Details of WuRittSolva

Standard Application Package for Wu-Ritt Process

By
Huashan Liu,
Department of Mathematics, Tianjin Polytechnic University, P.R.C.
E – Mail : liukaitianpidi @sina.com
HomePage : http : // magicm .51.net
Instructed by
Prof. Huang Dongwei
Department of Mathematics, Tianjin Polytechnic University, P.R.C.

March.4th, 2005 – March.15th, 2005

Section WRS_I: Some Functions Defined for List Manipulation

■ Element postion

```
MaxElementPos[vector_] := Position[vector, Max[vector]][[1, 1]];
                          位置              最大值

MinElementPos[vector_] := Position[vector, Min[vector]][[1, 1]];
                          位置              最小值
```

■ Check Constants in List

```
IsConstantsIn[list_] :=
 Module[{tmp, res},
 模块
  tmp = Select[list, IntegerQ];
        选择         整数判定
  res = If[Length@tmp > 0, True, False];
        …  长度              真    假
  Return[res];
  返回
 ]
```

■ Fix the Variables of *poly* wrt *const* as constansts

```
PolyVariables[poly_, const___] := Module[{tmp}, tmp = Complement[Variables[poly], const];
                                  模块                 补集       变量
  Return[tmp];]
  返回
```

Section WRS_II:Computation of Class and MainVariable

■ Computation of Class wrt ord

```
FixedClass[poly_, ord_] :=
 Module[{},
 模块
  tmp = Exponent[poly, #] & /@ ord;
        最高次数
 mid = Last[Flatten[Position[tmp, Last@Select[tmp, Positive]]]];
      最…  压平     位置             最…  选择             正数判定
Return[ord[[mid]][[2]]];
返回
 ]

AutoClass[poly_, const___] :=
Module[{tmp, mid},
模块
tmp = PolyVariables[poly, const];
(*mid=Exponent[poly,#]&/@tmp;
res=MaxElementPos[mid];*)
res = Last[tmp][[2]];
      最后一个
Return[res];
返回
]

Class[poly_, ord_: False, const___] :=
                  假
 Module[{tmp},
 模块
  tmp = If[ArrayQ[ord], FixedClass[poly, ord], AutoClass[poly, const]];
        … 数组判定
  Return[tmp];
  返回
 ]
```

■ Computaion of MainVariable wrt ord

```
FixedMainVariable[poly_, ord_] :=
 Module[{},
 模块
  tmp = FixedClass[poly, ord];
  Return[ord[[tmp]]];
  返回
 ]

AutoMainVariable[poly_, const___] :=
 Module[{tmp, mid},
 模块
  tmp = AutoClass[poly, const];
  mid = PolyVariables[poly, const];
  Return[mid[[tmp]]];
  返回
 ]
```

```
MainVariable[poly_, ord_ : False, const___] :=
                          假
 Module[{tmp},
 模块
  tmp = If[Not[ArrayQ[ord]], AutoMainVariable[poly, const],
         …    …     数组判定
    FixedMainVariable[poly, ord]];
  Return[tmp];
  返回
 ]
```

Some Related Functions

```
MainVariableExponent[poly_, ord___] :=
 Module[{tmp},
 模块
  tmp = Exponent[poly, MainVariable[poly, ord]];
        最高次数
  Return[tmp];
  返回
 ]

Initial[poly_, ord___, const___] :=
 Module[{tmp, mid, res},
 模块
  tmp = MainVariable[poly, ord, const];
  mid = Exponent[poly, tmp];
        最高次数
  res = Coefficient[poly, tmp^mid];
        系数
  Return[res];
  返回
 ]

LeadCoefficient[poly_, var_] :=
 Module[{tmp},
 模块
  tmp = Last@CoefficientList[poly, var];
        最…   系数列表
  Return[Expand[tmp]];
  返回   展开
 ]
```

```
IsPolyReduced::"Reduced" = "`1` is  reduced to `2`.";
IsPolyReduced::"NotReduced" = "`1` is not reduced to `2`.";
IsPolyReduced[lhspoly_, rhspoly_, ord___, const___] :=
 Module[{tmp},
 模块
  tmp = If[Class[rhspoly, ord, const] < Class[lhspoly, ord, const] ||
        如果
     (Class[rhspoly, ord, const] == Class[lhspoly, ord, const] &&
       MainVariableExponent[rhspoly, ord, const] <
        MainVariableExponent[lhspoly, ord, const]),
    (*Message[IsPolyReduced::"Reduced",rhspoly,lhspoly];*)True,
                                                          真
    (*Message[IsPolyReduced::"NotReduced",rhspoly,lhspoly];*)False];
                                                             假
  Return[tmp];
  返回
 ]
```

Section WRS_III:WuRittSolva Main Procedure

■ Poly2Poly Pseudo Division: Remainder & Resolution

```
PseudoRemainder[lhspoly_, rhspoly_, ord___, const___] :=
 Module[{polyquo, polyrem, tmppolyquo, tmppolyrem, polytmplcm, polypseudoquo,
 模块
   polypseudorem, i = 0},
  polyquo = Together@PolynomialQuotient[lhspoly, rhspoly,
            归并      多项式的商
     MainVariable[rhspoly, ord, const]];
  polyrem = Together@PolynomialRemainder[lhspoly, rhspoly,
            归并      多项式余数
     MainVariable[rhspoly, ord, const]];

  tmppolyquo = Denominator[polyquo];
               分母
  tmppolyrem = Denominator[polyrem];
               分母

  polytmplcm = PolynomialLCM[tmppolyquo, tmppolyrem];
               多项式的最小公倍式
  polypseudorem = polytmplcm * polyrem;

  Return[polypseudorem];
  返回
 ]
```

```
PseudoResolution[lhspoly_, rhspoly_, ord___, const___] :=
 Module[{polyquo, polyrem, tmppolyquo, tmppolyrem, polytmplcm, polypseudoquo,
 模块
   polypseudorem, i = 0},
  polyquo = Together@PolynomialQuotient[lhspoly, rhspoly,
            归并      多项式的商
     MainVariable[rhspoly, ord, const]];
  polyrem = Together@PolynomialRemainder[lhspoly, rhspoly,
            归并      多项式余数
     MainVariable[rhspoly, ord, const]];
  tmppolyquo = Denominator[polyquo];
               分母
  tmppolyrem = Denominator[polyrem];
               分母
  polytmplcm = PolynomialLCM[tmppolyquo, tmppolyrem];
               多项式的最小公倍式
  polypseudoquo = polytmplcm * polyquo;
  polypseudorem = polytmplcm * polyrem;

  (*Print[{polytmplcm,tmppolyquo,tmppolyrem}];
  While[i≤10&&
    Expand@Simplify[Initial[rhspoly,ord,const]^i*lhspoly]=!=
     Expand@Simplify[polypseudoquo*rhspoly+polypseudorem],
   i++;
   Print[{Initial[rhspoly,ord,const]^i,Expand[Initial[rhspoly,ord]^i lhspoly],
     Expand[polypseudoquo*rhspoly+polypseudorem],i}];
  ];
  i=Max[{MainVariableExponent[rhspoly,ord,const]-MainVarExponent[rhspoly,ord,const]+1,
     0}];*)
  Return[Expand@{(*Initial[rhspoly,ord,const],i*)polytmplcm, polypseudoquo,
  返回    展开
     polypseudorem}];
 ]
```

Test for above procedure

```
ff = x1^2 x2^3 - x2;
gg = x1^3 x2 - 2;

PseudoResolution[ff, gg, {x1, x2}]
```

Poly2Polyset Pseudo Division:Remainder & Resolution

```
AuxPseudoRemainder[poly_, polyset_, ord___, const___] :=
 Module[{len, tmp, mid, i = 1},
 模块
  len = Length@polyset;
           长度
  tmp = poly;
  While[i <= len, (mid = Evaluate@PseudoRemainder[tmp, polyset[[i]], ord, const];
  While循环                   计算
    tmp = mid;
    (*Print@tmp;*)
    i++;
   )
  ];
  Return[tmp];
  返回
 ]

AuxPseudoResolution[poly_, polyset_, ord___, const___] :=
 Module[{len, tmp, mid, i = 1},
 模块
  len = Length@polyset;
           长度
  tmp = poly;
  While[i <= len, (mid = PseudoResolution[tmp, polyset[[i]], ord, const];
  While循环
    tmp = Last@mid;
          最后一个
    (*Print@mid;*)
    i++;
   )
  ];
  Return[mid];
  返回
 ]
```

PolySet2PolySet Pseudo Division:Remainder & Resolution

```
PseudoRemainderSet[lhspolyset_, rhspolyset_, ord___, const___] :=
 Module[{tmp},
 模块
  tmp = AuxPseudoRemainder[#, rhspolyset, ord, const] & /@ lhspolyset;
  Return[tmp];
  返回
 ]

PseudoResolutionSet[lhspolyset_, rhspolyset_, ord___, const___] :=
 Module[{tmp},
 模块
  tmp = AuxPseudoResolution[#, rhspolyset, ord, const] & /@ lhspolyset;
  Return[tmp];
  返回
 ]
```

Checking Ascending Set

```
AscSetCheck[polyset_, ord___] :=
 Module[{tmp, tmpset},
 模块
  tmpset = Sort@polyset;
           排序
  tmp = Variables /@ tmpset;
        变量
  If[IsConstantsIn[tmpset] || MemberQ[Complement[ord, #] & /@ tmp, ord],
  如果                              成员判定   补集
   Return[False], Return[tmpset]];
   返回    假       返回
  Return[{True, tmpset}];
  返回     真
 ]
```

Test for AscSetCheck

```
poly1 = x1^2 - 2 x2^3 x3 + 1;
poly2 = x1 x2 + x3^2;
poly3 = 3 x1^2 - 2 x3^2;
polyset = {poly1, poly2, poly3};
ord = {x1, x2, x3};

AscSetCheck[polyset, ord]

ff = x1^2 x2^3 - x2;
gg = x1^3 x2 - 2;

PseudoResolution[ff, gg, {x2, x1}]

LeadCoefficient[ff, x1]

PseudoRemainder[ff, gg]
```

Section WRS_IV:WuRittSolva Characteristic Set

- **Split Polynomials into Groups wrt Main variables**

```
SplitPolySet[polyset_, ord___, const___] :=
 Module[{lst, tmplst, mid, midlst, res},
 模块
  lst = {MainVariable[#, ord, const], #} & /@ polyset;
  tmplst = Transpose[Sort[lst]];
           转置      排序
  mid = Split@tmplst[[1]];
        分割
  midlst = Length /@ mid;
           长度
  res =
   Table[Take[tmplst[[2]], {1 + Apply[Plus, Take[midlst, i - 1]],
   表格  选取                    应用   加    选取
      Apply[Plus, Take[midlst, i]]}], {i, 1, Length@midlst}];
      应用  加    选取                        长度
  Return[res];
  返回
 ]
```

- **BasicSet or MiniSet of polyset wrt to ord with const as constants**

```
BasicSet[polyset_, ord___, const___] :=
 Module[{tmp, mid, res},
 模块
  tmp = SplitPolySet[polyset, ord, const];
  mid =
   MinElementPos /@
    (Exponent[#, MainVariable[#, ord, const]] & /@ SplitPolySet[polyset, ord, const]);
     最高次数
  res = Table[tmp[[i]][[mid[[i]]]], {i, 1, Length[mid]}];
        表格                              长度
  Return[res];
  返回
 ]
```

■ CharacteristicSet of polyset wrt ord with const as constatns

```
IsNewComponent[lhspoly_] := Module[{tmp, res}, tmp = FactorList[lhspoly];
                            模块                      因子列表
  res = If[Length[tmp] > 2,
        ··· 长度
    Table[
    表格
     Print[{StyleForm[StringJoin["A New Component:", ToString[i]],
     打印             连接字符串                          转换为字符串
        FontColor → RGBColor[0, 0.6, 0]],
        字体颜色     RGB颜色
       StyleForm[tmp[[2 + i]][[1]]^tmp[[2 + i]][[2]], FontColor → RGBColor[1, 0, 0.5]]}],
                                                      字体颜色     RGB颜色
     {i, 1, Length[tmp] - 2}];
            长度
    True, False];
    真    假
  Return[res];]
  返回
```

```
CharacteristicSet[polyset_, switch_ : False, cnt_ : 20, ord___, const___] :=
                                      假
 Module[{tmppolys, tmpbass, tmprems, mid, i = 1, oldtmpbass, tmp, j = 0},
 模块
  tmppolys = polyset;
  tmpbass = BasicSet[tmppolys];
  mid = PseudoRemainderSet[tmppolys, tmpbass];
  tmprems = Union[Select[mid, Not[IntegerQ[#]] &]];
            并集  选择        …   整数判定
  While[Length[tmprems] > 0 && Length[tmp] < Length[polyset] && i < cnt + 1,
  Whil… 长度                   长度          长度
   tmpbass = BasicSet[tmppolys];
   If[Length[oldtmpbass] == Length[tmpbass], tmp = oldtmpbass - tmpbass;];
   … 长度                    长度
   mid = PseudoRemainderSet[tmppolys, tmpbass];
   tmprems = Union[Select[mid, Not[IntegerQ[#]] &]];
             并集  选择        …   整数判定
   tmppolys = Union[tmpbass, tmprems];
              并集
   If[switch,
   如果
    Print[{StyleForm[StringJoin["CS_STEP:", ToString[i]], FontColor → RGBColor[0, 0, 1]],
    打印             连接字符串               转换为字符串     字体颜色    RGB颜色
      StyleForm[Simplify@tmpbass, FontColor → RGBColor[1, 0, 0]]}]];
                化简               字体颜色    RGB颜色
   (*Print[{tmprems,i}];*)(*Print[{tmppolys,tmpbass,tmprems,i}];*)
   Map[If[IsNewComponent[#], ++j] &, tmpbass];
   映射 如果
   oldtmpbass = tmpbass;
   i++;];
  If[j > 0,
  如果
   Print[{StyleForm[StringJoin["Total ", ToString[j], " New Components Discovered"],
   打印             连接字符串              转换为字符串
      Subsubtitle, FontColor → RGBColor[0, 0.6, 0]]}]];
                   字体颜色    RGB颜色
  Return[tmpbass];]
  返回
```

New Characteristic Set

```
intTest[lst_] := Module[{tmp},
                 模块
  tmp = Select[lst, Not[IntegerQ[#]] &];
        选择        …   整数判定
  Return[tmp];
  返回
 ]
```

```
NewCharacteristicSet[polyset_, switch_ : False, ord___, const___] :=
                                          假
 Module[{tmppolys, tmpbass, tmprems, mid, i = 1, oldtmpbass, tmp, j = 0, midres,},
 模块
  tmppolys = polyset;
  tmpbass = BasicSet[tmppolys];
  mid = PseudoRemainderSet[tmppolys, tmpbass, ord, const];
  tmprems = Union[intTest[mid]];
            并集
  (*midres=Length@intTest[PseudoRemainderSet[polyset,Reverse@tmpbass,ord,const]];*)
  While[Length[tmprems] > 0(*&&midres>=0*) && i < 10 000
  Whil··· 长度
   (*SHOULD BE MODIFYED FOR COMPLEX SITUATION,i.e. Infinity*),
   tmpbass = BasicSet[tmppolys];
   mid = PseudoRemainderSet[tmppolys, Reverse@tmpbass, ord, const];
                                      反向
   tmprems = Union[intTest[mid]];
             并集
   tmppolys = Union[tmpbass, tmprems];
              并集
   If[switch,
   如果
    Print[{StyleForm[StringJoin["CS_STEP:", ToString[i]], FontColor → RGBColor[0, 0, 1]],
    打印             连接字符串            转换为字符串  字体颜色      RGB颜色
      StyleForm[Simplify@tmpbass, FontColor → RGBColor[1, 0, 0]]}]];
                化简             字体颜色      RGB颜色
   Map[If[IsNewComponent[#], ++j] &, tmpbass];
   映射 如果
   (*midres=Length@intTest[PseudoRemainderSet[polyset,Reverse@tmpbass,ord,const]];*)
   i++;
  ];
  If[j > 0,
  如果
   Print[{StyleForm[StringJoin["Total ", ToString[j], " New Component(s) Discovered"],
   打印             连接字符串           转换为字符串
     Subsubtitle, FontColor → RGBColor[0, 0.6, 0]}]];
                  字体颜色     RGB颜色
  Return[tmpbass];
  返回
 ]
```

■ OLD Characteristic Form

```
InsertDoubleZeros[lst_, checklst_] :=
 Module[{i = 1, tmp, mid},
 模块
  mid = lst;
While[i <= Length[checklst],
While循环    长度
   tmp = Insert[mid, "00", checklst[[i]][[2]]];
         插入
   i++;
   mid = tmp;
   (*Print@mid*)
  ];
  Return[mid]
  返回
 ]

HasInsertDoubleZeros[varlst_] :=
 Module[{tmp, mid, res, i},
 模块
  tmp = Table[Complement[Union @@ Take[varlst, i], Flatten@Take[varlst, {i + 1, i + 1}]],
        表格   补集        并集     选取                压平    选取
    {i, 1, Length[varlst] - 1}];
           长度
  mid = Table[InsertDoubleZeros[Rest[varlst][[i]], tmp[[i]]],
        表格                    去掉第一个
    {i, 1, Length[Rest[varlst]]}];
           长度    去掉第一个
  res = Insert[mid, InsertDoubleZeros[First@varlst,
        插入                          第一个
     Complement[Apply[Union, varlst], First@varlst]], 1];
     补集        应用   并集             第一个
  Return[res];
  返回
 ]
```

```
CharacteristicForm[cspolyset_, const___] :=
 Module[{vars, maxlen, tmp, mid, res, tmpcs},
 模块
  vars = HasInsertDoubleZeros@Map[PolyVariables[#, const] &, cspolyset];
                                映射
  maxlen = Max[Length /@ vars];
              … 长度
  tmp = Map[PadRight[#, maxlen, "00"] &, vars];
        映射 右填充
  tmpcs = MapThread[ExpandAll[#1, Last[#2]] &, {cspolyset, vars}];
          映射线程   展开全部      最后一个
  mid = MapThread[{#1, #2} &, {tmpcs, tmp}];
        映射线程
  (*Print@{Length@vars,maxlen,tmp};*)
  res = MatrixForm[mid];
        矩阵格式
  Return[res];
  返回
 ]
```

■ New Characteristic Form

```
toDoubledZeros[cspolyset_, const___] :=
 Module[{csvarlst, varlst, i = 1},
 模块
  csvarlst = Map[PolyVariables[#, const] &, cspolyset];
             映射
  varlst = Union@Flatten[csvarlst];
           并集   压平
  tmp =
   Select[Flatten[Table[If[Not@MemberQ[csvarlst[[i]], varlst[[j]]], {i, j}],
   选择   压平    表格   … …  成员判定
      {i, 1, Length[csvarlst]}, {j, 1, Length[varlst]}], 1], # =!= Null &];
             长度                     长度                        空
  For[i = 1, i <= Length[tmp], i++,
  For循环        长度
   csvarlst = Insert[csvarlst, "00", tmp[[i]]];
              插入
  ];
  Return[csvarlst];
  返回
 ]

NewCharacteristicForm[cspolyset_, const___] :=
 Module[{tmp, mid, res, tmpcs},
 模块
  tmp = toDoubledZeros[cspolyset, const];
  mid = MapThread[{#1, #2} &, {cspolyset, tmp}];
        映射线程
  res = MatrixForm[mid];
        矩阵格式
  Return[res];
  返回
 ]
```

■ WuRitt Equations Solvor

```
WuRittEqnsSolve[cspolyset_, const___] :=
 Module[{tmp, mainvars, sollen, j = 2, mid, result, cslen = Length@cspolyset, sols},
 模块                                                                  长度

   sols = Table[0, {i, 1, Length[cspolyset]}];
          表格              长度

   mainvars = Last /@ (PolyVariables[#, const] & /@ cspolyset);
              最后一个

   tmp = MapThread[Solve[#1 == 0, #2] &, {cspolyset, mainvars}];
         映射线程     解方程

   sols[[1]] = Flatten[First[tmp]];
               压平     第一个

   sollen = Length[sols[[1]]];
            长度

   (*Print[{solsts,sols,sollen,cslen}];*)

   While[j <= cslen,
   While循环

    mid =
     Simplify@
     化简
      Flatten@
      压平
       Table[ReplaceAll[tmp[[j]], Flatten@Map[Take[#, {i, i}] &, Take[sols, j - 1]]],
       表格   全部替代                  压平     映射 选取                 选取
        {i, 1, sollen}];
    sols[[j]] = mid;
    (*Print@mid;*)
    j++;
   ];
   (*res=Flatten[Partition[Transpose@tt,{1,cslen}],2];*)
   res = Flatten /@ Table[Map[Take[#, {i, i}] &, sols], {i, 1, sollen}];
         压平         表格   映射 选取

   Return[res];
   返回
  ]
```

Test Example

```
h1 = (u1 - x1)^2 + x2^2 - x1^2 - x2^2;
h2 = (u2 - x1)^2 + (u3 - x2)^2 - x1^2 - x2^2;
h3 = (u4 - x1)^2 + (x3 - u2)^2 - x1^2 - x2^2;
h4 = (x3 - x5) u3 + (u4 - x4) (u2 - u1);
h5 = x5 (u2 - u1) - u3 (x4 - u1);
h6 = (x3 - x7) u3 + (u4 - x6) u2;
h7 = x7 u2 - x6 u3;
hset = {h1, h2, h3, h4, h5, h6, h7};

cset = CharacteristicSet[hset, True]
                               真
```

```
CharacteristicForm[cset, {u_1, u_2, u_3, u_4}]

WuRittEqnsSolve[cset, {u_1, u_2, u_3, u_4}]
```

Test Example 1

```
poly_1 = x_1^2 - 2 x_1 x_3 + 1;
poly_2 = x_1 x_2 + x_3^2;
poly_3 = -3 x_2^2 + 2 x_3^2;
polyset = {poly_1, poly_2, poly_3};
ord = {x_1, x_2, x_3};
const = {};

cset = CharacteristicSet[polyset, ord, const, TracePrintOn → True]
                                                     真

CharacteristicForm[cset, ord, {u_1, u_2, u_3, u_4}]

WuRittEqnsSolve[cset, ord, const]
```

Test Example 2

```
poly_1 = x_1^2 - 2 x_2 x_3 + 1;
poly_2 = x_1 x_2 + x_3^2;
poly_3 = 3 x_1^2 - 2 x_3^2;
poly_4 = 2 x_1 + 2 x_2 x_1;
poly_5 = 2 x_1 + 2 x_2^2 x_1^2;
poly_6 = 3 x_1^2 + 2 x_3^3;
poly_7 = 3 x_1 x_2 + 1;
polyset = {poly_1, poly_2, poly_4, poly_3, poly_6, poly_5, poly_7};
ord = {x_1, x_2, x_3};
const = {u_1, u_2, u_3, u_4};

cset = CharacteristicSet[{poly_1, poly_2, poly_3}, ord, const, TracePrintOn -> True]
                                                                 真

CharacteristicForm[cset, ord, const]

WuRittEqnsSolve[cset, ord, const]

cset = CharacteristicSet[polyset, ord, const, TracePrintOn → True]
                                                     真

CharacteristicForm[cset, ord, const]

WuRittEqnsSolve[cset, ord, const]
```

Section WRS_V:Polynomial Rank and Some Other Functions

- **Polnomial Rank of lhspoly**

```
PolynomialRank[lhspoly_, ord___, const___] :=
 Module[{res},
 模块
  res = {Class[lhspoly, ord, const], MainVariableExponent[lhspoly, ord, const]};
  Return[res];
  返回
 ]

PolynomialRank /@ polyset
```

- **Is Rank Equal of lhspoly to rhspoly**

```
IsRankEqual[lhspoly_, rhspoly_, ord___, const___] :=
 Module[{res},
 模块
  res = If[PolynomialRank[lhspoly, ord, const] == PolynomialRank[rhspoly, ord, const],
        如果
    True, False];
    真     假
  Return[res];
  返回
 ]

IsRankEqual[poly1, poly2]
```

- **Is Rank Less of lhspoly to rhspoly**

```
IsRankLess[lhspoly_, rhspoly_, ord___, const___] :=
 Module[{lhsrank, rhsrank, res},
 模块
  lhsrank = PolynomialRank[lhspoly, ord, const];
  rhsrank = PolynomialRank[rhspoly, ord, const];
  res =
   If[lhsrank[[1]] < rhsrank[[1]] ||
   如果
     (lhsrank[[1]] == rhsrank[[1]] && lhsrank[[2]] < rhsrank[[2]]), True, False];
                                                                    真     假
  Return[res];
  返回
 ]

IsRankLess[poly1, poly2]
```

Is Rank Greater of lhspoly to rhspoly

```
IsRankGreater[lhspoly_, rhspoly_, ord___, const___] :=
 Module[{res},
 模块
  res = If[Not[IsRankEqual[lhspoly, rhspoly, ord, const]] &&
        … 逻辑非
      Not[IsRankLess[lhspoly, rhspoly, ord, const]], True, False];
      逻辑非                                            真      假
  Return[res];
  返回
 ]

IsRankGreater[poly5, poly4]
```

Section WRS_VI: WuRittProver

WuRittProver for geometry theorem proof

```
AuxProverRemainder[poly_, polyset_, switch_, ord___, const___] :=
 Module[{len, tmp, mid, i = 1}, len = Length[polyset];
 模块                              长度
  tmp = poly;
  While[i ≤ len, (mid = Evaluate[PseudoRemainder[tmp, polyset[[i]], ord, const]];
  While循环             计算
    tmp = mid;
    If[switch,
    如果
     Print[{StyleForm[StringJoin["WRP_STEP:", ToString[i]],
     打印       连接字符串              转换为字符串
        FontColor → RGBColor[0, 0, 1]], StyleForm[tmp, FontColor → RGBColor[1, 0, 0]]}]];
        字体颜色       RGB颜色                          字体颜色       RGB颜色
    i++;)];
  Return[Expand[tmp]];
  返回    展开
 ]

WuRittProver[lhspolyset_, rhspoly_, switch_, ord___, const___] :=
 Module[{tmp, res},
 模块
  tmp = AuxPseudoRemainder[rhspoly, lhspolyset, switch, ord, const];
  res = If[tmp === 0, True, False];
        如果           真     假
  Return[res];
  返回
 ]
```

Example 1(Paralell Square Theorem)

```
H1 = x1 - u1 - u2;
H2 = x2 - u3;
H3 = -u1 u3 x1 + u1 u3 x3 - u2 u3 x3 + u3 x1 x3;
H4 = x4 (u2 - u1) - (x3 - u1) u3;
HSet = {H1, H2, H3, H4};
G1 = x1^2 - 2 x1 x3 - 2 x4 x2 + x2^2;
G2 = 2 x3 u1 - 2 x3 u2 - 2 x4 u3 - u1^2 + u2^2 + u3^2;

CS = CharacteristicSet[HSet, True]
                                   真

WuRittProver[Reverse@CS, G1]
                       反向

WuRittProver[Reverse@CS, G2]
                       反向
```

Example2(Desargus Theorem)

```
H1 = x1 x6 - x2 x3;
H2 = x4 (x8 - x6) - x7 (x5 - x3);
H3 = (x4 - x1) x8 - x5 (x7 - x2);
HSet = {H1, H2, H3};
G = x4 x8 - x5 x7;

CS = CharacteristicSet[HSet, True]
                                   真

WuRittProver[Reverse@CS, G]
                       反向
```

Example3(Simon Theorem)

```
H1 = (u1 - x1)^2 + x2^2 - x1^2 - x2^2;
H2 = (u2 - x1)^2 + (u3 - x2)^2 - x1^2 - x2^2;
H3 = (u4 - x1)^2 + (x3 - x2)^2 - x1^2 - x2^2;
H4 = (x3 - x5) u3 + (u4 - x4) (u2 - u1);
H5 = x5 (u2 - u1) - u3 (x4 - u1);
H6 = (x3 - x7) u3 + (u4 - x6) u2;
H7 = x7 u2 - x6 u3;
HSet = {H1, H2, H3, H4, H5, H6, H7};
G = x7 (x4 - u4) - (x6 - u4) x5;

defCS = CharacteristicSet[Expand@HSet, True]
                                 展开         真

WuRittProver[Reverse@defCS, G, True]
                       反向           真

CS = { H1, H2, H3, PolyPRemainder[H4, H5, x5], H5, PolyPRemainder[H6, H7, x7], H7} // Expand
                                                                                       展开

WuRittProver[Reverse@CS, G, True]
                       反向        真
```

Example4(Algebra Relations Discovery)

```
H1 = x1^2 + x3^2 - x5^2;
H2 = x2^2 + x4^2 - x5^2;
H3 = x1^2 + (x3 - x5)^2 - (x2 - x1)^2 - (x3 - x4)^2;
H4 = x1^2 + (x3 - x5)^2 - 4 x2^2;
HSet = {H1, H2, H3, H4};

CharacteristicSet[HSet, True] // Simplify
                        真        化简
```

Relevent Resources

Some resources are available for developing the WuRittSolva Tools,and these coresponding notebooks are listed below:

[1]. Demonstration of WuRittSolva.nb
[2]. WuRittSolva User Guide.nb
[3]. Demonstration of WuRittSolva in Elementary Geometry.nb
[4]. A Collection of Testing Problems.nb
[5]. WuRittSolva for Concrete Geometric Configurations in Elementary Geometry.nb
[6]. WuRittSolva User Manual.nb