ブーリアン・グレブナー基底による集合制約解消の効率性について

On the Efficiency of Set Constraint Solving via Boolean Gröbner Bases

京都大学理学研究科 数学教室　　西村 進 [*1]
Susumu Nishimura
Department of Mathematics, Kyoto University

Abstract

Boolean Gröbner bases can solve set constraints, by converting them into Boolean polynomials with singleton set parameters and finding singleton solutions. This paper proposes a method to improve the efficiency of this process. The proposed method decomposes the input set of polynomials into a direct product of $\mathbb{GF}_2$ boolean polynomials by partitioning them according to equivalence classes derived from the coefficients of the input polynomials. This reduces the computation of redundant Boolean Gröbner bases and narrow down the search space for finding singleton solutions, thereby achieving an improved efficiency.

Using this improved procedure for constraint solving, we implemented a topological consistency checker for distributed programs written with set pattern matching. In addition to this implementation, we also implemented another consistency checker that uses SAT solver instead of Gröbner bases and compared their performance. The results show that the consistency checker that uses Gröbner bases as the set constraint solver outperforms the one that uses the SAT solver in memory efficiency, particularly for larger target programs, and provides more favorable results in terms of execution time.

1　はじめに

ブール多項式集合，すなわちブール環の要素を係数とする多項式集合の解は，そのブーリアン・グレブナー基底を計算することによって得ることができる．[11] 特に，冪集合のブール環を考えることで，集合制約をこれと等価なブール多項式集合に変換し，これの可解性を計算機によって系統的に判定することができる．[12] この集合制約解消手法は数独パズルの求解 [11, 10] や並行分散プログラムの整合性検査 [13] に適用されてきた．しかしながら，本著者が先行研究として行った [13] の計算効率は十分に満足できるものではなかった．

本稿では，ブーリアン・グレブナー基底を用いた集合制約解消の効率改善手法について議論し，その手法の適用による効率改善の効果について報告する．並行分散プログラムの整合性検査で扱う集合制約解消手続は，個々の集合制約が生成する多項式集合のサイズは大きくない代わりに，多数の集合制約を処理する必要がある，という特性を持つ．そのため，ブーリアン・グレブナー基底計算アルゴリズムの改善方向としては，入力サイズが巨大なものや基底計算が極端に難しいものではなく，サイズが比較的小さく，基底計算もそれほど難しくない多項式集合の多項式の処理を平均的に効率化することが主眼となる．

[*1] 〒 606–8502 京都市左京区北白川追分町　E-mail: susumu@math.kyoto-u.ac.jp

本著者による先行研究 [13] では，整合性検査器が重複して生成する集合制約の削除や，単純な集合制約に対するグレブナー基底計算の回避などの工夫により一定の最適化を行なっていたが，グレブナー基底の計算は冪集合のブール環を係数とする多項式集合に対して行なっていたため，この部分がボトルネックとなり，十分な実行効率が得られていなかった．このボトルネックに関しては，後述する先行研究 [2, 9] による手法が有効であることを [14] で確認している．

本稿では，集合制約の生成は先行研究 [13, 14] と同様のままで整合性検査器の実行効率を改善するため，更なるブーリアン・グレブナー基底計算手続きの改良を提案する．具体的には，ブール多項式に含まれる係数から導かれる同値類によって，既存手法より粗く解空間を分割することで基底計算による計算を削減する方法を提案し，この改良された基底計算アルゴリズムによる集合制約解消を実装する．さらに，別の集合制約解消手法として SAT ソルバを用いた検査器も実装し，２つの手法の実行効率を比較する．実装実験の結果，特に対象となるプログラムのサイズが大きい場合において，ブーリアン・グレブナー基底計算による制約解消が SAT ソルバによるものより明らかな優位を示したことを報告する．これらふたつの異なる手法による実行効率の差異の要因について，実行ベンチマークから推察される所見を述べる．

2　組合せトポロジー論に基づく集合パターンマッチによる並行分散プログラミング

本節では，集合パターンマッチを用いた並行分散プログラミングと，パターンマッチを用いて記述された並行分散プログラムの整合性検査のおおまかな概要について簡潔に述べる．

並行分散計算の計算可能性について，単体的複体を用いた組合せトポロジーモデル [5] を用いた議論が行われてきた．[13] において本著者は，この枠組みを用いて並行分散プログラムを集合パターンマッチを用いて記述し，プログラムの整合性を集合制約の解消によって検査する手法を提案した．

集合パターンマッチによる並行分散プログラミングは，Herlihy と Shavit による Asynchronous Computability Theorem (ACT)[6] を土台としている．ACT は，wait-free な Read-Write 共有分散メモリ並行分散システムで実現可能な計算は，単体的複体の標準色付き細分と単体写像の合成で必ず表現されることを保証する，並行分散計算の組合せトポロジー論における重要定理である．

例として 2 プロセスの 2 値 quasi-consensus 問題のプログラミングを考えよう．quasi-consensus 問題とは，ふたつのプロセスがそれぞれ T,F いずれかの値を入力とするとき，計算結果として両プロセスとも同一の値 (T もしくは F) を出力とするか，そうでなければ予め定められた一方のプロセスが T，もう一方が F を出力 (逆は認めない) を出力とするような分散プログラムを与えよ，というものである．

この問題を単体的複体の写像として表現すると以下の様に，1 次元単体，すなわちグラフ構造の二段階の変形として図示できる．

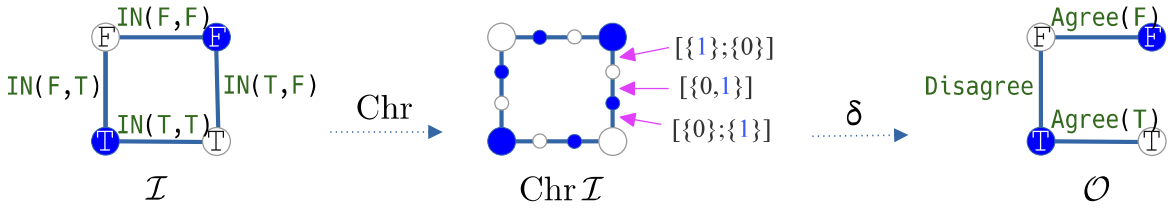

図 1: 単体的複体の変形としての 2 プロセスの quasi-consensus 問題

入力複体 $\mathcal{I}$ は，4 通りの可能性 (2 つのプロセスが独立に 2 種類の値を入力とする) を表す四辺形である．(なお，異なる色のグラフの頂点で異なる 2 つのプロセスを区別する．) この入力は適切な並行分散アルゴ

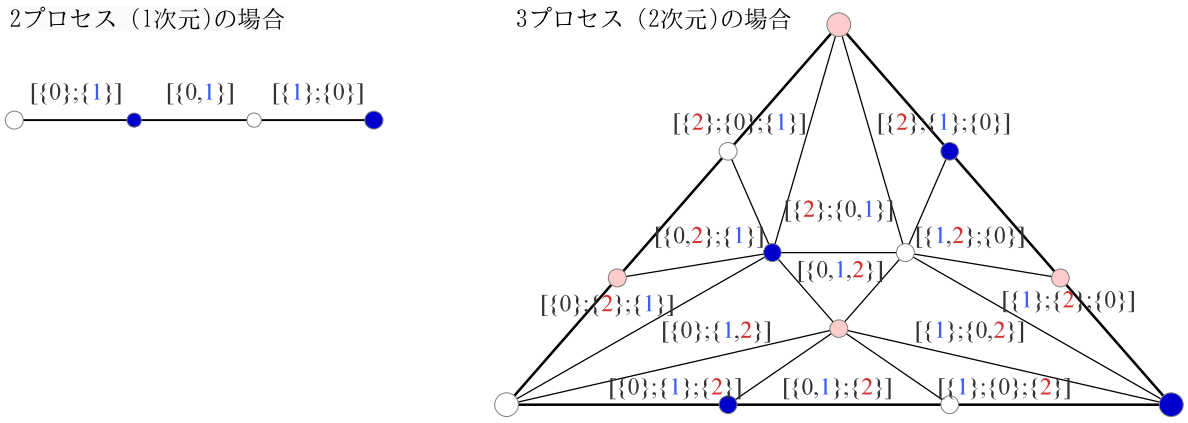

図 2: 標準色付き細分の例

リズムによってより細かな複体に細分することができる．この細分を標準色付き細分 [6] といい $\mathrm{Chr}\,\mathcal{I}$ で表す．この標準色付き細分に対して，頂点の色を保存する単体写像 δ を与えることで並行分散計算がひとつ定まる．いま quasi-consensus 問題をプログラミングするには，図 1 に示すように，標準色付き細分 δ によって 12 本のグラフ辺を適切な出力複体 $\mathcal{O}$ に対応付ければよい．この場合 $\mathcal{O}$ は，3 辺からなる逆コの字型のグラフである．

このような quasi-consensus 問題を定める単体写像 δ を与えるには，標準色付き細分から出力複体への辺の対応を，プログラム上で記号を用いて曖昧性なく記述する必要がある．このために，標準色付き細分の持つ組合せ構造を利用する．相異なるプロセス識別子の集合 $0,\ldots,n$ で区別される $n+1$ 並行分散プロセスによって構成される n 次元単体は，標準色付き細分によってより細かなファセット (n 次元単体) に分割されるが，これらは $\{0,\ldots,n\}$ の順序付き集合分割 (ordered set partition) $[S_1;\ldots;S_r]$ と 1 対 1 の対応関係を持つことが知られている．[8] ここで，順序付き集合分割 $[S_1;\ldots;S_r]$ とは，$\bigcup_{i=1}^{r} S_i = \{0,\ldots,n\}$ をみたすような $\{0,\ldots,n\}$ の互いに素な空でない部分集合の列 $S_1,\ldots,S_r$(但し，並び順を区別する) のことである．

例えば, 2 プロセスの場合，1 次元単体の標準色付き細分に含まれるファセットは，図 2 左図に示すように $\{0,1\}$ の 3 つの順序付き集合分割 $[\{0\};\{1\}]$, $[\{0,1\}]$, $[\{1\};\{0\}]$ にする．この標準色付き細分と順序付き集合分割の関係はより高次元でも成り立ち，たとえば 3 プロセスの場合，図 2 右図に示すように 2 次元単体は標準色付き細分によって 13 個のファセットに分割され，これらはそれぞれ順序付き集合分割によって一意に特定される．

標準色付き細分の順序付き集合分割との対応関係を利用して，順序付き集合分割に対するパターンマッチで単体写像を記号的に記述することで並行分散計算を定義できる．例えば，quasi-consensus 問題を定める単体写像 δ は，集合のパターンマッチ構文を用いて図 3 のようにプログラミングできる．

プログラム 2 行目のパターン節は，図 1 で示した入力複体 $\mathcal{I}$ の辺 `IN(F,F)`(四辺形の上辺)`Agree(F)` の細分によるファセットのうちパターン `[_]` に合致するファセットを出力複体 $\mathcal{O}$ のファセット `Agree(F)`(逆コの字の上辺) に写す．パターン `[_]` は辺 `IN(F,F)` の任意の順序付き集合分割，すなわち任意の細分に合致するので，`IN(F,F)` の 3 つの細分を全てすべて同じ辺 `Agree(F)` に写す．プログラム 5–7 行のパターン節は，$\mathcal{I}$ の辺 `IN(T,F)`(四辺形の右縦辺) の 3 つの細分を上から順にそれぞれ $\mathcal{O}$ の上辺 `Agree(F)`, 縦辺 `Disagree`, 下辺 `Agree(T)` に写す．

このように順序付き集合分割に対するパターンマッチを用いることで，プログラムを記号的に記述することができるが，これが並行分散プログラムとして有効であるためには，パターンで定義した単体写像 δ が幾何的整合性を満たしていることを検査しなければならない．すなわち，単体写像 δ によって幾何的に不

```
1: let δ = function
2:   | IN(F,F), [_] -> Agree(F)
3:   | IN(F,T), [_] -> Disagree
4:   | IN(T,T), [_] -> Agree(T)
5:   | IN(T,F), [{1};{0}] -> Agree(F)
6:   | IN(T,F), [{0,1}] -> Disagree
7:   | IN(T,F), [{0};{1}] -> Agree(T)
```

図 3: quasi-consensus 問題の集合パターンマッチによるプログラム記述

連続な変形が行われていないことを保証しなければならない．このような幾何的整合性の検査は，プログラムで与えられた集合パターンマッチ記述から集合制約を生成し，これの充足可能性を検査することによって達成することができる．

以下，集合パターンマッチ記述から集合制約の生成方法については本稿の主題ではないので省略し，検査すべき集合制約は所与のものであるとする．(集合制約生成の詳細については，先行論文 [13, 14] を参照されたい．)

3 ブーリアン・グレブナー基底による集合制約解消アルゴリズムとその改善

本節では，集合制約の充足可能性の，ブーリアン・グレブナー基底計算による解消手続きとその改良について議論する．

$(\mathbf{B}, +, \times, \mathbf{0}, \mathbf{1})$ を，$\mathbf{B}$ を台集合とするブール環とする．ブール環は乗法に関して冪等 $(SS = S)$ な可換環であり，$S + S = \mathbf{0}$ も成り立つ．本稿で扱うブール環は以下の 2 種類である．

- 冪集合環 $\mathbf{B} = \mathcal{P}(\{0, \ldots, n\})$
  乗法を集合積 $S \times T = S \cap T$, 加法を差積 $S + T = (S \setminus T) \cup (T \setminus S)$, 零元 $\mathbf{0}$ を空集合 $\mathbf{0} = \emptyset$, 単位元 $\mathbf{1}$ を全集合 $\{0, \ldots, n\}$ とするブール環.

- 2 元体 $\mathbb{GF}_2 = \{\mathbf{1}, \mathbf{0}\}$
  零元 $\mathbf{0}$ を偽, 単位元 $\mathbf{1}$ を真, 乗法を論理積 $S \times T = S \wedge T$, 加法を論理和 $S + T = S \vee T$ するブール環.

一般にブール環 $\mathbf{B}$ に係数を持つ変数 $\vec{X} = X_1, \ldots, X_m$ の多項式環 $\mathbf{B}[\vec{X}]$ について，その有限生成イデアル $\langle X_1^2 + X_1, \ldots, X_m^2 + X_m \rangle$ による剰余環はまたブール環である．これを**ブール多項式環**といい $\mathbf{B}(\vec{X})$ で表す.

イデアル $I \subseteq \mathbf{B}(\vec{X})$ について，任意の $f(\vec{X}) \in I$ について $f(\vec{b}) = \mathbf{0}$ であるとき，イデアル I は解 $\vec{b} = (b_1, \ldots, b_m)$ を持つという．一般に，有限生成イデアル I の可解性は，ブーリアン・グレブナー基底 G の消去イデアルについて $G \cap \mathbf{B} = \{\mathbf{0}\}$ が成り立つかどうかで判定できる．(ブーリアン・グレブナー基底の零点定理 [11])

3.1 冪集合環ブール多項式による集合制約表現

ブール環として冪集合環 $\mathbf{B} = \mathcal{P}(\{0, \ldots, n\})$ を考える．

$\vec{A} = A_1, \ldots, A_l$ を $\vec{X}$ と異なる，単元集合の上を動くパラメータとする．$\mathcal{P}(\{0, \ldots, n\})(\vec{A})(\vec{X})$ はブール多項式環 $\mathcal{P}(\{0, \ldots, n\})(\vec{A})$ に係数を持つ (変数 $\vec{X}$ の) ブール多項式環である．$\mathcal{P}(\{0, \ldots, n\})(\vec{A})(\vec{X})$ のイデアル I が (パラメータ $\vec{A}$ について) 単元集合解を持つとは

$$\forall f(\vec{X}, \vec{A}) \in I.\, f(\vec{S}, \vec{a}) = \mathbf{0}$$

をみたすような $S_1, \ldots, S_m \in \mathcal{P}(\{0, \ldots, n\})$ および $a_1, \ldots, a_l \in \{\{0\}, \ldots, \{n\}\}$ が存在することをいう．

集合に関する様々な制約を $\mathcal{P}(\{0, \ldots, n\})(\vec{A})(\vec{X})$ の多項式で表現することができる．[12, 11] 以下は，単元集合パラメータを含まないブール多項式の等式 $f(\vec{X}) = 0$ で表現可能な集合制約の例である．

(等値 $S = T$)	$S + T = \mathbf{0}$
(所属 $a \in S$)	$\{a\}S + \{a\} = \mathbf{0}$
(非所属 $a \notin S$)	$\{a\}S = \mathbf{0}$
(非交和 $S \cap T = \emptyset$)	$ST = \mathbf{0}$
(包含 $S \subseteq T$)	$ST + S = \mathbf{0}$

さらに，これらの制約の否定を単元集合パラメータ用いた多項式で表すこともできる．$f \neq \mathbf{0}$ と等価な多項式の等式を，fresh な単元集合パラメータ A を用いて $Af + A = \mathbf{0}$ で表すことができる．(例えば非空制約 $S \neq \emptyset$ を多項式で表すと $AS + A = \mathbf{0}$ となる．)

以上より，$\mathcal{P}(\{0, \ldots, n\})(\vec{A})(\vec{X})$ の多項式集合を用いることで，

$$\bigwedge_{j \in J} f_j(\vec{X}) = \mathbf{0} \wedge \bigwedge_{k \in K} f_k(\vec{X}) \neq \mathbf{0}$$

のように集合制約を表現することができる．この形で表される集合制約は，多項式で表される集合制約およびその否定の論理積に限定されており，一般の論理和を表すことはできない．

上記のような集合制約の充足可能性はこれと対応する以下の多項式集合

$$\{f_j(\vec{X}) \mid j \in J\} \cup \{A_k f_k(\vec{X}) + A_k \mid k \in K, A_k\text{: fresh}\}$$

の有限生成イデアルの多項式の可解性に帰着することで判定する．以下に詳述するように，多項式の可解性判定は，ブーリアン・グレブナー基底およびその消去イデアルの計算で行うことができる．

なお，$n + 1$ プロセスの並行分散プログラムの整合性検査が集合制約解消のために生成する個々のブール多項式集合は，以下のように比較的小さなサイズ・複雑さのものになる．

- 多項式の数は高々$O(n^2)$
- 多項式の次数は高々3
- 各多項式に含まれる項の数は高々$O(n)$，ただしそのほとんどは 1 個または 2 個

ただし，ひとつひとつのブール多項式集合は小さくても，整合性検査の過程において多数の集合制約に対してブール多項式集合のグレブナー基底計算を繰り返す必要がある．したがって，1 回々々のブーリアン・グレブナー基底の計算時間の積算が全体の実行時間の大きな部分を占めることになるため，基底計算の効率改善を図ることは全体の実行効率にとって重要である．

3.2 ブール多項式環 $\mathcal{P}(\{0,\ldots,n\})(\vec{A})(\vec{X})$ 上の求解アルゴリズム

多項式集合 $F \subseteq \mathcal{P}(\{0,\ldots,n\})(\vec{A})(\vec{X})$ の解は，以下の手続き SingSolve に示すように，まず有限生成イデアル $\langle f_1,\ldots,f_s\rangle$ のグレブナー基底 G_1 を計算し，その消去イデアル $G \cap \mathcal{P}(\{0,\ldots,n\})(\vec{A})$ の単元集合解を求めることで得られる．

入力 多項式の有限集合 $F \subset \mathcal{P}(\{0,\ldots,n\})(\vec{A})(\vec{X})$
出力 単元集合解の有無 (解有りの場合は解の一例，解無しの場合は $\perp$ を返す)

```
1: procedure SingSolve(F)
2:     G_1 ← ⟨F⟩ のブーリアン・グレブナー基底
3:     G_2 ← G_1 ∩ P({0,...,n})(A⃗)
4:     if G_2 の単元集合解を単元集合解アルゴリズム [7] により発見 then
5:         return 単元集合解の一例
6:     else
7:         return ⊥
```

ここで単元集合解アルゴリズム [7] は，グレブナー基底計算によって得られる多項式集合 $G_2 \subseteq \mathcal{P}(\{0,\ldots,n\})(\vec{A})$ から以下の代数的条件により得られる almost solution を利用して単元集合解を探索するアルゴリズムである．

- 以下のいずれかの代数的条件が成り立つとき，G_2 は almost solution $A_j + \{m\}$ (すなわち A_j に関する単元集合解 $A_j = \{m\}$) を持つという．
  - $\{m\}f_i = \{m\}A_j + \{m\}$ なる $f_i \in G_2$ が存在するかもしくは
  - 任意の $m' \in \{0,\ldots,n\} \setminus \{m\}$ について，$\{m'\}f_i = \{m'\}A_j$ なる $f_i \in G_2$ $(1 \le i \le s)$ が存在する．

単元集合解アルゴリズム [7] は，上記の代数的条件をみたす almost solution $A_{i_1} + \{m_1\},\ldots,A_{i_1} + \{m_k\}$ $(k \ge 1)$ を取得し，これらをもとのグレブナー基底に加えた $G_2 \cup \{A_{i_1} + \{m_1\},\ldots,A_{i_1} + \{m_k\}\}$ に対してさらにブーリアン・グレブナー基底を計算を繰り返すことで，単元集合解を漸進的に確定させていくアルゴリズムである．ただし，almost solution が常に存在するとは限らないので，存在しない場合，解が未定の各単元集合パラメータ A_j について，適当な解 $A_j + \{m\}$ を仮定して求解を行い，解が見つからなければ別の解候補を試す網羅的探索を行う．

Inoue の単元集合解アルゴリズム [7] は，総じて軽量に単元集合解の探索を行える場合が多いものの，網羅的探索に陥る可能性があるため，常に解探索が高速に行えるわけではないことに注意が必要である．

3.3 $\mathbb{GF}_2^{n+1}$ への分解による求解アルゴリズム

上記 3.2 節で示した求解アルゴリズムは，冪集合環の係数計算に要する集合演算のコストが大きい．この問題については，冪集合環 $\mathcal{P}(\{0,\ldots,n\})$ をこれと同型な 2 元体のブール環の直積 $\mathbb{GF}_2^{n+1}$ に置き換え，2 元体を係数とする多項式のデータ構造を ZDD で処理する [2] ことで大きな効率改善が達成できることが知られている．Nagai と Inoue[9] は，この方法を上記の単元集合解アルゴリズムに適用した，効率の良い集合制約解消手法を提案している．

冪集合環から 2 元体の直積へのあきらかな同型写像 $\iota : \mathcal{P}(\{0,\ldots,n\}) \to \mathbb{GF}_2^{n+1}$ を以下で定義する．

$$\iota(S) = (b_0,\ldots,b_n), \quad \text{ただし } b_k = \mathbf{1} \Longleftrightarrow k \in S$$

ι は多項式 $f \in \sum_{i=1}^{m} S_i t_i$ ($S_1, \ldots, S_m \in \mathcal{P}(\{0, \ldots, n\})$, 各 t_i は単項式) に作用する写像として以下のように自然に拡張できる．

$$\iota(f) = \left(\sum_{i=1}^{m} \iota(\{0\}S_i)t_i, \ldots, \sum_{i=1}^{m} \iota(\{n\}S_i)t_i\right)$$

この拡張は $\mathcal{P}(\{0, \ldots, n\})(\vec{A})(\vec{X})$ から $\mathbb{GF}_2(\vec{A})(\vec{X})^{n+1}$ への同型写像である．

この同型写像により 3.2 節の求解アルゴリズムを以下のように改変することで，冪集合環ではなく，2 元体の環に関するグレブナー基底計算を用いてより効率的に集合制約を解消する手続き SingSolveGF2 が得られる．

入力 多項式の有限集合 $F \subset \mathcal{P}(\{0, \ldots, n\})(\vec{A})(\vec{X})$
出力 単元集合解の有無 (解有りの場合は解の一例，解無しの場合は $\perp$ を返す)

1: **procedure** SingSolveGF2(F)
2: $(F_0, \ldots, F_n) \leftarrow \iota(F)$
3: $(G_0, \ldots, G_n) \leftarrow F_0, \ldots, F_n$ のブーリアン・グレブナー基底の直積
4: $(H_0, \ldots, H_n) \leftarrow (G_0 \cap \mathbb{GF}_2(\vec{A}), \ldots, G_n \cap \mathbb{GF}_2(\vec{A}))$
5: **if** $(H_0, \ldots, H_n)$ の単元集合解を発見 **then**
6: **return** 単元集合解の一例
7: **else**
8: **return** $\perp$

同型写像 ι で入力した多項式集合 F を $\mathbb{GF}_2(\vec{A})(\vec{X})$ の多項式集合の直積 $(F_0, \ldots, F_n)$ の形に分解 (2 行目) し，直積の各要素についてそのグレブナー基底を [2] のアルゴリズムで計算 (3 行目) する．直積の各要素の消去イデアルを計算し (4 行目)，消去イデアルの直積 $(H_0, \ldots, H_n)$ について単元集合解を探索する．

単元集合解の探索は既出の手続き SingSolve のときと同様に，almost solution を漸進的に追加していく方法で行う．ただし，almost solution の発見と追加は，グレブナー基底の直積 $(H_0, \ldots, H_n)$ について以下のように行う．

グレブナー基底の直積 $(H_0, \ldots, H_n)$ は，以下の条件を満たすときに almost solution $A_j + \{k\}$ $(0 \leq k \leq n)$ を持つという．

- $A_j + \mathbf{1} \in H_k$ であるかもしくは
- 任意の $k' \in \{0, \ldots, n\}$ について， $k' \neq k$ ならば $A_j \in H_{k'}$.

また，網羅的探索の際に almost solution $A_j + \{p\}$ $(0 \leq p \leq n)$ を単元集合解候補として加えるには直積 $(H_0, \ldots, H_n)$ の各成分を以下のように更新すればよい．

- 第 p 成分 H_p に $A_j + \mathbf{1}$ を追加，すなわち $H_p \leftarrow H_p \cup \{A_j + \mathbf{1}\}$ で更新
- それ以外の第 p' 成分 $H_{p'}$ には A_j を追加，すなわち $H_{p'} \leftarrow H_{p'} \cup \{A_j\}$ で更新

3.4 同値類分割による求解アルゴリズムの改善

3.3 節に示した単元集合解アルゴリズムの改変版 [9] は大きく効率を改善するが，この方法には更なる改良の余地がある．例えば，多項式集合 $F = \{\{1, 2, 3\}A + Y, \{3\}Y\}$ の求解を考えよう．これの同型写像による $\mathbb{GF}_2(\vec{A})(\vec{X})^{n+1}$ への分解は $\iota(F) = (\{Y\}, \{A + Y\}, \{A + Y\}, \{A + Y, Y\}, \{Y\}, \ldots)$ となるが，他の成分と全く同じ $\mathbb{GF}_2(\vec{A})(\vec{X})$ の多項式集合が重複して現れる．このような場合，改変版アルゴリズムでは同一の多項式集合に対して全く同じグレブナー基底計算が繰り返されることになる．$\mathbb{GF}_2$ を係数とする多項式環

のグレブナー基底計算は [2] の方法によって軽量化が図られているとはいえ，最悪で指数計算時間を要する [4] ためこのような無用な計算の重複は避けるべきである．

以下，多項式 $f \in \mathbf{B}(\vec{A})(\vec{X})$ の係数集合を $\mathit{Coeff}(f)$ で表す．すなわち $f = \sum_{i=1}^{k} a_i t_i$ $(k \geq 1, a_1, \ldots, a_k \neq \mathbf{0}$, 各 t_i は単項式) のとき $\mathit{Coeff}(f) = \{a_1, \ldots, a_k\}$ である．多項式集合 $F \subseteq \mathcal{P}(\{0, \ldots, n\})(\vec{A})(\vec{X})$ の係数集合を $\mathit{Coeff}(F) = \bigcup_{f \in F} \mathit{Coeff}(f)$ で定義する．

同型写像による 2 つの分解成分が同一の多項式集合となるための簡便な十分条件を，以下のように与えることができる．

命題 1
多項式集合 $F \subseteq \mathcal{P}(\{0, \ldots, n\})$ およびその分解 $\iota(F) = (G_0, \ldots, G_n)$ について，$\{0, \ldots, n\}$ 上の同値関係 $\sim$ を F に含まれる係数による $\{0, \ldots, n\}$ の部分集合族 $\mathit{Coeff}(F)$ が定める最も粗い同値関係，すなわち

$$i \sim j \;\Leftrightarrow\; \text{すべての } S \in \mathit{Coeff}(F) \text{ について } i, j \in S \text{ または } i, j \notin S.$$

で定める．このとき，

$$i \sim j \text{ ならば } G_i = G_j$$

が成り立つ．

同値関係 $\sim$ の同値類を $[i] = \{j \mid i \sim j\}$ で表したとき，多項式集合 F はこの同値類によって添字づけられた，係数を $\mathbb{GF}_2$ とする多項式集合 $\{G_{[i]} \mid i \in \{0, \ldots, n\}\}$ の直積の形に分解できる．各 $G_{[i]}$ は代表元 i の取り方に依らないから，以下 $G_{[i]}$ を G_i と書いて差し支えない．

上記の粗い分解を用いることで，3.3 節の求解アルゴリズム SingSolveGF2 を以下のように改良することができる．(3.3 節の方法は，最も細かな同値関係である恒等関係 $=$ を採用した特殊な場合とみることができる．)

入力　多項式の有限集合 $F \subset \mathcal{P}(\{0, \ldots, n\})(\vec{A})(\vec{X})$
出力　単元集合解の有無 (解有りの場合は解の一例，解無しの場合は $\perp$ を返す)

1: **procedure** SingSolveEqvc(F)
2: 　$[k_1], \ldots, [k_m] \leftarrow \mathit{Coeff}(F)$ が定める同値関係 $\sim$ による相異なる同値類全体
3: 　$(E_{k_1}, \ldots, E_{k_m}) \leftarrow (F_{k_1}, \ldots, F_{k_m})$ ただし，$(F_0, \ldots, F_n) = \iota(F)$
4: 　$(G_{k_1}, \ldots, G_{k_m}) \leftarrow E_{k_1}, \ldots, E_{k_m}$ のブーリアン・グレブナー基底の積
5: 　$(H_1, \ldots, H_m) \leftarrow (G_{k_1} \cap \mathbb{GF}_2(\vec{A}), \ldots, G_{k_m} \cap \mathbb{GF}_2(\vec{A}))$
6: 　**if** $(H_1, \ldots, H_m)$ の単元集合解を発見 **then**
7: 　　**return** 単元集合解の一例
8: 　**else**
9: 　　**return** $\perp$

$[k_1], \ldots, [k_m]$ を F に含まれる係数によって定まる相異なる同値類 (2 行目) としたとき，$(E_{k_1}, \ldots, E_{k_m})$ をこれら同値類の代表元に対応した $\mathbb{GF}_2(\vec{A})(\vec{X})$ の多項式集合の直積とする．(3 行目) この直積のそれぞれの要素についてブーリアン・グレブナー基底とその消去イデアルを計算し (4, 5 行目) $\mathbb{GF}_2(\vec{A})$ の多項式集合の直積 $(H_1, \ldots, H_m)$ を得る．この直積 $(H_1, \ldots, H_m)$ について，3.3 節と同様に almost solution を漸進的に追加していく方法で単元集合解を探索する．

almost solution の発見と追加は以下のように行う．グレブナー基底の直積 $(H_1, \ldots, H_m)$ は，以下の条件を満たすときに almost solution $A_j + \{k_p\}$ $(1 \leq p \leq m)$ を持つという．

- $A_j + \mathbf{1} \in H_p$ であるかもしくは

- 任意の $p' \in \{0, \ldots, m\} \setminus \{p\}$ について $A_j \in H_{p'}$.

網羅的探索の際に，同値類 $[k_q]$ の代表元 k_q を解として選んで almost solution $A_j + \{k_q\}$ $(0 \leq q \leq m)$ を単元集合解候補として加えた場合，同値類 $[k_q]$ は単元集合 $\{k_q\}$ および ($[k_q]$ 自体が単元集合でない場合) その補集合 $[k_q] \setminus \{k_q\}$ に細分される．したがって，直積 $(H_1, \ldots, H_m)$ の更新は以下のように行えばよい．

- $[k_q]$ が単元集合のとき
  - 第 q 成分 H_q に $A_j + \mathbf{1}$ を追加，すなわち $H_q \leftarrow H_q \cup \{A_j + \mathbf{1}\}$ で更新
  - それ以外の第 q' 成分 $H_{q'}$ には A_j を追加，すなわち $H_{q'} \leftarrow H_{q'} \cup \{A_j\}$ で更新
- $[k_q]$ が単元集合でない (2 個以上の要素を持つ) とき
  $(H_1, \ldots, H_m)$ に同値類 $[k_q] \setminus \{k_q\}$ に対応する新たな成分 H_{m+1} を追加した直積 $(H_1, \ldots, H_m, H_{m+1})$ を以下のように構成する．
  - 第 q 成分 H_q に $A_j + \mathbf{1}$ を追加，すなわち $H_q \leftarrow H_q \cup \{A_j + \mathbf{1}\}$ で更新
  - 新たな成分 H_{m+1} を H_q に A_j を加えたものとする，すなわち $H_{m+1} \leftarrow H_q \cup \{A_j\}$
  - これら以外の第 q' 成分 $H_{q'}$ には A_j を追加，すなわち $H_{q'} \leftarrow H_{q'} \cup \{A_j\}$ で更新

ここで，almost solution の追加は $[k_q]$ の代表元 k_q の取り方に依らないことに注意せよ．この性質によりアルゴリズム SingSolveEqvc は，同一の多項式集合に対するグレブナー基底計算を省略するだけでなく，単元集合解の網羅的探索においても同じ同値類に属する値を almost solution の候補から外すことで，3.3 節の SingSolveGF2 と比べて探索空間を狭めることができる．

4　整合性検査器の 2 つの実装と比較

並行分散プログラムの整合性を，対応する集合制約の解消によって検査する整合性検査器を実装した．集合制約解消は，前節までで議論した改良されたブーリアン・グレブナー基底を用いたものに加えて，SAT ソルバを用いた集合制約解消手法によるものも実装し，これらの実行効率を比較した．整合性検査器の実装言語は，静的型付き関数型言語 OCaml[1] である．

4.1　ブーリアン・グレブナー基底による集合制約解消

本研究での実装過程において観察された実行時間の効率改善の効果は以下のようであった.（いずれも概数）

- Buchberger 判定基準の導入 — 約 1.2 倍の効率改善
- $\mathbb{GF}_2^{n+1}$ 分解 [9] — さらに約 5 倍の効率改善
- 同値類を考慮した $\mathbb{GF}_2$ 分解 (本研究) — さらに約 1〜1.14 倍の効率改善

先行研究 [9] による効率改善が最も著しいが，同値類を考慮することでさらなる改良が加えられることがわかる．なお効率改善に幅があるのは，同値類による解空間の分割の個数の違いによるものである．一般に，元の冪集合ブール環 $\mathcal{P}(S)$ における集合 S の要素数 $n+1$ が十分大きければ同値類による分割が粗（同値類の個数が少）となる可能性も大きくなり効率改善に繋がりやすいが，要素数が小さい場合は粗な分割となることはあまりないため，効率改善の効果も小さくなる傾向がある．

k-Schlegel	BG 実行時間/メモリ消費量/基底計算回数			SAT 実行時間/メモリ消費量	
2	0.009s	3.8MB	7	0.008s	3.6MB
3	0.026s	5.6MB	112	0.012s	6.4MB
4	0.085s	5.8MB	527	0.038s	12MB
5	0.32s	6.8MB	2,151	0.16s	112MB
6	1.34s	9.5MB	8,270	1.2s	1.7GB
7	5.7s	86.2MB	31,256	20.3s	10.3GB
8	2.57s	1.37GB	118,725	∞	15GB 超
9	196.9s	10.0GB	457,201	–	–

– は未計測，∞ は計測不能を表す

表 1: ブーリアン・グレブナー基底アルゴリズム (BG) と SAT ソルバ (SAT) による整合性検査性能の比較

4.2 SAT ソルバによる集合制約解消

SAT ソルバとは，SAT 問題と呼ばれる論理式の充足可能性問題を解くための制約解消器である．SAT 問題は NP 完全問題であるため，すべての SAT 問題を効率的に解くことは不可能だが，一定の範囲の入力に対してこれを効率的に解くための手法が開発され，盛んに研究されている．

SAT 問題が扱うのは CNF (conjunctive normal form) と呼ばれる形式で表現される論理式である．CNF は，論理和で結ばれたリテラルの集合を節としたとき，このような節の論理積として表現される．ただし，リテラルとは命題変数もしくはその否定である．この CNF 形式を用いて集合制約を次のように表現することができる． $\{0,\ldots,n\}$ の部分集合 X について，$0,\ldots,n$ に対応する命題変数を $p_{X,0},\ldots,p_{X,n}$ と定め，$i \in X$ およびそのときに限り $p_{X,i}$ が真であると解釈する．このような解釈のもとで，例えば集合制約 $X \subseteq Y$ は CNF 形式で $(\neg p_{X,0} \vee p_{Y,0}) \wedge \cdots \wedge (\neg p_{X,n} \vee p_{Y,n})$ のように表される．

本研究では，全く同じに生成された集合制約をブール多項式および CNF の両方に変換し，それぞれの解消手法を適用することで，これらふたつの異なる方式による整合性検査器を実装した．なお SAT の制約解消には，標準的な SAT ソルバである MiniSat[3] を用いた．

4.3 グレブナー基底と SAT ソルバの実行効率の比較と考察

上記のふたつの整合性検査器の効率を比較するために，その実行時間およびメモリ使用量を測定した．実行効率の比較のためにベンチマークとしては，並行分散 k-Schlegel 問題のプログラムを用いた．並行分散 k-Schlegel 問題は，2 節で示した 2 プロセスの quasi-consensus 問題を一般化したものであり，集合パターンマッチ記述を用いて各 $k = 2, 3, 4, \ldots$ に対して系統的にプログラミングすることができる.[1)] ただし，一般の k-Schlegel 問題のプログラム記述は $O(2^k)$ 個のパターンマッチ節を要し，ある程度大きな k に対する整合性検査には相当の計算時間を要する．

k-Schlegel 問題 ($2 \leq k \leq 9$) それぞれについての計測結果を表 1 に示す．実験環境は MacBook Pro (Apple M4, 32GB 物理メモリ) である．表には，集合制約の解消をブーリアン・グレブナー基底と SAT ソルバに

[1)] 2 プロセスの quasi-consensus 問題は，1 次元正軸体の複体 (図 1 の四辺形 $\mathcal{I}$) を入力とし，この正軸体のあるひとつのファセットに他のファセットを投影して得られる Schlegel 図と呼ばれる細分を出力の複体 (図 1 の逆コの字型 $\mathcal{O}$) とする並行分散計算として幾何的に定式化できる．これを一般化して，k-Schlegel 問題は，$k-1$ 次元正軸体から Schlegel 図への投影に対応する k プロセス並行分散計算問題として定式化できる．Schlegel 図について詳細は [5] 16 章を参照されたい．

よって行った場合それぞれについて，その実行時間と物理メモリ使用量を示した．(なお，グレブナー基底については基底計算の回数も付記した．)

計測結果が示すように，k が小さい ($k \leq 6$) 場合，グレブナー基底による整合性検査は SAT ソルバによるものと比べて最大で 2 倍程度長い実行時間を要しており，効率が悪いように見える．しかし，より大きい $k = 7$ においてはグレブナー基底は SAT を大きく逆転している．これは，グレブナー基底による整合性検査は SAT ソルバに比べてメモリ使用量が圧倒的に少なく (100 倍以上省メモリ)，このためメモリシステムへの負荷が小さいためであると考えられる．このグレブナー基底の省メモリ性はいくらか k を大きくしても維持されており，SAT は $k = 8$ でメモリ不足のため実行計測不能となったのに対して，グレブナー基底は $k = 9$ でも数分程度の実行時間で整合性検査を完了することができた．

この実験結果より，ブーリアン・グレブナー基底計算は SAT ソルバと比較してメモリ効率性に関して優位であり，入力のサイズが大きい場合は実行時間においても優位となることが読み取れる．このように SAT ソルバがメモリ効率性に関して劣後する主たる要因としては，集合制約を CNF 形式の論理式に変換する過程で論理式のサイズが大きく膨張することが考えられる．集合制約から CNF への変換過程では，ひとつの集合変数 X について複数 (プロセスの個数分) の命題変数 $p_{X,0}, \ldots, p_{X_n}$ を導入する必要がある．さらに，集合制約の直接的な論理式への変換結果が CNF 形式でない場合には，これを論理積と和の分配法則に基づいて CNF 形式に変換する必要があり，大きくサイズを膨張させる原因となる．一方，ブーリアン・グレブナー基底計算の場合，集合制約からブール多項式への変換は，各集合変数 X はそのまま多項式変数 X に対応させることで素直に達成でき，この変換の過程で入力のサイズが大きく増えることはない．このような集合制約からの変換によるサイズの増加率の違いがメモリ効率性に関する顕著な差を生み出しているのではないかと推察される．

5　まとめ

ブーリアン・グレブナー基底計算を用いた集合制約解消手続きについて，単元集合解アルゴリズムと 2 元体の直積分解を組み合わせた手法 [9] をもとにこれを改良したアルゴリズムを提案した．このアルゴリズムでは，入力の多項式集合に含まれる係数集合から導かれる最も粗い同値関係が導く同値類により，解空間をより少ない 2 元体の直積に分解する．これによって，集合制約解消手続きのボトルネックとなるブーリアン・グレブナー基底計算の実行回数を削減するとともに，同じ同値類に属する解の候補に関する網羅的探索の探索空間を狭めることで，より効率の良い集合制約解消を達成した．

この改良されたアルゴリズムを組み込んだ，並行分散プログラムの幾何的整合性検査器を実装した．整合性検査器においては，個々のサイズはそれほど大きくはないが，多数の集合制約解消を繰り返し実行する必要があるため，集合制約解消の手続きの性能が全体の効率に大きく影響する．比較として，まったく同じ集合制約を SAT ソルバを用いて解消する手法も実装し，これら 2 つの手法の実行効率を比較した．その結果，ブーリアン・グレブナー基底計算を用いた検査器の方が，SAT ソルバを用い検査器よりあきらかに優位なメモリ効率性を示した．特に対象となるプログラムのサイズが大きい場合においては，ブーリアン・グレブナー基底計算の方が実行時間でも SAT ソルバの性能を上回った．

このようなメモリ効率性の差は，SAT ソルバが扱う CNF 論理式に比べ，ブール多項式の方が集合制約をより直接的に短く表現できることに起因すると考えられる．このことは，本稿で扱った並行分散プログラムの幾何的整合性検査の他にも，ブーリアン・グレブナー基底計算が SAT ソルバより効率的に働く問題領域が存在することを示唆していると考えられる．

参 考 文 献

[1] OCaml homepage. `https://ocaml.org/`.

[2] Michael Brickenstein and Alexander Dreyer. PolyBoRi: A framework for Gröbner-basis computations with boolean polynomials. *Journal of Symbolic Computation*, 44:1326–1345, 2009.

[3] Niklas Eén and Niklas Sörensson. The MiniSat page. `http://minisat.se/`.

[4] Sicun Gao. Counting zeros over finite fields using gröbner bases. Master's thesis, Carnegie Mellon University, 2009.

[5] Maurice Herlihy, Dmitry N. Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2013.

[6] Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *Journal of the ACM*, 46(6):858–923, 1999.

[7] Shutaro Inoue. Efficient singleton set constraint solving by boolean Gröbner bases. *Communications of JSSAC*, 1:27–37, 2012.

[8] Dmitry N. Kozlov. Chromatic subdivision of a simplicial complex. *Homology, Homotopy and Applications*, 14(2):197–209, 2012.

[9] A Nagai and S Inoue. An implementation method of boolean Gröbner bases and comprehensive boolean Gröbner bases on general computer algebra systems. In *Mathematical Software — ICMS 2014*, volume 8592 of *LNCS*, pages 531–536. Springer, 2014.

[10] Tetsuo Nakano, Kenji Arai, and Hiromasa Watanabe. On the Inoue invariants of the puzzles of Sudoku puzzles. *Communications of JSSAC*, 2:1–14, 2016.

[11] Yosuke Sato, Shutaro Inoue, Akira Suzuki, Katsusuke Nabeshima, and Ko Sakai. Boolean Gröbner bases. *Journal of Symbolic Computation*, 46:622–632, 2011.

[12] Yosuke Sato, Satoshi Menju, and Ko Sakai. Solving constraints over sets by boolean Gröbner bases. ICOT Tehcnical Report TR-680, Institute for New Generation Computer Technology, 1991.

[13] 西村 進. Boolean gröbner 基底を用いた並行分散プログラムの整合性検査. In *Computer Algebra — Foundations and Applications 2024,* **数理解析研究所講究録**, number 2320, pages 95–105. `https://www.kurims.kyoto-u.ac.jp/~kyodo/kokyuroku/contents/pdf/2320-10.pdf`.

[14] 西村 進. ブーリアン・グレブナー基底を用いた集合制約解消による関数型並行分散プログラムの幾何的整合性検査. In **日本ソフトウェア科学会第** *42* **回大会**, pages 3A–1, 2024.