Maximaの 規則を利用する話

東芝インフォメーションシステムズ株式会社

横田博史

規則って何？

- Maximaのある条件を満たす式に対する変換規則
  - tan(x) -> sin(x)/cos(x)等々
- rulesを入力するとMaxima上で定義された規則の一覧が表示され, disprule(<規則名>)で内容が表示される

```
(%i1) rules;
(%o1) [trigrule0, trigrule1, trigrule2, trigrule3, trigrule4,
htrigrule1,
                                                htrigrule2, htrigrule3,
htrigrule4]
(%i2) disprule(trigrule0);
                                                   sin(a)
(%o2)                    trigrule0 : tan(a) -> ------
                                                   cos(a)
(%i3)
```

規則について

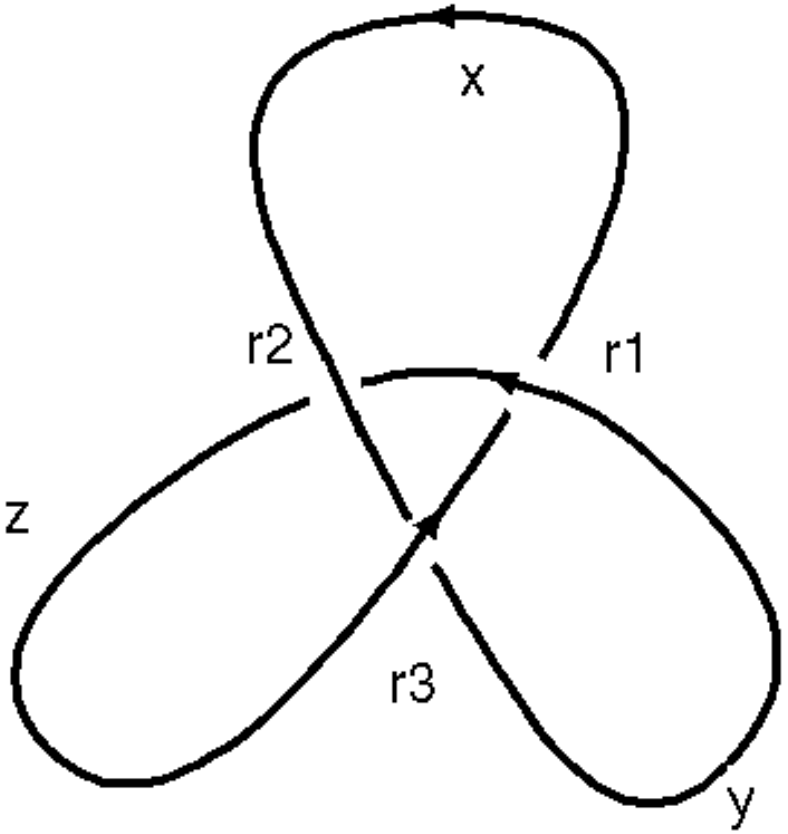
- Maximaは式の並びに対して, 規則を基にした処理が行える
- Maximaの規則は利用者が自由に定義出来る
  - defrule函数やlet函数を利用して規則を定義
- 規則は入力した式に直ちに適用されない
  - apply1やletsimp等の函数で式を評価してはじめて適用される

defruleを使った規則の定義

- defruleによる規則の定義では, 式の並びを指定する際に予め宣言した変数を用いなければ, 規則の定義で利用した式の並びに対してのみ規則が適用される
- 一般的な式の並びの指定ではmatchdeclare函数で宣言した変数を用いる

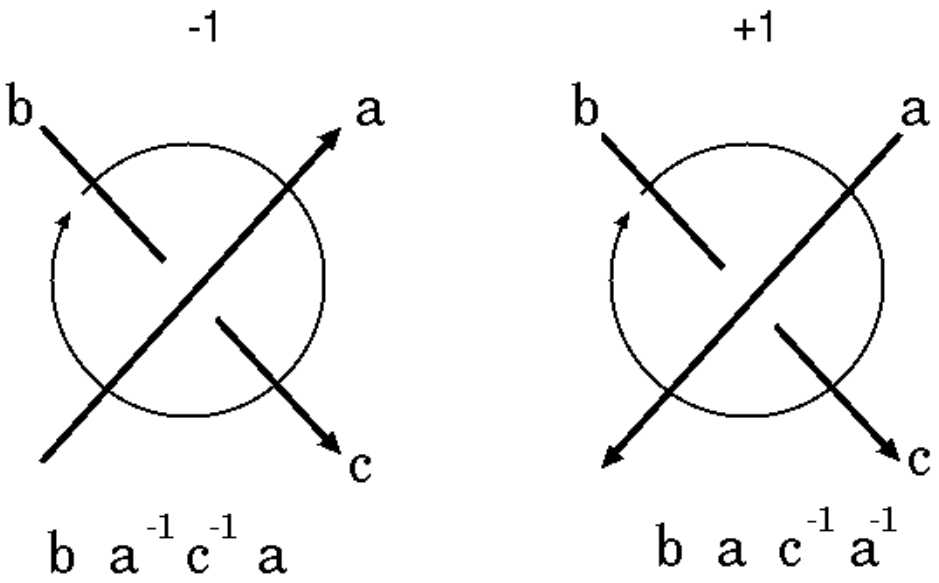
```
(%i3) infix("neko")$
(%i4) defrule(NEKOProd,a neko (c*b),a neko c + a neko b);
(%o4)          NEKOProd : a neko (b c) -> a neko c + a neko b
(%i5) apply1( a neko(b*c));
(%o5)                          a neko (b c)
(%i6) apply1( a neko(b*c),NEKOProd);
(%o6)                       a neko c + a neko b
(%i7) apply1( x neko(y*z),NEKOProd);
(%o7)                          x neko (y z)
```

例:結び目のアレキサンダー多項式



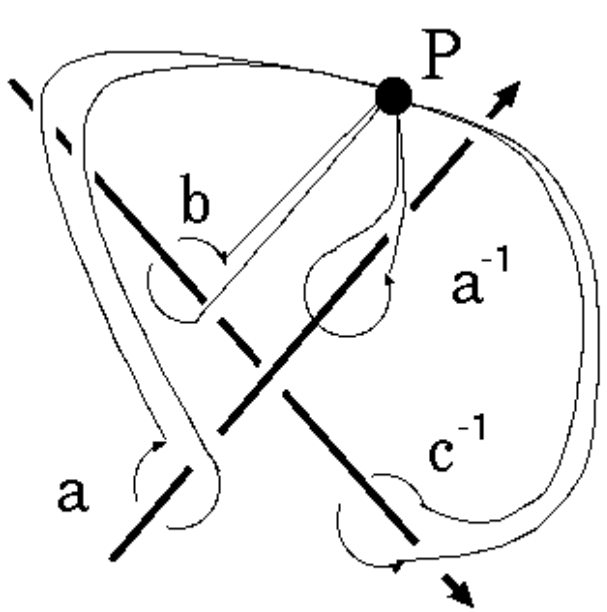
- 結び目群(結び目の補空間の基本群)の不変量
- Z[t^(-1),t]のイデアルの生成元
- フォックスの微分子によるヤコビアンから計算可能

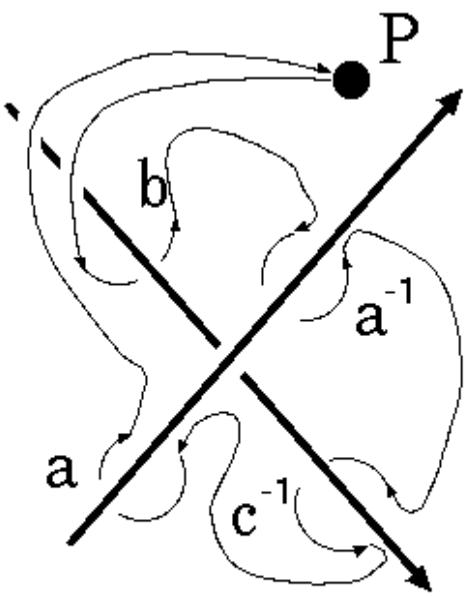
結び目群のヴィルティンガー表示



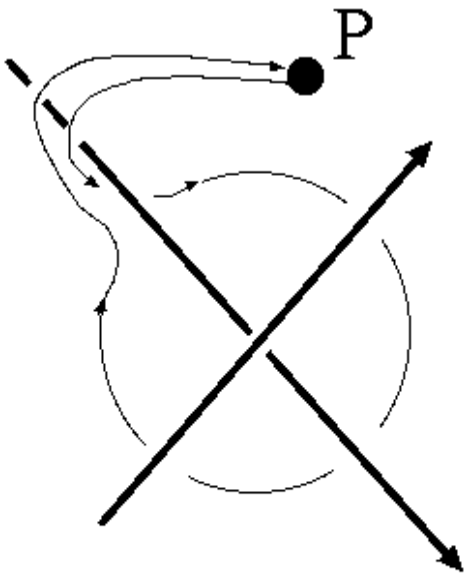
- 結び目の射影図から求められる
- 生成元は上道
- 関係子は交差点で決定される(左図参照)

ヴィルティンガー表示の意味





$b\ a^{-1}\ c^{-1}\ a$



$b\ a^{-1}\ c^{-1}\ a$

フォックスの微分子

Fox の微分子 Δ

- $\Delta(x+y) = \Delta(x) + \Delta(y)$
- $\Delta(xy) = t(y)\Delta(x) + x\Delta(y)$

Fox の微分子 Δ の性質

- $\Delta(n) = 0, n \in \mathbb{Z}$
- $\Delta(x^n) = \sum_{i=0}^{n-1} x^i \Delta(x), n \in \mathbb{N}$
- $\Delta(x^{-n}) = -\sum_{i=-n}^{-1} x^i \Delta(x), n \in \mathbb{N}$

- Δ -> d/dx1,..., d/dxn (x1,..,xn:生成元)
- dxj/dxi= δ ij

計算手順

- 生成元x1,..,xn,関係子r1,..,rn
- d/dxiを[r1,..,rn]に作用させてヤコビアンを計算
- 非可換積を可換積に置換する
- 生成元を変数tで置き換える
- ヤコビアンの余因子行列を計算
- 各成分の最大公約因子がアレクサンダー行列

Maximaでの表現

- 語はMaximaの非可換積(dot積.)とその冪を利用
  - x.y.z^^2.w^^(-1)
- フォックスの微分子は語に対して定義
- prefix函数を用いて前置式演算子として宣言
- declare函数を用いて線形性を宣言

```
(%i1) prefix("D_fox:");
(%o1)                                D_fox:
(%i2) declare("D_fox:",linear);
(%o2)                                 done
```

演算子と被演算子の束縛力

- Maximaでは演算子の被演算子に対する束縛力を指定可能
- 指定しない場合，束縛力はデフォルトで180
  - +の場合: 左右の束縛力は100
  - *の場合:左束縛力は120のみ

```
(%i60) infix("mike",130,130)$
(%i61) 1+3 mike 4+5;
(%o61)                          3 mike 4 + 6
(%i62) infix("mike",90,130)$
(%i63) 1+3 mike 4+5;
(%o63)                          4 mike 4 + 5
```

並びの設定

- matchdeclare函数を用いて並びの照合を定義
- 同時に並びの照合の述語函数を指定する
- 特に述語函数の必要がなければtrueを指定

述語函数wordp

```
wordp(w):=if atom(w) then true
else if member(inpart(w,0),[".","^^","","*"]) then true
else false;
```

```
(%i3) matchdeclare([_x,_y],wordp)$
```

defruleによる規則の設定

- defrule(<規則名>,<式の並び>,<変換後の式>)
- defruleによる規則はapply1,apply2,applyb1函数で与式に適用

```
(%i12) defrule(Dfox_Prod,D_fox:(_x._y),D_fox:_x*t1(_y)+_x.D_fox:_y);
(%o12) Dfox_Prod : D_fox: (_x . _y) -> D_fox: _x t1(_y) + _x . D_fox: _y
(%i13) apply1(D_fox:(x.y),Dfox_Prod);
(%o13)                  D_fox: x t1(y) + x . D_fox: y
```

非可換積演算子の結合律

- 非可換積演算子としてMaximaのdot演算子を利用
- dot演算子は結合律をデフォルトで満たす
- x.(y.z)は結合律を満たす場合にx.y.zとなる。内部表現では(. x y z)となりフォックスの微分子で 処理が出来ない

```
(%i14) D_fox:((x.y).z);
(%o14)                    D_fox: (x . y . z)
(%i15) dotassoc:false;
(%o15)                          false
(%i16) D_fox:((x.y).z);
(%o16)                    D_fox: ((x . y) . z)
```

dotassocをfalseに設定して, 結合律を除外

fox.mc

AlexanderPoly.mc

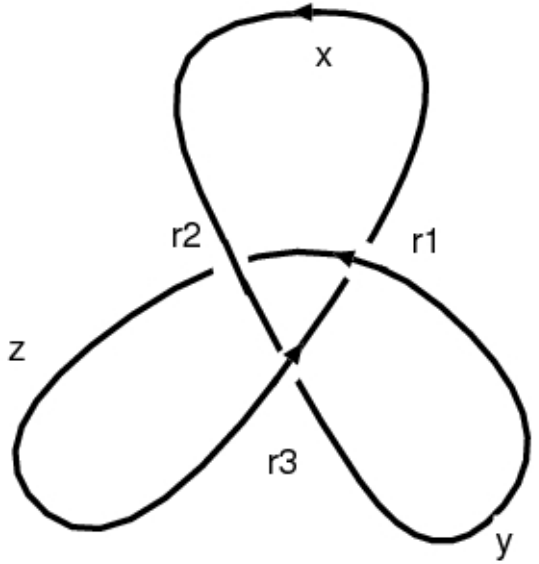
maxima-init.mac

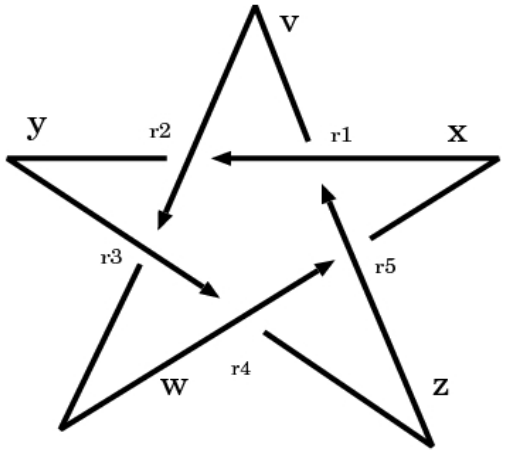
- Maximaの初期化ファイル
- 函数の読込や大域変数の設定等が行える
- Maximaを起動するディレクトリ上に置く

```
/* maxima-init.macの内容例 */

load("fox.mc")$
load("AlexanderPoly.mc")$
```

計算例





- t^2-t+1
- t^4-t^3+t^2-t+1

Maximaマニュアルの事

- 個人の頁http://www.bekkoame.ne.jp/~ponpoko で修正版を公開予定(連休前?)