C-SemigroupsToolbox

Descripción : Este notebook contiene un conjunto de funciones desarrolladas en Mathematica para trabajar, visualizar y crear ejemplos de C-semigrupos en $\mathbb{N}^2$. Para ello, nos ayudamos del paquete Normaliz. Es una herramienta de código abierto para cálculos en monoides afines, configuraciones vectoriales, politopos de retículo y conos racionales. Normaliz también calcula poliedros algebraicos, es decir, poliedros definidos sobre extensiones algebraicas reales de $\mathbb{Q}$. https://www.normaliz.uni-osnabrueck.de/

Autores :

- Sanchez Loureiro, Adrián.
- Vigneron Tenorio, Alberto.

Repositorio : https://github.com/asanlou/C-SemigroupsToolbox

Funciones

En caso de que ejecutemos este notebook en **Linux**, *operaringSystem* deberá tener el valor True.
En caso de ser ejecutado en **Windows**, *operaringSystem* deberá

tener el valor False.

In[248]:=
```
operatingSystem = True;
```

En este archivo daremos por hecho que Normaliz y el notebook se encuentran en el mismo directorio. Además, el programa Normaliz se ejecutará a partir del archivo “normaliz” o “normaliz.exe” si es Linux o Windows respectivamente.

Generar un cono a partir de puntos dados

Auxiliares

In[249]:=
```
toRat[x_] := Module[{num, den, xx, a, b},
   (*Pasa los racionales a numerador/denominador*)
   (*Output: el par {numerador, denominador}*)
   xx = Rationalize[x];
   a = Numerator[xx];
   b = Denominator[xx];
   Return[{a, b}]
  ];
```

In[250]:=
```
RationalToInt[V_] :=
  Module[{l = Length[V], i, mcm, V1},
   (*Pasa un vector racional al menor
    proporcional entero*)
   V1 = Table[{toRat[V[[i]]][[1]], toRat[V[[i]]][[2]]}, {i, l}];
   mcm = LCM @@ Table[V1[[i, 2]], {i, l}];
   V1 = Table[mcm * V1[[i]][[1]] / V1[[i]][[2]], {i, l}];
   Return[V1]
  ];
```

Final

In -> Lista de puntos “puntos” con que definir el cono, el directorio “dirTab” del notebook y el booleano “isLinux” que define si usamos Linux o en Windows.
Out -> Empleando *normaliz*, se devuelve la lista {{“Hib”},{“SupHyp”}} donde “Hib” es la base de Hilbert y “SupHyp” los hiperplanos soportes del cono.

In[251]:=
```
ConeGenSupHyp[puntos_, dirTrab_, isLinux_] :=
  Module[{Nf, Nc, origen, f, i, cad = "", ratray,
    suphyp, gen, st1, st2, caux, lineas = {}},
   (*Obtenemos las dimensiones de la lista
    puntos*)
   {Nf, Nc} = Dimensions[puntos];
   (*Origen en función de la dimensión*)
   origen = Table[0, {i, Nc}];

   ratray = Table[RationalToInt[puntos[[i]]], {i, Nf}];

   (*Creamos/escribimos el archivo aux.in
```

```
 para ejecutar con Normaliz con los datos
 dados*)
cad = "amb_space " <> ToString[Nc] <> "\n";
cad = cad <> "cone " <> ToString[Nf] <> "\n";
For[i = 1, i ≤ Nf, i ++,
 cad = cad <> StringReplace[ToString[ratray[[i]]],
     {", " → " ", "{" → "", "}" → ""}] <> "\n";
];
cad = cad <> "vertices " <> ToString[1] <> "\n";
cad = cad <> StringReplace[ToString[origen],
   {", " → " ", "{" → "", "}" → " "}] <> "1";
f = OpenWrite[dirTrab <> "aux1.in"];
WriteString[f, cad];
Close[f];

(*Una vez preparado el fichero con los
 datos dados, ejecutamos Normaliz*)
If[isLinux,
     Run[dirTrab <> "/normaliz -c -N -a " <>
   dirTrab <> "aux1.in"],
     Run["normaliz -c -N -a " <> dirTrab <>
   "aux1.in"]
];

(*Leemos y procesamos lo generado por
 Normaliz*)
cad = ReadString[dirTrab <> "aux1.gen"];
st1 = StringToStream[cad];
caux = ReadLine[st1];
caux = ReadLine[st1];
caux = ReadLine[st1];
```

```
While[Characters[caux] ≠ {},
      AppendTo[lineas, caux];
      caux = ReadLine[st1];
 ];
gen = Flatten[ImportString[#, "Table"] & /@ lineas,
   1];
gen = Select[gen, #[[Length[gen[[1]]]]] == 0 &];
For[i = 1, i ≤ Length[gen], i ++,
 gen[[i]] = Delete[gen[[i]], Nc + 1];
];

(*Seleccionamos los generadores*)
cad = "";
cad = ReadString[dirTrab <> "aux1.cst"];
st2 = StringToStream[cad];
lineas = {};
origen = Interpreter["Number"][ReadLine[st2]];
caux = ReadLine[st2];
i = 1;
While[i ≤ origen,
 caux = ReadLine[st2];
 AppendTo[lineas, caux];
 i ++
];

(*Seleccionamos los planos soporte*)
suphyp =
 Flatten[ImportString[#, "Table"] & /@ lineas,
   1];
suphyp = Select[suphyp,
   #[[Length[suphyp[[1]]]]] == 0 &];
For[i = 1, i ≤ Length[suphyp], i ++,
```

```
    suphyp[[i]] = Delete[suphyp[[i]], Nc + 1];
   ];

   (*Devuelve base de Hilbert (gen) e hiperplanos
    soportes (suphyp) *)
   Return[{gen, suphyp}]
  ];
```

Comprobar si un punto pertenece a un cono dado

In -> Un punto del cono “pto” y las ecuaciones del cono “Eq”.
Out -> Devuelve “True” si el punto pto pertenece al cono y “False” en caso contrario.

In[252]:=

```
InCone[pto_, Eq_] := Module[{prod, verif},
   (*Verifica si un pto está en el cono dado
    por las inecuaciones Eq*)
   (*Aplica las ecuaciones Eq en pto*)
   prod = Eq.pto;
   verif = Table[prod[[i]] ≥ 0, {i, Length[prod]}];
   verif = Union[verif];
   If[verif == {True}, Return[True], Return[False]];
  ];
```

Conjuntos de un $\mathcal{C}$-semigrupo

Apery(S,b)

Apery(S,b) conocido Gaps y Eq Cono

In -> Un punto del $\mathcal{C}$-semigrupo “b”, el conjunto de huecos Gaps del $\mathcal{C}$-semigrupo y las ecuaciones del cono “Eq”.
Out -> Devuelve Apery(S,”b”).

In[253]:=
```
GetAperyEq[b_, Gaps_, Eq_] :=
  Module[{nGaps, i, aux, Apery},
   (*Compruebo que b no es un hueco*)
   If[(MemberQ[Gaps, b]),
    (*If*)
       Print[b, "no pertenece al C-semigrupo."];
       Return[{}],

    (*Else*)
       (*Compruebo que b está en el cono*)
       If[¬ InCone[b, Eq],
             Print[b, "no pertenece al cono"];
             Return[{}]
     ];
   ];

   (*Número de huecos y inicializando conjunto
    Apery*)
   nGaps = Length[Gaps];
   Apery = {};
```

```
   (*Puntos
    Apery : punto x del C-semigrupo tal que x-
      b pertenece a Gaps.*)
   (*Equivalentemente,
   los puntos g de Gaps tal que g+
    b está en el cono y no pertenece a Gaps.*)
   (*Calculamos el Apery a partir de estos
    últimos:*)
   For[i = 1, i ≤ nGaps, i++,
    aux = b + Gaps[[i]];
    If[(¬ MemberQ[Gaps, aux]),
     Apery = Append[Apery, aux];
    ];
   ];

   (*Devolvemos Apery obtenido*)
   Return[Apery]

  ];
```

Apery(S,b) conocido geners del CSemi y Gaps y FileDirectory

In -> Un punto del $\mathcal{C}$-semigrupo “b”, los generadores “GenSemig” y el conjunto de huecos “Gaps” del $\mathcal{C}$-semigrupo, y la ruta del notebook “dirTrab”.
Out -> Devuelve Apery(S,”b”).

In[254]:=
```
GetApery[b_, GenSemig_, Gaps_, dirTrab_] :=
  Module[{nGaps, i, aux, Apery, T1, LineasT1,
    vectorsinrays, T1L, Eq},
```

```
(*Compruebo que b no es un hueco*)
If[( MemberQ[Gaps, b]),
 Print[b, "no pertenece al C-semigrupo."];
 Return[{}]
];

(*Calculamos ecuaciones del cono para
 usar función InCone*)
(**Vectores de los rayos extremales para
 sacar ecuaciones del cono**)
T1 = ConvexHullMesh[Join @@ {{{0, 0}}, GenSemig}];
LineasT1 = MeshPrimitives[T1, 1];
 T1L = Select[Level[LineasT1, {2}],
   MemberQ[#, {0, 0}] &];
vectorsinrays = Select[Flatten[T1L, 1],
   # ≠ {0, 0} &];

(**Ecuaciones del cono para comprobar si
 punto está en el cono**)
Eq = ConeGenSupHyp[vectorsinrays, dirTrab,
    operatingSystem][[2]];

(*Compruebo que b está en el cono*)
If[¬ InCone[b, Eq],
 Print[b, "no pertenece al cono"];
 Return[{}]
];

(*Inicializando conjunto Apery*)
Apery = {};

(* Puntos
```

```
      Apery : punto x del C-semigrupo tal que x-
        b pertenece a Gaps.
          De forma similar los puntos g de
         Gaps tal que g+
        b está en el cono y no pertenece a Gaps.
          Calculamos el Apery a partir de
         estos últimos: *)
    nGaps = Length[Gaps];

    For[i = 1, i ≤ nGaps, i++,
     aux = b + Gaps[[i]];
     If[(¬ MemberQ[Gaps, aux]),
      Apery = Append[Apery, aux];
     ];
    ];

    (*Devolvemos Apery obtenido*)
    Return[Apery]
   ];
```

PF(S) a partir de generadores y huecos

In ->Los generadores “GenSemig” y el conjunto de huecos “Gaps” del $\mathcal{C}$-semigrupo.
Out -> El conjunto PF(⟨”GenSemig”⟩)

In[255]:=
```
GetPseudoFrobenius[GenSemig_, Gaps_] :=
  Module[{nGaps, i, j, nGens, PseuFrobs},
   (*Calcularemos el número de generadores
    y huecos*)
    nGens = Length[GenSemig];
   nGaps = Length[Gaps];

   (*Inicializamos el conjunto PF(〈"GenSemig"〉)*)
   PseuFrobs = {};

   (*Comprobamos que huecos x∈
    G(S) verifican que x+(S\{0})⊂S*)
   For[i = 1, i ≤ nGaps, i++,
        j = 1;


    While[(j ≤ nGens) ∧
      (¬ MemberQ[Gaps, Gaps[[i]] + GenSemig[[j]]]),
           j ++
        ];

        If[j == nGens + 1,
             PseuFrobs = Append[PseuFrobs, Gaps[[i]]]
        ];
   ];

   (*Devolvemos los puntos obtenidos*)
   Return[PseuFrobs]

  ];
```

SG(S)

SG(S) a partir de Pseudos y Huecos

In -> El conjunto de Pseudo-Frobenius “PseuFrobs” y el conjunto de huecos “Gaps” del $\mathcal{C}$-semigrupo.
Out -> El conjunto SG⟨”GenSemig”⟩)

In[256]:=
```
GetEspGaps[PseuFrobs_, Gaps_] :=
  Module[{EspGaps, i, nPseu},
   (*Calcularemos el número de pseudo-Frobenius*)
   nPseu = Length[PseuFrobs];

   (*Inicializamos los huecos especiales*)
   EspGaps = {};

   (*Comprobamos que x∈PF(S) son tales que 2x∈S*)
   For[i = 1, i ≤ nPseu, i ++,
        If[¬ MemberQ[Gaps, 2 * PseuFrobs[[i]]],

         EspGaps = Append[EspGaps, PseuFrobs[[i]]]
           ];
   ];

   (*Devolvemos los SG(S) obtenidos*)
   Return[EspGaps]
  ];
```

SG(S) y nº de ellos a partir de Generadores y Huecos

In -> Los generadores “GenSemig” y el conjunto de huecos “Gaps” del $\mathcal{C}$-semigrupo.
Out -> El conjunto SG⟨”GenSemig”⟩)

In[257]:=
```
GetEspGapsGen[GenSemig_, Gaps_] :=
  Module[{nGaps, i, j, nGens, EspGaps},
   (*Número de generadores y de huecos*)
    nGens = Length[GenSemig];
   nGaps = Length[Gaps];

   (*Inicializamos el conjunto de huecos
    especiales*)
   EspGaps = {};

   (*Comprobamos que huecos son SG("GenSemig"))*)
   For[i = 1, i ≤ nGaps, i++,
        j = 1;


    While[(j ≤ nGens) ∧
      (¬ MemberQ[Gaps, Gaps[[i]] + GenSemig[[j]] ]) ,
             j++

        ];



    If[ (j == nGens + 1) ∧ (¬ MemberQ[Gaps, 2 * Gaps[[i]]]),
             EspGaps = Append[EspGaps, Gaps[[i]]]
        ];
   ];

   (*Devolvemos el conjunto obtenido*)
   Return[EspGaps]
  ];
```

FG(S) a partir de huecos

In -> El conjunto de huecos “Gaps” del $\mathcal{C}$-semigrupo.
Out -> El conjunto PF(⟨”GenSemig”⟩)

In[258]:=
```
GetFundGaps[Gaps_] := Module[{FundGaps, i, nGaps},
   (*Calculamos el número de huecos*)
   nGaps = Length[Gaps];

   (*Inicializamos el conjunto de huecos
    fundamentales*)
   FundGaps = {};

   (*Calculamos los x∈G(S) tales que 2x,3x∈S,
   es decir, 2x,3x∉"Gaps"*)
   For[i = 1, i ≤ nGaps, i++,

    If[(¬ MemberQ[Gaps, 2*Gaps[[i]]]) ∧
       (¬ MemberQ[Gaps, 3*Gaps[[i]]]),

      FundGaps = Append[FundGaps, Gaps[[i]]]
          ];
   ];

   (*Devolvemos el conjunto obtenido*)
   Return[FundGaps]
  ];
```

$I(n) = \{ s \in S : s \leqslant_{C} n \}$

I(n) conocido Gaps y Eq Cono

In -> Dado un “n” natural, el conjunto de generadores minimales “GenSemig” y loshuecos “Gaps” del C-semigrupo en el cono con ecuaciones “Eq”.
Out -> I(“n”)

In[259]:=
```
GetIsetEq[n_, GenSemig_, Gaps_, Eq_] :=
  Module[{Iset, T1, LineasT1, vectorsinrays,
    T1L, i, j},
   (*Puntos I(n) :
    puntos x del C-semigrupo tal que n -
     x pertenece al cono.*)
   (*Equivalentemente,
   los (x1,x2) con x1 ≤ n1 y x1 ≤ n2,
   que que cumplen la definicion de I(n)*)
   (*Calculamos el I(n) bajo ese critero:*)

   (*Compruebo que n está en el cono*)
   If[¬ InCone[n, Eq],
        Print[n, "no pertenece al cono"];
        Return[{}]
   ];

   (*Inicializando I(n) →
    coordenadas menores que las de n.*)
   Iset = Flatten[ParallelTable[{i, j}, {i, 0, n[[1]]},
      {j, 0, n[[2]]}], 1];

   (*Tomando puntos del cono*)
```

```
    Iset = Select[Iset, InCone[#, Eq] &];

    (*Tomando puntos de Iset*)
    Iset = Complement[Iset, Gaps];

    Iset = Select[Iset, InCone[n - #, Eq] &];

    (*Devolviendo Iset*)
    Return[Iset]
  ];
```

I(n) conocido geners del CSemi y Gaps y FileDirectory

In -> Dado un “n” natural, el conjunto de generadores minimales “GenSemig” y loshuecos “Gaps” del $\mathcal{C}$-semigrupo en el cono con ecuaciones “Eq”.
Out -> El conjunto PF(⟨”GenSemig”⟩)

In[260]:=

```
GetIsetNoEq[n_, GenSemig_, Gaps_, dirTrab_] :=
  Module[{Iset, T1, LineasT1, vectorsinrays,
     nGaps, T1L, Eq, auxGaps, i, j},
    (*Puntos I(n) :
     puntos x del C-semigrupo tal que n -
      x pertenece al cono.*)
    (*Equivalentemente,
    los (x1,x2) con x1 ≤ n1 y x1 ≤ n2,
    que pertenecen al cono y no pertenecen
     a Gaps.*)
    (*Calculamos el I(n) bajo ese critero:*)

    (*Vectores de los rayos extremales para
     sacar ecuaciones del cono*)
```

```
T1 = ConvexHullMesh[Join @@ {{{0, 0}}, GenSemig}];
LineasT1 = MeshPrimitives[T1, 1];
 T1L = Select[Level[LineasT1, {2}],
   MemberQ[#, {0, 0}] &];
vectorsinrays = Select[Flatten[T1L, 1],
   # ≠ {0, 0} &];

(*Ecuaciones del cono para comprobar si
 punto está en el cono*)
Eq = ConeGenSupHyp[vectorsinrays, dirTrab,
    operatingSystem][[2]];

(*Compruebo que n está en el cono*)
If[¬ InCone[n, Eq],
     Print[n, "no pertenece al cono"];
     Return[{}]
];

(*Inicializando I(n) →
 coordenadas menores que las de n.*)
Iset = Flatten[ParallelTable[{i, j}, {i, 0, n[[1]]},
    {j, 0, n[[2]]}], 1];

(*Tomando puntos del cono*)
Iset = Select[Iset, InCone[#, Eq] &];

(*Tomando puntos de Iset*)
Iset = Complement[Iset, Gaps];

Iset = Select[Iset, InCone[n - #, Eq] &];

(*Devolviendo Iset*)
```

```
    Return[Iset]
  ];
```

$C(S_i) = \{ h \in SG(S) : h \notin S_i \}$ para cualquier $S_i \in \mathcal{I}(S)$

In -> Dado “Semig”, la lista {{Generadores}, {Huecos}} de un semigrupo S y el conjunto de huecos “DescGaps” de un $S_i \in \mathcal{I}(S)$.
Out -> El conjunto $C(S_i)$.

In[261]:=
```
IrreducibleCSi[Semig_, DescGaps_] := Module[{espGaps,cSi},

    (*Calculamos los huecos especiales de Semig*)
    espGaps = GetEspGaps[GetPseudoFrobenius[Semig[[1]],Semig[[

    (*Devolvemos los huecos especiales también pertenecien
    cSi = Intersection[espGaps, DescGaps];

    (*Devolviendo C(Si)*)
    Return[cSi]
  ];
```

$D(X) = \{ a \in \mathcal{C} : na \in X$ para algún $n \in \mathbb{N} \}$ $\forall X \subset \mathcal{C}$

In -> Dado “A”$\subset \mathcal{C}$ con $\mathcal{C}$ un cono natural con ecuaciones “Eq” y el booleano “verb” para devolver o no por pantalla lo realizado por la función.
Out -> El conjunto D(“A”).

In[262]:=
```
GenerateDX[A_,Eq_,verb_]:= Module[{a,i,gcd,k,aux,aux2,auxgcd,
```

```
If[verb, Print["---------------------------"]];
(*Comprueba que los elementos de X están en el cono*)
For[k=1,k≤Length[A],k++,
    aux = ¬InCone[A[[k]],Eq];

    If[verb,
        Print["A[[k]]",A[[k]]];
        Print["¬InCone[x,Eq]->",aux]
    ];

    If[aux,
        If[verb, Print["Exist element out of cone ->",A
        Return[{}]];
];


conjDX=A;

(*Añadimos los elementos de D(X) que no están en X*)
For[k=1,k≤Length[A],k++,
    If[verb, Print["*******************"]];
    gcd=Divisors[Apply[GCD,A[[k]]]];
    auxgcd=Length[gcd];

    If[verb,
        Print["A[[k]]",A[[k]]];
        Print["gcd->",gcd];
        Print["Lenght[gcd]->",auxgcd]
    ];

    If[auxgcd ≠ 0,
        For[i=1,i≤auxgcd,i++,
            aux=A[[k]]/gcd[[i]];
            If[¬MemberQ[conjDX,aux],
```

```
                    Print["Element added->",aux,¬MemberQ[au
                    conjDX = AppendTo[conjDX,aux];
                ]
            ]
        ]

    ];

    (*Devolvemos el conjunto obtenido*)
    If[verb, Print["DX finished"]];
    Return[conjDX]
]
```

$\{ s \in \mathcal{C} \backslash X : s \leq_{\mathcal{C}} x$ para algún $x \in X \}$ para un $X \subset \mathcal{C}$

In -> Dado “setX”$\subset \mathcal{C}$ con $\mathcal{C}$ un cono natural con ecuaciones “coneEq”, “dirTab” el directorio del archivo y el booleano “verb” para devolver o no por pantalla lo realizado por la función.
Out -> El conjunto D(“A”).

In[263]:=
```
SetMinusXFewer[setX_,coneEq_,dirTrab_,verb_]:=Module[{auxX,or
    auxX=Length[setX];
    orderedX=ReverseSort[LexicographicSort[setX]];
    If[verb,Print["Ordered"]];

    maxCoords=Table[MaximalBy[orderedX, #[[i]] &][[1]],{i,1,Lengt
    If[verb,Print["maxCoords -> ",maxCoords]];

    elemtMiddle=Select[orderedX, maxCoords[[1,1]] ≤ #[[1]] ≤ ma
    If[verb,Print["elemtMiddle -> ",elemtMiddle]];
```

```
    (*Calcularemos auxDesired: puntos s de C\X con s≤Cx sie
    (*Inicializando el conjunto deseado → coordenadas meno
    auxDesired=Flatten[ParallelTable[{i,j},{i,0,maxCoords[[1,
    If[verb,Print["auxDesired All-> ",auxDesired]];
    (*Tomando puntos de C\X*)
    auxDesired = Complement[auxDesired,setX];
    If[verb,Print["auxDesired Complement-> ",auxDesired]];

    auxMiddle=Select[auxDesired, maxCoords[[2,1]] ≤ #[[1]] && m
    If[verb,Print["auxMiddle creation -> ",auxMiddle]];

    auxDesired=Complement[auxDesired,auxMiddle];
    If[verb,Print["auxMiddle \\ auxMiddle -> ",auxDesired]];

    For[k=1,k≤Length[elemtMiddle],k++,
        If[verb,Print["elemtMiddle[[k]] -> ",elemtMiddle[[k]]]];
        auxDesired = Union[auxDesired,Select[auxMiddle, ele
    ];
    If[verb,Print["auxDesired after middles-> ",auxDesired]]

    (*Tomando puntos pertenecientes al cono*)
    auxDesired=Select[auxDesired,InCone[#,coneEq] &];

    Return[auxDesired]
]
```

$\{\,(x,s) \in X \times \mathcal{C}\backslash X :\; s \leq_{\mathcal{C}} x\,\}$ para un $X \subset \mathcal{C}$

In[264]:=
```
SetMinusXTimesCX[setX_,coneEq_,dirTrab_,verb_,verbMore_]:=Mo
    auxDesired = SetMinusXFewer[setX,coneEq,dirTrab,verbMo
    If[verb,Print["setDesired-> ",auxDesired]];
    setDesired={};

    For[k=1,k≤Length[setX],k++,
        If[verb,Print["setX[[k]] -> ",setX[[k]]]];

        setDesired = Join[setDesired,ParallelTable[{setX[[k]],
        If[verb,Print["setDesired-> ",setDesired]];
    ];

    Return[setDesired]
]
```

Vector Frobenius para distintos órdenes y conjunto N(S)

Lexicografico graduado inverso

In[265]:=
```
FrobeniusDRLexi[hole_] :=
  Module[{i, j, x, X, MonHole, Frob},
   (*hole lista de huecos*)
   (*Devuelve Frobnenius respecto orden fijado*)
   X = Table[x_i, {i, Length[hole[[1]]]}];
   MonHole =
    Sum[Product[(x_i)^hole[[j]][[i]],
      {i, 1, Length[hole[[j]]]}], {j, 1, Length[hole]}];
   (*Print["Polinomio= ",MonHole];*)
   MonHole = MonomialList[MonHole, X,
     DegreeReverseLexicographic];
   Frob = MonHole[[1]];
   (*Éste es el Fröbnenius respecto al orden
    fijado*)
   Frob = Table[Exponent[Frob, x_i],
     {i, Length[hole[[1]]]}];
   Return[Frob];
  ];
```

Lexicografico graduado

In[266]:=
```
FrobeniusDLexi[hole_] :=
  Module[{i, j, x, X, MonHole, Frob},
   (*hole lista de huecos*)
   (*Devuelve Frobnenius respecto orden fijado*)
   X = Table[x_i, {i, Length[hole[[1]]]}];
   MonHole =
    Sum[Product[(x_i)^hole[[j]][[i]],
      {i, 1, Length[hole[[j]]]}], {j, 1, Length[hole]}];
   (*Print["Polinomio= ",MonHole];*)
   MonHole = MonomialList[MonHole, X,
     DegreeLexicographic];
   Frob = MonHole[[1]];
   (*Éste es el Fröbnenius respecto al orden
    fijado*)
   Frob = Table[Exponent[Frob, x_i],
     {i, Length[hole[[1]]]}];
   Return[Frob];
  ];
```

Lexicografico

In[267]:=
```
FrobeniusLexi[hole_] :=
  Module[{i, j, x, X, MonHole, Frob},
   (*hole lista de huecos*)
   (*Devuelve Frobnenius respecto orden fijado*)
   X = Table[x_i, {i, Length[hole[[1]]]}];
   MonHole =
    Sum[Product[(x_i)^hole[[j]][[i]],
      {i, 1, Length[hole[[j]]]}], {j, 1, Length[hole]}];
   (*Print["Polinomio= ",MonHole];*)
   MonHole = MonomialList[MonHole, X,
     Lexicographic];
   Frob = MonHole[[1]];
   (*Éste es el Fröbnenius respecto al orden
    fijado*)
   Frob = Table[Exponent[Frob, x_i],
     {i, Length[hole[[1]]]}];
   Return[Frob];
  ];
```

Dando matriz de pesos

In[268]:=
```
Frobenius2[hole_, MatrizOrden_] :=
  Module[{i, p = Length[hole], MonHole, MonHole2,
    MonHole3, Frob},
   (*hole lista de huecos*)
   (*Devuelve Frobnenius respecto orden
    fijado por la matriz*)
   MonHole = Table[MatrizOrden.hole[[i]], {i, 1, p}];
   MonHole2 = Sort[MonHole];
   (*Print["Polinomio= ",MonHole,MonHole2];*)
   MonHole3 = Flatten[Position[MonHole, MonHole2[[p]]],
     1];
   (*Print["Polinomio= ",MonHole3];*)
   Frob = hole[[MonHole3[[1]]]];
   (*Éste es el Fröbnenius respecto al orden
    fijado*)
   Return[Frob];
  ];
```

$N(S) = \{x \in S \mid x \preceq F(S)\}$

Lexicógrafo graduado inverso

In[269]:=
```
NFrobeniusDRLexi[GenSemig_, Gaps_] :=
  Module[{semig, L, T1, LineasT1, T1L,
    vectorsinrays, t, ConeC, i, j,
    Frob1, x, X, MonHole, MonHoleFixed,
    elemLength},
   (*Calcula N(S) a partir de sus generadores
```

```
 minimales, huecos y el Frobenius.
  Para ello, calcula parte del semigrupo
 y se ordena quedándonos con N(S)*)

semig = GenSemig;
L = Length[semig];

T1 = ConvexHullMesh[Join @@ {{{0, 0}}, semig}];
LineasT1 = MeshPrimitives[T1, 1];
T1L = Select[Level[LineasT1, {2}],
   MemberQ[#, {0, 0}] &];
vectorsinrays = Select[Flatten[T1L, 1],
   # ≠ {0, 0} &];

t = Ceiling[Max[Join[semig, Gaps]]];
(*Print[Max[Join[semig,Gaps]]," ",t];*)
ConeC =
  Flatten[ParallelTable[{i, j}, {i, 0, t}, {j, 0, t}],
   1];
(*Print[ConeC];*)

T1 = ConvexHullMesh[
   Join @@ {{{0, 0}}, 20 * vectorsinrays}];

ConeC = Select[ConeC, Element[#, T1] &];

elemLength = Length[ConeC[[1]]];

ConeC = Union[ConeC[[2 ;;]], Gaps];

X = Table[x_i, {i, elemLength}];
MonHole =
```

```
    Sum[Product[(x_i)^ConeC[[j]][[i]], {i, 1, elemLength}],
     {j, 1, Length[ConeC]}];

   MonHole = MonomialList[MonHole, X,
     DegreeReverseLexicographic];

   MonHoleFixed =
    Table[Table[Exponent[MonHole[[j, i]], x_i],
      {i, elemLength}], {j, Length[MonHole]}];

   Frob1 = FrobeniusDRLexi[Gaps];

   If[Position[MonHoleFixed, Frob1] ≠ {},
    MonHoleFixed =
     MonHoleFixed[[
      Position[MonHoleFixed, Frob1][[1, 1]] + 1 ;;]];
    Return[Complement[MonHoleFixed, Gaps]],
    Return[{}]
   ]


  ];
```

Lexicógrafo graduado

In[270]:=
```
NFrobeniusDLexi[GenSemig_, Gaps_] :=
  Module[{semig, L, T1, LineasT1, T1L,
    vectorsinrays, t, ConeC, i, j,
    Frob1, x, X, MonHole, MonHoleFixed,
    elemLength},
   (*Calcula N(S) a partir de sus generadores
    minimales, huecos y el Frobenius.
```

```
  Para ello, calcula parte del semigrupo
 y se ordena quedándonos con N(S)*)

semig = GenSemig;
L = Length[semig];

T1 = ConvexHullMesh[Join @@ {{{0, 0}}, semig}];
LineasT1 = MeshPrimitives[T1, 1];
T1L = Select[Level[LineasT1, {2}],
   MemberQ[#, {0, 0}] &];
vectorsinrays = Select[Flatten[T1L, 1],
   # ≠ {0, 0} &];

t = Ceiling[Max[Join[semig, Gaps]]];
(*Print[Max[Join[semig,Gaps]]," ",t];*)
ConeC =
  Flatten[ParallelTable[{i, j}, {i, 0, t}, {j, 0, t}],
   1];
(*Print[ConeC];*)

T1 = ConvexHullMesh[
   Join @@ {{{0, 0}}, 20*vectorsinrays}];

ConeC = Select[ConeC, Element[#, T1] &];

elemLength = Length[ConeC[[1]]];

ConeC = Union[ConeC[[2 ;;]], Gaps];

X = Table[x_i, {i, elemLength}];
MonHole =
  Sum[Product[(x_i)^ConeC[[j]][[i]], {i, 1, elemLength}],
```

```
      {j, 1, Length[ConeC]}];

    MonHole = MonomialList[MonHole, X,
      DegreeLexicographic];

    MonHoleFixed =
     Table[Table[Exponent[MonHole[[j, i]], x_i],
       {i, elemLength}], {j, Length[MonHole]}];

    Frob1 = FrobeniusDLexi[Gaps];

    If[Position[MonHoleFixed, Frob1] ≠ {},
     MonHoleFixed =
      MonHoleFixed[[
       Position[MonHoleFixed, Frob1][[1, 1]] + 1 ;;]];
     Return[Complement[MonHoleFixed, Gaps]],
     Return[{}]
    ]


   ];
```

Lexicógrafo

In[271]:=
```
NFrobeniusLexi[GenSemig_, Gaps_] :=
  Module[{semig, L, T1, LineasT1, T1L,
    vectorsinrays, t, ConeC, i, j,
    Frob1, x, X, MonHole, MonHoleFixed,
    elemLength},
   (*Calcula N(S) a partir de sus generadores
    minimales, huecos y el Frobenius.
     Para ello, calcula parte del semigrupo
```

```
 y se ordena quedándonos con N(S)*)

semig = GenSemig;
L = Length[semig];

T1 = ConvexHullMesh[Join @@ {{{0, 0}}, semig}];
LineasT1 = MeshPrimitives[T1, 1];
T1L = Select[Level[LineasT1, {2}],
   MemberQ[#, {0, 0}] &];
vectorsinrays = Select[Flatten[T1L, 1],
   # ≠ {0, 0} &];

t = Ceiling[Max[Join[semig, Gaps]]];
(*Print[Max[Join[semig,Gaps]]," ",t];*)
ConeC =
  Flatten[ParallelTable[{i, j}, {i, 0, t}, {j, 0, t}],
   1];
(*Print[ConeC];*)

T1 = ConvexHullMesh[
   Join @@ {{{0, 0}}, 20*vectorsinrays}];

ConeC = Select[ConeC, Element[#, T1] &];

elemLength = Length[ConeC[[1]]];

ConeC = Union[ConeC[[2 ;;]], Gaps];

X = Table[x_i, {i, elemLength}];
MonHole =
  Sum[Product[(x_i)^ConeC[[j]][[i]], {i, 1, elemLength}],
   {j, 1, Length[ConeC]}];
```

```
MonHole = MonomialList[MonHole, X,
   Lexicographic];

MonHoleFixed =
  Table[Table[Exponent[MonHole[[j, i]], x_i],
    {i, elemLength}], {j, Length[MonHole]}];

Frob1 = FrobeniusLexi[Gaps];

If[Position[MonHoleFixed, Frob1] ≠ {},
  MonHoleFixed =
   MonHoleFixed[[
    Position[MonHoleFixed, Frob1][[1, 1]] + 1 ;;]];
  Return[Complement[MonHoleFixed, Gaps]],
  Return[{}]
 ]


];
```

Generadores minimales de un C-semigrupo dado

In[272]:=
```
MinGenGeneral[gen_,holes_,Eq_]:= Module[{i,k,msg={},xx,genOrde
(*Elimina generadores no minimales de gen1 y huecos holes.
    genOrdenado=Sort[gen];
    i=2;
    msg={genOrdenado[[1]]};
    While[i≤Length[genOrdenado],
        seguir=True;
        For[k=1,k<i ,k++,
            xx=genOrdenado[[i]]-genOrdenado[[k]];
            If[(MemberQ[holes,xx] ∨ !InCone[xx,Eq]),seguir=Tru
                seguir=False;
                Break[];
            ];
        ];
        If[seguir,
            AppendTo[msg,genOrdenado[[i]]];
            i++;
            ,
            genOrdenado=Delete[genOrdenado,i];
        ];
    ];
    Return[{msg,holes}]
    ];
```

Graficando C-semigrupos

Dibujando Cono con huecos y generadores

minimales

In[273]:=
```
Plot2DSemig[GenSemig_, Gaps_] :=
  Module[{PtosEnS, semig, coeficientes, i, j,
    t, L, T1, LineasT1, T1L, vectorsinrays,
    T2, ConeC},
   (*Dibuja el C-semigrupo de N^2 generado
     por Gen_Semig, con huecos Gaps*)
   semig = GenSemig;
   L = Length[semig];

   T1 = ConvexHullMesh[Join @@ {{{0, 0}}, semig}];
   LineasT1 = MeshPrimitives[T1, 1];
   T1L = Select[Level[LineasT1, {2}],
     MemberQ[#, {0, 0}] &];
   vectorsinrays = Select[Flatten[T1L, 1],
     # ≠ {0, 0} &];
   Print["Vectores de los rayos extremales= ",
    vectorsinrays];

   t = Ceiling[Max[Join[semig, Gaps]]];
   (*Print[Max[Join[semig,Gaps]]," ",t];*)
   ConeC =
    Flatten[ParallelTable[{i, j}, {i, 0, t}, {j, 0, t}],
     1];
   (*Print[ConeC];*)
   T1 = ConvexHullMesh[
     Join @@ {{{0, 0}}, 20*vectorsinrays}];

   ConeC = Select[ConeC, Element[#, T1] &];
   (*Print[ConeC];*)
```

```
  vectorsinrays =
   ParallelTable[{{0, 0}, 20*vectorsinrays[[i]]},
    {i, Length[vectorsinrays]}];
  (*Print[vectorsinrays];*)

  Print[
   Show[
    {
     ListPlot[ConeC, PlotStyle → Red],
     ListPlot[Gaps, PlotStyle → Black,
      PlotMarkers → "OpenMarkers"],
     ListPlot[GenSemig, PlotStyle → Yellow,
      PlotMarkers → "OpenMarkers"],
     Graphics[{Red, Thick, Line[vectorsinrays]}]
     (*Graphics3D[{Red,Line /@ T1L }]*)
    },
    AxesOrigin → {0, 0}, AspectRatio → Automatic,
    Axes → True, AxesLabel → {X, Y}
    (*,AxesStyle→{Black,Red,Blue}*)
   ]
  ];
  Return[]
 ];
```

Dibujando Cono con todo

In[274]:=
```
Plot2DSemigAll[GenSemig_, Gaps_] :=
  Module[{PtosEnS, semig, coeficientes, i, j,
    t, L, T1, LineasT1, T1L, vectorsinrays,
    T2, ConeC, PseuFrobs, EspGaps},
```

```
(*Dibuja el C-semigrupo de N^2 generado
  por Gen_Semig, con huecos Gaps,
pseudos PseuFrobs y especiales EspGaps*)
semig = GenSemig;
L = Length[semig];

PseuFrobs = GetPseudoFrobenius[GenSemig, Gaps];
EspGaps = GetEspGaps[PseuFrobs, Gaps] ;

T1 = ConvexHullMesh[Join @@ {{{0, 0}}, semig}];
LineasT1 = MeshPrimitives[T1, 1];
T1L = Select[Level[LineasT1, {2}],
  MemberQ[#, {0, 0}] &];
vectorsinrays = Select[Flatten[T1L, 1],
  # ≠ {0, 0} &];
Print["Vectores de los rayos extremales= ",
 vectorsinrays];

t = Ceiling[Max[Join[semig, Gaps]]];
(*Print[Max[Join[semig,Gaps]]," ",t];*)
ConeC =
 Flatten[ParallelTable[{i, j}, {i, 0, t}, {j, 0, t}],
  1];
(*Print[ConeC];*)

T1 = ConvexHullMesh[
  Join @@ {{{0, 0}}, 20*vectorsinrays}];

ConeC = Select[ConeC, Element[#, T1] &];

vectorsinrays =
 ParallelTable[{{0, 0}, 20*vectorsinrays[[i]]},
```

```
      {i, Length[vectorsinrays]}];
    (*Print[vectorsinrays];*)

    Print[
     Show[
      {
       ListPlot[ConeC, PlotStyle → Red],
       ListPlot[Gaps, PlotStyle → White,
        PlotMarkers → "OpenMarkers"],
       ListPlot[PseuFrobs, PlotStyle → Brown,
        PlotMarkers → "OpenMarkers"],
       ListPlot[EspGaps, PlotStyle → Green,
        PlotMarkers → "OpenMarkers"],
       ListPlot[GenSemig, PlotStyle → Yellow,
        PlotMarkers → "OpenMarkers"],
       Graphics[{Red, Thick, Line[vectorsinrays]}]
       (*Graphics3D[{Red,Line /@ T1L }]*)
      },
      AxesOrigin → {0, 0}, AspectRatio → Automatic,
      Show → True, AxesLabel → {X, Y}
      (*,AxesStyle→{Black,Red,Blue}*)
     ]
    ];
    Return[]
   ];
```

Dibujando Cono con todo símbolos

In[275]:=

```
Plot2DSemigAllBW[GenSemig_,Gaps_]:=Module[{PtosEnS,semig,coe
(*Dibuja el C-semigrupo de N^2 generado por Gen_Semig, con
semig=GenSemig;
```

```
L=Length[semig];

PseuFrobs = GetPseudoFrobenius[GenSemig,Gaps];
EspGaps= GetEspGaps[PseuFrobs,Gaps] ;

T1=ConvexHullMesh[Join @@ {{{0,0}},semig}];
LineasT1=MeshPrimitives[T1,1];T1L=Select[Level[LineasT1,{2}],
vectorsinrays=Select[Flatten[T1L,1],# ≠{0,0}&];
Print["Vectores de los rayos extremales= ",vectorsinrays];

t=Ceiling[Max[Join[semig,Gaps]]];
(*Print[Max[Join[semig,Gaps]]," ",t];*)
ConeC=Flatten[ParallelTable[{i,j},{i,0,t},{j,0,t}],1];
(*Print[ConeC];*)

T1=ConvexHullMesh[Join @@ {{{0,0}},20*vectorsinrays}];

ConeC=Select[ConeC,Element[#,T1]&];

vectorsinrays=ParallelTable[{{0,0},20*vectorsinrays[[i]]},{i,Le
(*Print[vectorsinrays];*)

Print[
Show[
{
ListPlot[Complement[ConeC,GenSemig],PlotStyle→Red],
ListPlot[Gaps,PlotStyle→White,PlotMarkers→"OpenMarkers"],
ListPlot[Complement[PseuFrobs,EspGaps],PlotStyle→Black,Plot
ListPlot[EspGaps,PlotStyle→Blue,PlotMarkers→"□"],
ListPlot[GenSemig,PlotStyle→Orange,PlotMarkers→"▾"],
Graphics[{Black,Thin,Line[vectorsinrays]}]
(*Graphics3D[{Red,Line /@ T1L }]*)
},
AxesOrigin→{0,0},AspectRatio→Automatic,Show→True,AxesLabel
```

```
(*,AxesStyle→{Black,Red,Blue}*)
]
];
Return[]
];
```

Dibujando Cono símbolos ajustando

In[276]:=
```
Plot2DSemigAllBW2[GenSemig_,Gaps_]:=Module[{PtosEnS,semig,co
(*Dibuja el C-semigrupo de N^2 generado por Gen_Semig, con
semig=GenSemig;
L=Length[semig];

PseuFrobs = GetPseudoFrobenius[GenSemig,Gaps];
EspGaps= GetEspGaps[PseuFrobs,Gaps] ;

T1=ConvexHullMesh[Join @@ {{{0,0}},semig}];
LineasT1=MeshPrimitives[T1,1];T1L=Select[Level[LineasT1,{2}],
vectorsinrays=Select[Flatten[T1L,1],# ≠{0,0}&];
Print["Vectores de los rayos extremales= ",vectorsinrays];
(*t=Ceiling[Max[Join[semig,Gaps]]]*)
t=29;
(*Print[Max[Join[semig,Gaps]]," ",t];*)
ConeC=Flatten[ParallelTable[{i,j},{i,0,t},{j,0,t}],1];
(*Print[ConeC];*)

T1=ConvexHullMesh[Join @@ {{{0,0}},20*vectorsinrays}];

ConeC=Select[ConeC,Element[#,T1]&];

vectorsinrays=ParallelTable[{{0,0},20*vectorsinrays[[i]]},{i,Le
(*Print[vectorsinrays];*)
```

```
Print[
Show[
{
ListPlot[Complement[ConeC,GenSemig],PlotStyle→Red],
ListPlot[Gaps,PlotStyle→White,PlotMarkers→"OpenMarkers"],
ListPlot[Complement[PseuFrobs,EspGaps],PlotStyle→Black,Plot
ListPlot[EspGaps,PlotStyle→Blue,PlotMarkers→"□"],
ListPlot[GenSemig,PlotStyle→Orange,PlotMarkers→"▼"],
Graphics[{Black,Thin,Line[vectorsinrays]}]
(*Graphics3D[{Red,Line /@ T1L }]*)
},
AxesOrigin→{0,0},AspectRatio→Automatic,Show→True,AxesLabel
(*,AxesStyle→{Black,Red,Blue}*)
]
];
Return[]
];
```

Dibujando puntos dado vectores cono

In[277]:=
```
Plot2DConeDotsSmallValues[Dots_, vectorConeRays_] :=
  Module[{vectorRays, PtosEnS, semig, coeficientes,
     i, j, t, L, T1, LineasT1, T1L, vectorsinrays,
     T2, ConeC, PseuFrobs, EspGaps},
    (*Dibuja el C-semigrupo de N^2 generado
      por Gen_Semig, con huecos Gaps,
    pseudos PseuFrobs y especiales EspGaps*)

    vectorRays = vectorConeRays;
    t = 2*Ceiling[Max[Join[Dots, vectorRays]]];
    (*Print[Max[Join[semig,Gaps]]," ",t];*)
    ConeC =
```

```
   Flatten[ParallelTable[{i, j}, {i, 0, t}, {j, 0, t}],
    1];
  (*Print[ConeC];*)

  T1 = ConvexHullMesh[
    Join @@ {{{0, 0}}, 10*vectorRays}];

  ConeC = Select[ConeC, Element[#, T1] &];

  vectorRays = ParallelTable[
    {{0, 0}, 10*vectorRays[[i]]},
    {i, Length[vectorRays]}];
  (*Print[vectorsinrays];*)
  Print[vectorRays];
  Print[
   Show[
    {
     ListPlot[ConeC, PlotStyle → Red],
     ListPlot[Dots,
      PlotStyle → {Yellow, PointSize[Large]}],
     Graphics[{Red, Thick, Line[vectorRays]}]
     (*Graphics3D[{Red,Line /@ T1L }]*)
    },
    AxesOrigin → {0, 0}, AspectRatio → Automatic,
    Show → True, AxesLabel → {X, Y}
    (*,AxesStyle→{Black,Red,Blue}*)
   ]
  ];
  Return[]
 ];
```

In[278]:=
```
Plot2DConeDotsBigValues[Dots_, vectorConeRays_] :=
```

```
Module[{vectorRays, PtosEnS, semig, coeficientes,
  i, j, t, L, T1, LineasT1, T1L, vectorsinrays,
  T2, ConeC, PseuFrobs, EspGaps},
 (*Dibuja el C-semigrupo de N^2 generado
   por Gen_Semig, con huecos Gaps,
 pseudos PseuFrobs y especiales EspGaps*)

 vectorRays = vectorConeRays;
 t = 2 * Ceiling[Max[Join[Dots, vectorRays]]];
 (*Print[Max[Join[semig,Gaps]]," ",t];*)
 ConeC =
  Flatten[ParallelTable[{i, j}, {i, 0, t}, {j, 0, t}],
   1];
 (*Print[ConeC];*)

 T1 = ConvexHullMesh[
   Join @@ {{{0, 0}}, 2 t / 3 * vectorRays}];

 ConeC = Select[ConeC, Element[#, T1] &];

 vectorRays = ParallelTable[
   {{0, 0}, 2 t / 3 * vectorRays[[i]]},
   {i, Length[vectorRays]}];
 (*Print[vectorsinrays];*)
 Print[vectorRays];
 Print[
  Show[
   {
    ListPlot[ConeC, PlotStyle → Red],
    ListPlot[Dots,
     PlotStyle → {Yellow, PointSize[Large]}],
    Graphics[{Red, Thick, Line[vectorRays]}]
```

```
      (*Graphics3D[{Red,Line /@ T1L }]*)
     },
     AxesOrigin → {0, 0}, AspectRatio → Automatic,
     Show → True, AxesLabel → {X, Y}
     (*,AxesStyle→{Black,Red,Blue}*)
    ]
   ];
   Return[]
  ];
```

Dibujando puntos dado vectores cono y veces múltiplo del vector

In[279]:=
```
Plot2DConeDotsRegSize[Dots_, vectorConeRays_,
   multiple_] :=
  Module[{vectorRays, PtosEnS, semig, coeficientes,
    i, j, t, L, T1, LineasT1, T1L, vectorsinrays,
    T2, ConeC, PseuFrobs, EspGaps},
   (*Dibuja el C-semigrupo de N^2 generado
     por Gen_Semig, con huecos Gaps,
   pseudos PseuFrobs y especiales EspGaps*)

   vectorRays = vectorConeRays;
   t = multiple *
     Ceiling[Max[Join[Dots, vectorRays]]];
   (*Print[Max[Join[semig,Gaps]]," ",t];*)
   ConeC =
    Flatten[ParallelTable[{i, j}, {i, 0, t}, {j, 0, t}],
     1];
   (*Print[ConeC];*)
```

```
    T1 = ConvexHullMesh[
      Join @@ {{{0, 0}}, 10 * vectorRays}];

    ConeC = Select[ConeC, Element[#, T1] &];

    vectorRays = ParallelTable[
      {{0, 0}, 10 * vectorRays[[i]]},
      {i, Length[vectorRays]}];
    (*Print[vectorsinrays];*)
    Print[vectorRays];
    Print[
     Show[
      {
       ListPlot[ConeC, PlotStyle → Red],
       ListPlot[Dots,
        PlotStyle → {Yellow, PointSize[Large]}],
       Graphics[{Red, Thick, Line[vectorRays]}]
       (*Graphics3D[{Red,Line /@ T1L }]*)
      },
      AxesOrigin → {0, 0}, AspectRatio → Automatic,
      Show → True, AxesLabel → {X, Y}
      (*,AxesStyle→{Black,Red,Blue}*)
     ]
    ];
    Return[]
   ];
```

Algoritmo 2 y 3 -> Obtener descomposición en irreducibles y

descomposición minimal en irreducibles

Auxiliares

GetSInfo

In -> Generadores y huecos de un semigrupo S
Out -> Devuelve Lista { {“Índices de huecos que son PF(S)”}, {“Índice de huecos que son SGS”}, nº SG(S) }

In[280]:=
```
GetSInfo[GenSemig_,Gaps_] := Module[{nGaps,i,j,nGens,EspGaps
nGens = Length[GenSemig];
nGaps = Length[Gaps];
PseuFrobs={};
EspGaps={};

For[i=1,i≤ nGaps,i++,
    j=1;

    While[(j ≤ nGens)∧ (¬MemberQ[Gaps,Gaps[[i]]+GenSemig[[j]] ]) ,
        j++
    ];

    If[ j == nGens+1,
        AppendTo[PseuFrobs,i];
        If[¬MemberQ[Gaps,2*Gaps[[i]]],
            AppendTo[EspGaps,i];
        ]
    ];
];
Print["Out getsinfo -> ",{PseuFrobs,EspGaps,Length[EspGaps]}]
Return[{PseuFrobs,EspGaps,Length[EspGaps]}]
];
```

GetPFandSGLemma

In -> Dado de S inicial los Geners, Huecos, índice del hueco especial “a” a añadir y Los elementos añadidos anteriormente y las Ecuaciones del Cono.
Out -> Devolvemos de S’ = S ∪ { a } la lista {{“Los indices de elementos que ya no son huecos”}, {“Los índices de sus pseudos”}, {“los índices de sus sgaps”}}

In[281]:=
```
GetPFandSGLemma[GS_,HS_,sg_,G_,PF_,Eq_,verb_]:=Module[{PsF,Sp
```

```
If[verb,Print["In GetPFandSGLemma-> \n Hueno especial a

PsF={};
SpG={};

nPF=Length[PF];
nG=Length[G];
nGS=Length[GS];
p=0;

(*Elementos incluidos en S' a partir de los índices de
GapsAdded=ParallelTable[HS[[G[[j]]]],{j,1,nG}];

(* Casos 1 y 2 Lema → A partir Pseudofrobenius anterio
For[i=1,i≤nPF,i++,
    If[PF[[i]]≠sg,
        (*Caso 1 Lema*)
        pf = HS[[ PF[[i]] ]] ;
        pf2 = HS[[ PF[[i]] ]] + HS[[sg]];

        (*Comprobamos si posible pf verifica pf + a no
        If[(¬MemberQ[HS, pf2 ])∨ (MemberQ[GapsAdded, pf2
            (*Añadimos el índice del pf a PsF*)
            AppendTo[PsF,PF[[i]]];
            (*Comprobamos si es huecos especial*)
            If[(¬MemberQ[HS, 2*pf])∨(MemberQ[GapsAdded, 2*
                (*Añado índice para los nuevos especia
                AppendTo[SpG,PF[[i]]];
                p++
            ],
        (*else*)
        (*CASO 2 LEMA*)
            aux=Position[HS,pf2,1,1][[1,1]];
            If[verb,Print["aux caso 2-> ",aux]];
```

```
                    If[(¬MemberQ[PF,aux]),
                        (*Añadimos el índice del pf2 a PsF*)
                        AppendTo[PsF,aux];
                        (*Comprobamos si es hueco especial*)
                        If[(¬MemberQ[HS, 2*pf2])∨(MemberQ[GapsAdde
                            AppendTo[SpG,Last[PsF]];
                            p++
                        ]
                    ]
                ]
            ]
        ];

(* CASO 3 LEMA → A partir de Geners y Huecos anteriore
For[i=1,i≤nGS,i++,
    pf= HS⟦sg⟧ - GS⟦i⟧;
    If[InCone[pf,Eq],
    (*Comprobamos si el candidato pf es hueco de S y no
    aux=Position[HS,pf,1,1]⟦1,1⟧;
    If[verb,Print["aux caso 3-> ",aux, " pf-> ",pf]];
    If[((MemberQ[HS,pf])∧(¬MemberQ[GapsAdded, pf ]))∧(¬Membe
        pf2= pf + HS⟦sg⟧;
        (*Comprobamos si pf + a pertenece a S*)
        If[(¬MemberQ[HS,pf2])∨(MemberQ[GapsAdded,pf2]),
            (*Comprobamos si pf + Geners⟦j⟧ no es u
            If[verb,Print["  aux caso 3-> ",pf, " p
            For[j=1,j≤ nGS,j++,
                pf2 = pf + GS⟦j⟧;
                If[j==i,
                    j++,
                (*else*)
                    If[(MemberQ[HS,pf2])∧(¬MemberQ[Ga
                        If[verb,Print["     aux ca
```

```
                                                            (MemberQ[
                                            j++;
                                            Break[];
                                        ]
                                    ]
                                ]
                            ];

                (*Comprobamos si ha ido bien lo anterior*)
                If[ j == nGS+1,
                    (*Añadimos el índice del pf a PsF*)
                     AppendTo[PsF,aux];
                    (*Comprobamos si es nuevo special gap*)
                    If[(¬MemberQ[HS, 2*pf])∨(MemberQ[GapsAdded, 2*
                        AppendTo[SpG,Last[PsF]];
                        p++
                    ]
                ]
            ]
        ]
    ];

    If[verb,Print["Out GetPFandSGLemma-> ",{{Append[G,sg],PsF
    Return[{{Append[G,sg],PsF,SpG},p}]
];
```

FirstIandCsets

In -> Dado los Geners, Gaps, ... of S C-Semigroup.

Out -> Returns algorithm 1’s firsts I and C sets

In[282]:=
```
FirstIandCsets[GS_,HS_,PF_,SG_,Eq_,verb_]:=Module[{C,I,i,nSG2
    I={};
    C={};
    nSG=Length[SG];
    If[verb,Print["In FirstIandCsets-> "]];
    For[i=1,i≤nSG,i++,
        {Sets,nSG2}=GetPFandSGLemma[GS,HS,SG[[i]],{},PF,Eq,ver

        If[nSG2≤1,
            AppendTo[I,Sets],
        (*else*)
            AppendTo[C,Sets];
        ];

        Sets={}
    ];

    If[verb,Print["Out FirstIandCsets-> ",{I,C}]];
    Return[{I,C}]
]
```

CheckIfinI

In -> Dado Conjunto de huecos de S añadidos como elementos a S’ y el conjunto I.
Out -> Devuelve False si ∃ S’’ ∊ I tal que S’’ ⊂ S, True en caso contrario.

In[283]:=
```
CheckIfinI[GC_,I_]:=Module[{nI,aux,i},
    nI=Length[I];
    aux=True;

    For[i=1,((i≤nI)∧(aux)),i++,
        If[SubsetQ[I⟦i,1⟧,GC],
            aux=False
        ];
    ];

    Return[aux];
]
```

IandCsets

Bajo contexto Algoritmo 1 sobre un C-Semigrupo S con Geners ⇌ GS y huecos ⇌ HS.
In -> Dado conjunto I de irreducibles, C1 de reducibles, Geners GS y Huecos HS de S y las Ecuaciones Eq del cono.
Out -> Devolvemos aplicado el algoritmo 1, los nuevos conjuntos I y C obtenidos.

In[284]:=
```
IandCsets[I_,C1_,GS_,HS_,Eq_,verb_]:=Module[{C,cn,Iaux,Caux,a
    C=C1;
    If[verb,Print["In IandCsets-> "]];
    cn=Length[C];
    Caux={};
    Iaux={};
    SemigDone={};

    Print["Nº de ♯C ",cn];
```

```
While[1≤cn,
    nsg=Length[C⟦1,3⟧];
    For[j=1,j≤nsg,j++,
        aux=Sort[Append[C⟦1,1⟧, C⟦1,3,j⟧ ]];
        If[¬MemberQ[SemigDone,aux],
            If[CheckIfinI[ aux , I ],
                {Sets,nsg2}=GetPFandSGLemma[GS,HS,C⟦1,3

                If[nsg2==1,
                    (*Irreducible*)
                    If[Length[Sets⟦2⟧]==1,
                        (*Simetrico*)
                        AppendTo[Iaux,Append[Sets,1]],
                        (*PseudoSimetrico*)
                        AppendTo[Iaux,Append[Sets,0]],
                    ];
                    Sets={},
                (*else*)
                    (*No Irreducible*)
                    AppendTo[Caux,Sets];
                    Sets={}
                ]
            ];
            AppendTo[SemigDone,aux];
            If[verb,Print["SemigDone-> ",SemigDone]];
        ];
    ];

    C = C⟦2;;⟧;
    cn=cn-1;
];
If[verb,Print["Out IandCsets-> ",{Union[I,Iaux],Caux}]];
Return[{Union[I,Iaux],Caux}];
```

```
]
```

SearchSubset

In -> Dado Conjunto de huecos de S añadidos como elementos a S’ y el conjunto I.
Out -> Devuelve False si ∃ S’’ ∈ I tal que S’’ ⊂ S, True en caso contrario.

In[285]:=
```
SearchSubset[Setaux_,ListAux_,I_]:=Module[{nAux,i,j,sets,inde
    nAux=Length[ListAux];
    sets={};
    indexIs={};

    For[i=2,(i≤nAux),i++,
        If[SubsetQ[ListAux[[i]],Setaux],
            AppendTo[sets,{i}];
            aux=Position[I,ListAux[[i]]];
            AppendTo[indexIs,Table[{aux[[j,1]]},{j,1,Length[aux
            aux={};
        ];
    ];

    If[Length[sets]≥1,
        Return[{True,sets,indexIs}],
        Return[{False}]
    ]
]
```

In[•]:=
```
(*Si Special gaps con lenght>1*)
        If[(Length[listISG[[1]]] > 1),
            (*Compruebo si hay subconjuntos*)
            subsets = SearchSubset[listISG[[1]], listISG, I];
```

```
        (**Si hay subconjuntos los elemino de I y listISG**)
        If[subsets[[1]],
            listISG = Delete[listISG, subsets[[2]]];
            I = Delete[I, subsets[[3]]]
        ];

        If[verb,
            Print["   Length>1--"];
            Print["   subsets : ", subsets];
            Print["   I : ", I];
            Print["   listISG : ", listISG];
        ]
    ];
```

Part : Part specification listISG⟦1⟧ is longer than depth of object.

Part : Part specification listISG⟦1⟧ is longer than depth of object.

BetterOne

In[286]:=
```
BetterOne[Setaux_,I_]:=Module[{nAux,i,maxLength,indexIs,aux},
    nAux=Length[I];
    maxLength={0,0};
    indexIs={};

    For[i=1,(i≤nAux),i++,
        If[I〚i,2〛==Setaux,
            AppendTo[indexIs,{i}];
            If[I〚i,3〛>maxLength〚1〛,
                maxLength={I〚i,3〛,I〚i,1〛}
            ]
        ];
    ];

    Return[{maxLength〚2〛,indexIs}]
]
```

BestIrreducibles

CRITERIO: segunda parte de Algoritmo 2.

IDEA: algoritmo

Devuelvo los obtenidos.

In[287]:=
```
BestIrreducibles[I1_,SPG_,verb_]:=Module[{i,I,carI,CSi,A,PI,
    I=I1;
    carI=Length[I];
    PI=Range[carI];
    carSPG=Length[SPG];
```

```
    If[verb,
        Print["BestIrreducibles--"];
        Print["I= ",I];
        Print["PI= ",PI];
    ];

    CSi=Reverse[Sort[Table[{Complement[SPG,I⟦i,1⟧],i},{i,1,car

    If[verb,
        Print["CSi= ",CSi];
    ];

    For[i=2,i≤carI-1,i++,
        A=Subsets[PI,i];
        While[Length[A]>0,
            If[Length[Union[Flatten[CSi⟦A⟦1⟧,1⟧]]]==carSPG,
                (* Devuelve el conjunto encontrado que es
                Return[Flatten[CSi⟦A⟦1⟧,2⟧]];
            ];
            A=Delete[A,1];
        ];
    ];

    (*Devolvemos el conjunto completo*)
    Return[PI];
]
```

Algoritmo 2 -> GetIrreducibles

In -> Generadores “GensS” y huecos “GapsS” de un C-Semigrupo S; las ecuaciones “Eq” del cono al que pertenece S y el booleano “verb” para mostrar mensajes o no mientras se ejecuta el código. Out -> Devuelve Lista de { { {“Los indices de elementos que ya no son huecos”}, {“los índices de sus sgaps”} } ,... } de Irreducibles que componen S

In[288]:=
```
GetIrreducibles[GensS_,GapsS_,Eq_,verb_] := Module[{I,C,PFS,S
    {PFS,SGS,aux}=GetSInfo[GensS,GapsS];

    If[aux≤1,
        Return[{GensS,GapsS}]
    ];

    {I,C}=FirstIandCsets[GensS,GapsS,PFS,SGS,Eq,verb];

    If[verb,
        Print["I"];
        Print[I];
        Print["C"];
        Print[C];
        Print["dsp FirtI--"];
    ];

    While[C≠{},
        {I,C}=IandCsets[I,C,GensS,GapsS,Eq,verb];
        If[verb,
            Print["dsp IandC--"];
            Print["I",I];
            Print["C",C];
        ]
    ];

    If[verb,Print[I]];

    out=ParallelTable[{
```

```
    Union[ GensS , GapsS[[ i[[1]] ]] ],
    Delete[GapsS,Table[ {j},{j,i[[1]] } ] ],
    i[[4]] },
    {i,I}];

    Return[ out ]

]
```

Algoritmo 3 -> GetMinimalsIrreducibles

In -> Generadores “GensS” y huecos “GapsS” de un C-Semigrupo S; las ecuaciones “Eq” del cono al que pertenece S y los booleanos “verb” y “verbIrr” para mostrar mensajes o no mientras se ejecuta el código.
Out -> Devuelve Lista de { { {“Generadores Irreducible”}, {“Huecos Irreducibles”},”Simetrico == 1 o PseudoSim==0” } ,... } de Irreducibles que componen S

In[289]:=
```
GetMinimalsIrreducibles[GensS_,GapsS_,Eq_,verb_,verbIrr_] :=
    {PFS,SGS,aux}=GetSInfo[GensS,GapsS];

    If[aux≤1,
        Return[{GensS,GapsS}]
    ];

    {I,C}=FirstIandCsets[GensS,GapsS,PFS,SGS,Eq,verb];

    If[verb,
        Print["I"];
        Print[I];
        Print["C"];
        Print[C];
```

```
    Print["dsp FirtI--"];
];

While[C≠{},
    {I,C}=IandCsets[I,C,GensS,GapsS,Eq,verb];
    If[verb,
        Print["dsp IandC--"];
        Print["I",I];
        Print["C",C];
    ]
];


(*Obteniendo Irreducibles alg 2*)
out=I〚 BestIrreducibles[I,SGS,verbIrr] 〛;

If[verb,
    Print["Nº irreducibles 1 -> ",Length[I]];
    Print["Nº irreducibles 2 -> ",Length[out]];
];

minimals=ParallelTable[{MinGenGeneral[Union[GensS,GapsS〚

If[verb,
    Print["Alg 1 -----"];
    Print[ParallelTable[{
        Union[GensS,GapsS〚i〛],
        Delete[GapsS,Table[{j},{j,i}]]
        },{i,I〚;;,1〛}]];
    Print["Alg 2 -----"];
    Print[minimals]
];
```

```
    Return[minimals]
]
```

Algoritmo 4 -> Comprobar si $C\backslash X$ es C-semigrupo

Auxiliares

Construcción y comprobación D[X]

In-> Dado “A” subconjunto del cono con ecuaciones “Eq” y el booleano “verb” para indicar si mostrar los resultados que va calculando la función.

Out -> Devuelve D(“A”)

In[290]:=
```
IsDX[A_,Eq_,verb_]:= Module[{a,i,gcd,k,aux,aux2,auxgcd},
    (*Comprueba que los elementos de X están en el cono*)
    For[k=1,k≤Length[A],k++,
        aux = ¬InCone[A⟦k⟧,Eq];

        If[verb,
            Print["A[[k]]",A⟦k⟧];
            Print["¬InCone[x,Eq]->",aux]
        ];

        If[aux,
            If[verb, Print["Exist element out of cone ->",A
            Return[False]];
    ];
```

```
    (*Comprueba que X ⊂ D(X)*)
    For[k=1,k≤Length[A],k++,
        gcd=Divisors[Apply[GCD,A〚k〛]];
        auxgcd=Length[gcd];

        If[verb,
            Print["A[[k]]",A〚k〛];
            Print["gcd->",gcd];
            Print["Lenght[gcd]->",auxgcd]
        ];

        If[auxgcd ≠ 0,
            For[i=1,i≤auxgcd,i++,
                aux=A〚k〛/gcd〚i〛;
                aux2=¬MemberQ[A,aux];
                If[verb,
                    Print["A[[k]]/gcd[[i]]->",aux];
                    Print["¬MemberQ[A,A[[k]]/gcd[[i]]]->",aux2];
                ];
                If[aux2,
                    Print["X≠D(X) - This element causes thi
                    Return[False]
                ]
            ]
        ]

    ];

    If[verb, Print["X subset Cone"]];
    Return[True]
]
```

Final

In-> “X” subconjunto del cono natural C con vectores “VectorConeRay”, “dirTrab” el directorio del notebook y los booleanos “verb”/”verbMore” para indicar que muestre los resultados que va obteniendo la función.

Out -> Devuelve “True” si C\X es un C-semigrupo y “False” en caso contrario.

In[291]:=
```
IsSetMinusCS[X_,VectorConeRays_,dirTrab_,verb_,verbMore_] :=
    (*Ecuaciones del cono para comprobar si un punto está e
    Eq=ConeGenSupHyp[VectorConeRays,dirTrab,operatingSyste
    auxX = Length[X];

    (*Comprueba que X no sea el conjunto vacío*)
    If[X=={},
        If[verb, Print["X is void"]];
        Return[True]];
    If[verb, Print["X not void"]];

    (*Comprueba que X sea igual a D(X)*)
    If[¬IsDX[X,Eq,verbMore],
        If[verb, Print["X is not D(X) or X not subset Cone"]
        Return[False]];
    If[verb, Print["X subset Cone & X = D(X)"]];

    (*Ordenamos X para construir de sencillo a complejo.*)
    X1 = ReverseSort[LexicographicSort[X]];
    If[verb, Print["X ordered"]];

    (*Comprobaremos que se verifica Proposición 5.1: X1[[k]]-
    For[k=1,k≤auxX,k++,
```

```
        If[verbMore, Print["Element x -> ",X1[[k]]]];

        (*Calcularemos A: puntos s de C\X1 tales que s≤CX1[[
        (*Inicializando A → coordenadas menores que las de
        A=Flatten[ParallelTable[{i,j},{i,0,X1[[k]][[1]]},{j,0,X1[[k

        (*Tomando puntos pertenecientes al cono*)
        A=Select[A,InCone[#,Eq]&];

        (*Tomando puntos de C\X1*)
        A = Complement[A,X];
        A = Select[A,InCone[X1[[k]]-#,Eq]&];
        If[verbMore, Print["A computed -> ",A]];

        (*Comprueba x-a∈X para cada a∈A*)
        For[l=1,l≤Length[A],l++,
            If[¬MemberQ[X1,X1[[k]]-A[[l]]],
                If[verb, Print["x-a not in X ->", X1[[k]],"-",
                Return[False]]
        ];

        If[verbMore, Print["Element x correctly passed -> "
    ];

    (*Tras comprobar lo anterior, confirmamos C\X es C-semi
    If[verb, Print["C \ X is a C-semigroup"]];
    Return[True]
]
```

■ Algoritmo 5 -> Comprobar si $\mathcal{C}\backslash X$ es $\mathcal{C}$-semigrupo y generadores

minimales

In-> “X” subconjunto del cono natural C con generadores “GenCone” / vectores “VectorConeRays”, el orden total “Order”, “dirTrab” el directorio del notebook y los booleanos “verb”/”verbMore” para indicar que muestre los resultados que va obteniendo la función.

Out -> Devuelve “True” si C\X es un C-semigrupo y “False” en caso contrario.

In[292]:=

```
SetMinusCSLast[setX_,GenCone_,Order_,VectorConeRays_,dirTra
    (*Ecuaciones del cono para comprobar si un punto está e
    Eq=ConeGenSupHyp[VectorConeRays,dirTrab,operatingSyste
    genCone=Eq⟦1⟧;
    Eq=Eq⟦2⟧;

    (*Calculamos el tamaño de X*)
    (*Ordenamos X para construir de sencillo a complejo.*)
    X = ReverseSort[LexicographicSort[setX]];
    auxX = Length[X];

    (*Comprueba que los elementos de X están en el cono*)
    For[k=1,k≤auxX,k++,
        aux = ¬InCone[X⟦k⟧,Eq];
        If[verb, Print["InCone[x,Eq]->",aux]];
        If[aux, Return[False]]];
    If[verb, Print["X subset Cone"]];

    (*Comprueba si X es subconjunto de GenCone*)
    If[¬SubsetQ[A,X], Return[{}]];
    If[verb, Print["X not subset GenCone"]];

    (*Comprueba que X sea igual a D(X)*)
```

```
    If[¬IsDX[X,Eq,verbMore], Return[False]];
    If[verb, Print["X = D(X)"]];

    (*Comprueba que X⟦1⟧ pertenece a los generadores gemCon
    If[¬MemberQ[GenCone,X⟦1⟧], Return[{}]];

    (*Calcula los generadores minimales de C\{X⟦1⟧}*)
    A = MinGenGeneral[DeleteCases[genCone,X⟦1⟧],{X⟦1⟧},Eq];

    (*Si los huecos restantes pertenecen a C\{X⟦1⟧}, devolve
    If[SubsetQ[A⟦2⟧,X⟦2;;⟧],
        Return[ A⟦1⟧ ]
    ];

    (*A será el C-semigrupo que obtendremos en cada iterac
    For[k=2,k≤auxX,k++,
        (*Comprueba que X⟦k⟧ pertenece a los generadores de
        If[¬MemberQ[A⟦1⟧,X⟦k⟧], Return[{}]];

        (*Calcula los generadores minimales de A\{X⟦k⟧}*)
        A = MinGenGeneral[DeleteCases[A⟦1⟧,X⟦k⟧],Union[A⟦2⟧,X⟦

        (*Si los huecos restantes pertenecen a A\{X⟦k⟧}, dev
        If[SubsetQ[A⟦2⟧,X⟦k+1;;⟧],
            Return[ A⟦1⟧ ]
        ];
    ];
]
```

Ejemplos

Capítulo 1

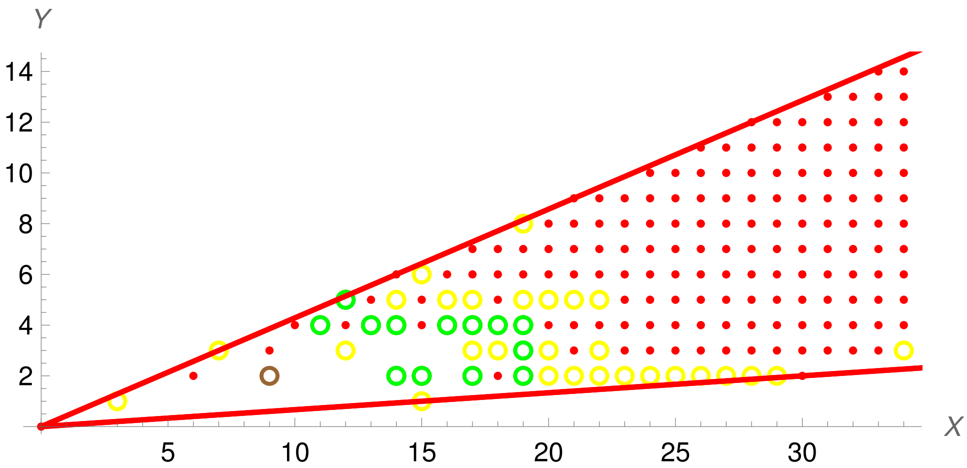
Ejemplo 1.1 en N^2

Semigrupo

In[293]:=
```
goodExample={{{3,1},{7,3},{12,3},{14,5},{15,1},{15,6},{16,5},{17,3}
{{4,1},{5,1},{5,2},{6,1},{7,1},{7,2},{8,1},{8,2},{8,3},{9,1},{9,2},{1(
```

In[294]:=
```
Plot2DSemigAll[goodExample⟦1⟧,goodExample⟦2⟧]
```

Vectores de los rayos extremales= {{15, 1}, {7, 3}}



In[295]:=
```
(* Vectores de los rayos extremales para sacar ecuaciones de
semig=goodExample⟦1⟧;
T1=ConvexHullMesh[Join @@ {{{0,0}},semig}];
LineasT1=MeshPrimitives[T1,1];
T1L=Select[Level[LineasT1,{2}],MemberQ[#,{0,0}]&];
vectorsinrays=Select[Flatten[T1L,1],# ≠{0,0}&]
```

Out[299]= {{15, 1}, {7, 3}}

Huecos, huecos pseudo-Frobenius y especiales

```
In[300]:= Print["G(S) -> ",goodExample[[2]]]
          gS = Length[goodExample[[2]]];
          Print["g(S) -> ",Length[goodExample[[2]]]]

          pseuFrobs = GetPseudoFrobenius[goodExample[[1]],goodExample[[2]]]
          tS=Length[pseuFrobs];
          Print["PF(S) -> ",pseuFrobs]
          Print["t(S) -> ",tS]

          espGaps=GetEspGaps[pseuFrobs,goodExample[[2]]];
          Print["SG(S) -> ",espGaps]

          espGaps == pseuFrobs

          G(S) -> {{4, 1}, {5, 1}, {5, 2}, {6, 1}, {7, 1}, {7, 2},
            {8, 1}, {8, 2}, {8, 3}, {9, 1}, {9, 2}, {10, 1}, {10, 2},
            {10, 3}, {11, 1}, {11, 2}, {11, 3}, {11, 4}, {12, 1},
            {12, 2}, {12, 5}, {13, 1}, {13, 2}, {13, 3}, {13, 4}, {14, 1},
            {14, 2}, {14, 3}, {14, 4}, {15, 2}, {15, 3}, {16, 2}, {16, 3},
            {16, 4}, {17, 2}, {17, 4}, {18, 4}, {19, 2}, {19, 3}, {19, 4}}

          g(S) -> 40

          PF(S) -> {{9, 2}, {11, 4}, {12, 5}, {13, 4}, {14, 2}, {14, 4}, {15, 2},
            {16, 4}, {17, 2}, {17, 4}, {18, 4}, {19, 2}, {19, 3}, {19, 4}}

          t(S) -> 14

          SG(S) -> {{11, 4}, {12, 5}, {13, 4}, {14, 2}, {14, 4}, {15, 2},
            {16, 4}, {17, 2}, {17, 4}, {18, 4}, {19, 2}, {19, 3}, {19, 4}}

Out[309]= False
```

Vector Frobenius y N(S) en distintos órdenes

In[310]:=
```
Print["Empleando orden Lexicográfico:"]
Print["v. Frob -> ",FrobeniusLexi[goodExample[[2]]]]
nLexi = NFrobeniusLexi[goodExample[[1]],goodExample[[2]]]

Print["Empleando orden Lexicográfico graduado:"]
Print["v. Frob -> ",FrobeniusDLexi[goodExample[[2]]]]
nDLexi = NFrobeniusDLexi[goodExample[[1]],goodExample[[2]]]

Print["Empleando orden Lexicográfico graduado inverso:"]
Print["v. Frob -> ",FrobeniusDRLexi[goodExample[[2]]]]
nDRLexi = NFrobeniusDRLexi[goodExample[[1]],goodExample[[2]]]

nLexi == nDLexi
nDLexi == nDRLexi
nLexi == nDRLexi
```

```
Empleando orden Lexicográfico:
v. Frob -> {19, 4}
```

Out[312]=
```
{{3, 1}, {6, 2}, {7, 3}, {9, 3}, {10, 4}, {12, 3}, {12, 4},
 {13, 5}, {14, 5}, {14, 6}, {15, 1}, {15, 4}, {15, 5},
 {15, 6}, {16, 5}, {16, 6}, {17, 3}, {17, 5}, {17, 6},
 {17, 7}, {18, 2}, {18, 3}, {18, 5}, {18, 6}, {18, 7}}
```

```
Empleando orden Lexicográfico graduado:
v. Frob -> {19, 4}
```

Out[315]=
```
{{3, 1}, {6, 2}, {7, 3}, {9, 3}, {10, 4}, {12, 3},
 {12, 4}, {13, 5}, {14, 5}, {14, 6}, {15, 1},
 {15, 4}, {15, 5}, {15, 6}, {16, 5}, {16, 6}, {17, 3},
 {17, 5}, {17, 6}, {18, 2}, {18, 3}, {18, 5}, {20, 2}}
```

```
Empleando orden Lexicográfico graduado inverso:
v. Frob -> {19, 4}
```

Out[318]= {{3, 1}, {6, 2}, {7, 3}, {9, 3}, {10, 4}, {12, 3},
{12, 4}, {13, 5}, {14, 5}, {14, 6}, {15, 1},
{15, 4}, {15, 5}, {15, 6}, {16, 5}, {16, 6}, {17, 3},
{17, 5}, {17, 6}, {18, 2}, {18, 3}, {18, 5}, {20, 2}}

Out[319]= False

Out[320]= True

Out[321]= False

Desigualdad g(S) ≤ t(S)·n(S)

```
In[322]:= gS ≤ tS * Length[nLexi]
          gS ≤ tS * Length[nDLexi]
          gS ≤ tS * Length[nDRLexi]
```

Out[322]= True

Out[323]= True

Out[324]= True

Capítulo 3

Estos dos ejemplos son obtenidos a partir del algoritmo 3 aplicado a un C-semigrupo.

Ejemplo 3.1 en N^2

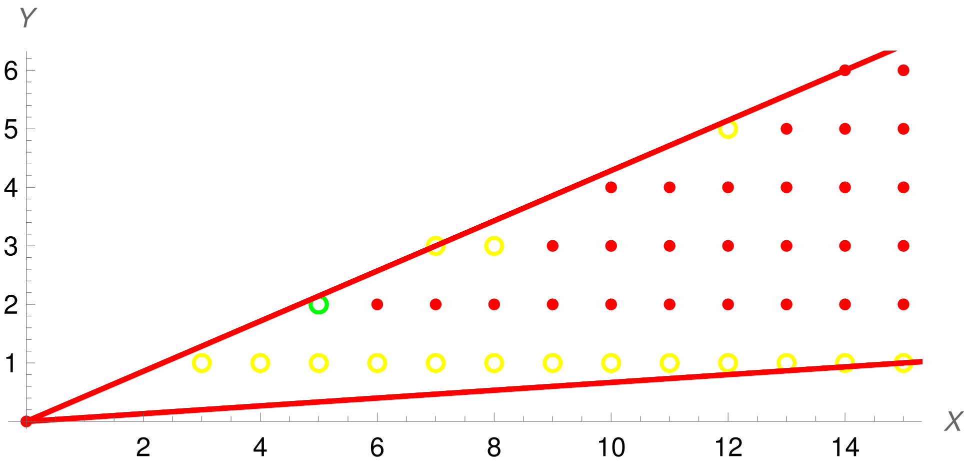
Semigrupo

In[325]:=
```
simExample={{{{3,1},{4,1},{5,1},{6,1},{7,1},{7,3},{8,1},{8,3},{9,1},{
```

Out[325]=
```
{{{{3, 1}, {4, 1}, {5, 1}, {6, 1}, {7, 1}, {7, 3},
    {8, 1}, {8, 3}, {9, 1}, {10, 1}, {11, 1}, {12, 1},
    {12, 5}, {13, 1}, {14, 1}, {15, 1}}, {{5, 2}}}, 1}
```

In[326]:=
```
Plot2DSemigAll[simExample[[1,1]],simExample[[1,2]]];
```

```
Vectores de los rayos extremales= {{15, 1}, {7, 3}}
```



Huecos, huecos pseudo-Frobenius y especiales

In[327]:=
```
Print["G(S) -> ",simExample[[1,2]]]
simgS = Length[simExample[[1,2]]];
Print["g(S) -> ",Length[simExample[[1,2]]]]

simpseuFrobs = GetPseudoFrobenius[simExample[[1,1]],simExample
simtS = Length[simpseuFrobs];
Print["PF(S) -> ",simpseuFrobs]
Print["t(S) -> ",simtS]

simespGaps=GetEspGaps[simpseuFrobs,simExample[[1,2]]];
Print["SG(S) -> ",simespGaps]

simespGaps == simpseuFrobs
```

```
G(S) -> {{5, 2}}
g(S) -> 1
PF(S) -> {{5, 2}}
t(S) -> 1
SG(S) -> {{5, 2}}
```

Out[336]= True

Vector Frobenius y N(S) en distintos órdenes

Al existir sólo un hueco, este será el vector Frobenius para cualquier orden fijado.

In[337]:=
```
simVFrob = simExample[[1,2,1]]
simnLexi = NFrobeniusLexi[simExample[[1,1]],simExample[[1,2]]]
```

Out[337]= {5, 2}

Out[338]= {{3, 1}, {4, 1}, {5, 1}}

Ap(S,b) para distintos órdenes

In[339]:=
```
(*Punto y directorio del archivo*)
fileDirectory=NotebookDirectory[];

simApery=GetApery[simExample[[1,1]][[1]],simExample[[1,1]],simExamp

simSubstractApery = Table[simApery[[i]] - simExample[[1,1]][[1]],{i,

MemberQ[simSubstractApery,simVFrob]
```

Out[340]= {{8, 3}}

Out[341]= {{5, 2}}

Out[342]= True

I(F(S)) y $\mathcal{F}$(S) para distintos órdenes

In[343]:=
```
fileDirectory=NotebookDirectory[];

simIFS = GetIsetNoEq[simExample[[1,2,1]],simExample[[1,1]],simExa
Length[simIFS]

sim𝓕S = Length[simIFS] + simgS
```

Out[344]= {{0, 0}}

Out[345]= 1

Out[346]= 2

In[347]:=
```
Length[simIFS] ≤ simgS
simgS == Length[simIFS]
```

Out[347]= True

Out[348]= True

In[349]:=
```
2*simgS == sim𝓕S
```

Out[349]= True

Ejemplo 3.2 en N^2

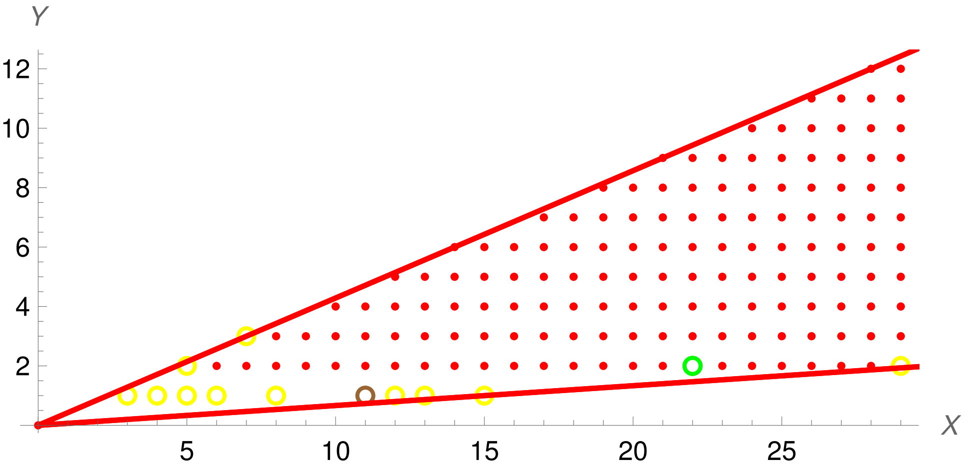
Semigrupo

In[350]:=
```
psimExample={{{{3,1},{4,1},{5,1},{5,2},{6,1},{7,3},{8,1},{12,1},{13,
```

Out[350]=
```
{{{{3, 1}, {4, 1}, {5, 1}, {5, 2}, {6, 1},
    {7, 3}, {8, 1}, {12, 1}, {13, 1}, {15, 1}, {29, 2}},
   {{7, 1}, {9, 1}, {10, 1}, {11, 1}, {14, 1}, {22, 2}}}, 0}
```

In[351]:=
```
Plot2DSemigAll[psimExample[[1,1]],psimExample[[1,2]]]
```

```
Vectores de los rayos extremales= {{15, 1}, {7, 3}}
```



Huecos, huecos pseudo-Frobenius y especiales

In[352]:=
```
Print["G(S) -> ", psimExample[[1,2]]]
psimgS = Length[psimExample[[1,2]]];
Print["g(S) -> ", Length[psimExample[[1,2]]]]

psimpseuFrobs = GetPseudoFrobenius[psimExample[[1,1]], psimExa
psimtS = Length[psimpseuFrobs];
Print["PF(S) -> ", psimpseuFrobs]
Print["t(S) -> ", psimtS]

psimespGaps=GetEspGaps[psimpseuFrobs, psimExample[[1,2]]];
Print["SG(S) -> ", psimespGaps]

psimespGaps == psimpseuFrobs
```

```
G(S) -> {{7, 1}, {9, 1}, {10, 1}, {11, 1}, {14, 1}, {22, 2}}
g(S) -> 6
PF(S) -> {{11, 1}, {22, 2}}
t(S) -> 2
SG(S) -> {{22, 2}}
```

Out[361]= False

Vector Frobenius y N(S) en distintos órdenes

In[362]:=
```
Print["Empleando orden Lexicográfico:"]
Print["v. Frob -> ",FrobeniusLexi[psimExample[[1,2]]]]
psimnLexi = NFrobeniusLexi[psimExample[[1,1]],psimExample[[1,2]]]

Print["Empleando orden Lexicográfico graduado:"]
Print["v. Frob -> ",FrobeniusDLexi[psimExample[[1,2]]]]
psimnDLexi = NFrobeniusDLexi[psimExample[[1,1]],psimExample[[1,

Print["Empleando orden Lexicográfico graduado inverso:"]
Print["v. Frob -> ",FrobeniusDRLexi[psimExample[[1,2]]]]
psimnDRLexi = NFrobeniusDRLexi[psimExample[[1,1]],psimExample[[

psimVFrob={22,2}

psimnLexi == psimnDLexi
psimnDLexi == psimnDRLexi
psimnLexi == psimnDRLexi
```

```
Empleando orden Lexicográfico:
v. Frob -> {22, 2}
```

Out[364]= {{3, 1}, {4, 1}, {5, 1}, {5, 2}, {6, 1}, {6, 2}, {7, 2}, {7, 3}, {8, 1}, {8, 2}, {8, 3}, {9, 2}, {9, 3}, {10, 2}, {10, 3}, {10, 4}, {11, 2}, {11, 3}, {11, 4}, {12, 1}, {12, 2}, {12, 3}, {12, 4}, {12, 5}, {13, 1}, {13, 2}, {13, 3}, {13, 4}, {13, 5}, {14, 2}, {14, 3}, {14, 4}, {14, 5}, {14, 6}, {15, 1}, {15, 2}, {15, 3}, {15, 4}, {15, 5}, {15, 6}, {16, 2}, {16, 3}, {16, 4}, {16, 5}, {16, 6}, {17, 2}, {17, 3}, {17, 4}, {17, 5}, {17, 6}, {17, 7}, {18, 2}, {18, 3}, {18, 4}, {18, 5}, {18, 6}, {18, 7}, {19, 2}, {19, 3}, {19, 4}, {19, 5}, {19, 6}, {19, 7}, {19, 8}, {20, 2}, {20, 3}, {20, 4}, {20, 5}, {20, 6}, {20, 7}, {20, 8}, {21, 2}, {21, 3}, {21, 4}, {21, 5}, {21, 6}, {21, 7}, {21, 8}, {21, 9}}

Empleando orden Lexicográfico graduado:

v. Frob -> {22, 2}

Out[367]= {{3, 1}, {4, 1}, {5, 1}, {5, 2}, {6, 1}, {6, 2}, {7, 2}, {7, 3}, {8, 1}, {8, 2}, {8, 3}, {9, 2}, {9, 3}, {10, 2}, {10, 3}, {10, 4}, {11, 2}, {11, 3}, {11, 4}, {12, 1}, {12, 2}, {12, 3}, {12, 4}, {12, 5}, {13, 1}, {13, 2}, {13, 3}, {13, 4}, {13, 5}, {14, 2}, {14, 3}, {14, 4}, {14, 5}, {14, 6}, {15, 1}, {15, 2}, {15, 3}, {15, 4}, {15, 5}, {15, 6}, {16, 2}, {16, 3}, {16, 4}, {16, 5}, {16, 6}, {17, 2}, {17, 3}, {17, 4}, {17, 5}, {17, 6}, {17, 7}, {18, 2}, {18, 3}, {18, 4}, {18, 5}, {18, 6}, {19, 2}, {19, 3}, {19, 4}, {19, 5}, {20, 2}, {20, 3}, {20, 4}, {21, 2}, {21, 3}}

Empleando orden Lexicográfico graduado inverso:

v. Frob -> {22, 2}

```
Out[370]= {{3, 1}, {4, 1}, {5, 1}, {5, 2}, {6, 1}, {6, 2}, {7, 2}, {7, 3},
  {8, 1}, {8, 2}, {8, 3}, {9, 2}, {9, 3}, {10, 2}, {10, 3},
  {10, 4}, {11, 2}, {11, 3}, {11, 4}, {12, 1}, {12, 2}, {12, 3},
  {12, 4}, {12, 5}, {13, 1}, {13, 2}, {13, 3}, {13, 4},
  {13, 5}, {14, 2}, {14, 3}, {14, 4}, {14, 5}, {14, 6},
  {15, 1}, {15, 2}, {15, 3}, {15, 4}, {15, 5}, {15, 6},
  {16, 2}, {16, 3}, {16, 4}, {16, 5}, {16, 6}, {17, 2},
  {17, 3}, {17, 4}, {17, 5}, {17, 6}, {17, 7}, {18, 2},
  {18, 3}, {18, 4}, {18, 5}, {18, 6}, {19, 2}, {19, 3},
  {19, 4}, {19, 5}, {20, 2}, {20, 3}, {20, 4}, {21, 2}, {21, 3}}

Out[371]= {22, 2}

Out[372]= False

Out[373]= True

Out[374]= False
```

Ap(S,b) para distintos órdenes

In[375]:=
```
(*Punto y directorio del archivo*)
fileDirectory=NotebookDirectory[];

psimApery=GetApery[psimExample[[1,1]][[4]],psimExample[[1,1]],psimE

psimSubstractApery = Table[psimApery[[i]] - psimExample[[1,1]][[4]]

MemberQ[psimSubstractApery,psimVFrob]
MemberQ[psimSubstractApery,psimVFrob/2]
```

Out[376]= {{12, 3}, {14, 3}, {15, 3}, {16, 3}, {19, 3}, {27, 4}}

Out[377]= {{7, 1}, {9, 1}, {10, 1}, {11, 1}, {14, 1}, {22, 2}}

Out[378]= True

Out[379]= True

I(F(S)) y $\mathcal{F}$(S) para distintos órdenes

In[380]:=
```
fileDirectory=NotebookDirectory[];

psimIFS=GetIsetNoEq[{22,2},psimExample[[1,1]],psimExample[[1,2]],i
Length[psimIFS]

psim𝓕S = Length[psimIFS] + psimgS
```

Out[381]= {{0, 0}, {8, 1}, {12, 1}, {13, 1}, {15, 1}}

Out[382]= 5

Out[383]= 11

In[384]:=
```
Length[psimIFS] ≤ psimgS
psimgS == Length[psimIFS] + 1
```

Out[384]= True

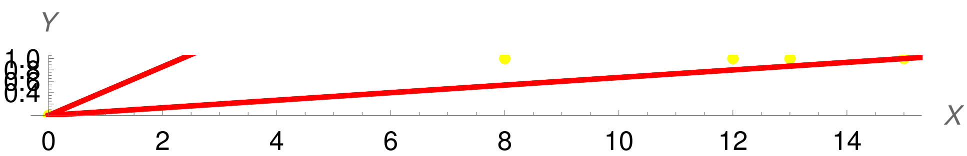
Out[385]= True

In[386]:=
```
2*psimgS == 1 + psimℱS
```

Out[386]= True

In[387]:=
```
(* Vectores de los rayos extremales para sacar ecuaciones de
semig=psimExample[[1,1]];
T1=ConvexHullMesh[Join @@ {{{0,0}},semig}];
LineasT1=MeshPrimitives[T1,1];
T1L=Select[Level[LineasT1,{2}],MemberQ[#,{0,0}]&];
vectorsinrays=Select[Flatten[T1L,1],# ≠{0,0}&]

(*Representación gráfica Apery*)
vectorsinrays=ParallelTable[{{0,0},20*vectorsinrays[[i]]},{i,Leng

Print[
Show[
{
ListPlot[psimIFS,PlotStyle→Yellow],
Graphics[{Red,Thick,Line[vectorsinrays]}]
},
AxesOrigin→{0,0},AspectRatio→Automatic,Show→True,AxesLabel→
]
];
```

Out[391]= {{15, 1}, {7, 3}}



Capítulo 4

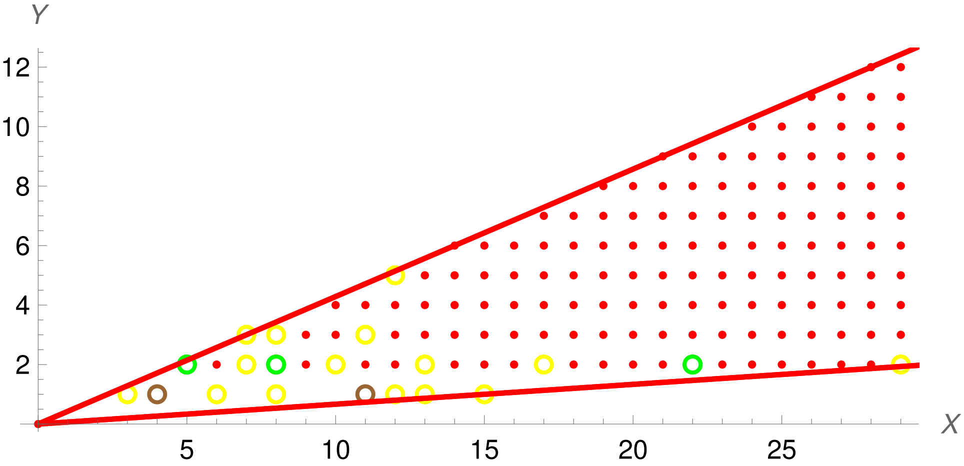
Ejecución Ejemplos 4.1 y 4.2 en N^2

Ejemplo base

In[394]:=
```
pruebaConPseudos={{{{3,1},{6,1},{7,2},{7,3},{8,1},{8,3},{10,2},{12,

Plot2DSemigAll[pruebaConPseudos⟦1,1⟧,pruebaConPseudos⟦1,2⟧];

Print["Huecos -> ",Length[pruebaConPseudos⟦1,2⟧]]
p = GetPseudoFrobenius[pruebaConPseudos⟦1,1⟧,pruebaConPseudo
Print["Ps -> ",Length[ GetPseudoFrobenius[pruebaConPseudos⟦1,
GetEspGaps[p,pruebaConPseudos⟦1,2⟧]
Print["Sg -> ",Length[ GetEspGaps[p,pruebaConPseudos⟦1,2⟧]]]
```

Out[394]=
```
{{{{3, 1}, {6, 1}, {7, 2}, {7, 3}, {8, 1},
    {8, 3}, {10, 2}, {12, 1}, {12, 5}, {13, 1},
    {13, 2}, {15, 1}, {17, 2}, {29, 2}, {11, 3}},
  {{4, 1}, {5, 1}, {5, 2}, {7, 1}, {8, 2}, {9, 1}, {10, 1},
    {11, 1}, {14, 1}, {22, 2}}}, {{-1, 15}, {3, -7}}}
```

Vectores de los rayos extremales= {{15, 1}, {7, 3}}



Huecos -> 10

Ps -> 5

Out[399]= {{5, 2}, {8, 2}, {22, 2}}

Sg -> 3

GetIrreducibles

In[401]:=
```
Timing [testirr = GetIrreducibles[pruebaConPseudos[[1,1]],prueb
Length[testirr]
```

Out getsinfo -> {{1, 3, 5, 8, 10}, {3, 5, 10}, 3}

Nº de ♯C 3

Nº de ♯C 8

Nº de ♯C 23

Nº de ♯C 47

Nº de ♯C 71

Nº de ♯C 76

Nº de ♯C 56

Nº de ♯C 28

Out[402]= 10

GetMinimalsIrreducibles

In[403]:=
```
Timing [testminirr = GetMinimalsIrreducibles[pruebaConPseudo
Length[testminirr]
Length[testirr]
```

```
Out getsinfo -> {{1, 3, 5, 8, 10}, {3, 5, 10}, 3}
Nº de ♯C 3
Nº de ♯C 8
Nº de ♯C 23
Nº de ♯C 47
Nº de ♯C 71
Nº de ♯C 76
Nº de ♯C 56
Nº de ♯C 28
```

Out[403]=
```
{0.812942,
 {{{{{3, 1}, {4, 1}, {5, 1}, {5, 2}, {6, 1}, {7, 3}, {8, 1},
      {12, 1}, {13, 1}, {15, 1}, {29, 2}},
     {{7, 1}, {9, 1}, {10, 1}, {11, 1}, {14, 1}, {22, 2}}}, 0},
  {{{{3, 1}, {5, 2}, {6, 1}, {7, 1}, {7, 2}, {7, 3}, {8, 1},
      {9, 1}, {10, 1}, {11, 1}, {12, 1}, {13, 1},
      {14, 1}, {15, 1}}, {{4, 1}, {5, 1}, {8, 2}}}, 0},
  {{{{3, 1}, {4, 1}, {5, 1}, {6, 1}, {7, 1}, {7, 3},
      {8, 1}, {8, 3}, {9, 1}, {10, 1}, {11, 1}, {12, 1},
      {12, 5}, {13, 1}, {14, 1}, {15, 1}}, {{5, 2}}}, 1}}}
```

Out[404]= 3

Out[405]= 10

Ejemplo 4.1 en N^2

Descomposicion simple

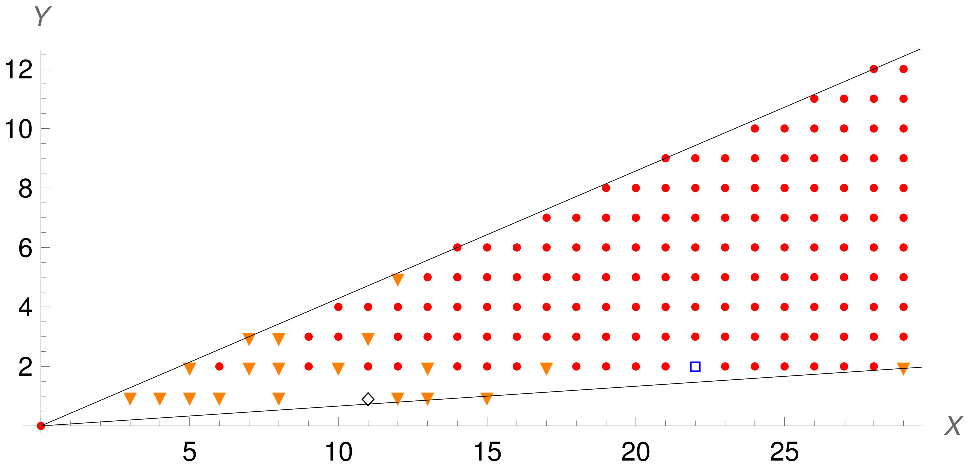
In[406]:=
```
pruebaConPseudos={{{3,1},{6,1},{7,2},{7,3},{8,1},{8,3},{10,2},{12,1
```

In[407]:=
```
testirr={{{{3,1},{4,1},{5,1},{5,2},{6,1},{7,2},{7,3},{8,1},{8,2},{8,3
```

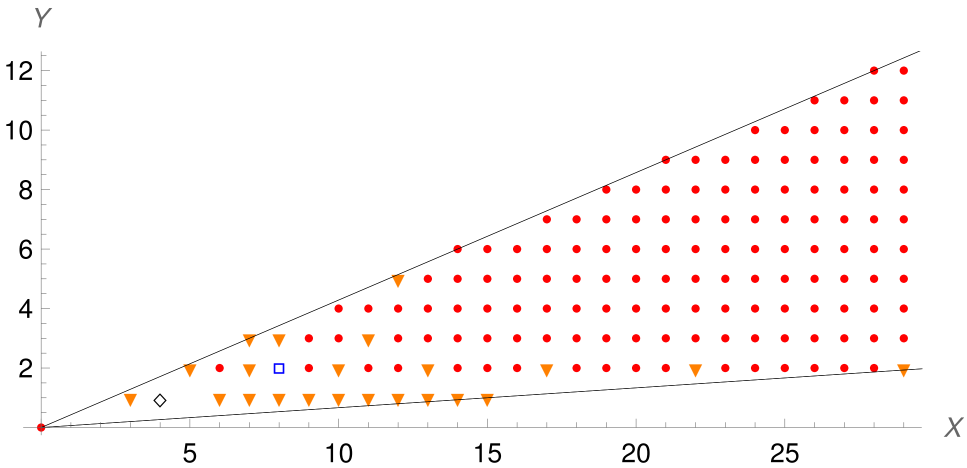
Graficando irreducibles

In[408]:=
```
Table[Plot2DSemigAllBW[testirr[[i,1]],testirr[[i,2]]],{i,1,Length[
```

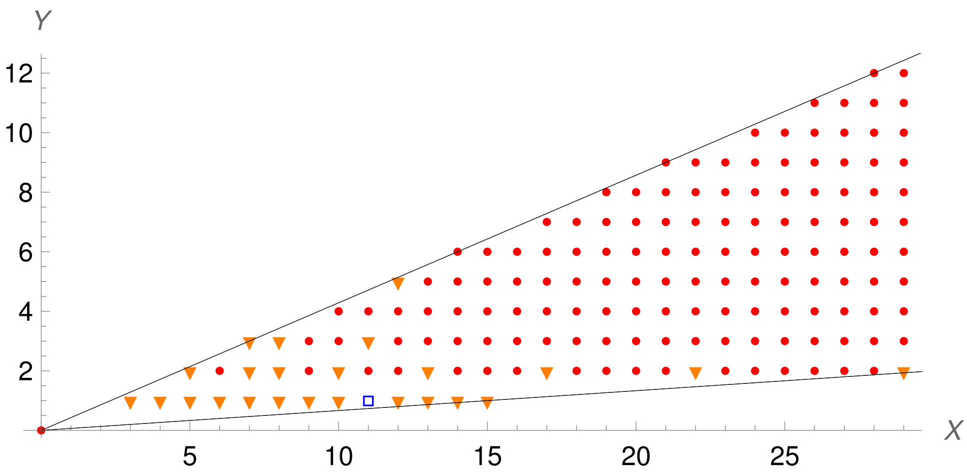
Vectores de los rayos extremales= {{15, 1}, {7, 3}}



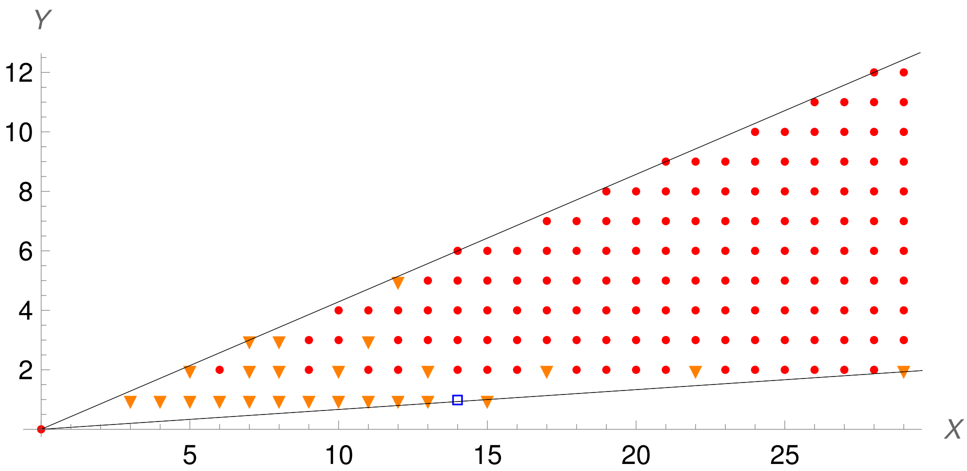
Vectores de los rayos extremales= {{15, 1}, {7, 3}}



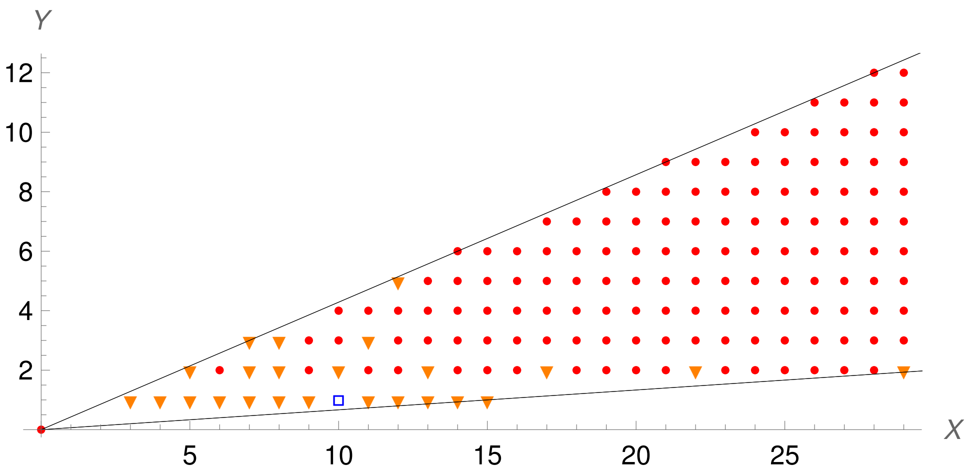
Vectores de los rayos extremales= {{15, 1}, {7, 3}}



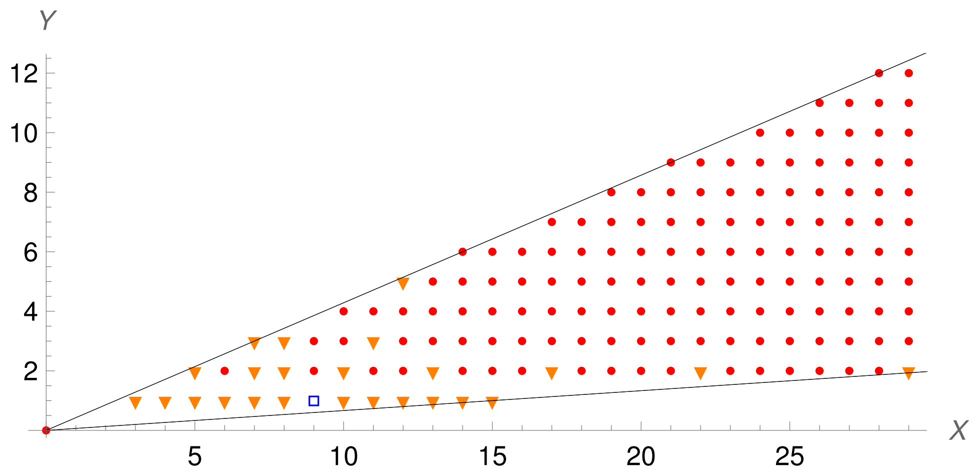
Vectores de los rayos extremales= {{15, 1}, {7, 3}}



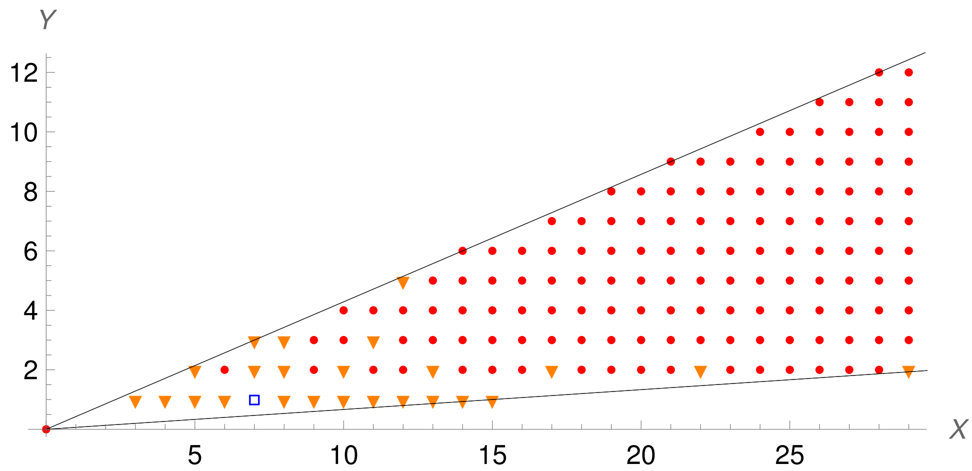
Vectores de los rayos extremales= {{15, 1}, {7, 3}}



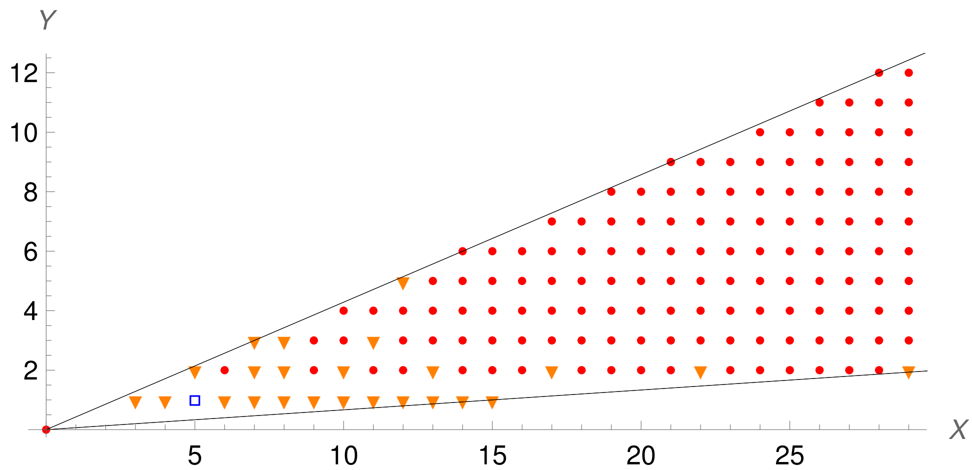
Vectores de los rayos extremales= {{15, 1}, {7, 3}}



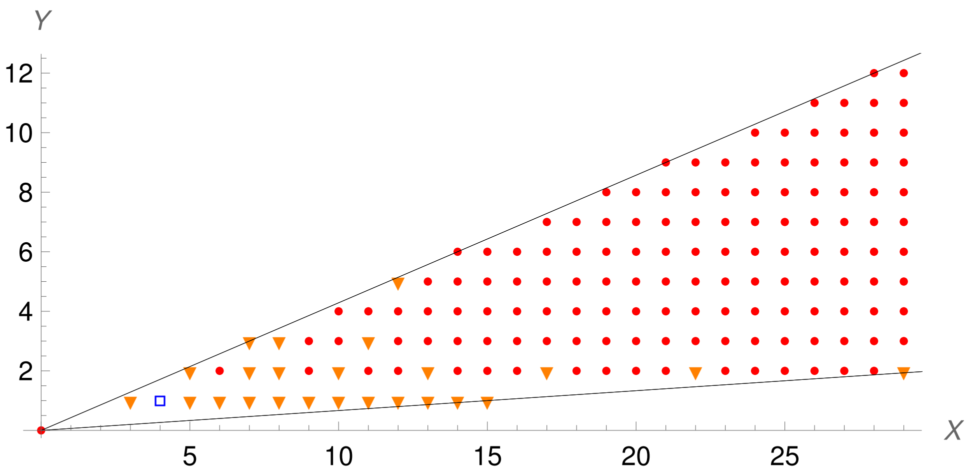
Vectores de los rayos extremales= {{15, 1}, {7, 3}}



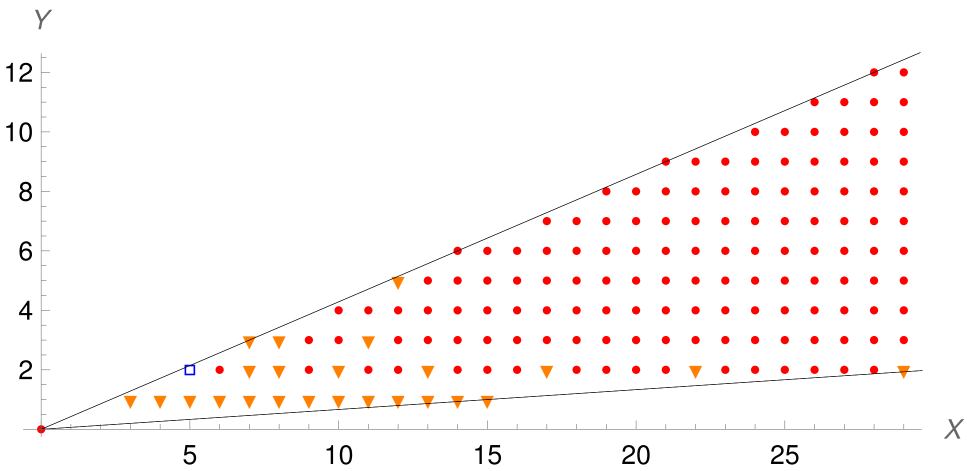
Vectores de los rayos extremales= {{15, 1}, {7, 3}}



Vectores de los rayos extremales= {{15, 1}, {7, 3}}




```
Vectores de los rayos extremales= {{15, 1}, {7, 3}}
```



C(S_i) ∀ i∈[t]

In[409]:=
```
Table[IrreducibleCSi[pruebaConPseudos, testirr[[i,2]]],{i,Lengt
```

Out[409]=
```
{{{22, 2}}, {{8, 2}}, {}, {}, {}, {}, {}, {}, {}, {{5, 2}}}
```

Ejemplo 4.2 en N^2

Descomposicion simple

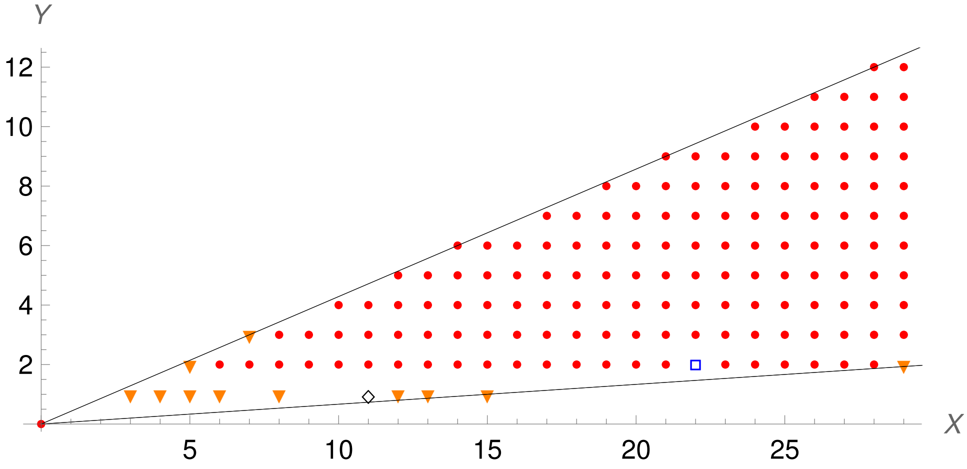
In[410]:=
```
pruebaConPseudos={{{3,1},{6,1},{7,2},{7,3},{8,1},{8,3},{10,2},{12,1
```

```
In[411]:= testminirr={{{{{3,1},{4,1},{5,1},{5,2},{6,1},{7,3},{8,1},{12,1},{13,1
          testminirr=Table[{testminirr[[i,1,1]],testminirr[[i,1,2]],testmin

Out[411]= {{{{{3, 1}, {4, 1}, {5, 1}, {5, 2}, {6, 1},
               {7, 3}, {8, 1}, {12, 1}, {13, 1}, {15, 1}, {29, 2}},
             {{7, 1}, {9, 1}, {10, 1}, {11, 1}, {14, 1}, {22, 2}}}, 0},
           {{{{3, 1}, {5, 2}, {6, 1}, {7, 1}, {7, 2}, {7, 3}, {8, 1}, {9, 1},
               {10, 1}, {11, 1}, {12, 1}, {13, 1}, {14, 1}, {15, 1}},
             {{4, 1}, {5, 1}, {8, 2}}}, 0},
           {{{{3, 1}, {4, 1}, {5, 1}, {6, 1}, {7, 1}, {7, 3},
               {8, 1}, {8, 3}, {9, 1}, {10, 1}, {11, 1}, {12, 1},
               {12, 5}, {13, 1}, {14, 1}, {15, 1}}, {{5, 2}}}, 1}}

Out[412]= {{{{3, 1}, {4, 1}, {5, 1}, {5, 2}, {6, 1},
             {7, 3}, {8, 1}, {12, 1}, {13, 1}, {15, 1}, {29, 2}},
            {{7, 1}, {9, 1}, {10, 1}, {11, 1}, {14, 1}, {22, 2}}, 0},
           {{{3, 1}, {5, 2}, {6, 1}, {7, 1}, {7, 2}, {7, 3}, {8, 1}, {9, 1},
             {10, 1}, {11, 1}, {12, 1}, {13, 1}, {14, 1}, {15, 1}},
            {{4, 1}, {5, 1}, {8, 2}}, 0},
           {{{3, 1}, {4, 1}, {5, 1}, {6, 1}, {7, 1}, {7, 3},
             {8, 1}, {8, 3}, {9, 1}, {10, 1}, {11, 1}, {12, 1},
             {12, 5}, {13, 1}, {14, 1}, {15, 1}}, {{5, 2}}, 1}}
```

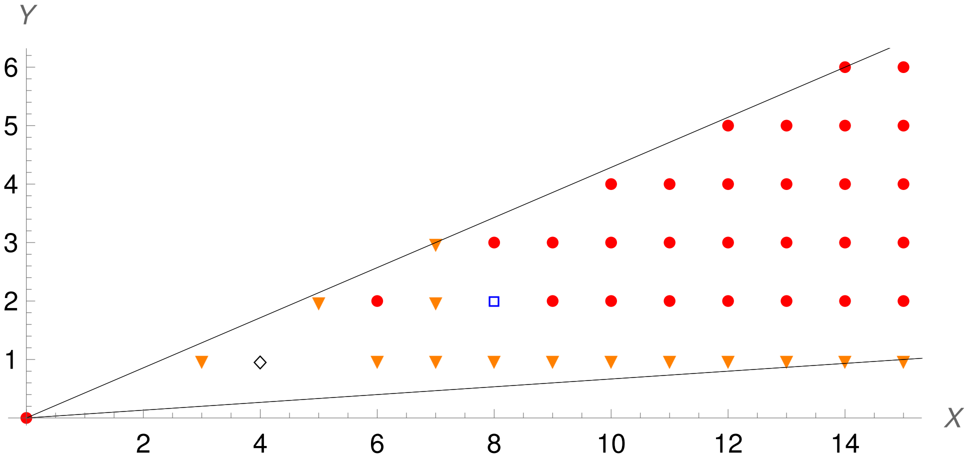
Graficando irreducibles

```
In[413]:= Table[Plot2DSemigAllBW[testminirr[[i,1]],testminirr[[i,2]]],{i,1,L
```

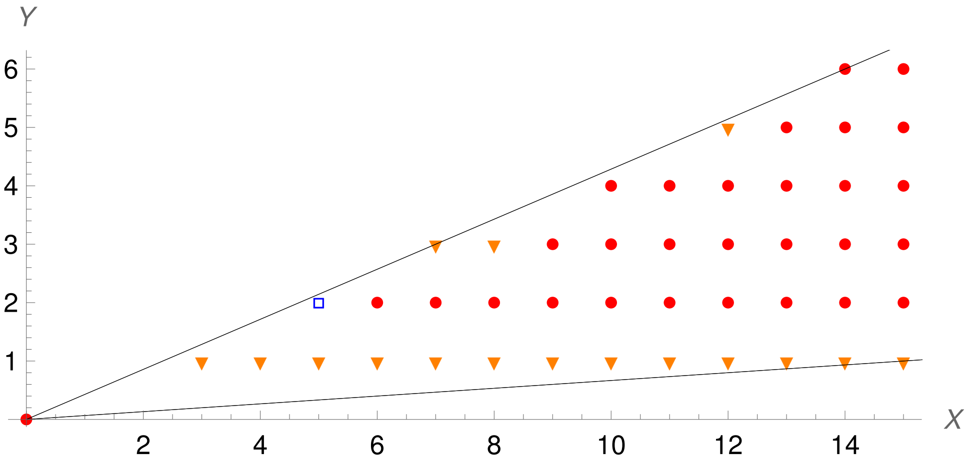
Vectores de los rayos extremales= {{15, 1}, {7, 3}}



Vectores de los rayos extremales= {{15, 1}, {7, 3}}



Vectores de los rayos extremales= {{15, 1}, {7, 3}}



$C(S_i)$ ∀ i∈[t]

In[414]:=
```
Table[IrreducibleCSi[pruebaConPseudos, testminirr[[i,2]]],{i,Le
espGaps = GetEspGaps[GetPseudoFrobenius[pruebaConPseudos[[1]],p
```

Out[414]= {{{22, 2}}, {{8, 2}}, {{5, 2}}}

Out[415]= {{5, 2}, {8, 2}, {22, 2}}

Capítulo 5

Ejemplo 5.1-3 en N^2

In[416]:=
```
fileDirectory=NotebookDirectory[];
```

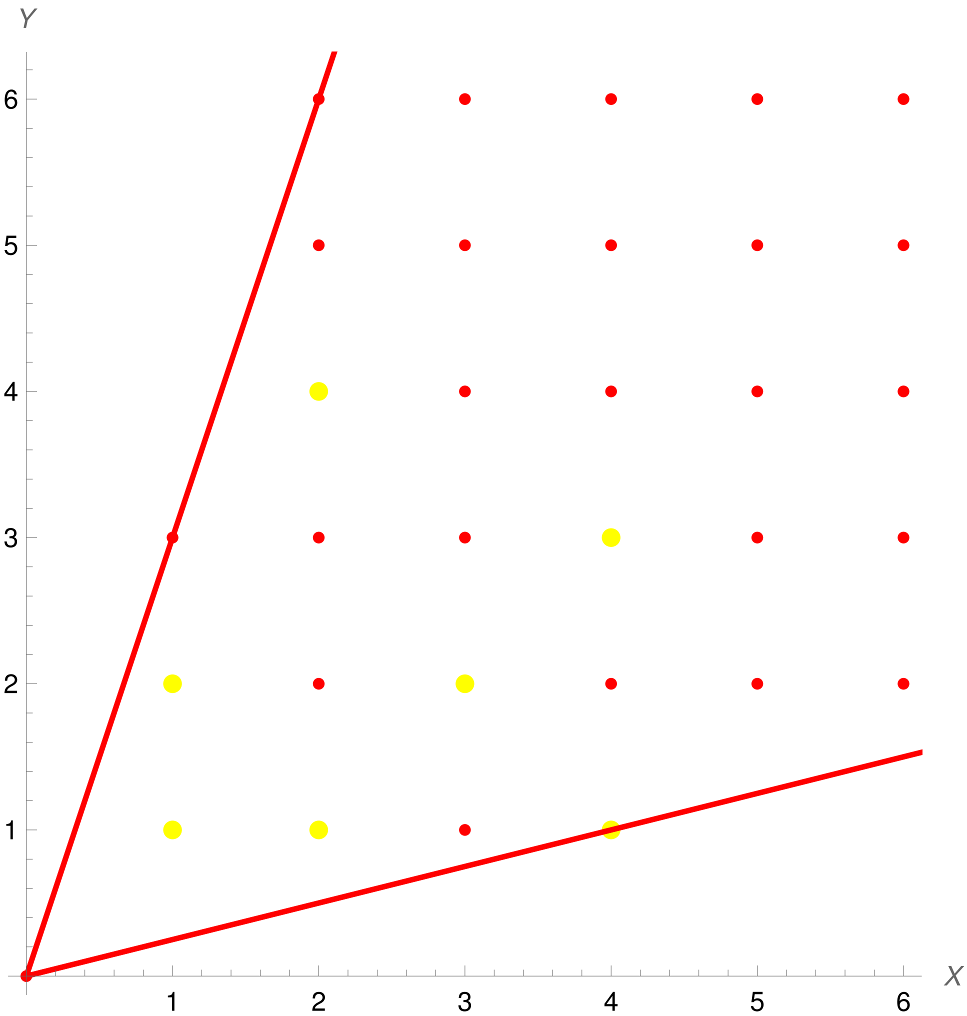
X conjunto de huecos de un C-semigrupo

In[417]:=
```
coneVectors = {{3,9},{4,1}};
geneqCone = {{{1,1},{1,2},{1,3},{2,1},{3,1},{4,1}},{{-1,4},{3,-1}}};
cminusEq = ConeGenSupHyp[coneVectors,fileDirectory,operating
```

In[420]:=
```
cminusX11={{{1,3},{2,2},{2,3},{2,5},{3,1},{3,3},{3,4},{4,2},{5,2},{6,
```

In[421]:=
```
Plot2DConeDotsRegSize[cminusX11[[2]],coneVectors,3/4]
```

{{{0, 0}, {30, 90}}, {{0, 0}, {40, 10}}}



In[422]:= IsSetMinusCS[cminusX11⟦2⟧,coneVectors,fileDirectory,False,Fa

Out[422]= True

In[423]:= GenerateDX[cminusX11⟦2⟧,cminusEq,False]

Out[423]= {{1, 1}, {1, 2}, {2, 1}, {2, 4}, {3, 2}, {4, 1}, {4, 3}}

In[424]:= SetMinusXFewer[cminusX11⟦2⟧,cminusEq,fileDirectory,True]

```
Ordered
maxCoords -> {{4, 3}, {2, 4}}
elemtMiddle -> {}
auxDesired All->
 {{0, 0}, {0, 1}, {0, 2}, {0, 3}, {0, 4}, {1, 0}, {1, 1}, {1, 2}, {1, 3},
  {1, 4}, {2, 0}, {2, 1}, {2, 2}, {2, 3}, {2, 4}, {3, 0}, {3, 1},
  {3, 2}, {3, 3}, {3, 4}, {4, 0}, {4, 1}, {4, 2}, {4, 3}, {4, 4}}
auxDesired Complement->
 {{0, 0}, {0, 1}, {0, 2}, {0, 3}, {0, 4}, {1, 0}, {1, 3}, {1, 4}, {2, 0},
  {2, 2}, {2, 3}, {3, 0}, {3, 1}, {3, 3}, {3, 4}, {4, 0}, {4, 2}, {4, 4}}
auxMiddle creation -> {{2, 3}, {3, 3}, {3, 4}, {4, 4}}
auxMiddle \ auxMiddle ->
 {{0, 0}, {0, 1}, {0, 2}, {0, 3}, {0, 4}, {1, 0}, {1, 3},
  {1, 4}, {2, 0}, {2, 2}, {3, 0}, {3, 1}, {4, 0}, {4, 2}}
auxDesired after middles->
 {{0, 0}, {0, 1}, {0, 2}, {0, 3}, {0, 4}, {1, 0}, {1, 3},
  {1, 4}, {2, 0}, {2, 2}, {3, 0}, {3, 1}, {4, 0}, {4, 2}}
```

Out[424]= {{0, 0}, {1, 3}, {2, 2}, {3, 1}, {4, 2}}

In[425]:=
```
setminuxtimes11=SetMinusXTimesCX[cminusX11⟦2⟧,cminusEq,fileD
```

```
setDesired-> {{0, 0}, {1, 3}, {2, 2}, {3, 1}, {4, 2}}

setX[[k]] -> {1, 1}

setDesired-> {{{1, 1}, {0, 0}}}

setX[[k]] -> {1, 2}

setDesired-> {{{1, 1}, {0, 0}}, {{1, 2}, {0, 0}}}

setX[[k]] -> {2, 1}

setDesired-> {{{1, 1}, {0, 0}}, {{1, 2}, {0, 0}}, {{2, 1}, {0, 0}}}

setX[[k]] -> {2, 4}

setDesired-> {{{1, 1}, {0, 0}}, {{1, 2}, {0, 0}},
  {{2, 1}, {0, 0}}, {{2, 4}, {0, 0}}, {{2, 4}, {1, 3}}}

setX[[k]] -> {3, 2}

setDesired-> {{{1, 1}, {0, 0}}, {{1, 2}, {0, 0}},
  {{2, 1}, {0, 0}}, {{2, 4}, {0, 0}}, {{2, 4}, {1, 3}}, {{3, 2}, {0, 0}}}

setX[[k]] -> {4, 1}

setDesired-> {{{1, 1}, {0, 0}}, {{1, 2}, {0, 0}}, {{2, 1}, {0, 0}},
  {{2, 4}, {0, 0}}, {{2, 4}, {1, 3}}, {{3, 2}, {0, 0}}, {{4, 1}, {0, 0}}}

setX[[k]] -> {4, 3}

setDesired-> {{{1, 1}, {0, 0}}, {{1, 2}, {0, 0}},
  {{2, 1}, {0, 0}}, {{2, 4}, {0, 0}}, {{2, 4}, {1, 3}}, {{3, 2}, {0, 0}},
  {{4, 1}, {0, 0}}, {{4, 3}, {0, 0}}, {{4, 3}, {2, 2}}, {{4, 3}, {3, 1}}}

Out[425]= {{{1, 1}, {0, 0}}, {{1, 2}, {0, 0}},
  {{2, 1}, {0, 0}}, {{2, 4}, {0, 0}},
  {{2, 4}, {1, 3}}, {{3, 2}, {0, 0}}, {{4, 1}, {0, 0}},
  {{4, 3}, {0, 0}}, {{4, 3}, {2, 2}}, {{4, 3}, {3, 1}}}

In[426]:= diffsetminus11=Union[ParallelTable[x⟦1⟧-x⟦2⟧,{x,setminuxtimes1
          SubsetQ[cminusX11⟦2⟧,diffsetminus11]

Out[426]= {{1, 1}, {1, 2}, {2, 1}, {2, 4}, {3, 2}, {4, 1}, {4, 3}}

Out[427]= True
```

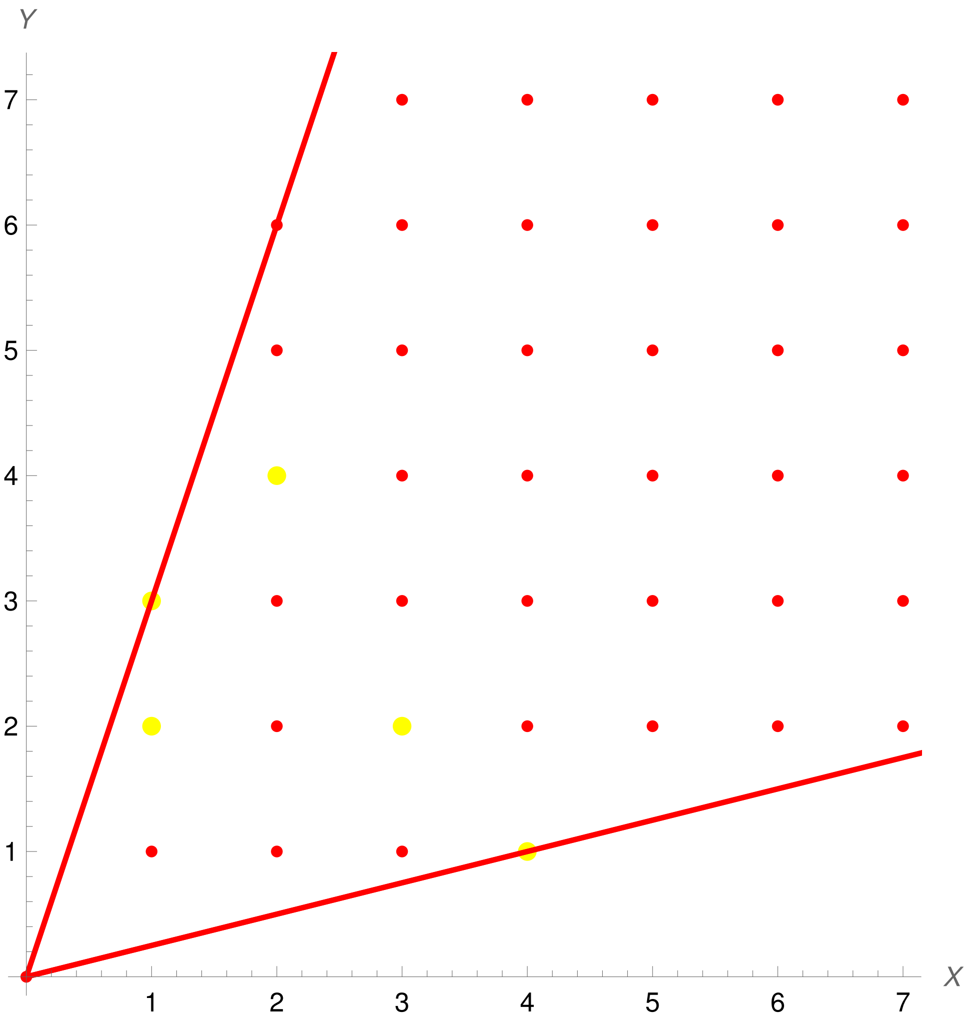
X con condición X ≠ D(X)

In[428]:=
```
coneVectors = {{3,9},{4,1}};
geneqCone = {{{1,1},{1,2},{1,3},{2,1},{3,1},{4,1}},{{-1,4},{3,-1}}};
cminusEq = ConeGenSupHyp[coneVectors,fileDirectory,operatin
```

In[431]:=
```
cminusX12 = {{2,4},{4,10},{3,2},{1,2},{1,3},{4,1}};
```

In[432]:=
```
Plot2DConeDotsRegSize[cminusX12,coneVectors,3/4]
```

```
{{{0, 0}, {30, 90}}, {{0, 0}, {40, 10}}}
```



In[433]:=
```
IsSetMinusCS[cminusX12,coneVectors,fileDirectory,True,True]
```

X not void

A[[k]]{2, 4}

¬InCone[x,Eq]->False

A[[k]]{4, 10}

¬InCone[x,Eq]->False

A[[k]]{3, 2}

¬InCone[x,Eq]->False

A[[k]]{1, 2}

¬InCone[x,Eq]->False

A[[k]]{1, 3}

¬InCone[x,Eq]->False

A[[k]]{4, 1}

¬InCone[x,Eq]->False

A[[k]]{2, 4}

gcd->{1, 2}

Lenght[gcd]->2

A[[k]]/gcd[[i]]->{2, 4}

¬MemberQ[A,A[[k]]/gcd[[i]]]->False

A[[k]]/gcd[[i]]->{1, 2}

¬MemberQ[A,A[[k]]/gcd[[i]]]->False

A[[k]]{4, 10}

gcd->{1, 2}

Lenght[gcd]->2

A[[k]]/gcd[[i]]->{4, 10}

¬MemberQ[A,A[[k]]/gcd[[i]]]->False

A[[k]]/gcd[[i]]->{2, 5}

¬MemberQ[A,A[[k]]/gcd[[i]]]->True

X≠D(X) - This element causes this->{4, 10}

X is not D(X) or X not subset Cone

Out[433]= False

In[434]:=
```
GenerateDX[cminusX12,cminusEq,False]
```

Element added->{2, 5}True

Out[434]= {{2, 4}, {4, 10}, {3, 2}, {1, 2}, {1, 3}, {4, 1}, {2, 5}}

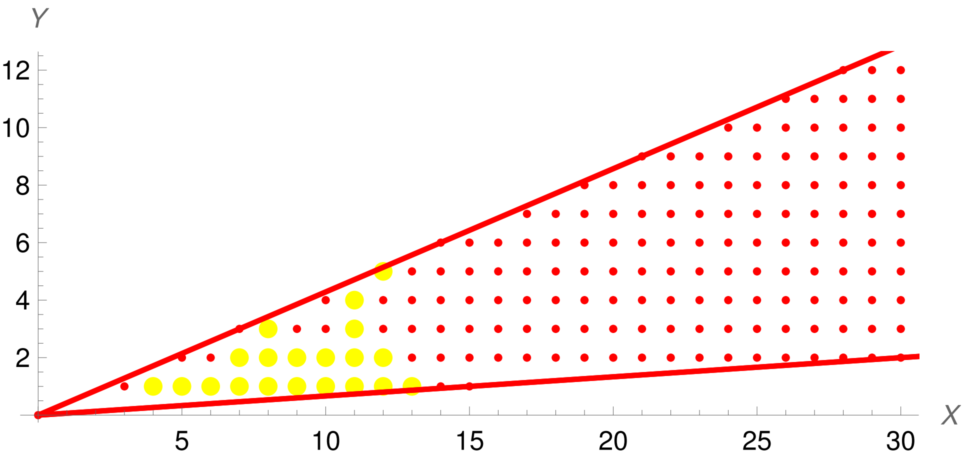
X con condición x-a ∉ S

In[435]:=
```
coneVectors = {{15,1},{7,3}};
geneqCone = {{3,1},{4,1},{5,1},{5,2},{6,1},{7,1},{7,3},{8,1},{9,1},{
cminusEq = {{-1,15},{3,-7}};
```

In[438]:=
```
cminusX13 = {{4,1},{5,1},{6,1},{7,1},{7,2},{8,1},{8,2},{8,3},{9,1},{
```

In[439]:=
```
Plot2DConeDotsBigValues[cminusX13,coneVectors]
```

{{{0, 0}, {300, 20}}, {{0, 0}, {140, 60}}}



In[440]:=
```
IsSetMinusCS[cminusX13,coneVectors,fileDirectory,True,False]
```

X not void

X subset Cone & X = D(X)

X ordered

x-a not in X ->{12, 5}-{5, 2}

Out[440]= False

In[441]:=
```
GenerateDX[cminusX13,cminusEq,False]
```

Out[441]=
```
{{4, 1}, {5, 1}, {6, 1}, {7, 1}, {7, 2}, {8, 1}, {8, 2},
 {8, 3}, {9, 1}, {9, 2}, {10, 1}, {10, 2}, {11, 1}, {11, 2},
 {11, 3}, {11, 4}, {12, 1}, {12, 2}, {12, 5}, {13, 1}}
```

In[442]:=
```
setminuxtimes13=SetMinusXTimesCX[cminusX13,cminusEq,fileDir
```

```
setDesired->
 {{0, 0}, {3, 1}, {5, 2}, {6, 2}, {7, 3}, {9, 3}, {10, 3}, {10, 4}}

setX[[k]] -> {4, 1}

setDesired-> {{{4, 1}, {0, 0}}}

setX[[k]] -> {5, 1}

setDesired-> {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}}

setX[[k]] -> {6, 1}

setDesired-> {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}}}

setX[[k]] -> {7, 1}

setDesired->
 {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}}, {{7, 1}, {0, 0}}}

setX[[k]] -> {7, 2}

setDesired-> {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}},
  {{6, 1}, {0, 0}}, {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}}}

setX[[k]] -> {8, 1}

setDesired-> {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}},
  {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}}, {{8, 1}, {0, 0}}}

setX[[k]] -> {8, 2}

setDesired-> {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}},
  {{6, 1}, {0, 0}}, {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}},
  {{7, 2}, {3, 1}}, {{8, 1}, {0, 0}}, {{8, 2}, {0, 0}}, {{8, 2}, {3, 1}}}

setX[[k]] -> {8, 3}
```

```
setDesired-> {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}},
  {{6, 1}, {0, 0}}, {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}},
  {{7, 2}, {3, 1}}, {{8, 1}, {0, 0}}, {{8, 2}, {0, 0}}, {{8, 2}, {3, 1}},
  {{8, 3}, {0, 0}}, {{8, 3}, {3, 1}}, {{8, 3}, {5, 2}}, {{8, 3}, {6, 2}}}
setX[[k]] -> {9, 1}
setDesired-> {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}},
  {{6, 1}, {0, 0}}, {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}},
  {{8, 1}, {0, 0}}, {{8, 2}, {0, 0}}, {{8, 2}, {3, 1}}, {{8, 3}, {0, 0}},
  {{8, 3}, {3, 1}}, {{8, 3}, {5, 2}}, {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}}}
setX[[k]] -> {9, 2}
setDesired->
 {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}}, {{7, 1}, {0, 0}},
  {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}}, {{8, 1}, {0, 0}}, {{8, 2}, {0, 0}},
  {{8, 2}, {3, 1}}, {{8, 3}, {0, 0}}, {{8, 3}, {3, 1}}, {{8, 3}, {5, 2}},
  {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}}, {{9, 2}, {0, 0}}, {{9, 2}, {3, 1}}}
setX[[k]] -> {10, 1}
setDesired-> {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}},
  {{6, 1}, {0, 0}}, {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}},
  {{8, 1}, {0, 0}}, {{8, 2}, {0, 0}}, {{8, 2}, {3, 1}}, {{8, 3}, {0, 0}},
  {{8, 3}, {3, 1}}, {{8, 3}, {5, 2}}, {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}},
  {{9, 2}, {0, 0}}, {{9, 2}, {3, 1}}, {{10, 1}, {0, 0}}}
setX[[k]] -> {10, 2}
setDesired-> {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}},
  {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}}, {{8, 1}, {0, 0}},
  {{8, 2}, {0, 0}}, {{8, 2}, {3, 1}}, {{8, 3}, {0, 0}}, {{8, 3}, {3, 1}},
  {{8, 3}, {5, 2}}, {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}}, {{9, 2}, {0, 0}},
  {{9, 2}, {3, 1}}, {{10, 1}, {0, 0}}, {{10, 2}, {0, 0}}, {{10, 2}, {3, 1}}}
setX[[k]] -> {11, 1}
setDesired->
 {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}}, {{7, 1}, {0, 0}},
  {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}}, {{8, 1}, {0, 0}}, {{8, 2}, {0, 0}},
  {{8, 2}, {3, 1}}, {{8, 3}, {0, 0}}, {{8, 3}, {3, 1}}, {{8, 3}, {5, 2}},
  {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}}, {{9, 2}, {0, 0}}, {{9, 2}, {3, 1}},
  {{10, 1}, {0, 0}}, {{10, 2}, {0, 0}}, {{10, 2}, {3, 1}}, {{11, 1}, {0, 0}}}
```

```
setX[[k]] -> {11, 2}

setDesired->
 {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}}, {{7, 1}, {0, 0}},
  {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}}, {{8, 1}, {0, 0}}, {{8, 2}, {0, 0}},
  {{8, 2}, {3, 1}}, {{8, 3}, {0, 0}}, {{8, 3}, {3, 1}}, {{8, 3}, {5, 2}},
  {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}}, {{9, 2}, {0, 0}}, {{9, 2}, {3, 1}},
  {{10, 1}, {0, 0}}, {{10, 2}, {0, 0}}, {{10, 2}, {3, 1}},
  {{11, 1}, {0, 0}}, {{11, 2}, {0, 0}}, {{11, 2}, {3, 1}}}

setX[[k]] -> {11, 3}

setDesired->
 {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}}, {{7, 1}, {0, 0}},
  {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}}, {{8, 1}, {0, 0}}, {{8, 2}, {0, 0}},
  {{8, 2}, {3, 1}}, {{8, 3}, {0, 0}}, {{8, 3}, {3, 1}}, {{8, 3}, {5, 2}},
  {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}}, {{9, 2}, {0, 0}}, {{9, 2}, {3, 1}},
  {{10, 1}, {0, 0}}, {{10, 2}, {0, 0}}, {{10, 2}, {3, 1}},
  {{11, 1}, {0, 0}}, {{11, 2}, {0, 0}}, {{11, 2}, {3, 1}},
  {{11, 3}, {0, 0}}, {{11, 3}, {3, 1}}, {{11, 3}, {5, 2}}, {{11, 3}, {6, 2}}}

setX[[k]] -> {11, 4}

setDesired-> {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}},
  {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}}, {{8, 1}, {0, 0}},
  {{8, 2}, {0, 0}}, {{8, 2}, {3, 1}}, {{8, 3}, {0, 0}}, {{8, 3}, {3, 1}},
  {{8, 3}, {5, 2}}, {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}}, {{9, 2}, {0, 0}},
  {{9, 2}, {3, 1}}, {{10, 1}, {0, 0}}, {{10, 2}, {0, 0}},
  {{10, 2}, {3, 1}}, {{11, 1}, {0, 0}}, {{11, 2}, {0, 0}},
  {{11, 2}, {3, 1}}, {{11, 3}, {0, 0}}, {{11, 3}, {3, 1}},
  {{11, 3}, {5, 2}}, {{11, 3}, {6, 2}}, {{11, 4}, {0, 0}},
  {{11, 4}, {3, 1}}, {{11, 4}, {5, 2}}, {{11, 4}, {6, 2}},
  {{11, 4}, {7, 3}}, {{11, 4}, {9, 3}}, {{11, 4}, {10, 3}}}

setX[[k]] -> {12, 1}
```

```
setDesired->
 {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}}, {{7, 1}, {0, 0}},
  {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}}, {{8, 1}, {0, 0}}, {{8, 2}, {0, 0}},
  {{8, 2}, {3, 1}}, {{8, 3}, {0, 0}}, {{8, 3}, {3, 1}}, {{8, 3}, {5, 2}},
  {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}}, {{9, 2}, {0, 0}}, {{9, 2}, {3, 1}},
  {{10, 1}, {0, 0}}, {{10, 2}, {0, 0}}, {{10, 2}, {3, 1}},
  {{11, 1}, {0, 0}}, {{11, 2}, {0, 0}}, {{11, 2}, {3, 1}},
  {{11, 3}, {0, 0}}, {{11, 3}, {3, 1}}, {{11, 3}, {5, 2}},
  {{11, 3}, {6, 2}}, {{11, 4}, {0, 0}}, {{11, 4}, {3, 1}},
  {{11, 4}, {5, 2}}, {{11, 4}, {6, 2}}, {{11, 4}, {7, 3}},
  {{11, 4}, {9, 3}}, {{11, 4}, {10, 3}}, {{12, 1}, {0, 0}}}

setX[[k]] -> {12, 2}

setDesired-> {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}},
  {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}}, {{8, 1}, {0, 0}},
  {{8, 2}, {0, 0}}, {{8, 2}, {3, 1}}, {{8, 3}, {0, 0}}, {{8, 3}, {3, 1}},
  {{8, 3}, {5, 2}}, {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}}, {{9, 2}, {0, 0}},
  {{9, 2}, {3, 1}}, {{10, 1}, {0, 0}}, {{10, 2}, {0, 0}},
  {{10, 2}, {3, 1}}, {{11, 1}, {0, 0}}, {{11, 2}, {0, 0}},
  {{11, 2}, {3, 1}}, {{11, 3}, {0, 0}}, {{11, 3}, {3, 1}},
  {{11, 3}, {5, 2}}, {{11, 3}, {6, 2}}, {{11, 4}, {0, 0}},
  {{11, 4}, {3, 1}}, {{11, 4}, {5, 2}}, {{11, 4}, {6, 2}},
  {{11, 4}, {7, 3}}, {{11, 4}, {9, 3}}, {{11, 4}, {10, 3}},
  {{12, 1}, {0, 0}}, {{12, 2}, {0, 0}}, {{12, 2}, {3, 1}}}

setX[[k]] -> {12, 5}
```

```
setDesired-> {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}},
  {{6, 1}, {0, 0}}, {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}},
  {{8, 1}, {0, 0}}, {{8, 2}, {0, 0}}, {{8, 2}, {3, 1}}, {{8, 3}, {0, 0}},
  {{8, 3}, {3, 1}}, {{8, 3}, {5, 2}}, {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}},
  {{9, 2}, {0, 0}}, {{9, 2}, {3, 1}}, {{10, 1}, {0, 0}},
  {{10, 2}, {0, 0}}, {{10, 2}, {3, 1}}, {{11, 1}, {0, 0}},
  {{11, 2}, {0, 0}}, {{11, 2}, {3, 1}}, {{11, 3}, {0, 0}},
  {{11, 3}, {3, 1}}, {{11, 3}, {5, 2}}, {{11, 3}, {6, 2}},
  {{11, 4}, {0, 0}}, {{11, 4}, {3, 1}}, {{11, 4}, {5, 2}},
  {{11, 4}, {6, 2}}, {{11, 4}, {7, 3}}, {{11, 4}, {9, 3}},
  {{11, 4}, {10, 3}}, {{12, 1}, {0, 0}}, {{12, 2}, {0, 0}},
  {{12, 2}, {3, 1}}, {{12, 5}, {0, 0}}, {{12, 5}, {3, 1}},
  {{12, 5}, {5, 2}}, {{12, 5}, {6, 2}}, {{12, 5}, {7, 3}},
  {{12, 5}, {9, 3}}, {{12, 5}, {10, 3}}, {{12, 5}, {10, 4}}}

setX[[k]] -> {13, 1}

setDesired-> {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}},
  {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}}, {{8, 1}, {0, 0}},
  {{8, 2}, {0, 0}}, {{8, 2}, {3, 1}}, {{8, 3}, {0, 0}}, {{8, 3}, {3, 1}},
  {{8, 3}, {5, 2}}, {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}}, {{9, 2}, {0, 0}},
  {{9, 2}, {3, 1}}, {{10, 1}, {0, 0}}, {{10, 2}, {0, 0}},
  {{10, 2}, {3, 1}}, {{11, 1}, {0, 0}}, {{11, 2}, {0, 0}},
  {{11, 2}, {3, 1}}, {{11, 3}, {0, 0}}, {{11, 3}, {3, 1}},
  {{11, 3}, {5, 2}}, {{11, 3}, {6, 2}}, {{11, 4}, {0, 0}},
  {{11, 4}, {3, 1}}, {{11, 4}, {5, 2}}, {{11, 4}, {6, 2}},
  {{11, 4}, {7, 3}}, {{11, 4}, {9, 3}}, {{11, 4}, {10, 3}},
  {{12, 1}, {0, 0}}, {{12, 2}, {0, 0}}, {{12, 2}, {3, 1}},
  {{12, 5}, {0, 0}}, {{12, 5}, {3, 1}}, {{12, 5}, {5, 2}},
  {{12, 5}, {6, 2}}, {{12, 5}, {7, 3}}, {{12, 5}, {9, 3}},
  {{12, 5}, {10, 3}}, {{12, 5}, {10, 4}}, {{13, 1}, {0, 0}}}
```

```
Out[442]= {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}},
  {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}},
  {{8, 1}, {0, 0}}, {{8, 2}, {0, 0}}, {{8, 2}, {3, 1}},
  {{8, 3}, {0, 0}}, {{8, 3}, {3, 1}}, {{8, 3}, {5, 2}},
  {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}}, {{9, 2}, {0, 0}},
  {{9, 2}, {3, 1}}, {{10, 1}, {0, 0}}, {{10, 2}, {0, 0}},
  {{10, 2}, {3, 1}}, {{11, 1}, {0, 0}}, {{11, 2}, {0, 0}},
  {{11, 2}, {3, 1}}, {{11, 3}, {0, 0}}, {{11, 3}, {3, 1}},
  {{11, 3}, {5, 2}}, {{11, 3}, {6, 2}}, {{11, 4}, {0, 0}},
  {{11, 4}, {3, 1}}, {{11, 4}, {5, 2}}, {{11, 4}, {6, 2}},
  {{11, 4}, {7, 3}}, {{11, 4}, {9, 3}}, {{11, 4}, {10, 3}},
  {{12, 1}, {0, 0}}, {{12, 2}, {0, 0}}, {{12, 2}, {3, 1}},
  {{12, 5}, {0, 0}}, {{12, 5}, {3, 1}}, {{12, 5}, {5, 2}},
  {{12, 5}, {6, 2}}, {{12, 5}, {7, 3}}, {{12, 5}, {9, 3}},
  {{12, 5}, {10, 3}}, {{12, 5}, {10, 4}}, {{13, 1}, {0, 0}}}
```

```
In[443]:= {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}},
   {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}},
   {{8, 1}, {0, 0}}, {{8, 2}, {0, 0}}, {{8, 2}, {3, 1}},
   {{8, 3}, {0, 0}}, {{8, 3}, {3, 1}}, {{8, 3}, {5, 2}},
   {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}}, {{9, 2}, {0, 0}},
   {{9, 2}, {3, 1}}, {{10, 1}, {0, 0}}, {{10, 2}, {0, 0}},
   {{10, 2}, {3, 1}}, {{11, 1}, {0, 0}}, {{11, 2}, {0, 0}},
   {{11, 2}, {3, 1}}, {{11, 3}, {0, 0}}, {{11, 3}, {3, 1}},
   {{11, 3}, {5, 2}}, {{11, 3}, {6, 2}}, {{11, 4}, {0, 0}},
   {{11, 4}, {3, 1}}, {{11, 4}, {5, 2}}, {{11, 4}, {6, 2}},
   {{11, 4}, {7, 3}}, {{11, 4}, {9, 3}}, {{11, 4}, {10, 3}},
   {{12, 1}, {0, 0}}, {{12, 2}, {0, 0}}, {{12, 2}, {3, 1}},
   {{12, 5}, {0, 0}}, {{12, 5}, {3, 1}}, {{12, 512}, {5, 2}},
   {{12, 5}, {6, 2}}, {{12, 5}, {7, 3}}, {{12, 5}, {9, 3}},
   {{12, 5}, {10, 3}}, {{12, 5}, {10, 4}}, {{13, 1}, {0, 0}}}

Out[443]= {{{4, 1}, {0, 0}}, {{5, 1}, {0, 0}}, {{6, 1}, {0, 0}},
   {{7, 1}, {0, 0}}, {{7, 2}, {0, 0}}, {{7, 2}, {3, 1}},
   {{8, 1}, {0, 0}}, {{8, 2}, {0, 0}}, {{8, 2}, {3, 1}},
   {{8, 3}, {0, 0}}, {{8, 3}, {3, 1}}, {{8, 3}, {5, 2}},
   {{8, 3}, {6, 2}}, {{9, 1}, {0, 0}}, {{9, 2}, {0, 0}},
   {{9, 2}, {3, 1}}, {{10, 1}, {0, 0}}, {{10, 2}, {0, 0}},
   {{10, 2}, {3, 1}}, {{11, 1}, {0, 0}}, {{11, 2}, {0, 0}},
   {{11, 2}, {3, 1}}, {{11, 3}, {0, 0}}, {{11, 3}, {3, 1}},
   {{11, 3}, {5, 2}}, {{11, 3}, {6, 2}}, {{11, 4}, {0, 0}},
   {{11, 4}, {3, 1}}, {{11, 4}, {5, 2}}, {{11, 4}, {6, 2}},
   {{11, 4}, {7, 3}}, {{11, 4}, {9, 3}}, {{11, 4}, {10, 3}},
   {{12, 1}, {0, 0}}, {{12, 2}, {0, 0}}, {{12, 2}, {3, 1}},
   {{12, 5}, {0, 0}}, {{12, 5}, {3, 1}}, {{12, 512}, {5, 2}},
   {{12, 5}, {6, 2}}, {{12, 5}, {7, 3}}, {{12, 5}, {9, 3}},
   {{12, 5}, {10, 3}}, {{12, 5}, {10, 4}}, {{13, 1}, {0, 0}}}
```

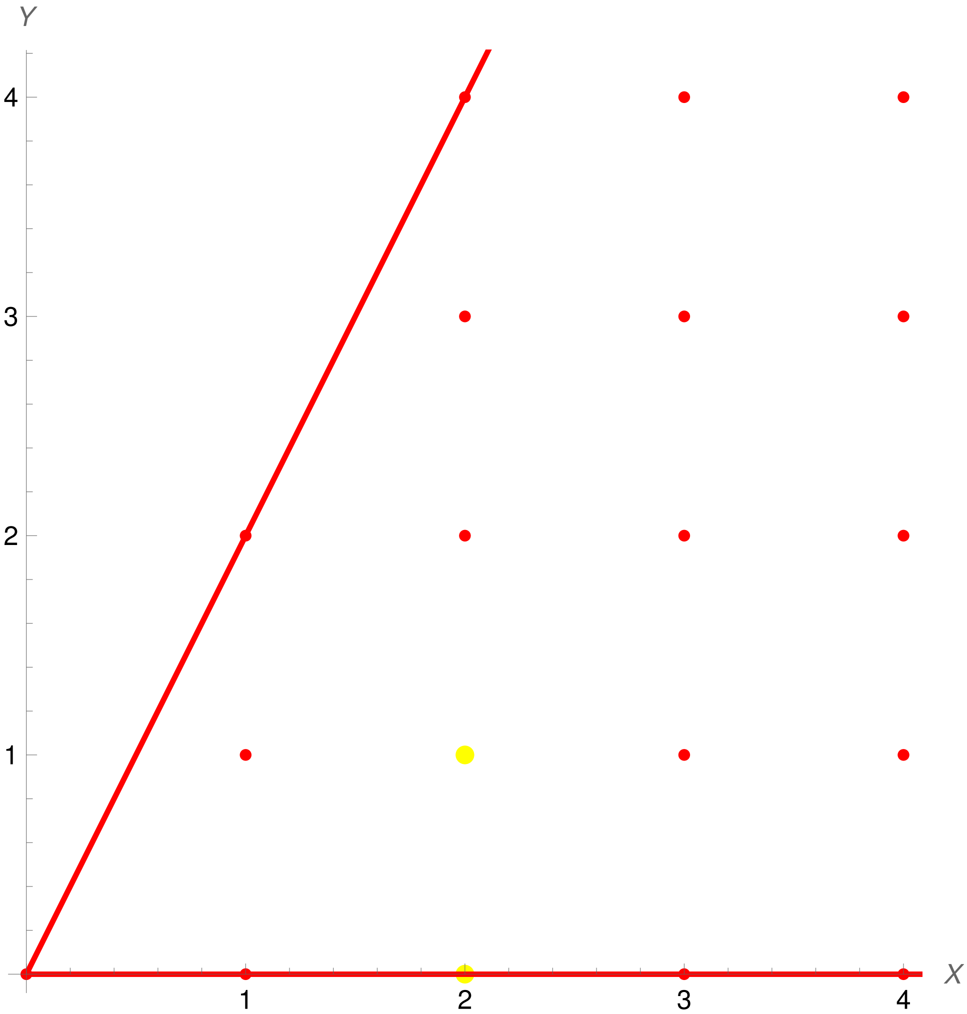
X con condición X ≠ D(X) (artículo)

In[444]:=
```
coneVectors = {{1,0},{1,2}};
```

In[445]:=
```
cminusX14 = {{2,0},{2,1}};
```

In[446]:=
```
Plot2DConeDotsSmallValues[cminusX14,coneVectors]
```

```
{{{0, 0}, {10, 0}}, {{0, 0}, {10, 20}}}
```



In[447]:=
```
IsSetMinusCS[cminusX14,coneVectors,fileDirectory,True,True]
```

```
X not void
A[[k]]{2, 0}
¬InCone[x,Eq]->False
A[[k]]{2, 1}
¬InCone[x,Eq]->False
A[[k]]{2, 0}
gcd->{1, 2}
Lenght[gcd]->2
A[[k]]/gcd[[i]]->{2, 0}
¬MemberQ[A,A[[k]]/gcd[[i]]]->False
A[[k]]/gcd[[i]]->{1, 0}
¬MemberQ[A,A[[k]]/gcd[[i]]]->True
X≠D(X) - This element causes this->{2, 0}
X is not D(X) or X not subset Cone
```

Out[447]= False

In[448]:=
```
cminusEq=ConeGenSupHyp[coneVectors,fileDirectory,operatingSy
GenerateDX[cminusX14,cminusEq,True]
```

```
---------------------------
A[[k]]{2, 0}
¬InCone[x,Eq]->False
A[[k]]{2, 1}
¬InCone[x,Eq]->False
*******************
A[[k]]{2, 0}
gcd->{1, 2}
Lenght[gcd]->2
Element added->{1, 0}True
*******************
A[[k]]{2, 1}
gcd->{1}
Lenght[gcd]->1
DX finished
```

Out[449]= {{2, 0}, {2, 1}, {1, 0}}

Ejemplo 5.4 en N^2

In[450]:=
```
fileDirectory=NotebookDirectory[];
```

Vectores del cono generado por ⟨(1,0), (1,1), (1,2)⟩

In[451]:=
```
genCone = {{1,0},{1,1},{1,2}};
```

In[452]:=
```
(* Vectores de los rayos extremales para sacar ecuaciones de
semig=genCone;
T1=ConvexHullMesh[Join @@ {{{0,0}},semig}];
LineasT1=MeshPrimitives[T1,1];
T1L=Select[Level[LineasT1,{2}],MemberQ[#,{0,0}]&];
vectorsinrays=Select[Flatten[T1L,1],# ≠{0,0}&]

(*Representación gráfica Apery*)
vectorsinrays=ParallelTable[{{0,0},20*vectorsinrays[[i]]},{i,Leng
```

Out[456]= {{1, 0}, {1, 2}}

In[458]:=
```
coneVectors = {{1,0},{1,2}};
```

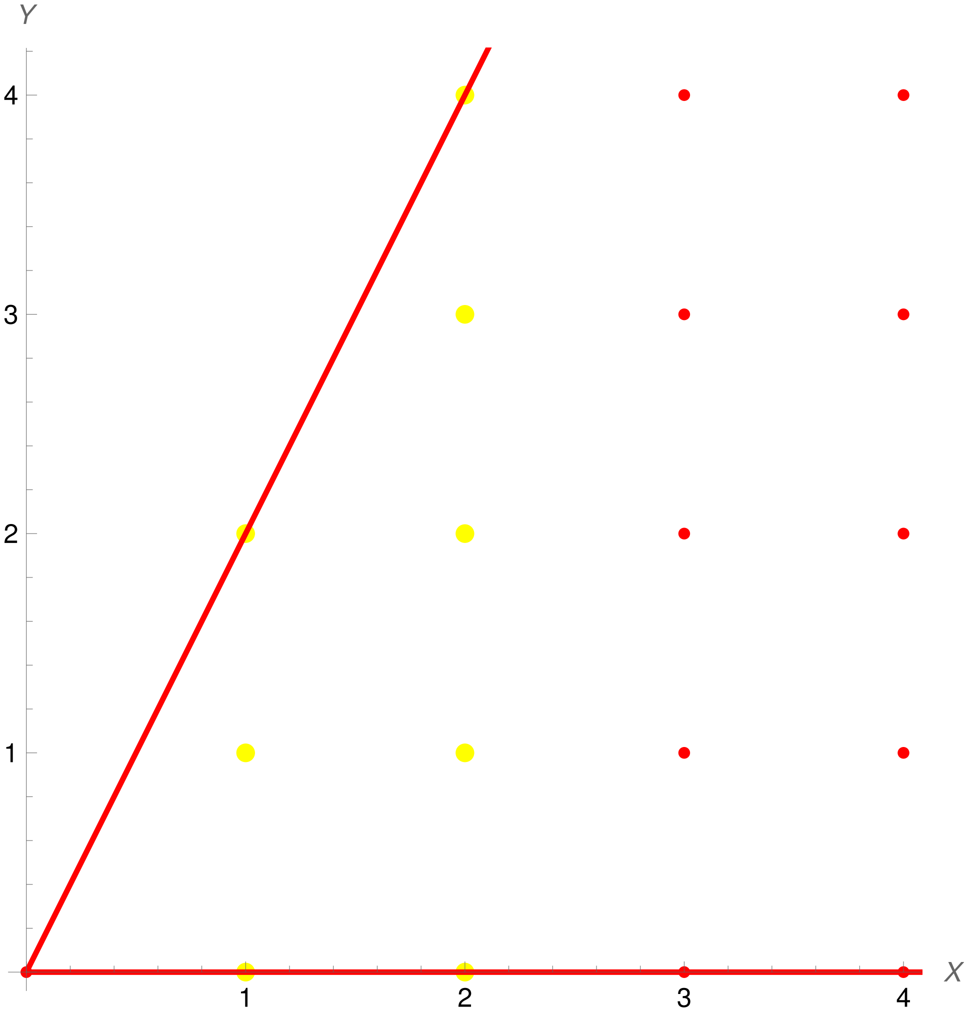
Ejemplo

In[459]:=
```
coneVectors = {{1,0},{1,2}};
```

In[460]:=
```
cminusX2 = {{1,0},{1,1},{1,2},{2,0},{2,1},{2,2},{2,3},{2,4}};
```

In[461]:=
```
Plot2DConeDotsRegSize[cminusX2,coneVectors,1]
```

```
{{{0, 0}, {10, 0}}, {{0, 0}, {10, 20}}}
```



In[462]:=
```
IsSetMinusCS[cminusX2,coneVectors,fileDirectory,False,False]
```

Out[462]=
```
True
```

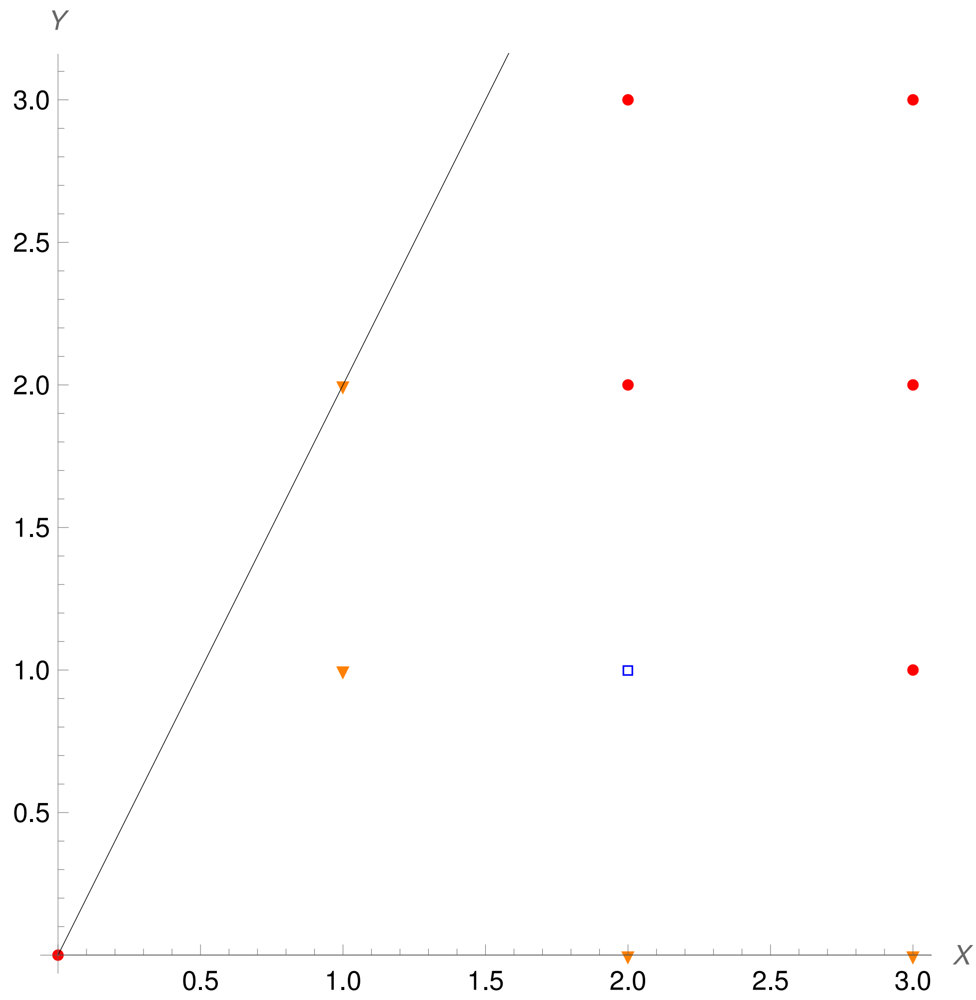
Capítulo 6

Ejemplo 6.1 en N^2

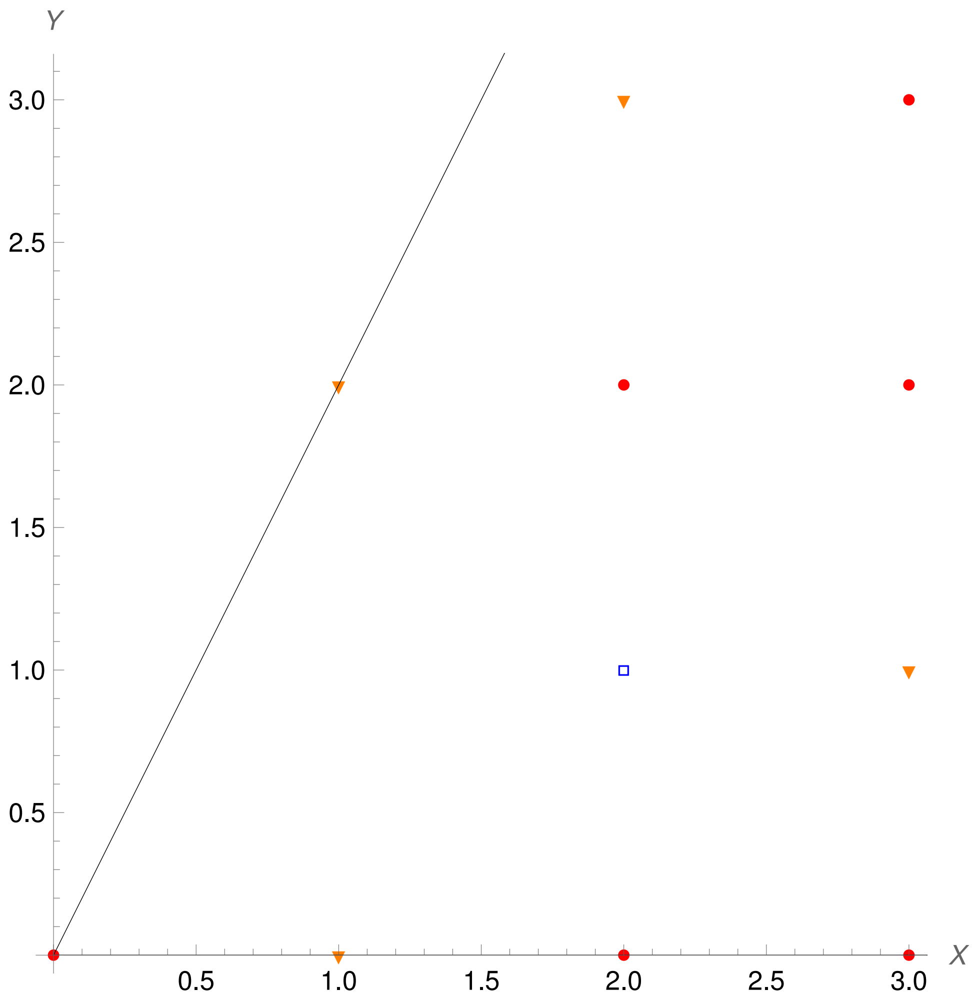
In[463]:=
```
vectCS={
{{{2,0}, {3,0}, {1,1}, {1,2}},{{1,0},{2,1}}},
{{{1, 0}, {3, 1}, {1, 2}, {2, 3}},{{1,1},{2,1}}},
{{{3, 0}, {4, 0}, {5, 0}, {1, 1}, {3, 1}, {1, 2}, {3, 2}},{{1,0}
{{{2, 0}, {3, 0}, {3, 1}, {4, 1}, {1, 2}, {2, 2}, {2, 3}, {3,
{{{3, 0}, {4, 0}, {5, 0}, {3, 1}, {4, 1}, {5, 1}, {1, 2}, {2,
{{{3, 0}, {4, 0}, {5, 0}, {3, 1}, {4, 1}, {5, 1}, {2, 2}, {3,
{{{1, 1}, {2, 3}, {2, 4}, {3, 0}, {3, 1}, {3, 2}, {3, 6}, {4,
{{{1, 0}, {2, 2}, {2, 3}, {2, 4}, {3, 1}, {3, 5}, {3, 6}},{{1,1}
{{{2, 0}, {2, 2}, {2, 3}, {2, 4}, {3, 0}, {3, 1}, {3, 2}, {3,
{{{1, 1}, {2, 0}, {2, 3}, {2, 4}, {3, 0}, {3, 2}, {3, 6}},{{1,0}
};
```

In[464]:=
```
Table[Plot2DSemigAllBW[cs[[1]],cs[[2]],{cs,vectCS}]
```

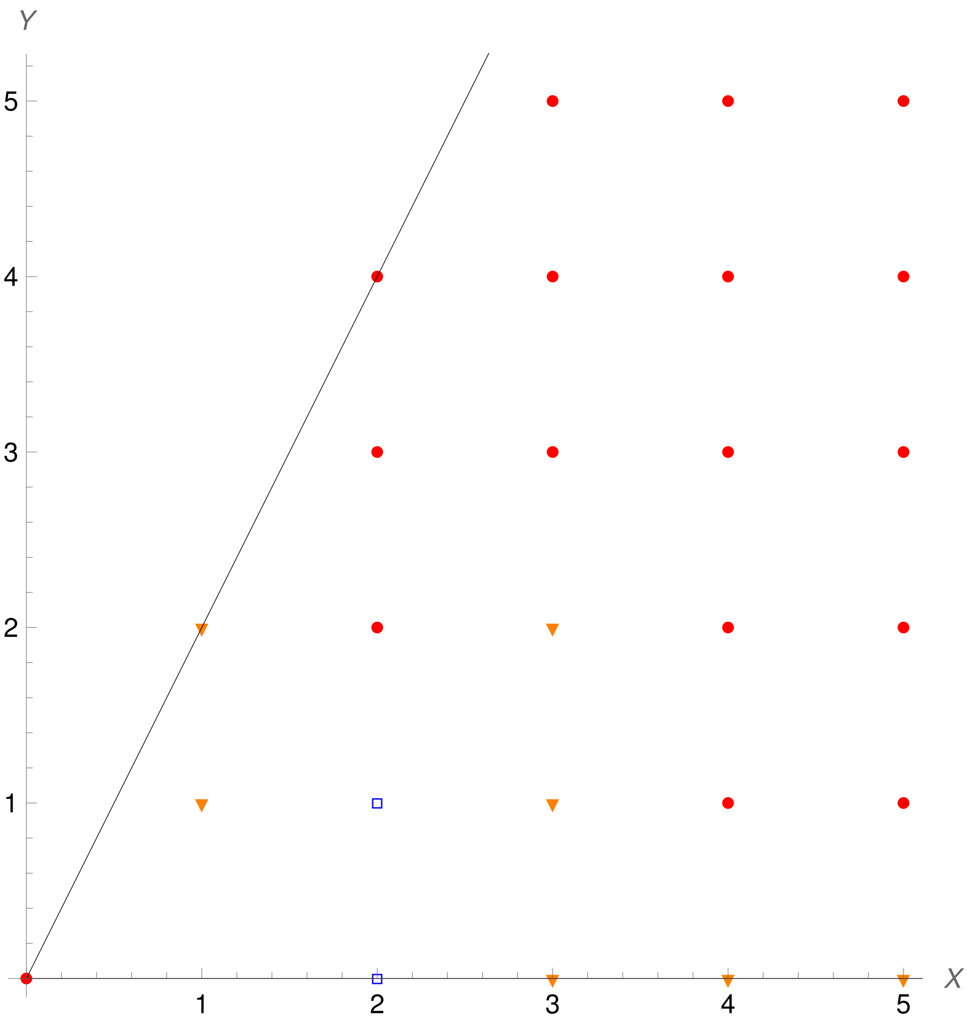
Vectores de los rayos extremales= {{3, 0}, {1, 2}}



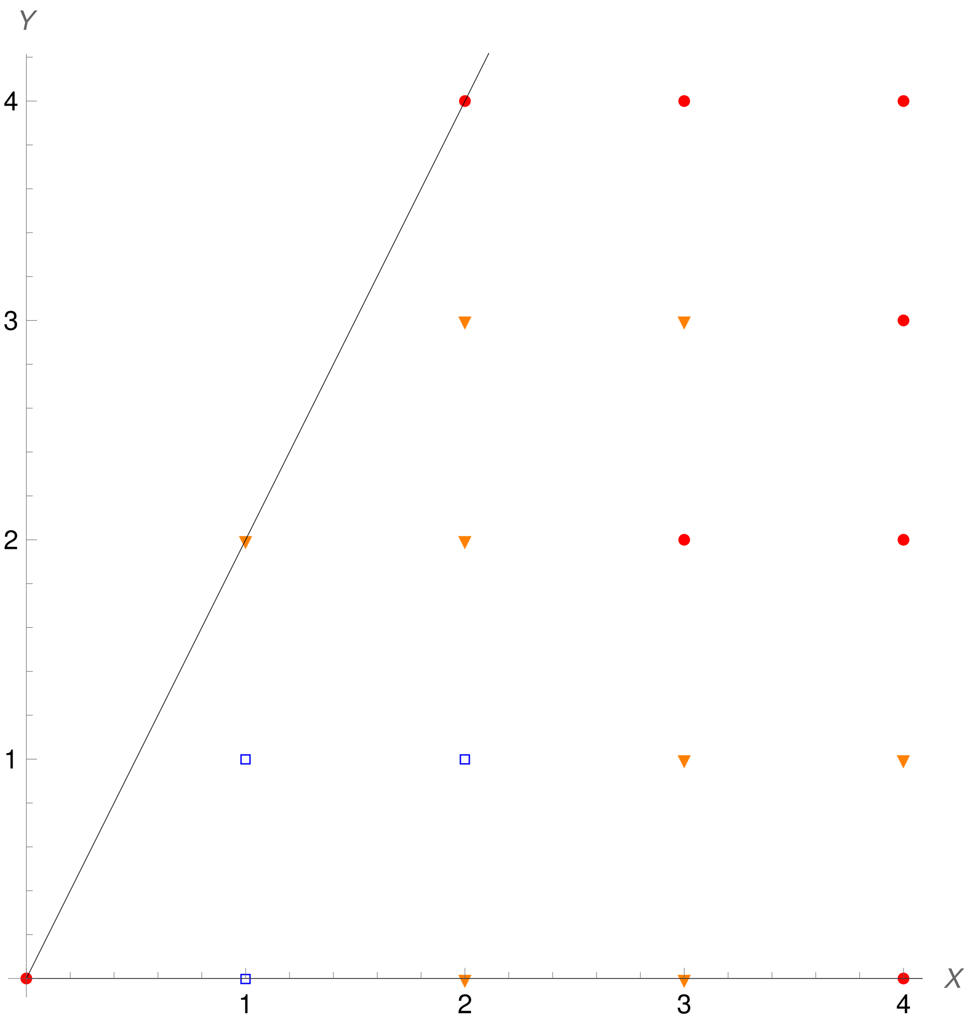
Vectores de los rayos extremales= {{1, 0}, {1, 2}}



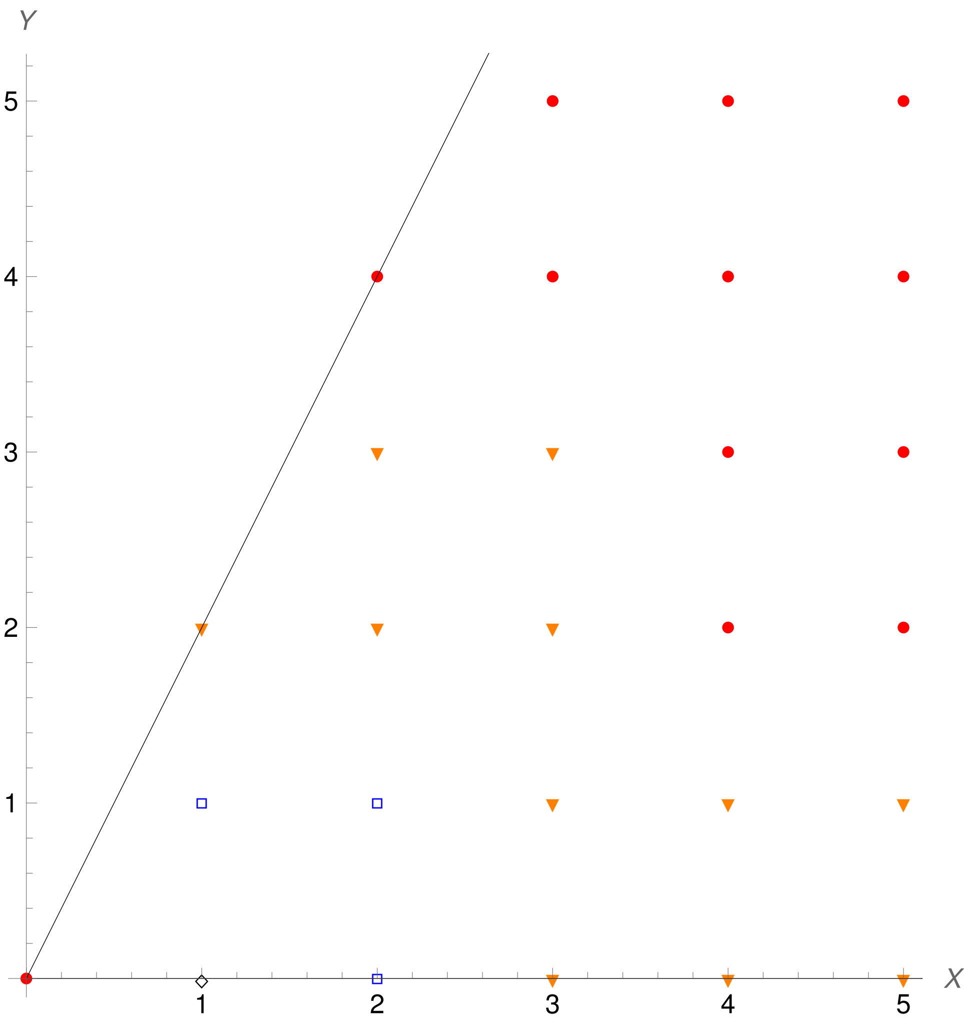
Vectores de los rayos extremales= {{5, 0}, {1, 2}}



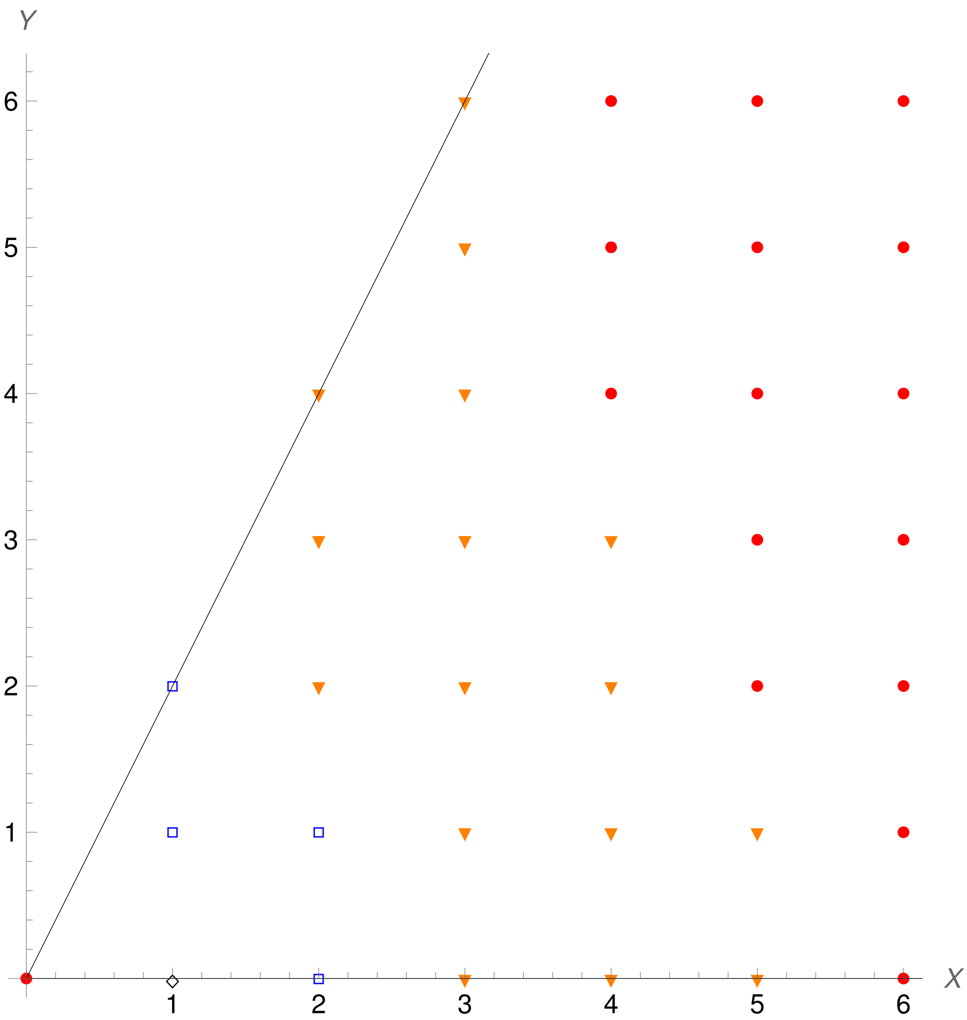
Vectores de los rayos extremales= {{3, 0}, {1, 2}}



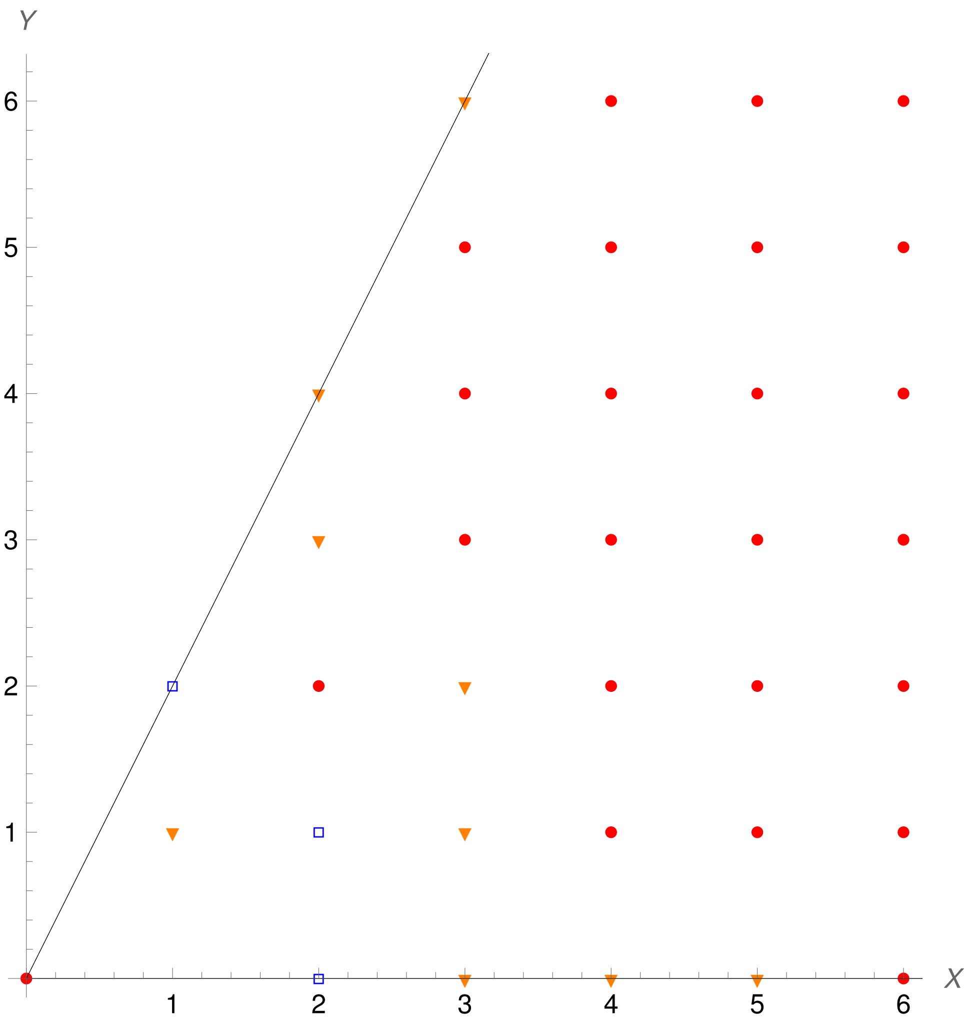
Vectores de los rayos extremales= {{5, 0}, {1, 2}}



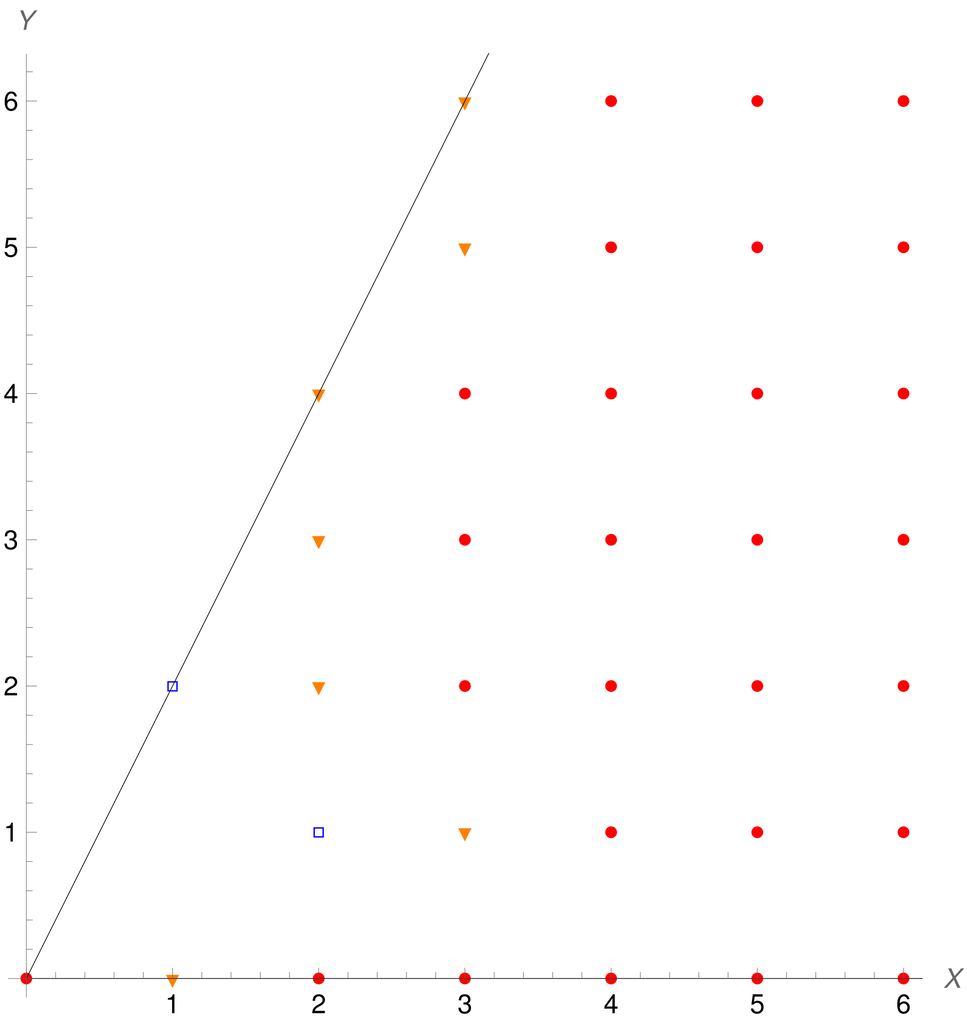
Vectores de los rayos extremales= {{5, 0}, {3, 6}}



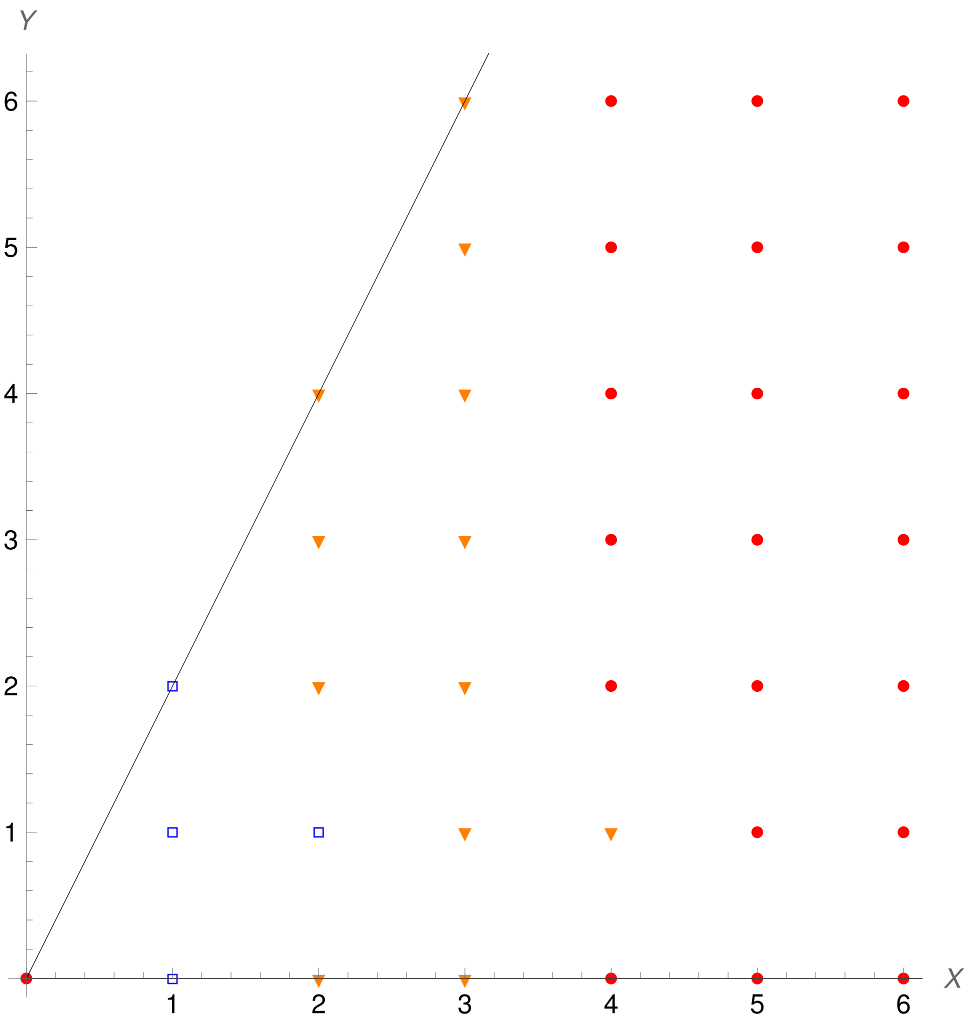

```
Vectores de los rayos extremales= {{5, 0}, {3, 6}}
```



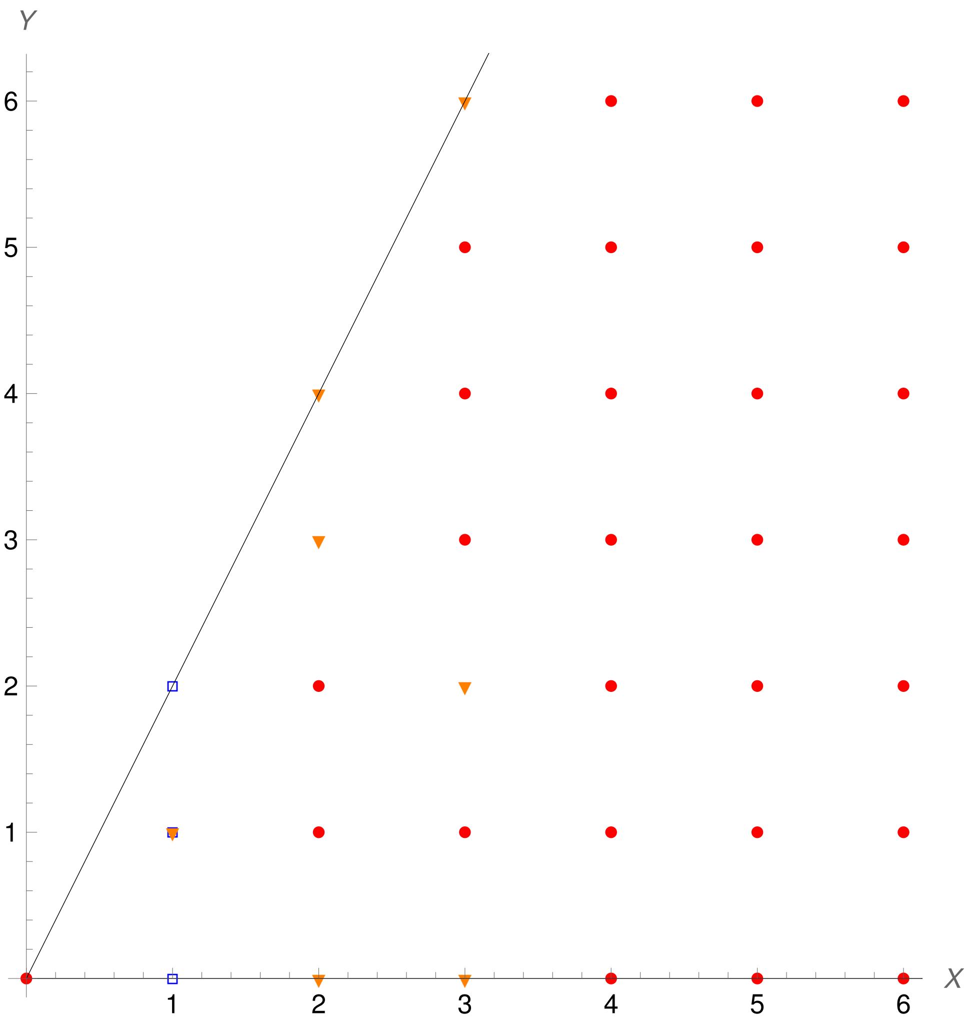

```
Vectores de los rayos extremales= {{1, 0}, {3, 6}}
```



Vectores de los rayos extremales= {{3, 0}, {3, 6}}



Vectores de los rayos extremales= {{3, 0}, {3, 6}}



Out[464]= {Null, Null, Null, Null,
 Null, Null, Null, Null, Null, Null}

Toolbox

Algunas funciones

C-Semigrupo a partir de dos conjuntos de generadores minimales

In[465]:=
```
(* Si quieres puedes meter un especial tienes
 otro semigrupo*)
MinGen[gen1_, gen2_, holes_, Eq_] :=
  Module[{i, j, k, msg = gen1, pos, xx, genOrdenado,
    seguir},
   (*Elimina generadores no minimales de
     gen1 U gen2. <gen1 U gen2>
    es el semigrupo igual al cono dado por
     las ecuaciones Eq menos los huecos
     hold. Los elementos de gen1 son minimales
     en gen1 U gen2 *)
   genOrdenado = Sort[Union[gen1, gen2]];
   pos =
    Flatten[Table[Position[genOrdenado, gen2[[i]]],
      {i, Length[gen2]}], 2];
   For[i = 1, i ≤ Length[pos], i++,
    seguir = True;
    For[k = 1, (k ≤ pos[[i]] - 1) , k++,
     xx = genOrdenado[[pos[[i]]]] - genOrdenado[[k]];
     If[(! InCone[xx, Eq] ∨ MemberQ[holes, xx]),
      seguir = True,
      seguir = False;
      Break[];
     ];
    ];
    If[seguir, AppendTo[msg, genOrdenado[[pos[[i]]]]]];
   ];
   Return[msg]
  ];
```

Algunas funciones para generar matrices de órdenes

In[466]:=
```
OrdenAleatorio[f_] := Module[{M},
    M = Join[RandomInteger[{1, 10}, {1, f}],
      RandomInteger[{-10, 10}, {f - 1, f}]];
    While[Det[M] == 0,
     M = Join[RandomInteger[{1, 10}, {1, f}],
        RandomInteger[{-10, 10}, {f - 1, f}]];
    ];
    Return[M];
   ];
```

In[467]:=
```
OrdenCrear[f_] := Module[{M},
    M = Join[RandomInteger[{1, 10}, {1, f}],
      RandomInteger[{-10, 10}, {f - 1, f}]];
    While[Det[M] == 0,
     M = Join[RandomInteger[{1, 10}, {1, f}],
        RandomInteger[{-10, 10}, {f - 1, f}]];
    ];
    Return[M];
   ];
```

Algo de conjetura de Wilf con Frobenius

In[468]:=
```
WilfMatrizOrdenAleatoria[gen_, hole_, dirTrab_,
    rayos_, info_] :=
```

```
(*Versión para No N^p completo,
poliedro ajustado a rayos para latte y
 normaliz!!*)
Module[{i, Frob, sumF, VertConv, pesos,
  p = Length[hole[[1]]], VertConvLatte, cad, f,
  st1, ptoIn = 0, lineas = {}, caux, ptohip,
  x, MonHip, MonHip2, posFrobHip, MatrizOrden,
  posFrobHip2, nombre},
 (*hole lista de huecos*)
 (*Devuelve todos los elementos de N^p por
  debajo de Frobenius respecto a un orden
  fijado "DegreeLexicographic"*)

 (*MatrizOrden={{1,1},{1,0}};*)
 MatrizOrden = OrdenAleatorio[p];
 If[info, Print["Matriz de orden= ",
   MatrizOrden]];

 Frob = Frobenius2[hole, MatrizOrden];
 (*Frobnenius*)
 pesos = MatrizOrden[[1]];
 sumF = Sum[Frob[[i]] * pesos[[i]], {i, p}];
 VertConv = Table[sumF * rayos[[i]],
   {i, Length[rayos]} ];
 VertConvLatte =
  Table[Prepend[VertConv[[i]], pesos.rayos[[i]]],
   {i, Length[rayos]}];
 (*Print["pesos= ",pesos, " Frob= ",Frob,
   " suma= ",sumF, " vértices= ",VertConv,
   " vertices para Latte= ",VertConvLatte];*)
 cad = ToString[Length[VertConvLatte] + 1] <>
   " " <> ToString[p + 1] <> "\n";
```

```
cad = cad <> ToString[1];
For[i = 1, i ≤ p, i ++,
  cad = cad <> "  " <> ToString[0];
];
cad = cad <> "\n";
For[i = 1, i ≤ Length[VertConvLatte], i ++,
  cad = cad <> StringReplace[
        ToString[VertConvLatte[[i]]],
        {", " → " ", "{" → "", "}" → ""}] <> "\n";
];
f = OpenWrite[dirTrab <> "auxlatte2.vrep_latte"];
WriteString[f, cad];
Close[f];
Pause[.2];
ptoIn =
  <<
   "!C:/latte-integrale-1.7.3/dest/bin/count.exe
     --vrep
     D:\Dropbox\Publicaciones\Csemigroup\
     calculos\auxlatte2.vrep_latte";
(*Print["Número de puntos en el convexo= ",
   ptoIn];*)
(*nombre=
  "D:/Dropbox/Publicaciones/Csemigroup/calculos/"<>
   " count --vrep auxlatte.vrep_latte";
Print[nombre];
Run[nombre];
(*Run["count.exe --vrep " <>dirTrab<>
    "auxlatte.vrep_latte"];*)
cad=ReadString[dirTrab<>"numOfLatticePoints"];
st1=StringToStream[cad];
ptoIn=ToExpression[ReadLine[st1]];
```

```
*)
(*METO SALIDA EN ptoIn*)
(*YA ESTARÍAN CONTADOS TODOS LOS NATURALES
 QUE ESTÁN EN EL HIPERPLANO Y POR DEBAJO
 DEL HIPERPLANO QUE CONTIENE A Frob*)
VertConvLatte =
 Table[Append[VertConv[[i]], pesos.rayos[[i]]],
  {i, Length[rayos]}];
(*Convierte para normaliz*)
cad = "";
cad = "amb_space " <> ToString[p] <> "\n";
cad = cad <> "vertices " <>
  ToString[Length[VertConvLatte]] <> "\n";
For[i = 1, i ≤ Length[VertConvLatte], i++,
 cad = cad <> StringReplace[
     ToString[VertConvLatte[[i]]],
     {", " → " ", "{" → "", "}" → ""}] <> "\n";
];
Pause[.1];
f = OpenWrite[dirTrab <> "auxhiper2.in"];
WriteString[f, cad];
Close[f];
Run["normaliz -c -N -a " <> dirTrab <>
  "auxhiper2.in"];
cad = ReadString[dirTrab <> "auxhiper2.gen"];
st1 = StringToStream[cad];
caux = ReadLine[st1];
caux = ReadLine[st1];
caux = ReadLine[st1];
While[Characters[caux] ≠ {},
     AppendTo[lineas, caux];
     caux = ReadLine[st1];
```

```
 ];
ptohip =
 Flatten[ImportString[#, "Table"] & /@ lineas,
   1];
ptohip = Select[ptohip,
   #[[Length[ptohip[[1]]]]] == 1 &];
For[i = 1, i ≤ Length[ptohip], i ++,
 ptohip[[i]] = Delete[ptohip[[i]], p + 1];
];

MonHip = Table[MatrizOrden.ptohip[[i]],
   {i, 1, Length[ptohip]}];
MonHip2 = Sort[MonHip];

(*MonHip=
 Flatten[Table[Position[MonHip,MonHip2[[i]]],
   {i,Length[MonHip2]}],1];
ptohip=
 Flatten[Table[ptohip[[MonHip[[i]]]],
   {i,Length[ptohip]}],1];
posFrobHip=Position[ptohip,Frob][[1]][[1]];
Print[MonHip2,MatrizOrden,Frob,
 MatrizOrden.Frob];*)

(*Print[" aqui-> ",MonHip2,MatrizOrden,
  Frob,MatrizOrden.Frob];*)
posFrobHip2 =
 Position[MonHip2, MatrizOrden.Frob][[1]][[1]];
(*Print["Ptos en hiperplano= ",ptohip,
 "\n ptos. en convexo= ",ptoIn];
Print["pos de Frob1= ",posFrobHip,
 "pos de Frob2 en hiperplano= ",posFrobHip2];*)
```

```
(*Print["Aqui2-> ",Length[gen]," ptoIn= ",
  ptoIn," longPtohip= ",Length[ptohip],
  " longHole= ",Length[hole]];*)

(*Print[(*"Semigrupo= " ,gen, " huecos= ",
  hole, " Frob= ",Frob," Orden= ",
  MatrizOrden,"\n *)"Huecos= ",Length[hole],
  ", e(S)= ",Length[gen],", n(S)= ",
  ptoIn-Length[ptohip]+posFrobHip2-Length[hole],
  ", N(F(S))+1= ",
  (ptoIn-Length[ptohip]+posFrobHip2+1),
  ", n(S)e(S)- (N(F(S))+1)= ",
  Length[gen]*
    (ptoIn-Length[ptohip]+posFrobHip2-
      Length[hole])-
   (ptoIn-Length[ptohip]+posFrobHip2+1),
  ", n(S)e(S)/ (N(F(S))+1)= ",
  N[Length[gen]*
    (ptoIn-Length[ptohip]+posFrobHip2-
       Length[hole])/
     (ptoIn-Length[ptohip]+posFrobHip2+1)]];*)
cociente = Join[cociente,
  {{{StringJoin["Huecos= ",
      ToString[Length[hole]], ", e(S)= ",
      ToString[Length[gen]], ", n(S)= ",
      ToString[ptoIn - Length[ptohip] +
         posFrobHip2 - Length[hole]],
      ", N(F(S))+1= ",
      ToString[
       (ptoIn - Length[ptohip] + posFrobHip2 + 1)],
      ", n(S)e(S)- (N(F(S))+1)= ",
```

```
          ToString[
           Length[gen] *
             (ptoIn - Length[ptohip] + posFrobHip2 -
               Length[hole]) -
            (ptoIn - Length[ptohip] + posFrobHip2 + 1)],
          ", n(S)e(S)/ (N(F(S))+1)= ",
          ToString[
           N[Length[gen] *
              (ptoIn - Length[ptohip] + posFrobHip2 -
                Length[hole]) /
             (ptoIn - Length[ptohip] + posFrobHip2 +
                1)]]},
        {N[Length[gen] *
            (ptoIn - Length[ptohip] + posFrobHip2 -
              Length[hole]) /
           (ptoIn - Length[ptohip] + posFrobHip2 +
              1)]}}];
    If[
     (Length[gen] *
         (ptoIn - Length[ptohip] + posFrobHip2 -
           Length[hole]) ≥
        (ptoIn - Length[ptohip] + posFrobHip2 + 1)) ≠
      True,
     Print["NO WILF!!!!!! "(*,"semigrupo= " ,
       gen, " huecos= ",hole, " Frob= ",Frob,
       " Orden= ",MatrizOrden *)];
    ];
    NotebookSave[];

    Return[
     Length[gen] *
       (ptoIn - Length[ptohip] + posFrobHip2 -
```

```
          Length[hole]) ≥
        (ptoIn - Length[ptohip] + posFrobHip2 + 1)];
  ];
```

Quita min gen en SN

In[469]:=
```
MimRemoveOneGen[gen_, rgen_, H_, Eq_] :=
  Module[{gen0, gen1, min, i},
   (*Calcula el sistema minimal de generadores
     de un semigrupo obtenido quitando al
     semigrupo mínimamente generado por
     "gen" el generador "rgen"
     ("rgen" tiene que pertenecer a "gen").
      Eq son las ecuaciones del cono que
     contiene a <gen>
    y H los elementos que le faltan a <gen>
    para ser el cono*)
   (*Print["Entrada MInRE..= ",gen," , ",rgen,
     " , ",H," , ",Eq];*)
   gen0 = Complement[gen, {rgen}];
   gen1 = Table[gen0[[i]] + rgen, {i, Length[gen0]}];
   gen1 = Union[gen1, {2 * rgen, 3 * rgen}];
   min = MinGen[gen0, gen1, Union[H, {rgen}], Eq];
   (*Manda los de entrada y los generados
    por separado*)
   Return[{min, Union[H, {rgen}]}]
   (*Salida: conjunto formado por
     {generadores minimales,
      huecos respecto al cono inicial} *)
  ];
```

SemigroupGenusToDFromGenEqEnRamaSinQuitarDeEjesYEnCirculo

In[470]:=
```
SemigroupGenusToDFromGenEqEnRamaSinQuitarDeEjesYE⋱
```

```
  nCirculo[conegen_, coneEq_, dirTrab_, d_,
 rayos_, info_, cotaInf_, cotaSup_] :=
Module[{hilb, Eq, SemGenD, i, j, allcase = {},
  allcase1 = {}, allcase2 = {}, todej, suc,
  SemGenD2, time, medGen, mini, W = 0, aux},
 (* dd saco "dd" para hacer una barra dinámica*)
 long = {};
 (* long variable global que recoge dimensión
  de inmersión de los semigrupos que
  aparecen *)
 (*SI MatrizOrden=Matriz de orden,
 comprueba desigualdad de Wilf extendida
  para el orden implementado, =
  False no hace nada*)
 hilb = conegen;
 (*Base de Hilbert del cono inicial*)
 Eq = coneEq;
 (*Inecuaciones del cono inicial*)
 SemGenD = {{0, hilb, {}}};
 If[info,
  Print["Número de semigrupos de ", 0,
    " huecos= ", 1,
    " número generadores minimals= ",
    Length[SemGenD[[1]][[2]]]];
 ];
 suc = {1};
 long = Join[long, {Length[SemGenD[[1]][[2]]]}];
 SemGenD = Join[SemGenD,
   {{1, ParallelMap[
       MimRemoveOneGen[hilb, #, {}, Eq] &, hilb]}}];
 (*El formato de SemGenD es uno inicial
  de las primeras pruebas. Ajustar para
  quitar el 0 y el 1 inicial. SemGenD2
```

```
 ya no tiene ese número entero inicial*)
suc = Join[suc, {Length[SemGenD[[2, 2]]]}];
SemGenD2 = SemGenD[[2, 2]];
(*Print["Semigrupos con n generadores= ",
  Select[SemGenD2,Length[#[[1]]]== 10&]];*)
(*Se toma un único semigrupo de forma
 aleatoria*)
SemGenD2 =
 SemGenD2[[RandomInteger[{1, Length[SemGenD2]}]]];
For[dd = 2, dd ≤ d, dd ++,
 If[info,
  Print["Semigrupo seleccionado= ",
    SemGenD2, " nº huecos= ",
    Length[SemGenD2[[2]]]];
 ];
 long = Join[long, {Length[SemGenD2[[1]]]}];
 time = SessionTime[];
 allcase = SemGenD2;

 NotebookSave[];

 SemGenD2 = {};
 (*Todos los semigrupos con dd-
  1 huecos respecto al cono inicial. Cada
   semigrupo está determinado por
   {sistema minimal de generadores, huecos}*)
 (*Print["SEMIGRUPOS DE ENTRADA= ",allcase];*)

 (*Tomo un elemento aleatorio de sistema
  minimal para "hacer hueco" fuera de
  los ejes*)
 aux = Select[allcase[[1]],
```

```
      (A[[1]][[2]]*#[[1]] - A[[1]][[1]]*#[[2]])*
          (A[[2]][[2]]*#[[1]] - A[[2]][[1]]*#[[2]]) ≠ 0 &&
        (#[[1]])^2 + (#[[2]])^2 ≤ cotaSup &&
        cotaInf < (#[[1]])^2 + (#[[2]])^2 &];
  If[aux ≠ {},
   (*Rama aelatoria a partir del nodo*)
   allcase2 = Join[allcase,
     {{aux[[RandomInteger[{1, Length[aux]}]]]}}];
   (*Print[
     "SEmigrupos con efectivos antes de
       quitar= ",allcase2,
     " longitud= ",Length[allcase2]];*)

   (*Se une a cada (generadores semi, hueco)
    los generadores efectivos,
   i.e. la estructura es
    (generadores semi, hueco,
     generadores efectivos)*)

   (*Print["semigrupos con efectivos= ",
    allcase2, " longitud= ",Length[allcase2]];
   Print["semigrupo 1= ",allcase2[[1]]];*)
   allcase1 = allcase2;
   NotebookSave[];
   (*Print["Mando a MimRemoveOneGen-> ",
     allcase1[[1]]," ",allcase1[[3,1]]," ",
     allcase1[[2]]];*)
   SemGenD2 = MimRemoveOneGen[allcase1[[1]],
     allcase1[[3, 1]], allcase1[[2]], Eq] ;
   (*Print[" Tiempo para ",dd,
    " huecos (cálculo completo)= ",
```

```
      SessionTime[]-time];
     Print[
      "Semigrupos con multiplicidad mínima
        generadores= ",
      Select[SemGenD2,
       Length[#[[1]]]== 2*Length[conegen[[1]]]&]];*)
     (*Print[
       "Semigrupos con máximo número de
         generadores= ",
       Select[SemGenD2,Length[#[[1]]]== maxi&]];*)
     (*Print["nuevo semigrupo= ",SemGenD2];*)
     NotebookSave[];
     ,
     Print[
      "No generadores minimales en la
        circunferncia de radio =", cota];
     Break[];
    ];
   ];
   (*Print[SemGenD2];*)
   Return[SemGenD2]
   (*SemGen2[1]= generadores del semigrupo,
   SemGen2[2]= huecos del semigrupo*)
  ];

(*En esta versión del programa sólo se guardan
 los semigrupos en ejecución*)
```

SemigroupGenusToDFromGenEqEnRa

ma

In[471]:=
```
SemigroupGenusToDFromGenEqEnRama[conegen_,
    coneEq_, dirTrab_, d_, rayos_, info_] :=
   Module[{hilb, Eq, SemGenD, i, j, allcase = {},
     allcase1 = {}, allcase2 = {}, todej, suc,
     SemGenD2, time, medGen, mini, W = 0},
    (* dd saco "dd" para hacer una barra dinámica*)
    long = {};
    (* long variable global que recoge dimensión
     de inmersión de los semigrupos que
     aparecen *)
    (*SI MatrizOrden=Matriz de orden,
    comprueba desigualdad de Wilf extendida
     para el orden implementado, =
     False no hace nada*)
    hilb = conegen;
    (*Base de Hilbert del cono inicial*)
    Eq = coneEq;
    (*Inecuaciones del cono inicial*)

    SemGenD = {{0, hilb, {}}};

    If[info,
     Print["Número de semigrupos de ", 0,
       " huecos= ", 1,
       " número generadores minimals= ",
       Length[SemGenD[[1]][[2]]]];
    ];

    suc = {1};
```

```
long = Join[long, {Length[SemGenD[[1]][[2]]]}];

SemGenD = Join[SemGenD,
  {{1, ParallelMap[
     MimRemoveOneGen[hilb, #, {}, Eq] &, hilb]}}];
(*El formato de SemGenD es uno inicial
 de las primeras pruebas. Ajustar para
 quitar el 0 y el 1 inicial. SemGenD2
 ya no tiene ese número entero inicial*)

suc = Join[suc, {Length[SemGenD[[2, 2]]]}];

SemGenD2 = SemGenD[[2, 2]];
(*Print["Semigrupos con n generadores= ",
  Select[SemGenD2,Length[#[[1]]]== 10&]];*)
(*Se toma un único semigrupo de forma
 aleatoria*)

SemGenD2 =
 SemGenD2[[RandomInteger[{1, Length[SemGenD2]}]]];

For[dd = 2, dd ≤ d, dd ++,
 If[info,
  Print["Semigrupo seleccionado= ",
    SemGenD2, " nº huecos= ",
    Length[SemGenD2[[2]]]];
 ];

 long = Join[long, {Length[SemGenD2[[1]]]}];
 time = SessionTime[];
 allcase = SemGenD2;
```

```
NotebookSave[];

SemGenD2 = {};
(*Todos los semigrupos con dd-
 1 huecos respecto al cono inicial. Cada
  semigrupo está determinado por
  {sistema minimal de generadores, huecos}*)
(*Print["SEMIGRUPOS DE ENTRADA= ",allcase];*)

(*Tomo un elemento aleatorio de sistema
 minimal para "hacer hueco"*)
(*Rama aelatoria a partir del nodo*)
allcase2 = Join[allcase,
  {{allcase[[1]][[RandomInteger[
      {1, Length[allcase[[1]]]}]]]}}];
(*Print[
  "SEmigrupos con efectivos antes de
    quitar= ",allcase2, " longitud= ",
  Length[allcase2]];*)

(*Se une a cada (generadores semi, hueco)
 los generadores efectivos,
i.e. la estructura es
 (generadores semi, hueco,
  generadores efectivos)*)

(*Print["semigrupos con efectivos= ",
 allcase2, " longitud= ",Length[allcase2]];
Print["semigrupo 1= ",allcase2[[1]]];*)
allcase1 = allcase2;
NotebookSave[];
(*Print["Mando a MimRemoveOneGen-> ",
```

```
      allcase1[[1]],"  ",allcase1[[3,1]],"  ",
      allcase1[[2]]];*)
    SemGenD2 = MimRemoveOneGen[allcase1[[1]],
      allcase1[[3, 1]], allcase1[[2]], Eq];
    (*Print[" Tiempo para ",dd,
     " huecos (cálculo completo)= ",
     SessionTime[]-time];
    Print[
     "Semigrupos con multiplicidad mínima
       generadores= ",
     Select[SemGenD2,
      Length[#[[1]]]== 2*Length[conegen[[1]]]&]];*)
    (*Print[
      "Semigrupos con máximo número de
        generadores= ",
      Select[SemGenD2,Length[#[[1]]]== maxi&]];*)
    (*Print["nuevo semigrupo= ",SemGenD2];*)
    NotebookSave[];
   ];
   (*Print[SemGenD2];*)
   Return[SemGenD2]
   (*SemGen2[1]= generadores del semigrupo,
   SemGen2[2]= huecos del semigrupo*)
  ];

(*En esta versión del programa sólo se guardan
 los semigrupos en ejecución*)
```

Generadores efectivos

In[472]:=
```
GetFrobeniusEfGen[gen_, hole_] :=
```

```
Module[{i, j, x, X, MonHole, Frob, EfGen,
  MonGen, t, HijosDGen},
 (*hole lista de huecos*)
 (*gen generadores minimales de un semigrupo
  S cuyos huecos respecto a N^p son hole*)
 (*Devuelve generadores no minimales de
  los semigrupos con un hueco más obtenidos
  de gen y hole "evitando" redundancias*)
 X = Table[x_i, {i, Length[gen[[1]]]}];
 MonGen =
  Sum[Product[(x_i)^gen[[j]][[i]], {i, 1, Length[gen[[j]]]}],
   {j, 1, Length[gen]}];
 MonGen = MonomialList[MonGen, X,
   DegreeLexicographic];
 MonHole =
  Sum[Product[(x_i)^hole[[j]][[i]],
    {i, 1, Length[hole[[j]]]}], {j, 1, Length[hole]}];
 (*Print["Polinomio= ",MonHole];*)
 MonHole = MonomialList[MonHole, X,
   DegreeLexicographic];
 (*Print["Lista monomios= ",MonHole];*)
 Frob = MonHole[[1]];
 (*Éste es el Fröbnenius respecto al orden
  fijado*)
 (*HijosDGen=MonGen;
 For[i=1,i≤Length[gen],i++,
  If[Frob== MonomialList[Frob+MonGen[[i]],All,
       Lexicographic][[1]],
    HijosDGen=Complement[HijosDGen,{MonGen[[i]]}];
   ];
 ];
 EfGen=Table[Exponent[HijosDGen[[j]],x_i],
```

```
    {j,Length[HijosDGen]},{i,Length[gen[[1]]]}];
  *)
  (*Print["Generadores en Frob= ",
    Table[Exponent[MonGen[[j]],xi],{j,Length[gen]},
     {i,Length[gen[[1]]]}]," huecos= ",hole,
    " Frobenius= ",
    Table[Exponent[Frob,xi],{i,Length[hole[[1]]]}]];*)
  t = Length[gen];
  For[i = 1, i ≤ Length[gen], i ++,
   (*Print["Frob= ",Frob," monomio más grande= ",
     MonomialList[Frob+MonGen[[i]],X,Lexicographic][[
      1]], " ¿Iguales?= ",
     Frob== MonomialList[Frob+MonGen[[i]],X,
        Lexicographic][[1]]];*)
   If[Frob == MonomialList[Frob + MonGen[[i]], X,
        DegreeLexicographic][[1]],
     t = i - 1;
     Break[]
    ];
  ];
  EfGen = Table[Exponent[MonGen[[j]], xi], {j, t},
    {i, Length[gen[[1]]]}];
  (*Table[MonGen[[i]],{i,t}];*)
  (*Print[" Generadores efectivos= ",EfGen];*)
  Return[EfGen];
 ];
```

Generando objetos

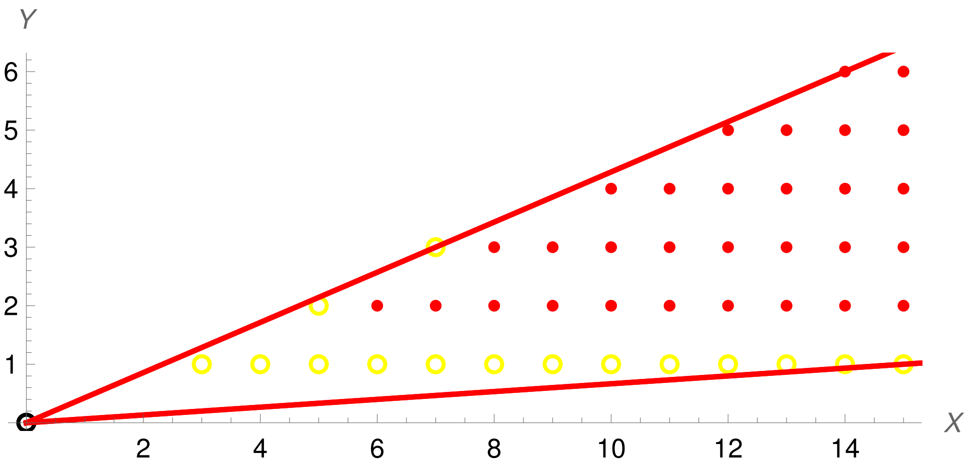
Conos

Cono generado por los elementos de A

In[473]:=
```
fileDirectory = NotebookDirectory[];
A = {{15, 1}, {7, 3}};
{hilb, Eq} = ConeGenSupHyp[A, fileDirectory,
   operatingSystem]
Plot2DSemig[hilb, {{0, 0}}]
```

Out[475]=
```
{{{3, 1}, {4, 1}, {5, 1}, {5, 2}, {6, 1}, {7, 1},
  {7, 3}, {8, 1}, {9, 1}, {10, 1}, {11, 1}, {12, 1},
  {13, 1}, {14, 1}, {15, 1}}, {{-1, 15}, {3, -7}}}
```

```
Vectores de los rayos extremales= {{15, 1}, {7, 3}}
```



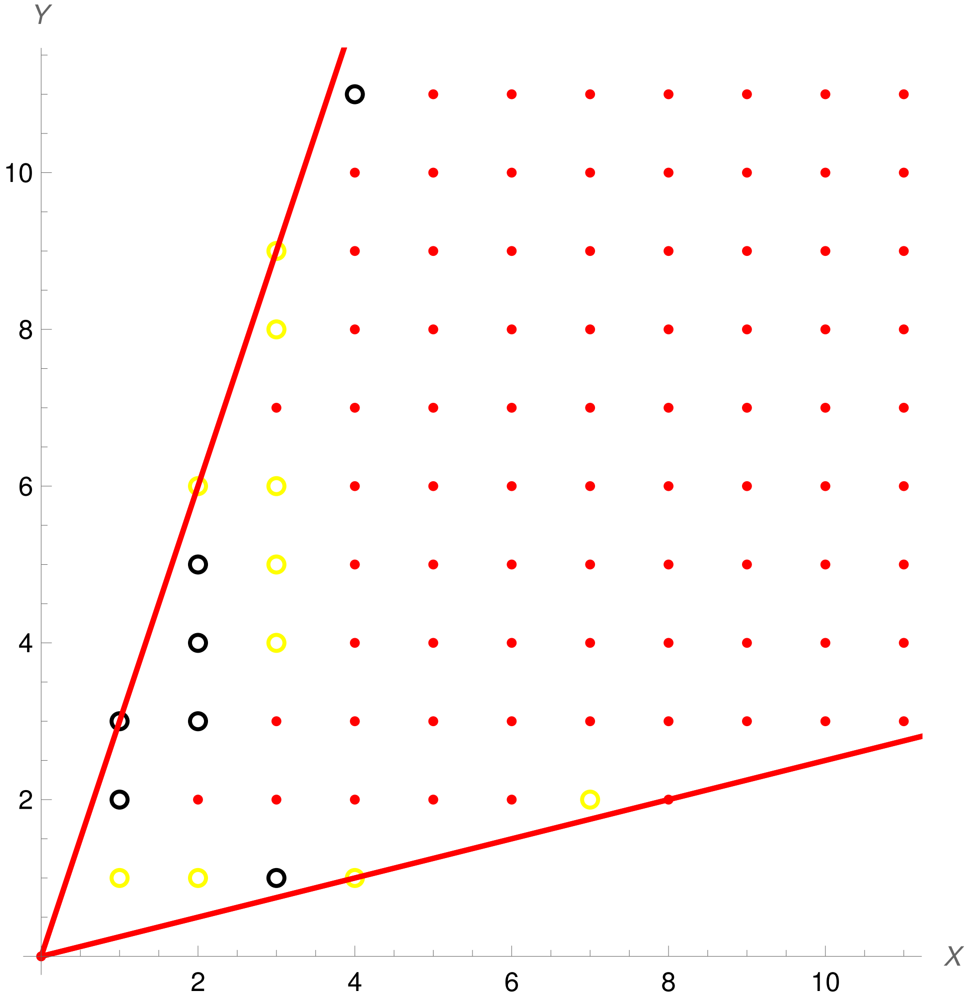
```
In[477]:= (*Verifica si un pto está en el cono dado por las inecuacior
          auxpto= {2,6}
          auxprod=Eq.auxpto
          auxverif=Table[auxprod[[i]]≥ 0,{i,Length[auxprod]}];
          auxverif=Union[auxverif];
          If[auxverif== {True},True,False]

Out[477]= {2, 6}

Out[478]= {88, -36}

Out[481]= False
```

Semigrupos

Al cono anterior, empleando las formulas obtenidas, generamos un semigrupo quitando huecos de manera aleatoria.

```
In[482]:= fileDirectory = NotebookDirectory[];
          A = {{3, 9}, {4, 1}};
          {hilb, Eq} = ConeGenSupHyp[A, fileDirectory,
              operatingSystem];

          nHuecos = 7
          quitaHuecos = SemigroupGenusToDFromGenEqEnRama[
            hilb, Eq, fileDirectory, nHuecos, A, 0]
          Plot2DSemig[quitaHuecos[[1]], quitaHuecos[[2]]]

Out[485]= 7

Out[486]= {{{1, 1}, {2, 1}, {2, 6}, {3, 5},
             {3, 6}, {3, 8}, {3, 9}, {4, 1}, {7, 2}, {3, 4}},
           {{1, 2}, {1, 3}, {2, 3}, {2, 4}, {2, 5}, {3, 1}, {4, 11}}}
```

Vectores de los rayos extremales= {{4, 1}, {3, 9}}



Semigrupos con radios

In[488]:=
```
fileDirectory = NotebookDirectory[];
A = {{30, 2}, {7, 3}};

{hilb, Eq} = ConeGenSupHyp[A, fileDirectory,
    operatingSystem];

nHuecos = 40;
{r1, r2} = {(4) ^ 2, (20) ^ 2};

KK =
 SemigroupGenusToDFromGenEqEnRamaSinQuitarDeEjesYEn⋱
   Circulo[hilb, Eq, fileDirectory, nHuecos, A,
  0, r1, r2]
If[KK ≠ {}, Plot2DSemigAll[KK⟦1⟧, KK⟦2⟧]]
```

Out[493]=
```
{{{3, 1}, {7, 3}, {12, 5}, {13, 4}, {15, 1},
  {15, 4}, {16, 2}, {17, 6}, {19, 4}, {20, 2}, {20, 3},
  {20, 4}, {20, 5}, {21, 2}, {21, 4}, {21, 5}, {22, 2},
  {22, 3}, {22, 6}, {23, 2}, {24, 2}, {25, 2}, {26, 2},
  {27, 2}, {28, 2}, {29, 2}, {34, 3}, {17, 5}},
 {{4, 1}, {5, 1}, {5, 2}, {6, 1}, {7, 1}, {7, 2}, {8, 1},
  {8, 2}, {8, 3}, {9, 1}, {9, 2}, {10, 1}, {10, 2}, {10, 3},
  {11, 1}, {11, 2}, {11, 3}, {11, 4}, {12, 1}, {12, 2}, {12, 3},
  {13, 1}, {13, 2}, {13, 3}, {14, 1}, {14, 2}, {14, 3},
  {14, 4}, {14, 5}, {15, 2}, {15, 3}, {16, 3}, {16, 4},
  {17, 2}, {17, 3}, {17, 4}, {18, 3}, {18, 4}, {19, 2}, {19, 5}}}
```

Vectores de los rayos extremales= {{15, 1}, {7, 3}}