La columna de matemática computacional

Sección a cargo de

Tomás Recio

En este número de La Gaceta*, presentamos... dos reseñas de acontecimientos recientes, ambos relacionados con esta* Columna*.*

El primer artículo narra el planteamiento y desarrollo de una estrecha colaboración entre matemáticos de diversos países, reunidos en un pueblecito de la Sierra de Cameros (La Rioja) durante una semana de febrero de 2010, para implementar, en el programa de cálculo simbólico CoCoA, una biblioteca de funciones que facilite el trabajo con ideales monomiales. Una iniciativa tan singular como interesante...

El segundo artículo se refiere a un hecho aún más actual: el anuncio por nuestro colega Francisco Santos, a principios de mayo de 2010, de la refutación de la conjetura de Hirsch; un anuncio que ha tenido amplísima cobertura en todos los medios de comunicación. En la segunda parte de esta Columna*, Santos nos ofrece, como primicia, su visión sobre dicha conjetura y una exposición simplificada de las ideas que subyacen a su contraejemplo. ¡Enhorabuena, Paco!*

Una semana monomial

por

Óscar Fernández-Ramos, Eva García-Llorente y Eduardo Sáenz-de-Cabezón

Durante la semana del 1 al 5 de febrero de 2010 la Universidad de La Rioja celebró unos días de trabajo sobre ideales monomiales organizados por el tercer autor. Los objetivos eran la difusión de la investigación sobre ideales monomiales y la implementación de algunos resultados de esta investigación para obtener un paquete de algoritmos enfocados al trabajo con ideales monomiales. En concreto, uno de los principales objetivos era poner las bases de una librería de funciones en CoCoA-CoCoALib [7, 8] similar a las existentes en otros sistemas como Macaulay2 [11] y Singular [12].

1. El plan

Los ideales monomiales son aquellos ideales de un anillo de polinomios en n variables que tienen un sistema generador mínimo formado por monomios.

En varios sentidos son los ideales más sencillos con los que se trabaja en álgebra conmutativa, pero además tienen importancia por, al menos, tres razones.

En primer lugar, la teoría de bases de Gröbner permite tratar ciertos problemas sobre ideales polinomiales generales mediante su traducción a la versión correspondiente (más sencilla) en ideales monomiales.

En segundo lugar, la estructura combinatoria de los ideales monomiales permite conectarlos con otras estructuras de tipo combinatorio, y resolver problemas a uno y otro lado de esta correspondencia mediante técnicas propias de las respectivas áreas. En este sentido han sido especialmente provechosas las correspondencias de ideales monomiales con hipergrafos, complejos simpliciales y redes [17, 18]. Estas correspondencias convierten a los ideales monomiales en herramientas útiles en distintas aplicaciones como el estudio de la fiabilidad de redes o la biología algebraica.

Y, en tercer lugar, de nuevo el carácter combinatorio de los ideales monomiales y sus especiales características los hacen particularmente adecuados para el desarrollo de algoritmos que trabajen sobre ellos y, a partir de aquí, generar algoritmos enfocados a ideales más generales o a las aplicaciones de los ideales monomiales en otras áreas.

Considerando esta situación, resulta algo sorprendente que los principales sistemas de álgebra computacional empleados en álgebra conmutativa y geometría algebraica, CoCoA [7], Macaulay2 [11] y Singular [12], tengan poco desarrollados tipos de datos y algoritmos específicos para ideales monomiales. De entre ellos es Macaulay2 el que más ha desarrollado un tratamiento específico para este tipo de ideales, aunque es mejorable. Ante esta carencia, nuestro objetivo inicial era, pues, dotarnos a la gente que trabajamos con ideales monomiales de un paquete de algoritmos específicos para este tipo de ideales.

Los algoritmos a los que queríamos dirigir nuestra atención eran los siguientes:

- Algoritmos de estructura: herramientas básicas para el trabajo con cualquier tipo de ideal, pero que en el caso monomial merecen algoritmos propios, pues el uso de los algoritmos generales desaprovecha la estructura propia de los ideales monomiales. Estos algoritmos son usados en múltiples ocasiones como subrutinas básicas de algoritmos más sofisticados.
  - Pertenencia de un elemento del anillo al ideal.
  - Sistema mínimo de monomios generadores del ideal (el sistema generador mínimo formado por monomios es único).
  - Intersecciones, cocientes, ...
- Verificación de ciertas propiedades: de nuevo, la comprobación de que un ideal general tiene ciertas propiedades particularmente interesantes es un problema algorítmicamente complejo que se simplifica enormemente al tratarse de ideales monomiales, con lo que es posible elaborar algoritmos eficientes.

  - Comprobar si un ideal es primo.
  - Comprobar si un ideal es primario.
  - Comprobar si un ideal es irreducible.
- Descomposiciones, construcciones, invariantes: los problemas del cálculo de los principales invariantes y descomposiciones de ideales monomiales, siendo en principio más sencillos que en el caso general, presentan también complicaciones. Para estos problemas el caso monomial admite técnicas combinatorias que pueden emplearse en conjunción con las técnicas algebraicas usuales para lograr algoritmos más eficientes.
  - Resoluciones, series de Hilbert, números de Betti, ...
  - Dimensión de Krull, dimensión proyectiva, regularidad, ...
  - Descomposición primaria e irreducible (para ideales monomiales la descomposición irreducible no redundante es única).

En cuanto al sistema de álgebra computacional sobre el que realizar este trabajo, el elegido fue CoCoA y, más en concreto, la librería CoCoALib [8]. Las razones para esta elección son las siguientes.

Al implementar nuevas funciones sobre el lenguaje de usuario de cualquier sistema de álgebra computacional (ya sea un sistema específico como CoCoA, Macaulay2 o Singular, o uno de propósito general, como Maple o Mathematica) el matemático tiene la ventaja de la sencillez del lenguaje y la disponibilidad de los tipos de datos y funciones matemáticas básicas sobre las que construir sus nuevos métodos. Sin embargo, el hecho de que el lenguaje sea interpretado hace que cuando se trata de experimentar sobre problemas de tamaño grande o de comparar el rendimiento de nuevos algoritmos con respecto a los ya existentes, la tarea sea poco menos que imposible. Los algoritmos programados sobre el lenguaje de usuario no pueden ser tan eficientes en términos de tiempo de ejecución como los implementados sobre el núcleo del sistema.

Por otro lado, programar sobre el núcleo del sistema es tarea normalmente difícil para el matemático. En primer lugar, porque los sistemas comerciales no brindan acceso al código, con lo que no podemos «extender» el sistema al mismo nivel que el núcleo. Además, el dominio del (los) lenguajes(s) de programación y el conocimiento del sistema en cuestión necesarios para hacer ese tipo de modificaciones está fuera del bagaje del matemático medio. Esto lleva a que, para implementar algoritmos competitivos, tenga que hacerlo «desde cero», con la carga ingente de trabajo que esto supone y la dificultad para que otras personas puedan usar los nuevos algoritmos directamente en su trabajo (a menos que los desarrolladores de alguno de los sistemas se decidan a incluirlo en él).

CoCoALib es una librería de C++ diseñada por el equipo de CoCoA para intentar cubrir este vacío. El lenguaje de programación a emplear será por tanto C++. En principio, para el uso de esta librería no son necesarios grandes conocimientos de C++, un conocimiento de los comandos más básicos es suficiente, y su aprendizaje es similar al de cualquiera de los lenguajes de usuario de los citados sistemas, cuya sintaxis está en muchas ocasiones inspirada en la de C.

CoCoALib nos permite usar, dentro de C++, las estructuras propias del álgebra conmutativa, como anillos, ideales, módulos,... así como muchas funciones para trabajar con ellos, tales como el cálculo de bases de Gröbner, funciones de Hilbert, etc. Por otro lado, el equipo de CoCoA ha diseñado un mecanismo (un programa en C++) llamado CoCoAServer que permite a CoCoA comunicarse con la librería CoCoALib, de modo que las funciones de ésta estén accesibles desde el sistema de álgebra computacional. Además, el mecanismo para hacer una función nueva de CoCoALib accesible desde CoCoA es sencillo y está descrito al detalle [9].

Con esto tenemos unidas las ventajas de poder «programar» nuestras funciones al nivel del núcleo del sistema (CoCoALib) y tenerlas accesibles como cualquier otra función nativa de CoCoA, de modo que puedan ser empleadas directamente por otros usuarios, puedan intervenir en programas realizados en CoCoA, y puedan ser comparadas con otras funciones del sistema.

2. Dramatis personae

Para realizar este plan de añadir nuevas funciones y tipos de datos a CoCoA y CoCoALib enfocadas al trabajo con ideales monomiales, se convocó un equipo de seis personas con diferentes perfiles. El equipo está formado por los desarrolladores de CoCoA y CoCoALib y por personas cuya investigación está directamente relacionada con los ideales monomiales, con distintos niveles de experiencia en programación. Damos a continuación una descripción algo más detallada de cada uno de los miembros del equipo.

John Abbott (Universidad de Génova, Italia): Desarrollador principal de CoCoA y CoCoALib. Autor del diseño de CoCoALib y de gran parte del código. Experto en C++ y en programación en general, así como en álgebra conmutativa, en particular en sus aspectos computacionales. Entre sus temas de investigación actuales están las bases de Gröbner y *border bases* aproximadas, sus propiedades y aplicaciones [1, 2].

Anna Bigatti (Universidad de Génova, Italia): Junto con John Abbott es desarrolladora principal de CoCoA y CoCoALib, experta en C++ y en programación, tiene una excelente carrera como investigadora en álgebra conmutativa, en la que destacan resultados relevantes como el Teorema de Bigatti-Hulett [5] y algunos algoritmos importantes, implementados en distintos sistemas, como son los del cálculo de la función y la serie de Hilbert, la dimensión o bases de Gröbner no homogéneas [6].

Óscar Fernández Ramos (Universidad de Valladolid): Estudiante de doctorado bajo la dirección de Ph. Gimenez. Con poca experiencia previa en el uso de CoCoA y ninguna en el uso de CoCoALib o C++. Su investigación se centra en el estudio de los invariantes de ideales monomiales asociados a grafos, en particular, sus números de Betti, relacionándolos con las caracerísticas combinatorias del grafo [10].

Eva García Llorente (Universidad de La Laguna): Estudiante de doctorado bajo la dirección de I. Bermejo. Con poca experiencia previa en el uso de CoCoA y ninguna en el uso de CoCoALib o C++. Es autora de un trabajo de investigación tutelado y una librería en Singular sobre algoritmos específicos para realizar cálculos con ideales monomiales [3].

Bjarke H. Roune (Universidad de Aarhus, Dinamarca): Autor del programa Frobby para hacer cálculos muy eficaces con ideales monomiales. Este programa ha sido conectado por él con CoCoALib. Experto en C++ y programación, es autor del algoritmo «Slice» para hacer diversos cálculos sobre ideales monomiales, en particular para obtener la descomposición irreducible de este tipo de ideales [13].

Eduardo Sáenz de Cabezón (Universidad de La Rioja): Usuario medio de CoCoA y CoCoALib, autor de algunas funciones. Su investigación trata principalmente sobre resoluciones de ideales monomiales, cálculo de invariantes y sus aplicaciones, en particular, al estudio de fiabilidad de sistemas y en la teoría formal de ecuaciones diferenciales [15, 16].

3. El desarrollo

Estas jornadas de trabajo se desarrollaron del lunes 1 al jueves 4 de febrero de 2010 y consistieron en dos actividades complementarias, una de tipo teórico y otra de tipo práctico:

3.1. Lunes: Charlas en el Departamento de Matemáticas y Computación

El primer día, los miembros del equipo de trabajo impartimos sendas charlas en el Departamento de Matemáticas y Computación de la Universidad de La Rioja con un doble objetivo: por un lado, difundir parte de la investigación que se hace actualmente sobre ideales monomiales, y por otro, conocer cada uno el trabajo de los otros miembros del equipo. Los títulos de las charlas impartidas fueron los siguientes: John Abbot y Anna Bigatti: *CoCoA and CoCoALib*; Óscar Fernández Ramos: *Betti diagram of the ideal associated to a simple graph*; Eva García Llorente: *Monomial ideals, basic algorithms and decompositions*; Bjarke Roune: *An algorithm on monomial ideals and simplicial complexes*; y Eduardo Sáenz-de-Cabezón: *Monomial resolutions, a partial survey.*

3.2. Martes a jueves: Cambios en la «estructura» de CoCoALib

De martes a jueves, los miembros del equipo de trabajo nos alojamos en una casa rural en un pequeño pueblo de montaña, *Ortigosa de Cameros*. La casa fue reservada de modo que fuésemos los únicos huéspedes alojados, junto con los dueños, que se encargaron de las tareas de limpieza y restauración, permitiéndonos concentrarnos en nuestra labor la casi totalidad del tiempo que allí estuvimos. El material técnico que trasladamos hasta el lugar consistió fundamen-

talmente en nuestros ordenadores portátiles, una selección de «libros de consulta» y el material ofimático necesario. El trabajo se desarrolló de manera continua durante los tres días. En resumen, el trabajo realizado fue el siguiente:

- Un ideal monomial es un ideal: El tipo «de trabajo» de ideales monomiales interno a CoCoALib es `PPVector`, que representa el sistema generador de un ideal monomial como un vector de monomios con una máscara que permite realizar rápidamente las dos operaciones básicas más comunes al tratar ideales monomiales: la división entre monomios y el test de divisibilidad entre dos monomios. Se decidió dar a este tipo una «interfaz» de tipo `ideal`, que es el tipo que CoCoALib usa para los ideales en general. Para ello se implementó un procedimiento interno `convert` para trabajar internamente con `PPVector` cuando el ideal sea monomial, aunque las funciones sean llamadas sobre ideales «a secas».
- Procedimiento para añadir funcionalidad a CoCoA y CoCoALib: en este aspecto, se mejoró el procedimiento para añadir funciones definidas por los usuarios a CoCoAServer, el programa que se encarga de la conexión entre CoCoA y CoCoALib. Se creó un fichero marco listo para poder agregar nuevas funciones simplemente añadiendo el código de las mismas, dándose ya hechos los procedimientos técnicos encargados de la conexión. También, el hecho de que en el equipo hubiera usuarios de sistemas operativos distintos (GNU-Linux, MacOS y Windows) permitió a los desarrolladores ver «en directo» las dificultades con que los usuarios pueden encontrarse al compilar la librería CoCoALib (que se desarrolla fundamentalmente en entornos MacOS y GNU-Linux). De este modo se implementaron algunas facilidades para la compilación[1]. En particular se trabajó en la compilación para el entorno Windows, usando Cygwin.
- Funciones básicas para el trabajo con ideales monomiales: En primer lugar se hizo un interfaz para algunas funciones ya existentes en CoCoALib y que tienen una implementación eficiente para ideales monomiales: `pertenencia`, `intersección`, `conjunto mínimo generador`, `ideal cociente`, `EsRadical?`, `radical`. Además, se implementaron nuevas funciones que no estaban todavía disponibles en CoCoALib: `EsPrimo?`, `EsIrreducible?`, `EsPrimario?`, `soporte`. También se incorporaron funciones nuevas a través de Frobby, tales como `Dimensión`, `primos asociados` y `descomposición primaria`, que se añadieron a las anteriormente disponibles en Frobby (`Dual de Alexander`, `Descomposición irreducible`, `monomios estándar maximales`).
- Finalmente se implementaron dos funciones nuevas para CoCoA-CoCoALib que usan las funciones optimizadas para ideales monomiales desarrolladas durante estos días. La primera, una función para el cálculo de ciclos inducidos de longitud impar en grafos a partir de las propiedades del ideal (monomial) asociado; esta función usa `conjunto mínimo generador`, `dual de Alexander` y

[1]Para la compilación de CoCoALib se precisa la librería numérica GMP. Además, de modo opcional, puede vincularse en la compilación la librería correspondiente al programa *Frobby* de B. Roune [14].

`primos asociados`. La segunda, una función para el cálculo de la regularidad de Castelnuovo-Mumford para cualquier ideal no necesariamente monomial (usando el algoritmo de Bermejo y Gimenez [4]); esta función utiliza `conjunto mínimo generador`, `soporte`, `Descomposición irreducible` y `dimensión`.

A modo de ejemplo mostramos el uso del procedimiento para calcular la intersección de dos ideales, una operación básica que se utiliza en multitud de otros cálculos. En el caso polinomial general, la intersección de dos ideales se calcula usando una variable auxiliar y calculando una base de Gröbner de un cierto ideal con un orden de términos que elimina la variable auxiliar. Aunque existen distintas alternativas, en cualquier caso han de usarse bases de Gröbner. En el caso monomial basta tomar el mínimo común múltiplo de cada generador del primer ideal con cada generador del segundo. Si se quiere el sistema generador monomial mínimo de la intersección, después habrá que eliminar los generadores redundantes. En el caso monomial podemos evitar pues el uso de bases de Gröbner, que puede tener un gran coste computacional.

La siguiente tabla muestra algunos tiempos de cálculo de dos intersecciones de ideales monomiales I y J:

CoCoA	*CoCoAMon*	*Singular-Poly*	*Singular-Mon*	*M2-Poly*	*M2-Mon*
–	0.0033/0.82	0.25/35.35	607.0	13.5	0.1033
–	0.016/2.06	7.36/–	–	1600.86	2.43

Los ideales usados en el primer ejemplo son ideales aleatorios en 50 variables, el conjunto mínimo de generadores de I consta de 50 generadores y el de J consta de 58. Los ideales del segundo ejemplo son ideales de 104 y 106 generadores en 100 variables. Los coeficientes de cada variable oscilan entre 0 y 50. Las columnas de la tabla indican el sistema de álgebra computacional usado. La columna *CoCoA* muestra los tiempos (en segundos) del comando general de CoCoA para ideales polinomiales. La columna *CoCoAMon* corresponde a nuestro algoritmo específico para ideales monomiales, los dos números que aparecen muestran el tiempo empleado sin y con eliminación de generadores redundantes respectivamente. La columna *SingularPoly* muestra los tiempos empleados por el algoritmo base de Singular (sin y con la opción de devolver la base mínima). La columna *Singular-Mon* muestra los resultados del mismo algoritmo especializado en ideales monomiales, pero implementado en el lenguaje interpretado de Singular, no a nivel del núcleo. Finalmente, las columnas *M2-Poly* y *M2-Mon* muestran los tiempos de los algoritmos del núcleo de Macaulay2, el general y el monomial. Este último es específico para ideales monomiales y está implementado a nivel del núcleo, de modo que esta columna junto con *CoCoA-Mon* es la que establece una comparación entre dos algoritmos equivalentes en dos sistemas distintos. Un guión en una celda significa que el sistema correspondiente no pudo realizar el cálculo. Todos los cálculos se realizaron sobre la misma máquina (procesador Intel Centrino de doble núcleo con sistema operativo GNU-Linux).

Los tiempos muestran claramente la ventaja de tener un algoritmo especializado en ideales monomiales sobre otro no específico, como queda patente al comparar los tiempos de las columnas *CoCoA* y *M2-Poly* con *CoCoAMon* y *M2-Mon*. En concreto se observa la mejora del algoritmo disponible en *CoCoA*, que no había

podido calcular los ideales del ejemplo. Por otro lado, los algoritmos *CoCoAMon* y *M2-Mon* están implementados a nivel del núcleo, mientras que *Singular-Mon* es una librería escrita en el lenguaje del sistema. La ventaja de los primeros sobre el segundo queda también clara. Así pues, la situación ideal se da cuando uno es capaz de programar sus funciones de modo específico y a nivel del núcleo del sistema, que fueron las razones por las que elegimos CoCoALib para esta semana monomial.

4. En conclusión

Aunque en principio el entorno y la forma de trabajo de esta semana era un tanto «inusual», la experiencia ha resultado muy satisfactoria. Es una buena forma de trabajo en cuanto que se propicia un ambiente muy productivo, y a la vez relajado, favoreciéndose un buen entorno para el trabajo conjunto y el intercambio de ideas directa o indirectamente relacionadas con el tema de trabajo. De hecho, durante los tres días de trabajo intenso, se fueron cumpliendo gran parte de los objetivos marcados e incluso algunos en principio no marcados.

Ortigosa de Cameros

El tema de trabajo suponía poner en marcha algo en parte nuevo, lo que implica tomar algunas decisiones, por ejemplo sobre el tipo de datos a usar para codificar ideales monomiales, el tipo de interfaz con las funciones nativas de CoCoA, tener en cuenta las posibles necesidades de los usuarios, el compromiso entre eficacia y sencillez... cuestiones que, al estar en un entorno «aislado» y «sin horarios», pudieron abordarse con calma dedicándoles el tiempo necesario. Así, se maduraron acuerdos sobre esos temas y se establecieron reglas comunes para la implementación de las nuevas funciones, el desarrollo de código, etc. Este tipo de acuerdos y decisiones han brindado al equipo la posibilidad de continuar un trabajo común con cierta estabilidad y coherencia.

Desde el punto de vista de los desarrolladores de CoCoA y CoCoALib, el trabajar directamente con usuarios de distintos niveles y en distintos entornos es extremadamente útil para no obviar las dificultades que quien no es experto se encuentra al manejar este tipo de librerías. Y esto tiene especial relevancia en el caso de CoCoA y CoCoALib, ya que la intención principal es ofrecer una manera sencilla de poder implementar funciones a nivel del núcleo del sistema.

Durante estos días hemos podido comprobar que los grandes esfuerzos hechos por el equipo de desarrollo en este sentido han sido muy acertados. Desde el punto de

vista del usuario, resulta interesante el comprobar que con un proceso de aprendizaje no mucho más complicado que el necesario para manejar un sistema de cálculo algebraico «normal» a nivel de usuario, se puede trabajar a nivel interno, obteniéndose resultados muy superiores.

El equipo está, pues, muy satisfecho, y se plantea seguir con nuevas jornadas de trabajo con una metodología similar, estudiando paquetes de funciones con temas específicos, como por ejemplo «resoluciones», «ideales de grafos», «ideales libres de cuadrado», etc.

Agradecimientos

Queremos agradecer al Centro de Investigación en Matemáticas, Informática y Estadística (CIME) de la Universidad de La Rioja el apoyo económico que ha permitido la celebración de estas jornadas de trabajo. Y también a Santiago y Teresa, los dueños de la casa rural «La Hoz del Encinedo» de Ortigosa de Cameros (La Rioja) por el magnífico trato y su contribución a que tuviéramos un ambiente de trabajo excepcional.

Referencias

[1] J. Abbott, A. Bigatti, M. Kreuzer y L. Robbiano, Computing ideals of points, *Journal of Symbolic Computation* **30 (4)** (2000), 341–356

[2] J. Abbott, C. Fassino y M.-L. Torrente, Stable border bases for ideals of points, *Journal of Symbolic Computation* **43 (12)** (2008), 883–894

[3] I. Bermejo, E. García-Llorente y Ph. Gimenez, `monomial.lib`, a library for computing with monomial ideals, Singular, 2009.

[4] I. Bermejo y Ph. Gimenez, Saturation and Castelnuovo-Mumford regularity, *Journal of Algebra* **303 (2)** (2006), 592–617.

[5] A. M. Bigatti, Upper Bounds for the Betti Numbers of a Given Hilbert Function, *Communications in Algebra* **21 (7)** (1993), 2317–2334.

[6] A. M. Bigatti, Computation of Hilbert-Poincaré series, *Journal of Pure and Applied Algebra* **119 (3)** (1997), 237–253.

[7] CoCoATeam, CoCoA: a system for doing Computations in Commutative Algebra, `http://cocoa.dima.unige.it`

[8] CoCoATeam, CoCoALib: a GPL C++ library for doing Computations in Commutative Algebra, `http://cocoa.dima.unige.it`

[9] `http://cocoa.dima.unige.it/cocoalib/doc/html/RegisterServerOps.html`

[10] O. Fernández-Ramos y Ph. Gimenez, First Nonlinear Syzygies of Ideals Associated to Graphs, *Communications in Algebra* **37 (6)** (2009), 1921–1933.

[11] D. R. Grayson y M. E. Stillman, Macaulay2, a software system for research in algebraic geometry, `http://www.math.uiuc.edu/Macaulay2/`

[12] G.-M. Greuel, G. Pfister y H. Schönemann, Singular 3-1-1 — A computer algebra system for polynomial computations, `http://www.singular.uni-kl.de`

[13] B. H. Roune, The Slice Algorithm for irreducible decomposition of monomial ideals, *Journal of Symbolic Computation* **44** (2009), 358–381.

[14] B. H. Roune, Frobby-Computations with monomial ideals, `http://www.broune.com/frobby/index.html`

[15] E. Sáenz de Cabezón, Multigraded Betti numbers without computing minimal free resolutions, *Applicable Algebra in Engineering, Communication and Computing* **20(5-6)** (2009), 481–495.

[16] E. Sáenz de Cabezón y H. P. Wynn, Betti numbers and minimal free resolutions for multi-state system reliability bounds, *Journal of Symbolic Computation* **44** (2009), 1311–1325.

[17] R. P. Stanley, *Combinatorics and Commutative Algebra*, Birkhäuser, 1996.

[18] R. Villarreal, *Monomial algebras*, Marcel Dekker, 2001.

Óscar Fernández-Ramos, Dpto. de Álgebra, Geometría y Topología, Universidad de Valladolid
Correo electrónico: `oscarf@agt.uva.es`

Eva García-Llorente, Dpto. de Matemática Fundamental, Universidad de La Laguna
Correo electrónico: `evgarcia@ull.es`

Eduardo Sáenz-de-Cabezón, Dpto. de Matemáticas, Universidad de La Rioja
Correo electrónico: `eduardo.saenz-de-cabezon@unirioja.es`
Página web: `https://belenus.unirioja.es/~esaenz-d`