1 環の定義

```
import("yang.rr");
yang.define_ring(["partial",[x,y]]);
F=yang.mul(dx,x);
```

微分作用素環 $\mathbf{Q}(x,y)\langle\partial_x,\partial_y\rangle$ を定義. `yang.mul` はこの環で掛け算をする. `F` に $\partial_x x = x\partial_x + 1$ を代入.

```
import("yang.rr");
yang.define_ring([x,y]);
F=yang.mul(dx,x);
```

$\theta_x = x\partial_x$, $\theta_y = y\partial_y$ (Euler 作用素) とする. 微分作用素環 $\mathbf{Q}(x,y)\langle\theta_x,\theta_y\rangle$ を定義. `yang.mul` はこの環で掛け算をする. `F` に $\theta_x x = \theta_x x + x$ を代入.

2 reduction

```
import("yang.rr");
yang.define_ring(["partial",[x,y]]);
yang.reduction(x*<<1,0>>+<<0,0>>,[(x+1)*<<1,0>>+<<0,0>>]);
```

Reduction (割り算) は分散多項式表現でおこなう. `<<i,j>>` が $\partial_x^i\partial_y^j$ に対応. 分散多項式表現と普通の表現の変換は `dp_ptod`, `dp_dtop` 関数で行う (asir manual 参照). この例は $f = x\partial_x + 1$ の $g = (x+1)*\partial_x + 1$ での reduction を上のコードは計算する.

$$\begin{aligned} f &\rightarrow f - \frac{x}{x+1}g \\ &= 1 - \frac{x}{x+1} \\ &= \frac{1}{x+1} \end{aligned}$$

出力は `[(1)*<<0,0>>,x+1]` となる. 0 番目の成分が分子, 1 番目の成分が分母.

複数の元で reduction する場合は 2 番目の引数のリストに複数の元を与えれば良い.

```
import("yang.rr");
yang.define_ring(["partial",[x,y]]);
V=[dx,dy];
F1=dp_ptod(dx^2+y^2,V);
F2=dp_ptod(dy^2+x^2,V);
F3=dp_ptod(x*dx-y*dy,V);
yang.reduction(F1,[F2,F3]);
```

出力は [0,1] つまり $0/1 = 0$ である. これは F1 を [F2,F3] で reduction した結果が 0 であることを意味する.

3 Gröbner basis と Pfaffian 方程式

```
import("yang.rr");
yang.define_ring(["partial",[x,y]]);
V=[dx,dy];
F1=dp_ptod(dx^2+y^2,V);
F2=dp_ptod(dy^2+x^2,V);
G=yang.buchberger([F1,F2]);
Std=yang.stdmon(G);
Pf=yang.pfaffian(map(dp_ptod,Std,V),G);
```

F1, F2 が生成する左イデアルのグレブナー基底を yang.buchberger で計算する. 出力は以下のようになる.

```
[2596] G[0];
[(-4*x)*<<1,0>>+(4*y)*<<0,1>>,-4*x]
[2597] G[1];
[(1)*<<0,2>>+(x^2)*<<0,0>>,1]
[2598] Std;
[dy,1]
[2599] Pf[0];
[ (1)/(x) -y*x ]
[ (y)/(x) 0 ]
[2600] Pf[1];
[ 0 -x^2 ]
[ 1 0 ]
```

Std が standard monomials, Pf[0] は Std を縦ベクトル $Q = (\partial_y, 1)^T$ とし

て見たときの

$$\partial_x Q - P_0 Q \equiv 0 \mod I$$

となる 2×2 行列 P_0. ここで I は F1, F2 が生成する左イデアル. Pf[1] は Std を縦ベクトル $Q = (\partial_y, 1)^T$ として見たときの

$$\partial_y Q - P_1 Q \equiv 0 \mod I$$

となる 2×2 行列 P_1.

定理 6.2.3 をこの場合に計算で確かめる.

```
 map(diff,Pf[0],y)+Pf[0]*Pf[1]
-(map(diff,Pf[1],x)+Pf[1]*Pf[0]);
```

結果は 0 行列.

G がグレブナー基底. 要素は reduction の出力形式に同じ. initial (または leading term) は asir の組み込み関数 dp_hm などを用いて取り出せる.

4 微分差分作用素環

以下の例ではより簡略化したコマンド yang.gr, yang.nf, yang.pf を用いる.

たとえば差分作用素環 $\mathbf{Q}(a,c)\langle T_a, T_c\rangle$, $T_a a = (a+1)T_a$, $T_c c = (c+1)T_c$ を定義するには次のように環を定義する.

```
Ring=["difference",[[a,1],[c,1]]];
yang.define_ring(Ring);
```

[a,1] の 1 は シフト作用素 da($= T_a$) のシフトの数を示す. この場合は yang.mul(da,a) の結果は (a+1)*da となる.

微分差分作用素環を定義するには次のようにすればよい.

```
Ring=["difference",[[a,1]],"partial",[z]];
yang.define_ring(Ring);
```

一般超幾何関数 ${}_1F_1(a,c;z)$ の満たす微分差分方程式系の例.

```
import("yang.rr")$

Ring=["difference",[[a,1]],"partial",[z]];
yang.define_ring(Ring);

// contiguity of 1F1(a,c;z)
L1 = a*da-(z*dz+a);
L2 = z*dz^2+(c-z)*dz-a;

L=[L1,L2];
G=yang.gr(L);
S=yang.stdmon(G);
Pf=yang.pf(S,G);

yang.nf(da*dz,G);
\begin{screen}
\begin{verbatim}
```

S は [dz,1] となる. I を L1, L2 が生成する左イデアルとする時 Pf[0],Pf[1] は次の Pfaffian 方程式の係数 P_0, P_1 である.

$$T_a \begin{pmatrix} \partial_z \\ 1 \end{pmatrix} \equiv P_0 \begin{pmatrix} \partial_z \\ 1 \end{pmatrix} \bmod I, \quad \partial_z \begin{pmatrix} \partial_z \\ 1 \end{pmatrix} \equiv P_1 \begin{pmatrix} \partial_z \\ 1 \end{pmatrix} \bmod I$$

Pfaffian 方程式の係数は normal form の計算により行う. たとえば, P_0 の一行目の係数の計算例が `yang.nf(da*dz,G);`