超幾何系の隣接関係式を求めるアルゴリズム

Algorithms to find contiguity relations of hypergeometric systems

日本大学 生物資源科学部　　中山洋将 *1
Hiromasa Nakayama
College of Bioresource Sciences, Nihon University

神戸大学 大学院理学研究科数学専攻 *　　高山信毅 *2
Nobuki Takayama
Department of Mathematics,
Kobe University

Abstract

This report provides a detailed explanation of the program we developed to derive contiguity relations for hypergeometric systems of Horn type. For unexplained terminology, details of the algorithm, a discussion of its mathematical correctness, and related literature, please refer to the arxiv preprint [2]. Our motivation for deriving contiguity relations is to utilize it for isomorphic classification of hypergeometric D-modules.

Contiguity relation 導出のために我々が開発したプログラムの詳しい解説をこの報告に記載する. 説明のない用語, アルゴリズムの詳細やその数学的な正しさの議論および関連する文献などは arxiv のプレプリント [2] を参照. Contiguity relation を求める我々の動機は超幾何 D 加群を同型分類するために活用することである.

1 Contiguity Relation

$I(\alpha)$ をパラメータ α を含む Horn 型超幾何系の生成する Weyl 代数 D の左イデアルとする. $D/I(\alpha)$ と $D/I(\alpha')$ が左 D 加群として同型の時にこの同型対応を与える Weyl 代数の元 C, つまり

$$D/I(\alpha) \ni \ell \mapsto \ell C \in D/I(\alpha') \tag{1}$$

が同型写像となるような C を求める問題を考えたい. 我々はこの C を contiguity relation (隣接関係) とよぶ. C は $\mathcal{F}$ 解空間の同型写像

$$\mathrm{Hom}_D(D/I(\alpha'), \mathcal{F}) \ni f \mapsto C \bullet f \in \mathrm{Hom}_D(D/I(\alpha), \mathcal{F}) \tag{2}$$

を与えるので, 超幾何関数の間の通常の意味での contiguity relation も与えている. ここで $\varphi \in \mathrm{Hom}_D(D/I(\alpha'), \mathcal{F})$ としたとき $f = \varphi([1]) \in \mathcal{F}$ なる対応で上記対応は解釈してほしい.

*1 〒 252-0880 神奈川県藤沢市亀井野 1866　E-mail: nakayama.hiromasa@nihon-u.ac.jp
*2 〒 657-8501, 神戸市灘区六甲台町　E-mail: takayama@math.kobe-u.ac.jp

例 1
一変数の Weyl 代数 $D=\mathbb{C}\langle x,\partial_x\rangle$ を考える．$L(\alpha)=x\partial_x+\alpha$, $\alpha\in\mathbb{Z}$ に対して左 D 加群 $D/L(\alpha)$ を同型なもので分類することを考える．

$$D/L(\alpha)\ni\ell\overset{\cdot x}{\longmapsto}\ell x\in D/L(\alpha+1)$$

は D 左準同型．$L(\alpha)x=(x\partial_x+\alpha)x=x(x\partial_x+\alpha+1)=xL(\alpha+1)$ なので well-defined.

$$D/L(\alpha+1)\ni\ell\overset{\cdot\partial_x}{\longmapsto}\ell\partial_x\in D/L(\alpha)$$

も D 左準同型．$L(\alpha+1)\partial_x=(x\partial_x+\alpha+1)\partial_x=\partial_x(x\partial_x+\alpha)$ なので well-defined．この 2 つの準同型を合成すると

$$D/L(\alpha)\ni\ell\mapsto\ell x\partial_x\in D/L(\alpha)$$

だが

$$x\partial_x\equiv-\alpha\mod L(\alpha)$$

となる．$\alpha\neq0$ なら x,∂_x は同型射を与える．しかし $\alpha=0$ では $D/L(0)$ と $D/L(1)$ をつなぐ同型射はつくれていない．$-\alpha$ が超幾何 b-関数のもっとも単純な例．

同型対応 ∂_x, x は解の contiguity relation を与える．つまり，$f=x^{-(\alpha+1)}$ を $L(\alpha+1)\bullet f=0$ なる解とすると，xf は $L(\alpha)$ の解．$f=x^{-\alpha}$ を $L(\alpha)\bullet f=0$ なる解とすると，$\partial_x\bullet f$ は $L(\alpha+1)$ の解．

以下の節では超幾何 D-加群の同型分類のための contiguity relation を求めるプログラムおよび同型分類のためのプログラムを紹介する．これらは

```
git clone https://github.com/nobuki-takayama/contiguity
```

で取得できる．

2 Contiguity を求めるプログラム—直接法

Asir プログラム `f2-cont.rr` は，Appell の 2 変数超幾何関数 $F_2(a,b,b',c,c';x,y)$ の contiguity を未定係数法により求めるプログラムである．例えば $F_2(a,b,b',c,c';x,y)$ の b についての contiguity を求める場合，計算方法は以下のようになる．

1. 隣接作用素 H,B' を未定係数 u,v （ただし u,v は複数）を使って

$$H=c_{00}(u)+c_{10}(u)\partial_x+c_{01}(u)\partial_y+c_{02}(u)\partial_y^2+c_{11}(u)\partial_x\partial_y$$
$$B'=d_{00}(v)+d_{10}(v)\partial_x+d_{01}(v)\partial_y+d_{02}(v)\partial_y^2+d_{11}(v)\partial_x\partial_y$$

   の形に表す．ここで $c_{ij}(u),d_{ij}(v)$ は，指定した全次数以下 (例えば，2 次以下）の x,y の多項式で未定係数 u,v を持つものとする．

2. $F_2(a,b,b',c,c';x,y)$ の満たす微分方程式に対応する微分作用素環の左イデアル $I(b)$, $F_2(a,b+1,b',c,c';x,y)$ の満たす微分方程式に対応する微分作用素環の左イデアル $I(b+1)$ とする．$I(b+1)H\subset I(b)$ となるように未定係数 u を定める．$I(b)B'\subset I(b+1)$ となるように未定係数 v を定める．

   未定係数 u の求め方を具体的に言えば，$I(b+1)$ の各生成元 P について PH を計算し，$I(b)$ のグレブナー基底 G を求めて PH を G で割った余りが 0 になるように未定係数 u を定める．

3. 先ほど求めた H,B' について，積 $B'H$ を計算し $I(b)$ のグレブナー基底 G で割り算した余り R を求める．この R が 0 でないとき，左 D 加群 $D/I(b)$ と $D/I(b+1)$ が同型になる．

■ `find_cont_auto(ParamPlus, Param)`

Appell F_2 の ParamPlus と Param の間の contiguity relation を未定係数法により求める．返り値の第 0 番目と第 1 番目の要素が隣接作用素であり，第 2 番目は D 加群の同型が成り立つとは限らないパラメータの条件である．アルゴリズムの詳細は [2, §3, 4] を参照.

```
[3133] load("f2-cont.rr");
[3298] L=find_cont_auto([a,b+1,bb,c,cc],[a,b,bb,c,cc]);
1.............+.........................................+.................+.....................
略
[3299] L[0];    <--- F_2 の b の上昇作用素 H(b)
((y*x^2+(y^2-y)*x)*dy+bb*x^2+(bb*y+1)*x)*dx+((y^2-y)*x+y^3-2*y^2+y)*dy^2+(((a+bb+1)*y-cc)*x+(a+bb+1)*y^2
+(-a-cc-bb-1)*y+cc)*dy+bb*a*x+bb*a*y-bb*a+(bb+1)*b
[3300] L[1];   <---- F_2 の b の下降作用素 B(b+1)
((y*x^2+(y^2-y)*x)*dy-a*x^2+(bb*y+a)*x)*dx+((y^2-y)*x+y^3-2*y^2+y)*dy^2
+((y-cc)*x+(a+bb+1)*y^2+(-a-cc-bb-1)*y+cc)*dy-a^2*x+bb*a*y+(-b+c-bb-1)*a-bb*b+bb*c-bb
[3301] L[2];   <---- B(b+1) H(b) を $I(b)$ のグレブナー基底で割った余り
((-bb-1)*b^2+((bb+1)*c-bb-1)*b)*a+(-bb^2-bb)*b^2+((bb^2+bb)*c-bb^2-bb)*b
[3302] fctr(L[2]);
[[-1,1],[bb+1,1],[b,1],[b-c+1,1],[a+bb,1]]
```

3　Contiguity を求めるプログラム—GKZ 系を用いる方法

2025-04-01-contiguity.rr, 2026-02-28-chg-by-gkz.rr, 2026-03-01-horn-by-gkz.rr が Horn 型超幾何系の contiguity relation を求める Risa/Asir によるプログラムである．これらのプログラムで定義されている主な関数の説明を以下に記す.

■ `set_A(A)`

行列 A に付随した GKZ 系およびそれの制限として得られる Horn 型超幾何方程式系を求め, contiguity relation の計算に必要な大域変数を設定する．大域変数の値は `show_globals()` で取得できる．特に `H_horn_str` に対応する Horn 型超幾何方程式が文字列として格納されており, このパラメータ `a1, a2, a3, ...` が伝統的な Horn 型超幾何方程式のパラメータとどう対応しているかをこの変数を調べることによりわかる．なお A は左側が単位行列となっている必要がある [2, §5].

例 2

Gauss の超幾何方程式.

```
[2134] load("2026-03-01-horn-by-gkz.rr");
0
[2692] set_A([[1,0,0,-1],[0,1,0,1],[0,0,1,1]]);;
[2693] show_globals();;
A matrix (H_A)=
[ 1 0 0 -1 ]
[ 0 1 0 1 ]
[ 0 0 1 1 ]
A is 3 * 4 matrix.
GKZ parameter (H_beta)=[b1,b2,b3]
GKZ x variables (H_gkz_xvar)=[x1,x2,x3,x4]
GKZ system (H_gkz)=[[-x4*dx4+x1*dx1-b1,x4*dx4+x2*dx2-b2,x4*dx4+x3*dx3-b3,-dx1*dx4+dx2*dx3],[x1,x2,x3,x4]]
------------------------
Horn system (H_horn_str)=
(-x4*dx4-a3)(-x4*dx4-a2) - dx4(x4*dx4+a1)
Horn sytem is stored in H_horn
Horn sytem xvar(H_horn_xvar) = [x4]
Horn system parameter (H_a) = [a1,a2,a3]
a parameter expressed by beta (H_a_rule)=[[a1,b1],[a2,-b2],[a3,-b3]]
beta parameter expressed by a (H_b_rule)=[[b1,a1],[b2,-a2],[b3,-a3]]
Other globals:
```

```
   H_horn_str2=[1*(-x4*dx4-a3)*(-x4*dx4-a2) - 1*dx4*(x4*dx4+a1)]
   H_set_A_data = gkz_natural_params return value
   H_horn_gb (GB of Horn)
   Horn_contiguity
------------------------------
0
[2694]
```

この場合の Horn 型超幾何系は Gauss の超幾何方程式である. `H_horn_str` は `(-x4*dx4-a3)(-x4*dx4-a2) - dx4(x4*dx4+a1)` なので, ${}_2F_1(\alpha,\beta,\gamma;x_4)$ のパラメータとの対応は

$$\alpha = \mathtt{a2},\ \beta = \mathtt{a3},\ \gamma - 1 = \mathtt{a1}$$

である. このパラメータ `a1, a2, a3, ...` は関数 `horn_contiguity_from_a_shift` に与えるパラメータとなる.

■ `T=horn_contiguity_from_a_shift(A_start, A_end)`
パラメータを `A_start` から `A_end` へ動かした場合 (解の方のパラメータの動きと解釈すること, D-加群の方では逆方向) の contiguity relation を `T[0]` に格納する. `T[1]` は超幾何 b-関数である. 戻り値は 5 つの要素からなるリスト `[Contiguity, Bf, Info, UV, Common_factor]` である. `Bf` はパラメータの多項式でありこれの零点集合は同型が成り立たないパラメータの必要条件を与える. 逆向きの場合の計算もすることによりこの二つの零点集合の和集合の外では同型が成り立つことが保証される.

この関数では GKZ 系の contiguity relation を求めるためにベクトル u, v および 対応する GKZ 系のパラメータ β を求める [2, §3.3]. 具体的には, 出発点と到着点のパラメータ差分から $\Delta\beta$ を計算し, $A(u-v) = \Delta\beta$ を満たす非負整数ベクトル u, v を逆算する. $\beta - Au$ が `A_start` に対応する GKZ 系のパラメータであるような β を求め

$$E\partial^u - b\partial^v \in H_A(\beta) \tag{3}$$

がなりたつ Weyl 代数の元 E と パラメータの多項式 b を求める [3]. ここで $H_A(\beta)$ はパラメーター β の GKZ イデアルを表す. $f(\beta;x)$ を GKZ 系 $H_A(\beta)$ の解とすると,

$$E\partial^u \bullet f(\beta;x) = b\partial^v \bullet f(\beta;x), \tag{4}$$

$$E \bullet f(\beta - Au;x) = bf(\beta - Av;x) \tag{5}$$

がなりたつ. E に対して制限アルゴリズム `lr_red` [2, §5.1, 5.2] を経て, Horn 系のグレブナー基底で正規化 (reduction) を行う (`horn_reduction`).

例 3
任意の退化パラメータを設定したシフトも計算可能である. 例えば, パラメータ間に $a_1 = p - 1, a_2 = p$ といった代数的な関係（特殊化）を持たせたまま, $p \to p + 1$ の contiguity を求める場合,

```
horn_contiguity_from_a_shift([p-1, p, q], [p, p+1, q])
```

と入力する. E, b を計算するアルゴリズムはパラメータに特殊な関係があっても generic な値で同型がなりたてば正しく E, b を求めるので [3], 正しい 1 階の作用素を堅牢に導出することができる.

```
openxm asir
[2134] load("2026-03-01-horn-by-gkz.rr");
0
[2691] set_A([[1,0,0,-1],[0,1,0,1],[0,0,1,1]]);;
[2692] horn_contiguity_from_a_shift([p-1,p,q],[p,p+1,q]);
uv_contiguity(UV=[[1,0,0,0],[0,1,0,0]],A,Beta)
A=[[1,0,0,-1],[0,1,0,1],[0,0,1,1]], Beta=[p,-p,-q]
```

```
  略
check_uv_contiguity(Cont,Dxu,Id,Xvar): Cont[1]*Dxu-Cont[0] = 0 mod Id?
  Cont=[(-p+q)*dx2,-x3*dx4-x1*dx2],Dxu=dx1,Id=[-x4*dx4+x1*dx1-p,x4*dx4+x2*dx2+p,x4*dx4+x3*dx3+q,-dx1*dx4+dx2*dx3],
  Xvar=[x1,x2,x3,x4]
check_uv_contiguity: OK.
  略
lr_red1_top((-x3-1)*dx4+(-x1-1)*dx2)-->(x4-1)*dx4+p
  略
[(x4-1)*dx4+p,-p+q,[[p-1,p,q], -> ,[p,p+1,q]],
 [-x3*dx4-x1*dx2,(-p+q)*dx2,dx1,dx2],1]
[2693]
```

u, v の取り方によっては contiguity relation と b に共通因子がある場合がある.

例 4

$L \bullet {}_2F_1(c, b, c; x) \sim {}_2F_1(c-1, b, c-1; x)$ となる contiguity relation L を下記の u, v で求めた場合.

```
[2699] T=horn_gauss_contiguity_by_GKZ([c,b,c],[c-1,b,c-1]); //旧版
// A=[[1,0,0,-1],[0,1,0,1],[0,0,1,1]];;
// Beta=[c-2,-c+1,-b+1];;
// C=uv_contiguity(UV=[[0,0,0,1],[0,0,1,0]],A,Beta);
// C を制限して Gauss 超幾何の contiguity にする
[((b-1)*x4^2+(-b+1)*x4)*dx4+(b^2-b)*x4+(-c+1)*b+c-1,(c-1)*b-c+1,
 [[c-1,c,b], -> ,[c-2,c-1,b]], // a1, a2, a3 での パラメータの変換
 [x3*x4*dx3^2+(x2*x3*dx1+(-c+2)*x4)*dx3+x2*dx1,((c-1)*b-c+1)*dx3,dx4,dx3]]
[2700] fctr(T[0]);
[[1,1],[b-1,1],[(x4^2-x4)*dx4+b*x4-c+1,1]]
[2701] fctr(T[1]);
[[1,1],[c-1,1],[b-1,1]]
```

$b-1$ が共通因子でありキャンセルしてよい. 本プログラムでは, 有理 Weyl 代数上での還元に加えて, Risa/Asir の可換 GCD 計算を援用することで, このようなパラメータ多項式に依存する共通因子を自動的に検知・約分し, 正規化された作用素を出力する機構を実装している.

a1, a2, a3, β, u, v の関係はやや複雑なのでこの場合にどう対応しているかその解説も記載しておく. ${}_2F_1(c, b, c; x)$ に対応する horn_contiguity_from_a_shift のスタート a パラメータは a1 $= c-1$, a2 $= c$, a3 $= b$ となる. H_b_rule によれば GKZ 系の β パラメータは $(c-1, -c, -b)$. $u = (0, 0, 0, 1)$ なので $Au = (-1, 1, 1)$ (本来は Au^T と書くべきだが適宜 ^{T} は省略する. その他のベクトルも同様.). したがって $\beta - Au = (c-1, -c, -b)$ を満たす β は $\beta = (c-2, -c+1, -b+1)$ となり, この値を用いて $E\partial^u - b \in H_A(\beta)$ となる E, b を求める.

3.1 最短シフト探索（ヒューリスティクス）による計算の高速化

GKZ 系を経由して contiguity を計算する際, シフトベクトル u, v は方程式 $A(u-v) = \Delta\beta$ を満たす解として求められる. 連立一次方程式の一般解（特解）として冗長なベクトルを選んだ場合, ベクトルの長さ（微分作用素 ∂^u の階数）が大きくなり, GKZ 系の自由加群におけるグレブナー基底計算に膨大な時間を要するケースがある (パラメータの飛びが大きいと計算時間がかかる問題).

これを回避するため, horn_contiguity_from_a_shift の内部では, 線形方程式ソルバに投げる前に以下のヒューリスティクスを適用している.

1. u, v の長さの和が 1 となる最短シフト ($U = e_j$ または $V = e_j$) を全探索する.
2. 見つからなければ, 長さの和が 2 となるシフトを全探索する.
3. それでも見つからない場合のみ, 連立一次方程式を解いて特解を求める.

この単純な最短シフト探索を挟むことで, 最も頻繁に計算される「パラメータを ± 1 動かす」基本的な contiguity は常に最小次数の作用素でアルゴリズムが回り, 計算時間の短縮と, 余分な因子の発生を未然に防ぐことが可能となっている.

4 SageMath との連携インターフェース

多面体計算や, 得られた超幾何方程式のパラメータ操作をより直感的に行うため, 我々は SageMath から Risa/Asir を透過的に呼び出すためのインターフェーススクリプト (`asir.py`, `horn_contiguity.py`) を提供している.

■ `set_horn_A(A_matrix)`
SageMath 側から行列 A を Risa/Asir に送信し, 初期化を行う. その際, Asir 側で自動生成された変数リスト `H_horn_xvar` や `H_a` をパースし, SageMath 側の多項式環 (`PolynomialRing`) および, 変数文字列とジェネレータを対応付ける辞書を自動構築して返す. これにより, ユーザーは変数の宣言を手動で行うことなく, 即座に多項式としての計算を開始できる.

■ `get_horn_system(ring, dic)`
Risa/Asir 側で構築された Horn 型超幾何方程式の定義方程式を SageMath 側に取得する. 数式処理システム間で微分作用素をやり取りする際, 単純に多項式環の元としてパースすると完全に展開されてしまい, パラメータの対応構造が見えなくなる. そこで, Asir 側には乗算記号を省略しない厳密な文字列表現 `H_horn_str2` を用意し SageMath 側では Symbolic Ring (`SR`) を用いてパースを行っている. これにより, 例えば Gauss の超幾何方程式の作用素は

$$-(a_2+\partial_4 x_4)(a_3+\partial_4 x_4)-(a_1+\partial_4 x_4)\partial_4$$

のように, 因数分解された構造（未展開表現）を保ったまま SageMath の数式ツリーとして保持され, パラメータの代入 (`.subs()`) などの解析が極めて容易になる.

■ `horn_contiguity_by_GKZ(A_start, A_end, ring, dic)`
SageMath の変数を用いてパラメータのシフトを指定し, Asir 側の `horn_contiguity_from_a_shift` を実行して結果を Sage の多項式/有理式オブジェクトとして受け取るラッパー関数である. Asir が出力したリスト形式の文字列をカンマで単純分割するとネスト構造が破壊されるため, 一旦 Asir 側の内部変数に結果を保持させ, インデックス指定で要素ごとに抽出・パースする堅牢な実装となっている.

例 5
SageMath で Gauss 超幾何関数の contiguity relation を計算する.

```
$ sage
```

```
 SageMath version 10.6, Release Date: 2025-03-31
 Using Python 3.11.13. Type "help()" for help.
```

```
sage: load("horn_contiguity.py")
sage: load_horn_contiguity()
sage: A = [[1,0,0,-1], [0,1,0,1], [0,0,1,1]]
sage: R, dic = set_horn_A(A)
[*] Asir に行列 A をセットしました。
[*] 自動生成されたリング変数: x4,dx4,a1,a2,a3
sage: horn_system = get_horn_system(R, dic)
[*] Asir から Horn system の定義方程式を取得します（未展開)...
sage: print(horn_system)
```

```
[(dx4*x4 + a2)*(dx4*x4 + a3) - (dx4*x4 + a1)*dx4]
sage: a1, a2, a3 = dic['a1'], dic['a2'], dic['a3']
sage: L = horn_contiguity_by_GKZ([a1, a2, a3], [a1, a2+1, a3], ring=R, dic=dic)
sage: print(L)  # contiguity
[-x4*dx4 - a2, 1, '[[a1,a2,a3], -> ,[a1,a2+1,a3]]', '[dx2,dx2,1,dx2]', 1, Multivariate Polynomial Ring in
x4, dx4, a1, a2, a3 over Rational Field]
sage:
```

5　Horn 型超幾何系の同型分類を遂行するプログラム

本節では, 構築した contiguity 導出プログラムを用いて, Horn 型超幾何系のパラメータ空間（$\mathbb{Z}^m$ の格子）全体にわたる同型分類を自動的に遂行する SageMath スクリプトについて解説する. 関連するプログラム群は representative-2025-09-13/ にて公開されている. 以下は `2025-07-23-representative.py` の解説である[1)].

5.1　同型分類の基本戦略

一般に, パラメータ空間 $\mathbb{Z}^m$ において, 全てのパラメータ間で隣接関係式（同型射）が常に存在するとは限らない. 同型が崩れる（作用素が存在しない, あるいは有理関数の分母が 0 になる）のは, 超幾何 b-関数の一次因子が 0 になる超平面（壁）の上のみである.

そこで, パラメータ空間を以下の手順で探索し, セル分解を行うことで同型分類の代表元 (representative) を決定する.

1. 一般の格子点において contiguity relation を計算し, その際の b-関数を取得する.
2. b-関数の因子から, 同型を隔てる「壁の方程式（一次式）」を抽出する.
3. その「壁」と現在の格子の交わり（部分格子）を計算し, 次元を一つ落として再帰的に探索を続ける.

数学的な詳細は [2, §6] を参照.

5.2　主要関数 `representative` の解説

この探索アルゴリズムの心臓部となるのが, 再帰呼び出しを用いて格子空間の分類を行う関数 `representative(BT, S, conti_func, vars)` である.

■ `representative(BT, S, conti_func, vars)`

- `BT` (Basis of Translation): 現在探索している格子の基底ベクトル. 初期値としては標準基底（例えば Appell F_1 なら 4×4 の単位行列）を与える.
- `S` (Shift/Origin): 現在の格子の起点（シフト量）.
- `conti_func`: Risa/Asir を呼び出して contiguity を計算するラッパー関数（例: `f1_contiguity` などの SageMath 側関数）.

[1)] `https://github.com/nobuki-takayama/contiguity/Horn` に `horn_contiguity_by_GKZ` 関数対応の最新版 representative 関数を公開した.

- vars: 探索するパラメータのリスト（例: [a, b, c]）.

処理のフロー:

1. **探索方向の決定:** 現在の起点 S から, 格子基底 BT の各方向に動かした隣接パラメータのリスト (old, new_up, new_down) を生成する.
2. **Contiguity と b-関数の計算:** 各隣接パラメータへの移動について conti_func を呼び出し, 往復の contiguity を計算する. これらから得られた b-関数の最小公倍数 (lcm) をとり, 現在地の同型性を支配する多項式 bf を決定する.
3. **「壁（超平面）」の抽出:** bf を因数分解 (factor(bf)) し, 個々の一次因子（例: $a - c + 1$ など）を取り出す. これらが同型が壊れる可能性のある「壁」である.
4. **壁（部分格子）への再帰的探索:** 各因子について, その一次式を係数ベクトル AH に変換し, 外部関数 affine_lattice_basis2 を呼び出す. これにより,「現在の格子 $(BT) + S$」と「壁 $AH = 0$」の交わりが計算され, 次元が 1 つ下がった新たな基底 BT_new と起点 T が得られる. この新しい部分格子上で再び representative を再帰的に呼び出し, 分類を 0 次元の点や交わりが変化しなくなるまで進める.

なお現在の実装は壁に囲まれた領域の内部のすべての格子点の間の同型が上記の格子基底 BT の方向で必ず作れるという仮定 (つまり BT **が格子点の Markov 基底**となっている) のもとで実装されている. この仮定を取り去った実装は現在開発中である.

5.3 本プログラムの意義

本プログラムの最大の特長は, Risa/Asir 側で行われる **D-加群のグレブナー基底による b-関数の代数解析的な計算**と, SageMath 側で行われる **格子と超平面の交差計算という多面体・整数計画的な計算**を完全に融合させている点にある.

人間が手計算で「このパラメータが整数のときは同型が成り立たない」と場合分けして分類していた煩雑な作業を,

1. Risa/Asir に contiguity を計算させて「同型の障害物（b-関数）」を吐き出させる.
2. SageMath にその障害物の方程式を解かせて,「障害物の上」という低い次元の新しい格子空間を作る.
3. その低い次元の空間で, 再び Risa/Asir に contiguity を計算させる.

というループに落とし込むことで, 完全に自動化されたアルゴリズム的同型分類を実現している. これにより, パラメータの数が多い複雑な Horn 型超幾何系に対しても, 網羅的かつ見通しの良い分類が可能となる.

例 6

Gauss 超幾何の場合の分類の様子.

```
sage: load("2025-07-23-representative.py")

# 探索の開始: 基底は 3x3 の単位行列、起点は (0,0,0)。
# つまり、退化のない一般の 3 次元パラメータ空間 [a, b, c] で探索をスタートする。
sage: f=representative([[1,0,0],[0,1,0],[0,0,1]],[0,0,0],gauss_contiguity,vars=[a,b,c])

['representative [BT,S,conti_func,vars]',
```

```
 [[[1, 0, 0], [0, 1, 0], [0, 0, 1]], [0, 0, 0],
 <function gauss_contiguity at 0x7f70376b2e80>, [a, b, c]]]

# 一般パラメータ空間での contiguity の計算。
# a -> a+1 方向へのシフトを計算し、作用素と b-関数を求めている。
calling  [[a, b, c], [a + 1, b, c]] # shift の指定
[-x*dx-a,1,-x*dx-a,[[a,b,c],->,[a+1,b,c]],[dx2,dx2,1,dx2]]
Done
  略
#  （この後、b, c 方向へのシフトも計算し、同型が壊れる障害物＝「b-関数の因子（壁）」を全て収集する）
  略

sage: print(f[0])
['representative [BT,S,conti_func,vars]', [[[1, 0, 0], [0, 1, 0], [0, 0, 1]], [0, 0, 0],
 <function gauss_contiguity at 0x7f70376b2e80>, [a, b, c]]]

sage: print(f[-1])  # 結果リストの最後
# ========== 【次元低下 1】 2 次元の「壁」への突入 ==========
# 先の計算で見つかった b-関数の因子（例えば最初のパラメータが 0 になる壁）と
# 格子の交わりを計算し、基底が 2 次元 [[0, 1, 0], [0, 0, 1]] に減少している。
[['representative [BT,S,conti_func,vars]', [[[0, 1, 0], [0, 0, 1]], (0, 0, 0),
 <function gauss_contiguity at 0x7f70376b2e80>, [a, b, c]]],

 [[[[0, a, b], [0, a + 1, b]], # 2 次元壁の上での shift の指定。最初の成分が 0 に退化している。
  [(-a + b - 1, 1), (a, 1)],    # このシフトにおける b-関数の因子（往路・復路の lcm）
 [-x*dx - a, 1],                # 往路 (up) の contiguity (分子, 分母)
 [x^2*dx - x*dx + a - b + 1, -a^2 + a*b - a]], # 復路 (down) の contiguity (分子, 分母)

 [[[0, a, b], [0, a, b + 1]],   # 2 次元壁の上でのもう一つの方向の shift
  [(b, 1), (a - b, 1)],         # 得られた b-関数の因子
 [-x*dx + dx - a + b, -a*b + b^2],
 [x*dx + b, 1]]],

 [(-a + b - 1, 1), (-a + b, 1), (b, 1), (a, 1)], 0], # この 2 次元壁の中で同型を壊す因子のリスト
  略

# ========== 【次元低下 2】 1 次元の「線」への突入 ==========
# 2 次元壁の中でさらに同型を壊す因子（例えば 2 番目のパラメータも 0 になるなど）と交わり、
# 基底が 1 次元 [[0, 0, 1]] に減少している。
,[['representative [BT,S,conti_func,vars]', [[[0, 0, 1]], (0, 0, 0),
 <function gauss_contiguity at 0x7f70376b2e80>, [a, b, c]]],

 [[[[0, 0, a], [0, 0, a + 1]], # 1 次元の線上での shift の指定。パラメータが [0, 0, a] に退化。
  [(a, 2)],                     # 得られた因子は a^2
  [x*dx - dx - a, -a^2],        # この極端な退化空間での contiguity 作用素
 [x*dx + a, 1]]],

 [(a, 2)], 0],

# ========== 【次元低下 3】 0 次元の「点」への到達 ==========
# 最後に残った因子 'a' が 0 になる点、つまり原点 (0, 0, 0) に到達し、
# これ以上探索する次元がない（0-dim face）ため、このブランチの再帰探索が終了する。
 ['0-dim face', (0, 0, 0), (a, 2)]]]
```

謝辞:
Google AI アシスタント Gemini の gem 機能は特定の目的や役割に合わせて Gemini の挙動をカスタマイズ, 保存できる機能である. この原稿の執筆およびソフトウエア作成にあたり, 我々が開発した gem: Risa/Asir Programming Assistant v5 [1] を援用した.
This work was supported by the Research Institute for Mathematical Sciences, an International Joint Usage/Research Center located in Kyoto University.

参 考 文 献

[1] https://gemini.google.com/gem/1AtXP96KN9hUnUYgO9HnI4HoXB17wovxA?usp=sharing
なお Risa/Asir Programming Assistant の最新版は以下の v7 である.
https://gemini.google.com/gem/1-OoG766iQlM7YbW9hAS1aYMebKOpGEKc?usp=sharing

[2] Hiromasa Nakayama, Nobuki Takayama, Comprehensive Restriction Algorithm for Hypergeometric Systems, https://arxiv.org/abs/2510.05104

[3] M.Saito, Isomorphism Classes of A-Hypergeometric Systems, Compositio Mathematica 128, (2001) 323–338.