W. Brown. QEPCAD B: A program for computing with semi-algebraic sets using CADs. *ACM SIGSAM Bulletin*, **37**(4), pp. 97–108, 2003.

[3] C. W. Brown. Fast simplifications for Tarski formulas based on monomial inequalities. *J. Symbolic Comput*, **47**, pp. 859–882, 2012.

[4] C. W. Brown and A. Strzeboński. Black-Box/White-Box Simplification and Applications to Quantifier Elimination. *Proc. ISSAC*, ACM, pp. 69–76, 2010.

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

[6] D. Cox, J, Little, D. O'Shea. Ideals, Varieties, and Algorithms—An Introduction to Computational Algebraic Geometry and Commutative Algebra (Third Edition), Springer, 2007.

[7] A. Dolzmann, T. Sturm. Redlog [Computer software], `http://redlog.dolzmann.de/`, 2016/01/14 アクセス.

[8] 富士通研究所. SyNRAC [Computer software], `http://www.fujitsu.com/jp/group/labs/resources/tech/freeware/synrac/`, 2015/12/04 アクセス.

[9] Hoon Hong, C. W. Brown. QEPCAD B [Computer software], `http://www.usna.edu/CS/qepcadweb/B/QEPCAD.html`, 2016/01/14 アクセス.

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

[11] Maple, a division of Waterloo Maple Inc. Maple 18 [Computer software], Waterloo, Ontario, Canada.

[12] T. Strum and A. Dolzmann. Simplification of quantifier-free formulae over ordered fields, *J. Symbolic Comput*, **24**, pp. 209–231, 1997.