数理解析研究所講究録
第1572巻 2007年 131-141

Risa/Asir CGB関連パッケージの整備

倉田 陽介
YOSUKE KURATA
神戸大学 自然科学研究科 *

鈴木 晃
AKIRA SUZUKI
神戸大学 情報管理室 †

鍋島 克輔
KATSUSUKE NABESHIMA
RISC-Linz, Johannes Kepler Universität ‡

1 Introduction

Comprehensive Göbner Systems (CGS), あるいは, Comprehensive Gröbner Bases (CGB) とはパラメータを含む多項式イデアルに対するGröbner Basesを与える. CGS & CGBの概念は1992年にWeispfenning [19] によって示され, その後, Weispfenning自身による [20] (CCGB), Manubens-Montes [8, 7] (DISPGB), 鈴木・佐藤 [16, 15] (ACGB) などにより, 新しいアルゴリズムが発展している.

また, 2006年に鈴木・佐藤 [17] (SS_CGS) によりまったく新しいCGS & CGB計算のアルゴリズムが示され, これまでのアルゴリズム, 実装では計算できなかった問題が高速に計算できるようになり, CGSアルゴリズムの研究者のみならず注目を集めるところとなっている.

パラメータ付き多項式集合 $F \subset K[\bar{A}, \bar{X}]$ のCGS & CGBの計算は, K を体とし, L をその代数的閉包, また $\bar{A} = \{A_1, \ldots, A_m\}$ をパラメータ, $\bar{X} = \{X_1, \ldots, X_n\}$ を変数とすると,

1. $K(\bar{A})$ 上の多項式環 $K(\bar{A})[\bar{X}]$
2. K 上の多項式環 $K[\bar{A}, \bar{X}]$
3. von Neumann regular ring R 上の多項式環 $R[\bar{X}]$

のどれかの観点から計算をすることになる. WeispfenningのCGBやManubens-MontesのDISPGBは本質的には1.の上にCGS計算に特化したS多項式と単項簡約を定義している. 鈴木・佐藤のACGBはやや趣が変わり, 3.による方法である. この方法はどちらかというと計算効率における改良というよりは, CGS & CGBの理論的な定式化といえる. 最新の鈴木・佐藤のSS_CGSアルゴリズムは2.による計算法であり, 既存の体 K 上の多項式環におけるGröbner基底計算をそのまま流用できることから, 新たにCGS & CGBプログラムを開発する場合の敷居の低さもこの方法の大きな特徴になっている.

現在のCGS & CGBアルゴリズム研究は, 上記で紹介したWeispfenningグループ, Montesグループ, 佐藤グループが主流となって開発が行われている. それぞれのシステムは,

*kurata@math.kobe-u.ac.jp
†sakira@kobe-u.ac.jp
‡Katsusuke.Nabeshima@risc.uni-linz.ac.at

- Weispfenning (CGB) : 計算機代数システム REDUCE.
  `http://students.fim.uni-passau.de/~reduce/cgb/`
- Montes (DISPGB) : 数式処理システム Maple.
  `http://www-ma2.upc.es/~montes/`
- 鈴木・佐藤 (ACGB) : 計算機代数システム Risa/Asir. (未公開)
- 鈴木・佐藤 (SS_CGS) : 計算機代数システム Risa/Asir.
  `http://kurt.scitec.kobe-u.ac.jp/~sakira/CGBusingGB/`

の上で構築されている. 特に SS_CGS 計算とその周辺に関する研究は倉田・野呂 [6] や鍋島 [10] により進んでおり, 日本人研究者の活躍するところが大きい. そこで鈴木, 鍋島, 倉田のそれぞれの SS_CGS アルゴリズムに関連した研究成果を統合し, 計算機代数システム Risa/Asir [12] 上のパッケージとして提供することにした.

2 鈴木・佐藤の CGS アルゴリズムの復習 (SS_CGS)

最初に, アルゴリズムに無関係な CGS および, CGB の定義を与えておく. $<_{\bar{X}}$ で $T(\bar{X})$ 上の項順序を表すとする.

定義 1 (Comprehensive Gröbner System)
F を $K[\bar{A},\bar{X}]$ の部分集合とし, $S_1,\ldots,S_l$, $T_1,\ldots,T_l$ をそれぞれ $K[\bar{A}]$ の有限部分集合とする. このとき, 有限集合 $\mathcal{G}=\{(S_1,T_1,G_1),\ldots,(S_l,T_l,G_l)\}$ が F の項順序 $<_{\bar{X}}$ に関する *comprehensive Gröbner system* であるとは, $(\mathbf{V}(S_1)\setminus\mathbf{V}(T_1))\cup\cdots\cup(\mathbf{V}(S_l)\setminus\mathbf{V}(T_l))=L^m$ かつ, 任意の $\bar{a}\in\mathbf{V}(S_i)\setminus\mathbf{V}(T_i)$, $(i=1,\ldots,l)$ に対して $\sigma_{\bar{a}}(G_i)$ が $L[\bar{X}]$ のイデアル $\langle\sigma_{\bar{a}}(F)\rangle$ の $<_{\bar{X}}$ に関する *Gröbner basis* となるときのことを言う. また, 各 (S_i,T_i,G_i) あるいは $(\mathbf{V}(S_i)\setminus\mathbf{V}(T_i),G_i)$ を $\mathcal{G}$ の *segment* と呼ぶ.

定義 2 (Comprehensive Gröbner Bases)
$G\subset K[\bar{A},\bar{X}]$ が F の $<_{\bar{X}}$ に関する *comprehensive Gröbner basis* であるとは, 任意の $\bar{a}\in L^m$ に対して $\sigma_{\bar{a}}(G)$ が $\langle\sigma_{\bar{a}}(F)\rangle\subset L[\bar{X}]$ の $<_{\bar{X}}$ に関する *Gröbner basis* となるときのことを言う.

SS_CGS アルゴリズムを支えるのは, Kalkbrener [5] による以下の定理である.

定理 3 (Kalkbrener 1997)
有限集合 $F\subset K(\bar{A})[\bar{X}]$ の項順序 $<_{\bar{X}}$ に関する *Gröbner* 基底を $G=\{g_1,\ldots,g_l\}$ とし, 任意の $\bar{a}\in L^m$ に対応する *specialization homomorphism* $K(\bar{A})\longrightarrow L$ を $\sigma_{\bar{a}}$ とする. ここで, ある $\bar{a}\in L^m$ に対して,

$$G_{\bar{a}}=\left\{\, g\in G \;\middle|\; \sigma_{\bar{a}}(\mathrm{HC}_{<_{\bar{X}}}(g))\neq 0 \,\right\}$$

とするとき,
$\sigma_{\bar{a}}(G_{\bar{a}})$ が $\langle\sigma_{\bar{a}}(F)\rangle$ の $<_{\bar{X}}$ に関する *Gröbner* 基底になることと, 任意の $g\in G$ に対して, $\sigma_{\bar{a}}(g)\xrightarrow{*}_{\sigma_{\bar{a}}(G_{\bar{a}})}0$ となることは必要十分である.

したがって, 重要なことは **$K(\bar{A})[\bar{X}]$ 上での Gröbner 基底が得られることであって, その計算方法は問わない**. ということである. この定理から, **$K(\bar{A})$ 上の Gröbner 基底を block 順序を使って K 上で計算することを考えると**, 次の補題が成立する.

補題 4
有限集合 $F \subset K[\bar{A}, \bar{X}]$ の項順序 $<_{\bar{A},\bar{X}}$ に関する *Gröbner* 基底を G とする．もし，任意の $g \in G \setminus K[\bar{A}]$ と任意の $\bar{a} \in \mathbf{V}(G \cap K[\bar{A}])$ に対して，$\sigma_{\bar{a}}(\mathrm{HC}_{<_{\bar{X}}}(g)) \neq 0$ ならば，$\sigma_{\bar{a}}(G)$ は $\langle \sigma_{\bar{a}}(F) \rangle \subset L[\bar{X}]$ の $<_{\bar{X}}$ に関する *Gröbner* 基底である．ここで，項順序 $<_{\bar{A},\bar{X}}$ は $T(\bar{A}, \bar{X})$ 上の順序で，$\bar{A} \ll \bar{X}$ を満たす *block* 順序であり，$<_{\bar{A},\bar{X}}$ の $T(\bar{X})$ への制限は $<_{\bar{X}}$ に等しいとする．

したがって，$F \subset K[\bar{A}, \bar{X}]$ の項順序 $<_{\bar{A},\bar{X}}$ に関する Gröbner 基底を G とし，任意の $\bar{a} \in \mathbf{V}(G \cap K[\bar{A}]) \setminus \bigcup_{g \in G \setminus K[\bar{A}]} \mathbf{V}(\mathrm{HC}_{<_{\bar{X}}}(g))$ に対して，$\sigma_{\bar{a}}(G)$ は $\langle \sigma_{\bar{a}}(F) \rangle \subset L[\bar{X}]$ の $<_{\bar{X}}$ に関する Gröbner 基底となる．

SS_CGS アルゴリズムではこのアイデアを再帰的に適用して完全な CGS，あるいは CGB を得る．詳細は鈴木・佐藤 [17] を参照されたい．鈴木・佐藤アルゴリズムにはいくらかのバリエーションが考えられるが，基本形は以下に示す再帰的アルゴリズム CGS である．

Algorithm CGS

INPUT: A finite subset F of $K[\bar{A}, \bar{X}]$ and a term order $<_{\bar{A},\bar{X}}$.
OUTPUT: A finite set $\mathcal{H}$ of triples (S, T, G) of a set of polynomials S and T in $K[\bar{A}]$, and a Gröbner basis G in $K[\bar{A}, \bar{X}]$.

```
BEGIN
    G ← ReducedGröbnerBasis(F, <_{Ā,X̄});
    H ← {(F ∩ K[Ā], G ∩ K[Ā], {1})};
    IF 1 ∈ G THEN
        return H;
    END
    h ← SquareFree(∏_{g∈G\K[Ā]} HC_{<_X̄}(g));
    {h_1, ..., h_l} ← Factors(h);
    H ← H ∪ {(G ∩ K[Ā], {h}, G \ K[Ā])};
    FOR i = 1, ..., l DO
        H ← H ∪ CGS(G ∪ {h_i}, <_{Ā,X̄});
    END
    return H;
END
```

3 開発動機

すでに，SS_CGS 計算法は従来からある方法に比べていくらかの例で高速に結果を出力することが分かっているが，実際のところ計算が速い理由が何かはっきりとは分かっていない．具体的には以下の疑問点がある．

- ベースシステム (REDUCE, Maple, Risa/Asir) の多項式演算速度の性能差の影響はどうか？ 特に整数演算部の性能差は確実に存在する．
- $K(\bar{A})[\bar{X}]$ 上の Gröbner 基底は $K(\bar{A})$ 上の多項式環で直接計算するより，K 上の多項式環でブロック順序 $\bar{A} \ll \bar{X}$ で計算する方が良いとされているが，どの程度それがいえるのか？

さらに，このアルゴリズムの登場で，CGS 計算がずいぶん効率的になったと言われているが，それでも少しでも複雑な問題を入力するとすぐに計算不能に陥るのも事実である．そこで，改めて研究動機をまとめると以下のようになる．

1. **実際に利用できる CGS・CGB 計算プログラムの提供.**
   特にプログラムは多くの人に利用されることによって開発が進むので.

2. **研究成果の統合.**
   鈴木, 鍋島, 倉田とそれぞれで SS_CGS アルゴリズムに関連した研究をしており, その研究成果を統合する.

3. **CGS 計算過程・結果の解析用関数群の開発.**
   計算効率を上げる工夫, 計算時間等の解析に必要.

4. **Selection strategy の研究.**
   今後の研究課題の 1 つ. Selection strategy とは何か? は第 6 節にて触れる.

5. **先ほどの疑問点の検証用.**

4 開発プロジェクトと現在までの成果物

Risa/Asir PGB プロジェクトとして, 以下の URL にて研究開発成果の提供を行う.

`http://www.math.kobe-u.ac.jp/Asir/PGB/`

現在までの成果物としては,

1. SS_CGS アルゴリズムによる CGS・CGB 計算プログラム. `ss_cgs.rr`
2. 倉田・野呂 [6] アルゴリズムによる Discrete Comprehensive Gröbner Bases 計算プログラム. `yk_dcgr.rr`. このプログラムの詳細は [6] を参照してもらいたい.

5 CGS・CGB 計算プログラム "ss_cgs.rr"

- `ss_cgs.rr` は SS_CGS アルゴリズムを実装している.
- このプログラムで, パラメータ付き多項式集合 $F \subset \mathbb{Q}[\bar{A}, \bar{X}]$ の reduced CGS, reduced faithful CGS, reduced CGB が計算できる.
- 計算過程や計算時間, パラメータ空間の解析用関数が付属している.

5.1 パッケージの読み込み

Risa/Asir の通常のパッケージの読み込み方法と同じである.
local ディスクに `ss_cgs.rr` をダウンロードした後に,

`load("{ss_cgs.rr のあるディレクトリの full path}/ss_cgs.rr");`

をタイプして読み込む.
なお, 自動的に `gr`, `primdec` パッケージも読み込まれる.

5.2 CGS・CGB計算関数

5.2.1 ss_cgs.cgs_main, ss_cgs.fcgs_main

ss_cgs.cgs_main(*plist, vlist, homo, modular, order*)
::Comprehensive Gröbner System の計算.
ss_cgs.fcgs_main(*plist, vlist, homo, modular, order*)
::Faithful Comprehensive Gröbner System の計算.

return	内部表現 ss_branch のリスト.	*plist*	多項式のリスト.
vlist	不定元のリスト.	*order*	数, リストまたは行列.
homo	フラグ.	*modular*	フラグまたは素数.

- パラメータ付き多項式リスト *plist* の変数順序 *vlist*, 項順序型 *order* に関する reduced CGS, および reduced faithful CGS を求める.
- *vlist* に含まれない不定元はパラメータと見なされる.
- パラメータの変数順序, 項順序型 (通常は "0") および, ブロック順序は自動的に生成される.
- 内部の $\mathbb{Q}$ 上多項式環の Gröbner 基底計算では組み込み関数 nd_gr_trace を用いている.
- フラグ *homo* の値は内部で呼び出される nd_gr_trace の引数 *homo* に渡され, 斉次化経由の Gröbner 基底計算をするかしないかの制御をする.
- *modular* の値も内部で呼び出される nd_gr_trace の引数 *modular (p)* に渡され, Gröbner trace アルゴリズムの制御をする.
- 内部表現 ss_branch を読みやすい文字列に変換するには, ss_cgs.cgsprint を利用する.

計算例 1:
$\mathbb{Q}[z, r, l, s_1, c_1, s_2, c_2]$ のイデアル,

$$I = \langle r - c_1 + l(s_1 s_2 - c_1 c_2),\ z - s_1 - l(s_1 c_2 + s_2 c_1),\ s_1^2 + c_1^2 - 1,\ s_2^2 + c_2^2 - 1 \rangle$$

に対して, 変数 z, r, l をパラメータとする辞書式順序 $s_1 > c_1 > s_2 > c_2$ に関する reduced CGS を計算する. これは, The inverse kinematics problem for a simple robot で, [8] の Application 11.3 からである.

```
[301] F = [r-c1+l*(s1*s2-c1*c2), z-s1-l*(s1*c2+s2*c1), s1^2+c1^2-1, s2^2+c2^2-1]$
[302] G = ss_cgs.cgs_main(F,[s1,c1,s2,c2],0,1,2);

The CGS computation done....
UP=[ 0.03125 0.015625 ], RA=[ 0.015625 0 ], NZC=10, NZCT=[ 0.0625 0.03125 ],
ZC=2, ZCT=[ 0 0 ], AEC=4, AET=[ 0.015625 0 ], BEC=14, BET=[ 0.015625 0 ],
ALC=16, MaxDepth=4, MaxWidth=6

[{[s1,c1,s2,c2],2,[z,r,l],0,0},{[0],[ 0 0 ],1,{[],[(l^3-l)*r*z^3+(l^3-l)*r^3*z],
[-z^2-r^2+l^2+2*c2*l+1,z^4+(2*r^2-2*l^2-2)*z^2+r^4+(-2*l^2-2)*r^2+l^4+(4*s2^2-2)*l^2+1
```

⋮ ⋮

計算結果は reduced CGS を返すが，見ての通り内部表現の構造体を返す．計算の最後に，各部の計算に要した時間，不要なために除去された segment の数などの情報がレポートされる．内部表現で出力されたものは，ss_cgs.cgsprint で文字列に変換できる．

```
[303] ss_cgs.cgsprint(G);
[] == 0,  [(l)*(l-1)*(l+1)*(r)*(z)*(z^2+r^2)] != 0,
[-z^2-r^2+l^2+2*c2*l+1,
z^4+(2*r^2-2*l^2-2)*z^2+r^4+(-2*l^2-2)*r^2+l^4+(4*s2^2-2)*l^2+1,
(-r+2*c1)*z^2-2*s2*l*z-r^3+2*c1*r^2+(l^2-1)*r,
-r*z^3+2*s1*r*z^2+(-r^3+(l^2-1)*r)*z+2*s1*r^3+2*s2*l*r^2]

[l] == 0,  [(z^2+r^2-1)] != 0,
[1]

[l-1] == 0,  [(r)*(z)*(z^2+r^2)] != 0,
[-z^2-r^2+2*c2+2,z^4+(2*r^2-4)*z^2+r^4-4*r^2+4*s2^2,
(r-2*c1)*z^2+2*s2*z+r^3-2*c1*r^2,r*z^3-2*s1*r*z^2+r^3*z-2*s1*r^3-2*s2*r^2]

[l-1,r] == 0,  [(z)] != 0,
[-z^2+2*c2+2,z^4-4*z^2+4*s2^2,-c1*z+s2,z^2-2*s1*z]

[l-1,r,z] == 0,  [] != 0,
[c2+1,s2,c1^2+s1^2-1]

[l-1,z] == 0,  [(r)] != 0,
[-r^2+2*c2+2,r^4-4*r^2+4*s2^2,r^2-2*c1*r,s1*r+s2]

[z^2+r^2,l-1] == 0,  [(z),(r)*(z),(r)] != 0,
[1]
```

⋮ ⋮

となる．1 つの segment はパラメータ空間の零点条件と非零点条件，そして，対応する reduced Gröbner 基底の組である．例えば，

```
[l-1] == 0,  [(r)*(z)*(z^2+r^2)] != 0,
[-z^2-r^2+2*c2+2,z^4+(2*r^2-4)*z^2+r^4-4*r^2+4*s2^2,
(r-2*c1)*z^2+2*s2*z+r^3-2*c1*r^2,r*z^3-2*s1*r*z^2+r^3*z-2*s1*r^3-2*s2*r^2]
```

の意味する segment は，

$$\Big(\mathbf{V}(l-1)\setminus\mathbf{V}(rz(z^2+r^2)),\ \{-z^2-r^2+2c_2+2,\ z^4+(2r^2-4)z^2+r^4-4r^2+4s_2^2,\\ (r-2c_1)z^2+2s_2z+r^3-2c_1r^2, rz^3-2s_1rz^2+r^3z-2s_1r^3-2s_2r^2\}\Big)$$

であり，$\mathbf{V}(l-1)\setminus\mathbf{V}(rz(z^2+r^2))$ の任意の点に対して，$\{-z^2-r^2+2c_2+2,\ z^4+(2r^2-4)z^2+r^4-4r^2+4s_2^2,\ (r-2c_1)z^2+2s_2z+r^3-2c_1r^2, rz^3-2s_1rz^2+r^3z-2s_1r^3-2s_2r^2\}\subset L[s_1,c_1,s_2,c_2]$ が辞書式順序 $s_1>c_1>s_2>c_2$ に関する reduced Gröbner 基底であることを示している．

5.2.2 ss_cgs.cgs_main, ss_cgs.fcgs_main のオプション

`ss_cgs.cgs_main`, `ss_cgs.fcgs_main` はオプション指定により色々と動作を変更することができる．

pv, po: パラメータに関する明示的な指定を行う．*pv = list* でパラメータ変数順序を，*po = number, list or matrix* で項順序型をそれぞれ指定できる．

phom : *phom = flag* で，パラメータ部分のみの Gröbner 基底計算に斉次化を用いるかどうかを指定する．デフォルトでは *phom = 0* である．

pmod : *pmod = flag or prime number* でパラメータ部分のみの Gröbner 基底計算での trace アルゴリズムの制御を行う．デフォルトでは *pmod = 1* である．

ra : *ra = flag* で結果を reduced CGS にするかどうかの制御を行う．デフォルトでは *ra = 1* である．

elim : *elim = number* で，パラメータ空間がすでに計算されている別の segment のパラメータ空間に含まれた場合の事前除去のレベルを設定する．*elim = 0* の時は除去は行わず，*elim = -1* で根基所属判定を用いて完全な除去を行う．デフォルトでは *elim = 2* である．

prim : *prim = flag* で，素分解を使った計算を行うかどうかの制御を行う．デフォルトでは *prim = 0* である．

zrad : `ss_cgs.cgs_main` のみのオプション．*zrad = flag* で，0 次元根基イデアルを使った計算を行うかどうかの制御を行う．デフォルトでは *zrad = 0* である．

print : *print = flag* で，計算途中の中間情報を表示する．デフォルトは *print = 0* である．

5.2.3 ss_cgs.cgs, ss_cgs.hcgs, ss_cgs.fcgs, ss_cgs.hfcgs

`ss_cgs.cgs`(*plist, vlist, order*)
`ss_cgs.hcgs`(*plist, vlist, order*)
::Comprehensive Gröbner System の計算.
`ss_cgs.fcgs`(*plist, vlist, order*)
`ss_cgs.hfcgs`(*plist, vlist, order*)
::Faithful Comprehensive Gröbner System の計算.

return	内部表現 `ss_branch` のリスト.	*plist*	多項式のリスト.
vlist	不定元のリスト.	*order*	数，リストまたは行列.

- パラメータ付き多項式リスト *plist* の変数順序 *vlist*，項順序型 *order* に関する reduced CGS および，reduced faithful CGS を求める．
- `ss_cgs.cgs`(*plist, vlist, order*) は，`ss_cgs.cgs_main`(*plist, vlist, 0, 1, order*) の省略形である．
- `ss_cgs.hcgs`(*plist, vlist, order*) は，`ss_cgs.cgs_main`(*plist, vlist, 1, 1,* order) の省略形である．

- ss_cgs.fcgs(*plist, vlist, order*) は，ss_cgs.fcgs_main(*plist, vlist, 0, 1, order*) の省略形である．
- ss_cgs.hfcgs(*plist, vlist, order*) は，ss_cgs.fcgs_main(*plist, vlist, 1, 1, order*) の省略形である．

5.2.4 ss_cgs.cgb

ss_cgs.cgb(*plist, vlist, order*)
ss_cgs.hcgb(*plist, vlist, order*)
::Comprehensive Gröbner Basis の計算．

return	内部表現 ss_branch のリスト．	*plist*	多項式のリスト．
vlist	不定元のリスト．	*order*	数，リストまたは行列．

- パラメータ付き多項式リスト *plist* の変数順序 *vlist*，項順序型 *order* に関する reduced CGB を求める．
- ss_cgs.cgb(*plist, vlist, order*) は，ss_cgs.fcgs_main(*plist, vlist, 0, 1, order*) で faithful reduced CGS を計算し，結果の和集合を返す．
- ss_cgs.hcgb(*plist, vlist, order*) は，ss_cgs.fcgs_main(*plist, vlist, 1, 1, order*) で faithful reduced CGS を計算し，結果の和集合を返す．

計算例 2:
先ほどの計算例でのイデアル $\langle r - c_1 + l(s_1s_2 - c_1c_2),\ z - s_1 - l(s_1c_2 + s_2c_1),\ s_1^2 + c_1^2 - 1,\ s_2^2 + c_2^2 - 1\rangle \subset \mathbb{Q}[z, r, l, s_1, c_1, s_2, c_2]$ に対して，辞書式順序 $s_1 > c_1 > s_2 > c_2$ に関する reduced CGB を計算する．

```
[301] F = [r-c1+l*(s1*s2-c1*c2), z-s1-l*(s1*c2+s2*c1), s1^2+c1^2-1, s2^2+c2^2-1]$
[302] G = ss_cgs.cgb(F,[s1,c1,s2,c2],2);

The faithful CGS computation done....
UP=[ 0.015625 0.046875 ], RA=[ 0 0 ], NZC=15, NZCT=[ 0.234375 0.09375 ], ZC=2,
ZCT=[ 0 0 ], AEC=0, AET=[ 0 0 ], BEC=20, BET=[ 0 0 ], ALC=17, MaxDepth=4,
MaxWidth=6

[s2^2+c2^2-1,c1^2+s1^2-1,-c1*z+s1*r+s2*l,c1*z-s1*r-s2*l,-z^2-r^2+l^2+2*c2*l+1,
-z^2+2*s1*z-r^2+2*c1*r+l^2-1,z^2+r^2-l^2-2*c2*l-1,z^2-2*s1*z+r^2-2*c1*r-l^2+1,
(-r+2*c1)*z^2-2*s2*l*z-r^3+2*c1*r^2+(l^2-1)*r,
(r-2*c1)*z^2+2*s2*l*z+r^3-2*c1*r^2+(-l^2+1)*r,
                    ⋮                    ⋮
```

5.2.5 ss_cgs.cgsprint

ss_cgs.cgsprint(*brlist*)
::CGS の内部表現を読みやすい文字列に変換する．

return	文字列．	*brlist*	CGS の内部表現 ss_branch のリスト．

- 出力は文字列で，"[*slist*] == 0, [*tlist*] != 0, [*gblist*]" で 1 つの segment $(\mathbf{V}(slist) \setminus \mathbf{V}(tlist), \{gblist\})$ を表す.
- "[*slist*] == 0" は *slist* に含まれる**すべての多項式が 0 になる**ことを意味し，"[*tlist*] != 0" は *tlist* に含まれる多項式のうち，**少なくとも 1 つが 0 でない**ことを意味する.

5.3 CGS 解析用関数

あまり詳しくは述べないが，以下のような解析用関数を用意している.

ss_cgs.segprint : **ss_cgs.segprint**(*brlist*, *number*) で CGS（内部表現）に含まれる詳しい情報を文字列で返す.

ss_cgs.segno : **ss_cgs.segno**(*nolist*, *brlist*) で CGS（内部表現）のうち番号 *nolist* を持つ segment を抽出する.

ss_cgs.trmlseg : **ss_cgs.trmlseg**(*brlist*) で，CGS 計算の各分岐の最終 segment を抽出する.

ss_cgs.upperseg: **ss_cgs.upperseg**(*nolist*, *brlist*) で CGS（内部表現）のうち番号 *nolist* を持つ segment の上位 segment をすべて抽出する.

ss_cgs.seghdim : **ss_cgs.seghdim**(*brlist*) で，CGS（内部表現）の各 segment のパラメータ空間（零条件多様体）の次元を計算する.

これらの関数はどちらかと言えば，アルゴリズムのメンテナンス用関数で開発者向けのものであるが，いまだに多くの問題が CGS 計算で時間を大量に使うため，利用者が状況を解析するのにも役に立つと思われる.

6 CGS 計算における選択戦略

SS_CGS の中心である，アルゴリズム CGS は再帰的なアルゴリズムである．$F \subset K[\bar{A}, \bar{X}]$ に対して，G を項順序 $<_{\bar{A},\bar{X}}$ に関する $\langle F\rangle \subset K[\bar{A}, \bar{X}]$ の reduced Gröbner 基底とし，$\{h_1, \ldots, h_l\} = \{\mathrm{HC}_{<_{\bar{X}}}(g) \mid g \in G \setminus K[\bar{A}]\} \subset K[\bar{A}]$ とする．このとき，$(\mathbf{V}(G \setminus K[\bar{A}]) \setminus (\mathbf{V}(h_1) \cup \cdots \cup \mathbf{V}(h_l)),\ G)$ は F の項順序 $<_{\bar{X}}$ に関する CGS の 1 つの segment であるが，その他の segment を得るために，l 個の多項式集合 $F \cup \{h_1\}, \ldots, F \cup \{h_l\}$ に対して，再帰的にアルゴリズム CGS を適用していく.

このとき，アルゴリズムの動作として，$F \cup \{h_1\}, \ldots, F \cup \{h_l\}$ の内のどれから segment 計算をしていくかについては，これまで論じられてこなかった．しかし，計算中に不要な segment を適宜除去していく過程（すでに計算が完了した別の segment のパラメータ空間に新しい segment のパラメータ空間が含まれる場合に不要と判断して除去をする）を考えれば，どの多項式集合から計算していくか（選択戦略）は重要な問題といえる．例えば，

$$\langle F\rangle = \langle aX^3Y + cXY^2,\ X^4Y + 3dY,\ cX^2 + bXY,\ X^2Y^2 + aX^2,\ X^5 + Y^5\rangle \subset \mathbb{Q}[a, b, c, d, X, Y]$$

に対して，a, b, c, d をパラメータとし，X, Y を変数とするイデアルの $X > Y$ を満たす辞書式順序 $<_{\{X,Y\}}$ に関する CGS を今回の実装で計算することを考える．先ほどの $F \cup \{h_1\}, \ldots, F \cup \{h_l\}$ に対して，パラメータのみの項集合 $T(a, b, c, d)$ 上の全次数逆辞書式順序 $<_{\{a,b,c,d\}}$ で，$\mathrm{HT}_{<_{\{a,b,c,d\}}}(h_i)$ の小さい順に $F \cup \{h_i\}$ を選んできて，計算をした場合は，全計算時間は 0.688 秒，全 segment 数は 31 で，計算中に除去された segment 数は 100 であり，不要な segment の検出には 0.563 秒かかるが，大きい順に選んだ場合は，全計

算時間は2.594秒，全segment数は89で，計算中に除去されたsegment数は234であり，不要なsegmentの検出には2.234秒かかる．ちなみに，不要なsegmentの除去を行わなかった場合は，全計算時間は25.43秒，全segment数は7930にもなる．

7 最後に

今回は，Risa/Asir PGBプロジェクトの発足と，鈴木・佐藤のSS_CGSアルゴリズムの実装を中心に紹介をした．

また，SS_CGSアルゴリズムの最適化に関する新しい結果である，鍋島[10]の実装はまだ行っておらず，実装・評価の実施を予定している．Risa/Asir PGBプロジェクトでは，CGS・CGB計算利用者の方々からのご要望，そして問題提供をお待ちしております．

参 考 文 献

[1] Becker, T. and Weispfenning, V. *Gröbner Bases.* GTM 141, Springer, 1993.

[2] Cox, D., Little, J. and O'Shea, D. *Ideals, Varieties, and Algorithms, Third Edition.* UTM, Springer, 2007.

[3] Cox, D., Little, J. and O'Shea, D. *Using Algebraic Geometry, Second Edition.* GTM 185, Springer, 2005.

[4] Dolzmann, A., Sturm, T. and Neun, W. CGB: Comprehensive Gröbner Bases. `http://students.fim.uni-passau.de/~reduce/cgb/`. 2007.

[5] Kalkbrener, M. On the Stability of Gröbner Bases Under Specializations. *J. Symbolic Computation.* Vol. 24/1, pp. 51–58. 1997.

[6] Kurata, Y. and Noro, M. Computation of Discrete Comprehensive Gröbner Bases Using Modular Dynamic Evaluation. *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC '07).* ACM Press, New York, pp. 243–250. 2007.

[7] Manubens, M. and Montes, A. Improving the DISPGB algorithm using the discriminant ideal. *J. Symbolic Computation.* Vol. 41/11, pp. 1245–1263. 2006.

[8] Montes, A. A new algorithm for discussing Gröbner bases with parameters. *J. Symbolic Computation.* Vol. 33/2, pp. 183–208. 2002.

[9] Montes, A. DPGB: Discussing Parametric Gröbner Bases. `http://www-ma2.upc.es/~montes/`. 2007.

[10] Nabeshima, K. A Speed-Up of the Algorithm for Computing Comprehensive Gröbner Systems. *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC '07).* ACM Press, New York, pp. 299–306. 2007.

[11] Noro, M. and Yokoyama, K. *Computational Fundamentals of Gröbner Bases (in Japanese).* University of Tokyo Press, 2003.

[12] Noro, M. et al. A Computer Algebra System Risa/Asir. `http://www.math.kobe-u.ac.jp/Asir/asir.html`. 2007.

[13] Sato, Y. and Suzuki, A. Discrete Comprehensive Gröbner Bases. *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC '01)*, ACM Press, New York, pp. 292–296. 2001.

[14] Sato, Y., Suzuki, A and Nabeshima, K. Discrete Comprehensive Gröbner Bases II. *Computer Mathematics, Proc. 6th Asian Symposium (ASCM 2003)*, Lecture Notes Series on Computing Vol. 10, World Scientific, pp. 240–247, 2003.

[15] Sato, Y., Suzuki, A and Nabeshima, K. ACGB on Varieties. *Proc. 6th International Workshop on Computer Algebra in Scientific Computing (CASC 2003)*, pp. 313–318. 2003.

[16] Suzuki, A. and Sato, Y. An alternative approach to Comprehensive Gröbner Bases. *J. Symbolic Computation.* Vol. 36/3-4, pp. 649–667. 2003.

[17] Suzuki, A. and Sato, Y. A Simple Algorithm to Compute Comprehensive Gröbner Bases Using Gröbner Bases. *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC '06)*, ACM Press, New York, pp. 326–331. 2006.

[18] Weispfenning, V. Gröbner bases for polynomial ideals over commutative regular rings. *Proc. EUROCAL '87*, LNCS Vol. 378, Springer, pp. 336–347. 1989.

[19] Weispfenning, V. Comprehensive Gröbner bases. *J. Symbolic Computation.* Vol. 14/1, pp. 1-29. 1992.

[20] Weispfenning, V. Canonical Comprehensive Gröbner bases. *J. Symbolic Computation.* Vol. 36/3-4, pp. 669-683. 2003.