UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES
CUAUTITLAN

POSGRADO EN CIENCIA E INGENIERIA
DE LA COMPUTACION

" LOS ALGORITMOS CONCRETOS Y LOS PROBLEMAS ALGORITMICOS GENERALES EN LA TEORIA DE ALGEBRAS DE LIE CUANTICAS "

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS

P R E S E N T A :

VICTOR ENRIQUE GONZALEZ VARGAS

DIRECTOR DE LA TESIS :

DR. VLADISLAV K. KHARTCHENKO

MEXICO / 2001

A mis dos hijas:

***Viviana** y **Karen** por su amor y paciencia.*

Agradecimientos:

A la Universidad Nacional Autónoma de México por su presencia y calidad académica.

Al CONACYT por el apoyo económico que me facilitó, sin el cual me hubiera sido muy difícil el dedicarme de tiempo completo al posgrado.

A la DGAPA de la UNAM por el apoyo económico que recibí por medio del PAPIIT IN-102599.

Al Dr. Vladislav K. Kharchenko por el tema propuesto y por su colaboración al desarrollo de esta tesis, y por hacerme participe en el PAPIIT IN-102599.

CONTENIDO

◆

RESUMEN DE TESIS

Actualmente la existencia de software dedicado específicamente a temas de álgebra abstracta y en especial a las álgebras de Lie cuánticas es muy pobre. En este trabajo se pretende contribuir al desarrollo de los sistemas computacionales relacionados con las teorías de álgebras de Lie cuánticas. El desarrollo de algoritmos y la construcción de programas específicos ayuda a la mejor comprensión y comprobación de algunos teoremas matemáticos y se convierten en una herramienta valiosa para los investigadores.

El capítulo 1 plantea los objetivos y expone la relación que ha existido y existe entre álgebra y algoritmos, menciona algunos requerimientos para poder tratar los problemas computacionales no numéricos, presenta la importancia que tienen los grupos cuánticos (matemáticamente) y las álgebras de Lie, y se comentan algunos de los trabajos que han sido realizados en estos campos.

El capítulo 2 trata los aspectos de la programación simbólica y para esto, menciona los lenguajes más utilizados, además presenta una breve e interesante historia del álgebra y los algoritmos, y menciona algunos de los trabajos que enfocan los problemas fundamentales de la teoría de algoritmos desde el punto de vista matemático. También trata los problemas de las palabras de semigrupos.

El capítulo 3 junto con los apéndices A y B contienen los conceptos básicos matemáticos y cuestiones de manejo de palabras y polinomios que se cree son de importancia para quienes se inicien por primera vez en los temas del álgebra computacional. Se dan las bases para definir las álgebras libres.

Casi puramente matemático, el capítulo 4 describe en forma concreta, a manera de introducción, los grupos cuánticos (las álgebras de Hopf) y sus relaciones de importancia con las álgebras de Lie. Se podría prescindir de éste capítulo, pero se presenta por contener temas relativamente novedosos que podrían ser de utilidad para investigaciones posteriores.

Un enfoque combinatorio para la cuantificación de las álgebras de Lie de la serie clásica se analiza en el capítulo 5. Aquí se dan también algunos teoremas de manejo de palabras para su trato computacional así como los sistemas de Groebner-Shirshov y los generadores de Poincaré-Birkhoff-Witt.

Finalmente en el capítulo 6 se presentan algunos algoritmos concretos de aplicación desarrollados en esta tesis y ejemplos de programas como resultados de las bases planteadas en los capítulos previos. Los listados de los programas se muestran en el apéndice C.

1

Introducción

1.1 Objetivos

- Planteamiento de los problemas algorítmicos generales en la teoría de álgebras de Lie cuánticas.
- Desarrollo de algoritmos concretos de palabras combinatorias y álgebras de Lie cuánticas.
- Presentar las relaciones existentes entre álgebra y algoritmos para su posible uso en los grupos cuánticos.
- Utilizar herramientas computacionales para la comprobación de teoremas matemáticos.

1.2 Consideraciones Preliminares

El proceso de realizar programas para computadoras digitales implica recompensa económica y científica, pero también puede ser una forma de creación artística al igual que componer poesía o música.

Para el desarrollo de los programas se deben considerar los siguientes puntos:

a) Saber como trabajan los programas almacenados en una computadora digital, saber como se almacenan dichos programas en la memoria de la máquina, y como se ejecutan las instrucciones. Tener experiencia previa con lenguajes de máquina puede ser muy útil.

b) Se requiere la habilidad de exponer las soluciones a los problemas en términos lo suficientemente explícitos como para que la computadora pueda entenderlos. (Éstas máquinas no tienen sentido común; y no han aprendido a "pensar", y hacen exactamente lo que se les dice, no más y no menos.)

c) De inicio se requiere conocer técnicas elementales de computación, tales como iteraciones (realizar un conjunto de instrucciones repetidamente), el uso de funciones y el uso de registros de índices.

d) Manejar los conceptos comunes de la jerga de computadoras, por ejemplo memoria, registros, bits, punto flotante, sobreflujo, etc.

Las computadoras han sido asociadas tradicionalmente con la solución de problemas numéricos tales como el cálculo de las raíces de una ecuación, interpolación e integración numérica, etc. En años recientes, sin embargo, se han realizado trabajos empleando las computadoras para

problemas esencialmente no numéricos, tales como ordenar (clasificar), traducción de lenguajes, resolver problemas matemáticos en alta álgebra y análisis combinatorio, prueba de teoremas, el desarrollo de software (programas para facilitar la escritura de otros programas), y la simulación de varios procesos de la vida diaria. En dichos problemas hay ocurrencia de números solo por coincidencia, y la habilidad de la toma de decisiones de la computadora, se emplea más que su habilidad para realizar aritmética.

Los resultados de investigaciones recientes en análisis no numérico están dispersos en numerosos boletines técnicos, y algunos de ellos al momento de escribirlos no tienen una nomenclatura estandarizada, por lo cual se dificulta su estudio.

El término "análisis no numérico" es un tanto negativo para este campo de estudio, el término "proceso de información" es muy extenso, y el término "técnicas de programación" está muy restringido, por lo cual es mejor emplear el término "algoritmos". Este nombre implica propiedades de algoritmos de computadora particulares.

Muchas de las técnicas expuestas aquí eventualmente podrán ser reemplazadas por otras mejores y se espera que este trabajo estimule investigaciones adicionales.

Por su extensión y dificultad, muchos aspectos matemáticos fundamentales y avanzados no se tratan aquí, pero se da la bibliografía más adecuada para quien, si lo desea, pueda ahondar en esta serie de conocimientos. Solo se tratan aquí algunos aspectos matemáticos importantes y se intenta dar introducciones claras y específicas de los mismos.

Existe una relación obvia entre las computadoras y las matemáticas en los campos del análisis numérico, la teoría de números, y la estadística, pero la conexión entre ambas materias es más profunda. La construcción de un programa de computadora a partir de un conjunto de instrucciones básicas es muy similar a la construcción de una prueba matemática a partir de un conjunto de axiomas. Además, los problemas de las matemáticas puras siempre han sido desarrollados históricamente a partir del estudio de problemas prácticos que surgen en otros campos, entre ellos el advenimiento de las computadoras. Algunos problemas de ese tipo son: (a) el estudio de las propiedades estocásticas de algoritmos particulares (la determinación de que tan bien se espera que se desempeñen); (b) la construcción de algoritmos óptimos, por ejemplo, para ordenar o evaluar polinomios; y (c) la teoría de lenguajes junto con aplicaciones interesantes de herramientas matemáticas a los problemas de programación; existen también aplicaciones de computadoras a la exploración de conjeturas matemáticas, por ejemplo, en análisis y en álgebra combinatoria; y en muchos de éstos casos existe una considerable interacción entre la programación y las matemáticas clásicas. Los intentos en la mecanización de las matemáticas son también muy importantes, ya que conducen a un mayor entendimiento de conceptos que se pensaba eran conocidos (hasta que hubo que explicárselo a una computadora). Seguramente las

conexiones entre computadoras y matemáticas puras las cuales ya se han enumerado, tendrán un incremento muy importante.

Se necesita un lenguaje para especificar cualquier algoritmo de computadora para lo cual se considera lo siguiente:

a) Desde un punto de vista, se admite que es más fácil escribir programas en lenguajes de programación de alto nivel, y es considerablemente fácil verificar los programas. Los lenguajes algebraicos se adaptan mejor a problemas numéricos que a aquellos de problemas no numéricos ya mencionados; aunque los lenguajes de programación se mejoran gradualmente, aún no son apropiados para algunas tareas como son entrada-salida, generación de números aleatorios, búsqueda combinatoria, recursión, y algunas instrucciones al nivel de máquina.

b) Un programador puede ser muy influenciado por el lenguaje en el cual escribe sus programas; hay una gran tendencia a preferir construcciones las cuales son las más simples en esos lenguajes, más que aquellas que son mejores para la máquina. Escribiendo en un lenguaje orientado a máquina, el programador tenderá a usar métodos mucho más eficientes; lo que es más cercano a la realidad.

c) Los programas que se requieren para este trabajo, son mas bien cortos, y será necesario un lenguaje de máquina para la salida de algunos programas.

Por lo anterior, se considera que el lenguaje "C++" es el más indicado para el desarrollo de los ejemplos aquí presentados por su flexibilidad y, ya que siendo un lenguaje de alto nivel y de vanguardia, también maneja instrucciones a nivel de máquina, además de que es el más utilizado para la creación de algunos paquetes que manejan álgebra como lo son MAPLE y MATHEMATICA, los cuales son de amplio uso para la solución de problemas algebraicos.

1.3 Introducción General

Los grupos de Lie, han tenido un profundo impacto en todas las áreas de las matemáticas, tanto puras como aplicadas, así como en física, ingeniería y otras ciencias basadas en las matemáticas. Las aplicaciones de los grupos de simetría continuos de Lie incluyen diversos campos: la topología algebraica, la geometría diferencial, la teoría invariante, la teoría de bifurcación, las funciones especiales, el análisis numérico, la teoría del control, la mecánica clásica, la mecánica cuántica, la relatividad, la mecánica continua y así sucesivamente.

Un grupo de simetría de un sistema de ecuaciones diferenciales es un grupo el cual transforma las soluciones del sistema a otras soluciones. En el marco clásico de Lie, estos grupos consisten en transformaciones geométricas del espacio de variables independientes o dependientes para el

sistema, y actúan sobre las soluciones transformando sus gráficos. Algunos ejemplos típicos de lo anterior son los grupos de traslaciones y rotaciones, así como grupos de simetrías escaladas, pero esto ciertamente no agota el rango de posibilidades. La gran ventaja de ver los grupos de simetría continua, opuesto a la simetría discreta, es que todos ellos pueden ser encontrados usando explícitamente métodos computacionales.

Las álgebras envolventes universales cuánticas aparecieron en los famosos artículos de Drinfel'd [27] y Jimbo [43]. Desde entonces varios artículos y número de monografías se han dedicado a su investigación. Todas estas investigaciones se relacionan principalmente con la cuantificación de álgebras de Lie de la serie clásica. Esto es tomado en cuenta primero por el hecho de que éstas álgebras de Lie tienen aplicaciones e interpretaciones visuales en especulaciones físicas, y luego por el hecho de que aún no se ha elaborado una noción general y comúnmente aceptada como estándar de un álgebra envolvente universal cuántica (ver discusión detallada en [4, 68]).

Recientemente han aparecido numerosos intentos para introducir una noción general de un álgebra envolvente universal cuántica. Un primer planteamiento de Cesar Bautista [4], muy interesante y con perspectiva, está basado en la idea de la generalización directa de la noción de superálgebras de Lie de color, y a la fecha resulta efectivo principalmente para la cuantificación de álgebras de Lie clásicas del tipo A_n. Un segundo planteamiento de V. Kharchenko [56] está basado en las ideas combinatoriales cuando se considera que un álgebra de Lie esta dada por generadores y las relaciones que la definen. De esta forma casi todas las cuantificaciones conocidas están reunidas en una noción. Para nuestros propósitos, es muy importante que este planteamiento permita aplicar métodos computacionales por medio de la construcción de sistemas de relaciones que las definen de Groebner-Shirshov y utilizar el así llamado Lema de Diamante de Bokut'-Bergman [10, 8].

Los grupos cuánticos surgieron primero en la literatura física, particularmente con el trabajo de L.D. Faddeev y la escuela de San Petersburgo [31, 32, 33, 34, 35], a partir del 'método de dispersión inversa' (inverse scattering method), el cual había sido desarrollado para construir y resolver sistemas cuánticos 'integrables'. Tales grupos cuánticos han experimentado gran interés en los pocos años transcurridos debido a sus conexiones inesperadas con tales, a primera vista, partes no relacionadas de las matemáticas como la construcción de invariantes de nudo y la teoría de la representación de grupos algebraicos de característica p.

En su forma original, los grupos cuánticos son álgebras asociativas cuyas relaciones que las definen se expresan en términos de una matriz de constantes (dependiendo del sistema integrable en consideración) llamadas una matriz-R cuántica. Se encontró independientemente por V.G. Drinfel'd [27] y M. Jimbo [43] alrededor de 1985 que éstas álgebras son álgebras de Hopf, las cuales en muchos casos, son deformaciones de 'álgebras envolventes universales' de álgebras de

Lie. Un poco después, Yu.I. Manin [72] y S.L. Woronowicz [100] independientemente construyeron deformaciones no conmutativas del álgebra de funciones en los grupos $SL_2(\mathbf{C})$ y SU_2, respectivamente, y mostraron que muchos de los resultados clásicos acerca de los grupos algebraicos y topológicos admiten analogías en el caso no-conmutativo. "Uno de los desarrollos más interesantes debido a V. Kharchenko, J. Keller, S. Rodríguez-Romo [51] y S. Rodríguez-Romo [88, 89] utiliza éstas ideas para investigar "simetrías cuánticas" y "Hamiltonianos cuánticos en sistemas físicos."

El descubrimiento y desarrollo de cualquier nuevo tipo de transformaciones de simetría y las correspondientes estructuras matemáticas conducen a un aumento en las habilidades para describir y explicar fenómenos físicos complicados. Las transformaciones de simetría basadas en los grupos de Lie y en las álgebras de Lie son las más conocidas y las más explotadas en todas las ramas de la física y otros campos de la ciencia. Pero los problemas complicados de la física fundamental han requerido diferentes generalizaciones y desarrollo adicional de la concepción de simetría: han aparecido transformaciones locales, supertransformaciones (supergrupos de Lie y superálgebras de Lie) y álgebras de Lie de dimensión infinita.

La teoría de los sistemas integrables cuánticos ha iniciado un nuevo tipo de simetría y objetos matemáticos llamados *grupos cuánticos*. Sin embargo, el nombre "cuántico" puede algunas veces causar confusión con la usual cuantificación *física*. Esto no es accidental y es verdaderamente significativo. El punto es que estos nuevos objetos se relacionan con los grupos de Lie usuales como la mecánica cuántica se relaciona con su límite clásico. Y matemáticamente el procedimiento de derivación de grupos cuánticos a partir de los grupos de Lie clásicos es bastante análogo al bien conocido método de cuantificación de sistemas clásicos. Realmente, este procedimiento es una generalización altamente no trivial de la cuantificación usual, la cual toma en cuenta las propiedades geométricas y topológicas de una variedad de grupo y una estructura de grupo sobre él. En un sentido, el papel de la geometría no conmutativa y de los grupos cuánticos en mecánica cuántica parece ser análoga a la geometría diferencial usual y los grupos de Lie en la formulación de la Relatividad General de Einstein: ambas concepciones matemáticas proporcionan un formalismo tan natural para la descripción del fenómeno físico que, prácticamente, la distinción entre "geometría" y "física" desaparece y uno habla de "dinámica geométrica" o física geométrica." En la Relatividad General esto es bien conocido desde el tiempo de su creación. Contrario a esto, el paso esencial para la geometrización amplia del fenómeno cuántico, la invención de los grupos cuánticos, se ha realizado muy recientemente, mucho después de la formulación de los principios básicos de la mecánica cuántica.

Sin embargo, el significado físico de una cuantificación de grupo puede ser absolutamente diferente y el parámetro correspondiente de la cuantificación el cual es usualmente denotado por la letra q, en general no tiene nada en común con la constante de Plank $\hbar$.

La formulación matemática con importantes contribuciones por V.G. Drinfel'd, L.D. Faddeev, S.L. Woronowicz, Yu.I. Manin, M. Jimbo, N.Yu. Reshetikhin, L.A. Takhtajan, P.P. Kulish, E. Sklyanin, M.A. Semenov-Tian-Shansky, J. Wess, B. Zumino, Y.S. Soibelman y muchos otros matemáticos y físicos han resultado en la muy rica y bella teoría de objetos (q-deformados) cuánticos: q-espacios, q-cálculo diferencial, q-supergrupos, grupos trenzados, etc.

Existen diferentes planteamientos a la construcción de grupos cuánticos: el planteamiento de Drinfel'd [28] altamente basado en la deformación de una estructura de Poisson grupo de Lie; el planteamiento de Faddeev-Reshetikhin-Takhtajan de matriz-R [36] el cual es en un sentido dual al de Drinfel'd; el planteamiento de Manin [71, 72] con el objeto inicial siendo un álgebra cuadrática sobre espacios lineales cuánticos y el planteamiento de Woronowicz [100, 101, 102] con un fondo esencialmente diferente de la teoría del álgebra $\mathbf{C}^*$. Recientemente han surgido nuevos enfoques por Z. Oziewicz [82] y M. Durdevich [29] los cuales podrían estar basados en las ideas de la lógica no estándar y la teoría de la representación de categorías especiales.

Así, aunque muchos de los documentos fundamentales sobre grupos cuánticos están escritos en el lenguaje de los sistemas integrables, sus propiedades son accesibles por otras técnicas convencionales, tales como la teoría de los grupos topológicos y algebraicos y las álgebras de Lie. La intención es presentar la teoría de grupos cuánticos desde este último punto de vista. De hecho, se concentra en el estudio de las 'álgebras de Lie' de grupos cuánticos, las cuales parecen ser el planteamiento que se ha probado como el más poderoso, particularmente en aplicaciones.

Hasta el presente, la mayoría de la información sobre algoritmos para las álgebras de Lie puede solamente ser encontrada buscando en los documentos de tesis de doctorados. Parte de este material es relativamente fácil de comprender y parte es difícil; los ejemplos son escasos y las descripciones de los desarrollos son algunas veces incompletas. Por otro lado, la construcción de algoritmos requiere un profundo entendimiento de las bases matemáticas de los algoritmos. Se ha encontrado que entender más profundamente los algoritmos se realiza mejor escribiendo descripciones detalladas de las matemáticas subyacentes.

Ha habido un gran surgimiento de paquetes de software poderosos para efectuar los más duros cálculos en álgebra. El concepto de una base de Groebner es ciertamente muy útil por sí misma, pero con la ayuda de una computadora, realmente se hace muy apreciable.

Actualmente tenemos la posibilidad de seleccionar entre los muchos sistemas para álgebra computacional de primera clase. Uno puede utilizar, ya sea los muy convenientes paquetes universales tales como MAPLE y MATHEMATICA, o los más especializados (y más efectivos) tales como MACAULAY, COCOA (álgebra conmutativa), GROEBNER (no conmutativa), BERGMAN (casos conmutativos y no conmutativos.) GAP (teoría de grupos) y así sucesivamente.

♦

2

Álgebra Computacional

2.1 Introducción

El deseo de emplear una computadora para realizar una computación matemática en forma simbólica surge naturalmente cada vez que se requiere una larga y tediosa secuencia de manipulación. Algunos trabajos matemáticos requieren páginas y páginas de manipulación algebraica y horas (o tal vez días) de tiempo. Esta computación podría haber sido desarrollada para resolver un sistema lineal de ecuaciones exactamente donde una solución por aproximación numérica no habría sido apropiada. O podría haber sido desarrollada para resolver la integral indefinida de una función bastante complicada para la cual se esperaba que alguna transformación pusiera la integral en alguna de las formas que aparecen en la tabla de integrales. En el último de los casos, podríamos haber tropezado con una transformación apropiada o podríamos habernos rendido sin saber si la integral pudiese ser expresada en términos de funciones elementales o no. O podría haber sido cualquiera de los otros numerosos problemas que requieren una manipulación simbólica.

La idea de emplear computadoras para computaciones no numéricas es relativamente antigua, pero el uso de computadoras para los tipos específicos de computación matemática simbólica mencionados arriba es un desarrollo bastante reciente. Algunas de las computaciones no numéricas incluyen procesos tales como: compilación de lenguajes de programación, procesamiento de palabra, programación lógica, o inteligencia artificial en su sentido más amplio. El presente trabajo se relaciona con el uso de la computadora para computaciones matemáticas específicas, las cuales son realizadas simbólicamente.

2.2 Computación Simbólica Frente a Computación Numérica

Quizá sea útil considerar un ejemplo que ilustre el contraste entre computación numérica y simbólica. Los polinomios de Chebyshev los cuales surgen en análisis numérico son definidos recursivamente como sigue:

$$T_0(x) = 1\,; \quad T_1(x) = x\,;$$
$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \qquad \text{para } k \geq 2\,.$$

Los primeros cinco polinomios de Chebyshev se listan en la tabla 2.1

Tabla 2.1 Los primeros cinco polinomios de Chebyshev

k	$T_k(x)$
0	1
1	x
2	$2x^2 - 1$
3	$4x^3 - 3x$
4	$8x^4 - 8x^2 + 1$

Como una típica computación numérica que involucra los polinomios de Chebyshev, suponga que se desea calcular los valores de los primeros cinco polinomios de Chebyshev a uno o más valores de la variable x. El programa en FORTRAN en la figura 2.1 podría ser usado para este propósito. Si la entrada para este programa es el número 0.30 entonces como salida se tendrán los primeros cinco números siguientes:

1.000 0.3000 −0.8200 −0.7920 0.3448

```
C   PROGRAMA PARA POLINOMIOS DE CHEBYSHEV
      REAL T(5)
C
      READ(5,1) X
      T(1) = 1.0
      T(2) = X
      WRITE(6,2) T(1), T(2)
      DO 10 N = 3, 5
         T(N) = 2.0 * X * T(N – 1) – T(N – 2)
         WRITE(6,2) T(N)
   10 CONTINUE
      STOP
C
    1 FORMAT(F5.2)
    2 FORMAT(F9.4)
      END
```

Figura 2.1. Programa en FORTRAN que involucra los polinomios de Chebysyev

Ahora suponga que quitamos la declaración READ en el programa de la figura 2.1. Esto, por supuesto, ocasiona un error en FORTRAN. Sin embargo, una interpretación razonable de éste programa sin la instrucción READ podría consistir en que los primeros cinco polinomios de Chebyshev serán computados en forma simbólica. Esta interpretación es precisamente la que se puede realizar en un lenguaje para computación simbólica. La figura 2.2 muestra un programa en uno de los primeros lenguajes de manipulación simbólica, ALTRAN. La salida del programa ALTRAN se presenta en la figura 2.3. Como lo indica este ejemplo, FORTRAN fue diseñado para manipular números mientras que ALTRAN fue diseñado para manipular polinomios.

```
PROCEDURE MAIN

    ALGEBRAIC (X:4) ARRAY (0:4) T
    INTEGER N

    T(0) = 1
    T(1) = X
    WRITE T(0), T(1)
    DO N = 2, 4
      T(N) = 2 * X * T(N - 1) - T(N - 2)
      WRITE T(N)
    DOEND

END
```

Figura 2.2 Programa en ALTRAN que involucra los polinomios de Chebyshev.

```
#T(0)
     1
#T(1)
     X
#T(2)
     2 * X ** 2 - 1
#T(3)
     X * (4 * X ** 2 - 3)
#T(4)
     8 * X ** 4 - 8 * X ** 2 + 1
```

Figura 2.3 Salida del programa en ALTRAN de la figura 2.2.

ALTRAN puede ser pensado como una variante de FORTRAN con la adición de una declaración extra, la declaración tipo "algebraica". De acuerdo con su nombre, ALTRAN, se deriva de ALgebraic TRANslator, siguiendo la convención del nombre de FORTRAN (derivado de FORmula TRANslator). También, siguiendo el espíritu de FORTRAN, ALTRAN fue diseñado para un modo de procesamiento por lotes. Más tarde, con el advenimiento de la calculadora moderna de mano, en la mitad de los sesentas, los sistemas de álgebra computacional, comenzaron a ser diseñados para uso interactivo, como un tipo de calculadora simbólica. Desde principios de los años setenta, casi todos los sistemas modernos de álgebra computacional han sido diseñados para uso interactivo. En la figura 2.4 se presenta un ejemplo de éste enfoque, mostrando una sesión interactiva corrida en uno de los sistemas modernos, MAPLE, para desarrollar la misma computación con los polinomios de Chebyshev.

```
> T[0] : = 1;
                    T[0] : = 1
> T[1] : = x;
                    T[1] : = x

> for n from 2 to 4 do
>    T[n] : = expand ( 2*x*T[n - 1] - T[n - 2] )
> od;
```

$$T[2] := 2x^2 - 1$$

$$T[3] := 4x^3 - 3x$$

$$T[4] := 8x^4 - 8x^2 + 1$$

Figura 2.4 Sesión de MAPLE involucrando los polinomios de Chebyshev.

2.3 Breve Historia de Álgebra y Algoritmos

El nacimiento y crecimiento de ambas ciencias álgebra y algoritmos están fuertemente interrelacionadós. Los orígenes de ambas disciplinas se remontan a Muhammed ibn-Mūsa al-Khwarizmi al-Quturbulli. Las contribuciones de al-Khwarizmi a las matemáticas arábicas y eventualmente occidentales (modernas) son múltiples: de él fue uno de los primeros esfuerzos por sintetizar las matemáticas axiomáticas griegas con las matemáticas algorítmicas hindúes. Los resultados fueron la popularización de los numerales hindúes, la representación decimal, los cálculos con símbolos, etc. Su tomo "al-Jabr wal-Muqabala", el cual fue traducido al latín por el inglés Robert of Chester bajo el título "Dicit Algoritmi", dio surgimiento a los términos álgebra (una corrupción de "al-Jabr") y algoritmo (una corrupción de la palabra "al-Khwarizmi").

Sin embargo, las dos ramas se desarrollaron en diferente proporción, entre dos diferentes comunidades. Mientras la disciplina de los algoritmos permaneció en su infancia suspendida por años, la rama del álgebra creció a una velocidad prodigiosa, y pronto dominó la mayoría de las matemáticas.

La formulación de la geometría en forma algebraica se facilitó con la introducción de la geometría de coordenadas por el matemático Descartes, y el álgebra capturó la atención de los matemáticos prominentes de la época. Los últimos diecinueve siglos contemplaron la teoría de funciones y el planteamiento topológico por Riemann, el planteamiento más geométrico de Brill y Noether, y el planteamiento puramente algebraico de Kronecker, Dedekind y Weber. El álgebra

creció más rica y más profunda, con el trabajo de muchos algebraístas ilustres y geómetras algebraicos.

Ahora, regresando a la ciencia de los algoritmos, vemos que aunque por muchos siglos hubo mucho interés por mecanizar el proceso de calcular, en ausencia de una computadora práctica, no hubo incentivos para estudiar los algoritmos de propósito general. En los 1670s, Gottfried Leibniz inventó su así llamada "Rueda de Leibniz", la cual podía sumar, restar, multiplicar y dividir.

Leibniz también buscó una *característica general*, un lenguaje simbólico, para ser utilizado en la traducción de métodos matemáticos y declaraciones en *algoritmos y fórmulas*. Muchas otras ideas de Leibniz influenciaron a las computadoras modernas, al cálculo y al razonamiento lógico.

A excepción de algunos otros desarrollos, la ciencia de la computación y los algoritmos permaneció mayormente abandonada en el último siglo. Escencialmente dos eventos dieron suspiro a éstas materias: Uno fue el estudio concerniente a los fundamentos de las matemáticas, como se estableció en el "programa de Hilbert", y este esfuerzo resultó en teoremas de incompletitud de Gödel [**41**], varios modelos computacionales adelantados por Church, Turing [**93**], Markov [**74**] y Post [**84, 85**], la interrelación de estos modelos, la existencia de una máquina "universal" y el problema de calculabilidad (*Entsheidungsproblem*). El otro evento fue el advenimiento de computadoras digitales de alta velocidad en el periodo de la posguerra. Durante la segunda guerra mundial, la viabilidad de una máquina de cálculo a gran escala fue demostrada por Colossus en el Reino Unido (bajo M.H.A. Newman) y la ENIAC en EE.UU. (bajo Von Newman, Eckert y Mauchly). Después de la guerra, se desarrollaron un gran número de más y más computadoras digitales potentes, iniciando con el diseño de la EDVAC en EE.UU. y Pilot ACE y DEDUCE en el Reino Unido.

Inicialmente los problemas manejados por éstas máquinas fueron puramente numéricos en su naturaleza, pero rápidamente se encontró que estas computadoras podían manejar objetos puramente simbólicos y hacer cálculos con ellos.

El siguiente gran paso fue la creación de lenguajes de programación de alto nivel en varias formas: como instrucciones, introducidas por Post; como producciones, introducidas independientemente por Chomsky y Backus; y como funciones, introducidas por Church. Estos lenguajes fueron rápidamente seguidos por el desarrollo de lenguajes más poderosos.

En paralelo, la ciencia del diseño y análisis de la complejidad de algoritmos combinatorios discretos ha crecido a una velocidad sin precedentes en las últimas tres décadas. Otras áreas tales como la geometría computacional, la teoría computacional de números, etc. han surgido en tiempos recientes, y han enriquecido la rama de los algoritmos. El campo del álgebra y la

geometría algebraica computacionales es relativamente nuevo, pero promete agregar una nueva dimensión a la rama de los algoritmos.

Después de un milenio, parece que las ramas de los algoritmos y el álgebra pueden converger finalmente y coexistir en una simbiosis fructífera. Uno debe preguntarse a sí mismo qué ejemplos pueden ser probados en una computadora, una pregunta que forza a uno a considerar los algoritmos concretos y tratar de hacerlos eficientes.

2.4 Sistemas de Álgebra Computacional

El campo del álgebra computacional ha ganado amplia atención en años recientes debido al incremento del uso de sistemas de cómputo algebraico en la comunidad científica. Estos sistemas, de los cuales los conocidos como mejores son DERIVE, MACSYMA, MAPLE, MATHEMATICA, REDUCE, y SCRATCHPAD, difieren marcadamente de los programas que existen para realizar únicamente cálculos científicos numéricos. A diferencia de los últimos programas, los sistemas de álgebra computacional pueden manipular objetos simbólicos matemáticos además de cantidades numéricas.

Ejemplos de otros sistemas de propósito especial los cuales aparecieron durante la década de los ochentas incluyen FORM por J. Vermaseren, para cálculos de física de alta energía, Lie, por A.M. Cohen para cálculos de álgebra de Lie, MACAULAY, por Michael Stillman, un sistema especialmente construido para computaciones en Geometría Algebraica y Álgebra Conmutativa, y PARI por H. Cohen en Francia, un sistema orientado principalmente para cálculos de teoría de números. Como para la mayoría de los nuevos sistemas de los ochentas, estos últimos dos están escritos en lenguaje C para portabilidad y eficiencia.

2.5 Problemas Fundamentales de la Teoría de Algoritmos [57]

El concepto matemático de algoritmo fue elaborado a mediados de la década de los treinta por D. Hilbert, K. Gödel, A. Church, S.C. Kleene, E.L. Post, A.M. Turing en dos formas. La primera solución se fundamentó en la noción de *función recursiva*, mientras que la segunda utilizó una clase de *proceso de cálculo* definido exactamente.

K. Gödel [41] con la ayuda de las ideas de D. Hilbert definió formalmente la clase de todas las funciones recursivas.

A. Church en 1936 comenzó con otras suposiciones completamente diferentes y obtuvo *exactamente la misma* clase de funciones. Esto le permitió formular la hipótesis de que la clase de las funciones recursivas es igual a la clase de funciones computables intuitiva y completamente

definidas. Actualmente esta hipótesis se conoce como la **tesis de Church** la cual establece que *existe un algoritmo para calcular los valores de una función si y solo si la función es recursiva.*

Esta tesis no puede ser demostrada ya que el concepto de algoritmo no es formal, mientras que el concepto de función recursiva es formal y matemáticamente correcto.

Esta tesis es considerada como una definición de una función algorítmicamente calculable.

Por medio de esta idea A. Church resolvió el problema algorítmico fundamental del predicado lógico y demostró que *no existe un algoritmo para definir si una proposición dada de primer orden es verdadera o falsa.*

Otro planteamiento fue desarrollado independientemente por E.L. Post [84] y A.M. Turing [93]. Este planteamiento está basado en la noción de la *máquina abstracta* de Post-Turing que esencialmente copia el trabajo de una computadora humana.

Las computadoras modernas también trabajan como algunas máquinas de Post-Turing. Por lo tanto la teoría de las máquinas de Turing-Post es muy importante para las ciencias de la computación. Es muy interesante e importante que la clase de funciones que son calculables por las máquinas de Turing-Post coincide con la clase de funciones recursivas. Este es otro argumento que apoya la tesis de Church.

En 1900 Hilbert propuso 23 problemas muy importantes. Entre ellos estaba el problema número 10 que fue formulado de la siguiente manera:

Sea una ecuación Diophantie dada con un número arbitrario de variables y coeficientes enteros. Es necesario encontrar un algoritmo que nos permita verificar en un número finito de pasos si la ecuación dada tiene soluciones enteras.

En ese tiempo todavía no había sido desarrollada la teoría de algoritmos, por lo tanto, en este problema solo fue posible hablar de la construcción del algoritmo en sentido intuitivo. Las grandes dificultades que han aparecido en las investigaciones de las ecuaciones con coeficientes enteros y sus soluciones enteras han llevado a la hipótesis de que no existe tal algoritmo. Esta hipótesis inesperada fue demostrada por Ju.V. Matijasevic [75], con la ayuda de los resultados de M. Davis, H. Putnam y J. Robinson [26], de la teoría moderna de los algoritmos.

Otro problema fundamental, el problema de la *igualdad de las palabras* para grupos, fue resuelto por P.S. Novikov [80] por medio de los métodos de la teoría de algoritmos. Este fue el primer problema que fue resuelto con la ayuda de la teoría de algoritmos y que surgió en las matemáticas independientemente de la lógica y de la teoría de algoritmos. Otras soluciones de este problema fueron obtenidas por W. Boone [15], J.L. Britton y G. Higman [42]. Todas estas soluciones son complicadas y por lo tanto no serán consideradas aquí. En lugar de eso se considerará el problema de la *igualdad de las palabras* para semigrupos.

El problema de la igualdad de palabras para semigrupos también tiene una solución negativa que fue encontrada por E.L. Post [85] y A.A. Markov [74].

Un *semigrupo* es un conjunto de elementos con una operación binaria asociativa. Por ejemplo el conjunto de los números naturales con la adición o con la multiplicación son semigrupos. Otro ejemplo muy importante es el **semigrupo libre.**

Si se considera al conjunto de variables

$$X = \{ x_1, x_2, \ldots, x_n \}$$

como un alfabeto, el conjunto de todas las palabras no vacías en este alfabeto forma un semigrupo con la multiplicación ordinaria de palabras $u \cdot v = uv$, por ejemplo:

$$x_1x_2x_1 \cdot x_1x_3x_3 = x_1x_2x_1x_1x_3x_3 = x_1x_2x_1^2x_3^2 \ .$$

Este semigrupo se dice que es un semigrupo libre con los generadores libres $x_1, x_2, \ldots, x_n$.

Ahora se considera la manera de definir un semigrupo con la ayuda de los *generadores y las relaciones definidas.*

Si se tiene un conjunto de igualdades formales de la forma

$$w_1 = u_1 \ , \ \ w_2 = u_2 \ , \ldots , \ w_k = u_k \ . \tag{2.5.1}$$

Una *transformación elemental* de una palabra V es una sustitución de la palabra u_i en la palabra V en lugar de la subpalabra w_i, o viceversa, una sustitución de la palabra w_i en la palabra V en lugar de la subpalabra u_i. Por ejemplo, si se considera el alfabeto con cinco letras $C = \{a, b, c, d, e\}$ y el sistema de siete relaciones formales

$$\begin{gathered} ac = ca \ , \ \ ad = da \ , \ \ bc = cb \ , \ \ bd = db \ , \\ eca = ce \ , \ \ edb = de \ , \ \ cca = ccae \ . \end{gathered} \tag{2.5.2}$$

entonces podemos tener algunos ejemplos de las transformaciones elementales de las palabras como son

$$ae\underline{ca}cadb \ \rightarrow \ aea\underline{cc}adb \ \rightarrow \ aeaccae\underline{db} \ \rightarrow \ aeaccade \ .$$

Dos palabras V, W se dice que son iguales en el semigrupo definido por las relaciones (2.5.1) si existe una secuencia de transformaciones elementales de la forma

$$V \rightarrow V_1 \rightarrow V_2 \rightarrow \ldots \rightarrow V_m \rightarrow W .$$

Por ejemplo en el semigrupo definido por las relaciones (2.5.2) las palabras $aecacadb$ y $aeaccade$ son iguales.

El problema de palabras de semigrupos tiene la siguiente forma:

Para el semigrupo dado encontrar un algoritmo que para cada dos palabras V, W indique si son iguales o no en el semigrupo.

Con la ayuda de las máquinas de Turing, E.L. Post y A.A. Markov han demostrado que *existen semigrupos que tienen el problema de palabras sin solución algorítmica.* Es posible demostrar que el semigrupo definido por las relaciones (2.5.2)

$$ac = ca,\ \ ad = da,\ \ bc = cb,\ \ bd = db$$
$$eca = ce,\ \ edb = de,\ \ cca = ccae\,.$$

es también un problema de palabras sin solución algorítmica [**92**].

2.5.1 Álgebra calculable.

El conjunto de funciones recursivas incluye a los tres tipos siguientes de las funciones calculables más sencillas

$$\mathbf{s}(x) = x + 1\,; \qquad \mathbf{o}(x) = 0\,; \qquad I_m^n(x_1, \ldots, x_n) = x_m\,,$$

y se define con la ayuda de los siguientes tres principales operadores:

1. Composición de funciones.
2. Operador de recursión primitiva.
3. Minimización.

Composición de funciones

$$\frac{\begin{array}{c} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_m(x_1, x_2, \ldots, x_n) \\ f(z_1, z_2, \ldots, z_m) \end{array}}{g(x_1, x_2, \ldots, x_n) = f(f_1(x_1, x_2, \ldots, x_n), f_2(x_1, x_2, \ldots, x_n), \ldots, f_m(x_1, x_2, \ldots, x_n))\,.}$$

Operador de la recursión primitiva

$g(x_1, x_2, \ldots, x_n)$
$h(x_1, x_2, \ldots, x_n, z_{n+1}, z_{n+2})$

La función $f(x_1, x_2, \ldots, x_n, y)$ *aparece por recursión primitiva* si y sólo si

$f(x_1, x_2, \ldots, x_n, 0) = g(x_1, x_2, \ldots, x_n)$,
$f(x_1, x_2, \ldots, y+1) = h(x_1, x_2, \ldots, x_n, y, f(x_1, x_2, \ldots, x_n, y))$

Este operador tiene la siguiente denotación:

$$f = \mathbf{R}(g, h)\,.$$

Ejemplos de funciones primitivamente recursivas:

1. $f(x, y) = x + y$.

$$x + 0 = x = I_1^1(x)$$
$$x + (y + 1) = (x + y) + 1 = S(x + y),$$
$$(g(x) = I_1^1,\ h(x, y, z) = S(z)).$$

2. $f(x, y) = xy$.

$$x \cdot 0 = \mathbf{o}(x)$$
$$x(y + 1) = xy + x,$$
$$(g(x) = \mathbf{o}(x), h(x, y, z) = z + x).$$

3. $f(x, y) = x^y$.

$$x^0 = \mathbf{1}$$
$$x^{y+1} = x^y x,$$
$$(g(x) = \mathbf{1}, h(x, y, z) = z \cdot x).$$

Minimización

$$f(x_1, \ldots, x_n)$$

Si se considera una ecuación de la forma

$$f(x_1, \ldots x_{n-1}, y) = x_n.$$

denotar

$\mu_y(f(x_1, \ldots x_{n-1}, y) = x_n)$ = mínima α, tal que
$f(x_1, \ldots x_{n-1}, \alpha) = x_n$.

se tiene una nueva función

$$\mu_y(f(x_1, \ldots x_{n-1}, y) = x_n) = g(x_1, \ldots, x_n)$$

que se obtiene de f *por el operador de minimización.*

Definición. Una función f se llama *recursiva* si se obtiene de **s**, **o**, I_m^n por un número finito de composiciones, recursiones primitivas y minimizaciones.

Tesis de Church. *Existe un algoritmo para calcular los valores de una función si y solo si la función es recursiva.*

♦

3

Conceptos Básicos del Álgebra Computacional y la Teoría de Lie

3.1 Conceptos Introductorios

A Conjunto. Un conjunto es una colección de objetos que no necesariamente tienen cualquier estructura o propiedades adicionales. Por ejemplo, una colección de n naranjas o bananas constituye un conjunto. Así n personas, así n puntos.

Para enlazar el álgebra y la geometría, se estudian los polinomios sobre un campo. Todos sabemos que son los polinomios, pero el término *campo* puede no ser familiar. La intuición básica es que un campo es un conjunto donde uno puede definir la adición, la substracción, la multiplicación, y la división con las propiedades usuales. Los ejemplos estándares son los números reales $\mathbb{R}$ y los números complejos $\mathbb{C}$, mientras que los enteros $\mathbb{Z}$ no son un campo ya que la división falla (3 y 2 son enteros pero su cociente 3/2 no lo es). Una definición formal de campo se encuentra en el apéndice A.

Una razón por la que los campos son importantes es que el álgebra lineal trabaja sobre *cualquier* campo. Así, aun si los cursos de álgebra lineal restringen a los escalares a caer en $\mathbb{R}$ o $\mathbb{C}$, la mayoría de los teoremas y técnicas se aplican a un campo arbitrario k.

Necesitamos tratar los polinomios en n variables $x_1, \ldots, x_n$ con coeficientes en un campo arbitrario k.

Definición 3.1.1. Un **monomio** *en* $x_1, \ldots, x_n$ *es un producto de la forma*

$$x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n},$$

donde todos los exponentes $\alpha_1, \ldots, \alpha_n$ *son enteros no negativos. El* **grado total** *de estos monomios es la suma* $\alpha_1 + \ldots + \alpha_n$.

Podemos simplificar la notación de los monomios como sigue: sea $\alpha = (\alpha_1, \ldots, \alpha_n)$ una n-túpla de enteros no negativos. Entonces establecemos

$$x^{\alpha} = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n}.$$

Cuando $\alpha = (0, \ldots, 0)$, observe que $x^{\alpha} = 1$. También $|\alpha| = \alpha_1 + \ldots + \alpha_n$ denota el grado total del monomio x^{α}.

Definición 3.1.2. *Un* **polinomio** *f en $x_1, \ldots, x_n$ con coeficientes en k es una combinación lineal finita (con coeficientes en k) de monomios. Escribiremos un polinomio f en la forma*

$$f = \sum_{\alpha} a_\alpha x^\alpha, \qquad a_\alpha \in k,$$

donde la suma es sobre un número finito de n-túplas $\alpha = (\alpha_1, \ldots, \alpha_n)$. El conjunto de todos los polinomios en $x_1, \ldots, x_n$ con coeficientes en k se denota $k[x_1, \ldots, x_n]$.

Cuando se trata con polinomios en un pequeño número de variables, usualmente se prescinde de los subíndices y se utilizan diferentes letras para las variables.

En esta sección se emplea la siguiente terminología.

Definición 3.1.3. *Sea $f = \sum_\alpha a_\alpha x^\alpha$ un polinomio en $k[x_1, \ldots, x_n]$.*

(i) *Llamamos a_α el* **coeficiente** *del monomio x^α.*

(ii) *Si $a_\alpha \neq 0$, entonces llamamos $a_\alpha x^\alpha$ un* **término** *de f.*

(iii) *El* **grado total** *de f, denotado* grad(f), *es la máxima $|\alpha|$ tal que el coeficiente a_α es no cero.*

La suma y el producto de dos polinomios es otra vez un polinomio. Decimos que un polinomio f *divide* a un polinomio g dado $g = fh$ para algún $h \in k[x_1, \ldots, x_n]$. Se puede mostrar que, bajo la adición y la multiplicación, $k[x_1, \ldots, x_n]$ satisface todos los axiomas del campo excepto por la existencia de inversos multiplicativos (porque, por ejemplo, $1/x_1$ no es un polinomio). Tal estructura matemática se llama un anillo conmutativo (ver apéndice A), y por esta razón nos referiremos a $k[x_1, \ldots, x_n]$ como un *anillo polinomial.*

3.2 Formas de Definir Álgebras de Dimensión Infinita

La forma más confortable de trabajar directamente con un álgebra es, después de todo, dando una base y una tabla de multiplicación sobre ella. Por ejemplo, esta es la forma de definir el álgebra *polinomial* y un *álgebra asociativa libre* (o sea el álgebra de polinomios sobre las variables no conmutativas). La base en estas dos álgebras está formada por los monomios (en el caso de álgebras libres, son en efecto llamadas a menudo palabras), mientras que la tabla de la multiplicación se da en la forma natural ya que el producto de monomios es otra vez un monomio. Por ejemplo, el producto del monomio xy por sí mismo, será el monomio $xyxy$, pero en la primera álgebra se podría escribir en la forma x^2y^2.

Con toda esta comodidad, este método de representación no es empleado tan frecuentemente, porque requiere más o menos las reglas de multiplicación del mismo tipo, debido a la infinidad de la "tabla de multiplicar". En la práctica, una situación mucho más frecuente se da cuando no es

posible describir aun la uniformidad de la base, sin mencionar las leyes de la multiplicación. En un caso, sin embargo, esta forma es utilizada regularmente; teniendo en mente las álgebras de grupo y de semigrupo.

Algunas veces es más confortable considerar, en lugar de un grupo G, su *álgebra de grupo* $K[G]$ sobre un campo K, el cual puede ser definido de la siguiente forma: los elementos de G son declarados para formar su base y la ley de la multiplicación en G (tabla de Cayley) no es más que la tabla de multiplicación en $K[G]$. Para evitar ambigüedad (especialmente cuando la ley de la multiplicación es escrita aditivamente) frecuentemente usaremos, en lugar de los elementos g, para la base, los símbolos e_g indexados por los elementos del grupo.

Por ejemplo, para el grupo $\mathbb{Z}$ de enteros, el álgebra de grupo tendrá la base e_i y la ley de la multiplicación $e_i e_j = e_{i+j}$ $(i, j \in \mathbb{Z})$.

Uno generalmente trata de definir las álgebras descriptivamente y directamente a través de sus propiedades que las definen. Esta es la característica definición "física".

En la mayoría de los casos tal descripción nos habilita para obtener una descripción de la base y la tabla de la multiplicación. Sin embargo, en este proceso, "la intuición física" se pierde y se hace un esfuerzo para no usar esta base hasta el inicio de los cálculos directos.

Un planteamiento más es una reducción a los clásicos o casi clásicos objetos. Por ejemplo muchos grupos infinitos surgen como grupos de matriz. Las álgebras de dimensión infinita también pueden ser vistas como álgebras de matriz, pero de dimensión infinita.

Otra forma es introducir nuevas operaciones sobre los objetos clásicos. Aquí están ejemplos de ésta clase (que por sí mismos se han vuelto clásicos).

Podemos hacer un álgebra de Lie A_L desde un álgebra asociativa A, introduciendo una nueva multiplicación: $[ab] = ab - ba$. Es fácil ver que satisface las identidades definidas para un álgebra de Lie:

$$[ab] = -[ba] \qquad \text{(la identidad de anti-conmutatividad)},$$

$$[[ab]c] + [[bc]a] + [[ca]b] = 0 \qquad \text{(la identidad de Jacobi)}.$$

3.3 Enfoque Combinatorio

La forma más importante de definir álgebras para nosotros consiste en describirlas en términos de generadores y definiendo relaciones. El método de generadores y relaciones es similar al método axiomático en la miniatura, donde el papel de los axiomas es realizado por las relaciones. Primero consideremos un ejemplo y entonces demos una definición exacta.

Asumamos que estamos estudiando un álgebra (asociativa) A definida por tres generadores a, b, c y las tres relaciones:

$$2ab - c = 0\,; \quad 2bc - a = 0\,; \quad 2ca - b = 0\,.$$

¿A qué se parece esta álgebra? Esta es un álgebra que automáticamente tiene tres elementos dados y también todos los productos posibles generados por ellos (por ejemplo cba, a^3 etc.), generalmente llamados palabras. Los productos de palabras se definen en la forma natural, digamos $ba \cdot ac = baac$ (o ba^2c en la forma abreviada). Sin embargo, algunas de éstas palabras son linealmente dependientes o más aun iguales. ¿Cuáles? Naturalmente aquellas que están incluidas en las relaciones que las definen, digamos $2ab = c$. Sin embargo, las otras relaciones derivables de las relaciones que las definen no son excluidas. Por ejemplo, si la igualdad $2ab = c$ se multiplica por c por la derecha, entonces obtenemos la igualdad $2abc = c^2$. Substituyendo $2bc$ por a por el lado izquierdo obtenemos la igualdad $a^2 = c^2$. Podemos obtener la igualdad $a^2 = b^2$ análogamente. Estas dos relaciones se siguen de las relaciones que las definen. Por lo tanto, cuando escribimos $A = \langle a,b,c \mid 2ab = c, 2bc = a, 2ca = b \rangle$, entonces tenemos en mente que, no solamente estas relaciones que las definen se cumplen en A, sino también aquellas que son sus consecuencias. Así esta propiedad permite el uso efectivo del método dado para definir álgebras. Sin embargo la efectividad en definir álgebras tiene su otro lado también. El hecho es que definiendo álgebras de este modo, muchas preguntas perfectamente naturales se vuelven no triviales y a menudo sin solución. Por ejemplo al tratar de obtener una respuesta a la pregunta ¿cuál es la dimensión del álgebra A definida arriba? Es difícil decir aun si A es de dimensión infinita o, por el otro lado, no-cero.

Es aun más difícil contestar preguntas acerca de los elementos concretos. Por ejemplo, ¿los elementos que representan las palabras ab y ba son iguales o diferentes en el álgebra A? No obstante, en la mayoría de los casos importantes es posible obtener respuestas satisfactorias a preguntas fundamentales.

Regresemos a la definición formal de un álgebra A definida por los generadores $x_1, x_2, \ldots, x_g$ y las relaciones que la definen

$$f_1 = 0\,, f_2 = 0\,, \ldots, f_r = 0\,.$$

(Ambos el conjunto generador así como el conjunto de las relaciones, generalmente hablando, pueden ser infinitos, pero ya que no hay diferencias importantes, utilizaremos solamente variantes finitas por conveniencia.)

Así que, consideremos un *álgebra libre* $\mathfrak{U}$ con el conjunto de generadores $X = \{x_1, x_2, \ldots, x_g\}$. Hacemos notar que estos elementos son polinomios de variables no conmutables x_i y la base consiste de palabras (monomios).

En particular todos los elementos f_j son elementos de esta álgebra. Podemos considerar el ideal I en ll generado por estos elementos (o sea, el ideal más pequeño que los contiene).

El álgebra definida tiene la propiedad universal correspondiente: para cada álgebra B con el mismo conjunto de generadores para la cual se satisfacen las mismas relaciones (y posiblemente algunas otras más también), existe un homomorfismo único A a B fijando los generadores.

Si el conjunto de generadores es finito, entonces el álgebra se llama *finitamente generada* y, además, el conjunto de relaciones que la definen es finito, entonces el álgebra se llama *finitamente presentada.*

3.4 Módulos y el Producto Tensor

A Módulos. Recuerde que un espacio vectorial V se llama un *módulo* sobre un álgebra A (o un módulo-A) si cada elemento del álgebra actúa como un operador lineal, mientras que la suma y el producto de los elementos son asignados a la suma y al producto de los operadores correspondientes. En otras palabras, para cada $a \in A$ y para cada $v \in V$, su producto $v * a$ está definido (y se llama la acción de a sobre v) y todas las propiedades naturales de linealidad y distributividad así como la de la asociatividad característica $(v * a) * b = v * (ab)$ para $a, b \in A$ se satisfacen. La noción de un módulo es una generalización natural de la noción de un espacio vectorial: si tomamos a A para ser el campo K, entonces un módulo-K es exactamente un espacio vectorial.

B El producto tensor. Recuerde que si V y W son dos espacios vectoriales con las bases e_1, e_2, ... y $f_1, f_2,$... respectivamente, entonces su producto tensor $V \otimes W$ es un espacio vectorial con la base denotada por $e_i \otimes f_i$. Aqui, si $v = \sum \alpha_i e_i \in V$ y $w = \sum \beta_j f_j \in W$, entonces la combinación lineal $\sum_{i,j} \alpha_i \beta_j e_i \otimes f_j$ es de otro modo denotada por $v \otimes w$, de esta manera las propiedades naturales de linealidad y distributividad del siguiente tipo son válidas: $a(v \otimes w) = av \otimes w = v \otimes aw$; $(a \in K)$; $(v_1 + v_2) \otimes w = v_1 \otimes w + v_2 \otimes w$ y la notación no depende de la elección de una base. Observe que no cada elemento en $V \otimes W$ es de la forma $x \otimes y$.

Si A y B son álgebras, entonces su producto tensor $A \otimes B$ puede ser también provisto con la estructura de un álgebra con la siguiente multiplicación:

$$(a \otimes b)(a' \otimes b') = aa' \otimes bb'.$$

3.5 Palabras Normales y una Base de Groebner de un Ideal de un Álgebra Libre

En el trabajo con un álgebra definida por generadores y definiendo relaciones, intuitivamente tendemos a trabajar con el lenguaje de las palabras, o sea los elementos de un álgebra libre. En esta sección se introducen las nociones de palabras normales y la descomposición de un álgebra libre en un ideal y su complemento normal. Después se introduce una noción importante de la base de Groebner (y su noción equivalente de sistema completo de relaciones).

3.5.1 Notación Básica, Grado y Orden.

Sea $\mathfrak{U} = K\langle X\rangle$ un álgebra asociativa libre con la unidad y S el conjunto de todas las palabras en el alfabeto X (incluyendo la palabra vacía Λ, la cual será identificada con la unidad 1). Para abreviar notación se generaliza la noción ordinaria de potencia. Si $f, g \in S$, entonces denotamos por grad $_f g$ el número de ocurrencias diferentes de la palabra f dentro de la palabra g. Por ejemplo, grad $_{xx}xxx = 2$, grad $_{xy}yx = 0$. Si $F \subseteq S$ es algún conjunto de palabras, entonces denotamos grad $_F g = \Sigma_{f \in F}$ grad $_f g$. Por ejemplo grad $_X g = |g|$ es la longitud de g. Para generalizar estas definiciones, asumamos que el conjunto de palabras en S está bien ordenado (o sea cada dos palabras diferentes son comparables y tenemos una posibilidad de inducción en el orden >) y el orden > se preserva después de la multiplicación:

$$f \geq g\ ;\ h \geq k \quad \Rightarrow fh \geq g\,k\ ;\ hf \geq kg\ .$$

La palabra más pequeña debe ser siempre la unidad.

A Palabra líder. Cada polinomio no conmutativo f en $x_1, \ldots, x_n$ es una combinación lineal de palabras $f = \Sigma \alpha_i u_i$. Denotamos por $\overline{f}$ una palabra líder la cual ocurre en esta descomposición con un coeficiente no-cero. En el caso general la palabra líder de un producto no es igual al producto de las palabras líderes de los factores. Por ejemplo, si $f = x_1 + x_1x_2$ y $g = x_3 + x_3x_2$ entonces $\overline{fg} = x_1x_2x_3 \neq x_1x_3 = \overline{f}\,\overline{g}$. Pero si la palabra líder de f no es el inicio de cualesquiera otra palabra en f, entonces

$$\overline{fg} = \overline{f}\,\overline{g}\ . \tag{3.5.1}$$

Verdaderamente, las desigualdades $\overline{f} > u_j$ se pueden multiplicar por la derecha por (posiblemente distintos) elementos $\overline{f}v_k > u_jv_s$. En particular, si f es un polinomio homogéneo, esto es, que todas las palabras u_i tienen la misma longitud, entonces la fórmula (3.5.1) es verdadera.

El conjunto de todas las palabras no es completamente ordenado ya que existen cadenas decrecientes infinitas — por ejemplo,

$$x_1 > x_1^2 > x_1^3 > \ldots > x_1^n > \ldots \tag{3.5.2}$$

pero, todos sus subconjuntos finitos son completamente ordenados. Esto nos permitirá emplear inducción sobre la palabra líder, dado que las restricciones son puestas en las longitudes de las palabras, $l(v)$, o en los grados del polinomio bajo consideración.

B Orden lexicográfico. En los ejemplos concretos de esta sección, el orden será como sigue: las palabras se ordenan primero por su longitud, y, si las longitudes son las mismas, entonces lexicográficamente. Llamamos a este orden *lexicográfico homogéneo*. Ahora, con cada elemento no-cero $u \in \mathfrak{U}$ podemos asociar su palabra líder $\hat{u}$ (en el orden >) y podemos extender > a un orden (parcial) sobre $\mathfrak{U}$: $u > v \Leftrightarrow \hat{u} > \hat{v}$. Además, si $U \subseteq \mathfrak{U}$, entonces $\hat{U} = \{\hat{u} : u \in U\}$ y si $v \in U$, entonces ponemos

$$\text{grad}_U v = \text{grad}_{\hat{U}} \hat{v} .$$

Por ejemplo, $\text{grad}_X v$ es la potencia ordinaria y

$$\text{grad}_{\{x^2-x,\, xy-yx\}} x^3y - x = 3 .$$

De hecho, el principal ejemplo del uso de la noción introducida de potencia es la igualdad $\text{grad}_U v = 0$, la cual es simplemente una forma corta del hecho de que ninguna de las palabras líderes de los elementos de U es una subpalabra de la palabra líder del elemento v.

3.5.2 Palabras Normales. Descomposición de un Álgebra Libre en un Ideal y su Complemento Normal.

Sea I un ideal del álgebra libre $\mathfrak{U}$, la cual consideraremos fija.

Definición 3.5.1. Una palabra $s \in S$ se llama *normal* (módulo ideal I), si s no es el término líder de cualquier elemento en I. La condición equivalente es $\text{grad}_I s = 0$. Denotemos por N la cubierta lineal del conjunto de palabras normales y le llamamos el *complemento normal del ideal* I. El nombre se justifica por lo siguiente:

Teorema 3.5.1. *La siguiente descomposición de suma directa de espacios vectoriales sostiene:* $\mathfrak{U} = N \oplus I$.

Prueba. Es obvio que $I \cap N = 0$. En consecuencia, es suficiente probar por inducción sobre > la representabilidad de cada palabra s en la forma $\overline{s} + y$, donde $\overline{s} \in N$, $y \in I$. Si s es normal, entonces $\overline{s} = s$, de otro modo, s es la palabra líder del elemento $u \in I$. Sea $u = \alpha s + v$. Entonces, por inducción, v es representable en la forma $\overline{v} + y$, por consiguiente $s = \alpha^{-1}(u - v) = \alpha^{-1}\overline{v} + \alpha^{-1}(u - y)$, la cual es la representación deseada.

Definición 3.5.2. Para cada $u \in \mathfrak{U}$, su *forma normal* $\overline{u}$ está definida para ser su imagen por la proyección natural $\mathfrak{U} \to N$. Claramente, $\overline{u} = 0 \Leftrightarrow u \in I$.

Corolario 3.5.1. *Definimos una nueva operación sobre N estableciendo* $s * t = \overline{st}$. *Entonces, N, junto con la operación introducida es isomórfica al álgebra factor* $A = \mathfrak{U}/I$.

De este modo vemos que, para trabajar con A en el marco del álgebra libre $\mathfrak{U}$, es necesario poder encontrar sus palabras normales y saber como reducir una palabra arbitraria a su forma normal. Desafortunadamente, este problema generalmente hablando, algorítmicamente no tiene solución, al igual que un problema más simple no tiene solución: para construir un algoritmo que determine si dos palabras dadas son iguales en el álgebra factor (o, equivalentemente, si ellas tienen la misma forma normal). Este último problema, llamado el *problema de la igualdad* para palabras es irresoluble aun para la siguiente álgebra concreta completa $a, b, c, d, e \mid ac = ca, ad = da, bc = cb, bd = db, eca = ce, edb = de, cca = ccae$.

Uno de los más efectivos enfoques a este problema, descubierto en varias formas por una serie de autores (Bergman [**8**], 1978), (Buchberger[**18**], 1983), (Bokut'[**10**], 1976), (Ufnarovskij [**96**], 1980), (Shirshov [**95**], 1962) es objeto de atención.

3.5.3 Base de Groebner. Sistema Completo de Relaciones.

Definición 3.5.3. Un subconjunto G del ideal I se llama una *base de Groebner* si, para toda $v \in I$, se satisface lo siguiente:

$$\text{grad}_G v > 0 .$$

El valor de la base de Groebner se muestra en el siguiente

Teorema 3.5.2. *Una palabra* s *es normal si y sólo si*

$$\text{grad}_G s = 0 .$$

Prueba. Por un lado, $G \subseteq I$ implica $\text{grad}_G s \leq \text{grad}_I s$. Por otro lado, si $\text{grad}_v s > 0$, $v \in I$, $\text{grad}_g v > 0$, $g \in G$, entonces obviamente también $\text{grad}_G s \geq \text{grad}_g s > 0$.

Existen, generalmente hablando, muchas bases de Groebner (por ejemplo el ideal por sí mismo es uno de ellos), sin embargo existe siempre un *mínimo* en el sentido de que ningún subconjunto de él es una base de Groebner. No es difícil ver que la condición de minimidad es equivalente a la declaración que, para cada $v \in G$ $\text{grad}_{G \setminus v} v = 0$ se conserva (esto es ninguna de las palabras líderes es una subpalabra de otra). Si además, la condición más fuerte es satisfecha, a saber que cada elemento $v \in G$ tiene la forma $f - \overline{f}$, donde $f = \hat{v}$ es la palabra líder, entonces la base se llama *reducida*. La base reducida es determinada únicamente y es fácil construirla iniciando con un mínimo, normalizándolo (para hacer que el coeficiente con el término líder sea uno) y reduciendo todas las palabras no líderes con la ayuda de la base misma en la forma normal. El reto es como construir una base mínima, y después de resolver ese problema, generalmente asumiremos que todas las bases de Groebner son reducidas.

Trabajando con el lenguaje del álgebra factor $A = \mathfrak{U}/I$, a menudo es más confortable hablar de relaciones, así introducimos un sinónimo más, llamando a un conjunto de relaciones $f_i = v_i$ el *sistema completo de relaciones* sobre A, si f_i son palabras, $f_i > v_i$ y $\{f_i - v_i\}$ es una base de Groebner.

Señalamos que una base de Groebner depende de la elección de los generadores X así como de la relación de orden $>$ y puede cesar de ser tal si una u otra se cambia. También señalamos que una base de Groebner siempre genera un ideal.

3.5.4 Reducción. Composición. El Lema de Composición.

Asumiendo que un ideal I es generado por un conjunto R, para obtener una base de Groebner, comenzando con R (y consecuentemente un sistema completo de relaciones del álgebra factor $\mathfrak{U}/I$), necesitamos transformar a R utilizando tres operaciones-estados.

I. *Normalización*. Es la sustitución de cada elemento mediante un proporcional, de la forma $f - w$, donde f es la palabra líder, $f > w$. En otras palabras, el coeficiente con la palabra líder se hace en 1. Después de que todos los elementos han sido normalizados, podemos ir al segundo estado.

II. *Reducción*. Si u y v son elementos normalizados, tal que grad $_v u > 0$, entonces sea $\hat{u} = g\hat{v}h$ la ocurrencia de la palabra líder de v en la palabra líder de u. Entonces por reducción significa la substitución de u por el elemento, obtenido a través de la normalización del elemento $u - gvh$. Tal vez sea más legible en el lenguaje de relaciones:

$$\left.\begin{array}{l} u = f - w \\ \\ v = k - l \end{array}\right\} \Rightarrow \begin{array}{c} f = gkh \\ \| \quad \| \\ w = glh \end{array}$$

Observamos que el elemento reducido es ya sea cero (en cuyo caso lo podemos quitar sin dificultad), o también, es más pequeño que el elemento con el que iniciamos. Esto garantiza que una serie de reducciones termina tarde o temprano. Si todas las reducciones han sido completadas (o sea $\forall\, u \in U$ grad $_{U \setminus u} u = 0$), nos podemos mover al tercer estado.

Una posible no unicidad de los resultados de éste estado, dependiendo del orden y el lugar de las reducciones, se vuelve sin importancia, después de todo. Una consideración análoga se aplica también al tercer estado.

III. *Composiciones*. La composición de un par u, v de elementos normalizados es una palabra f, tal que $\hat{u}$ es su comienzo y $\hat{v}$ es su final, donde las ocurrencias de las subpalabras declaradas intersectan. En otras palabras, $f = x \cdot y \cdot z$, donde $xy = \hat{u}$, $yz = \hat{v}$, $y \neq 1$.

El elemento obtenido normalizando el elemento $xv - uz$ se llama el *resultado de la composición.* Para claridad, un diagrama en el lenguaje de las relaciones: $u = xy - w$; $v = yz - l \Rightarrow$

$$\begin{array}{c} xyz \\ // \quad \backslash\backslash \\ wz = xl \end{array}$$

En el tercer estado, los resultados de todas las composiciones no consideradas deberían estar contiguos a U (observamos que inclusive un par puede dar varias composiciones) y regresar al estado de reducción.

El resultado de número infinito de repeticiones del segundo y el tercer estado es una base de Groebner mínima ya que se mantiene el siguiente:

A Lema Sobre Composición. *Si el conjunto U es tal que* $\forall\ u \in U$ grad $_{U \setminus u} u = 0$ *(o sea ninguna de las palabras lideres de los elementos* $u \in U$ *es una subpalabra de otra) y el resultado de cualquier composición se reduce a cero después de pocos pasos, entonces U es una base de Groebner mínima.*

Otro nombre para este lema es el lema del diamante; su prueba en varias formas puede encontrarse en cualquiera de los siguientes: (Bokut' [**10**], 1976), (Anick [**1**], 1986), (Bergman [**8**], 1978), (Ufnarovskij [**96**], 1980).

3.6 Bases de Groebner en el Caso Conmutativo

Una base de Groebner de un álgebra conmutativa, también puede ser infinita. Tiene por lo tanto sentido en este caso examinar todas las nociones consideradas e introducir otra base, la cual también llamamos una base de Groebner. Antes que todo, como punto de inicio, consideramos no un álgebra libre, sino el álgebra de polinomios $K[x_1, \ldots, x_n]$. Su base consiste del conjunto ordenado de palabras de la forma $x_1^{a_1} \ldots x_n^{a_n}$, el cual llamamos monomio.

El conjunto de monomios está provisto con un orden lineal >, pero las condiciones sobre él no son tan rígidas como en el caso no conmutativo: los requerimientos son únicamente que la unidad sea el elemento mínimo y que el orden se conserva después de la multiplicación: $f > g \Rightarrow fh > gh$. El buen ordenamiento no es necesario – en su lugar el teorema de la base de Hilbert se utiliza con éxito (Bokut', L'vov, Kharchenko [**11**], 1988); el uso del orden lexicográfico puro se permite en el caso cuando, digamos x es mayor que cada potencia de y.

La noción del grado grad $_f g$ es redundante aquí, y podemos manejarlo completamente con la noción de divisibilidad. Si $A = K[X] / I$, entonces un monomio normal es un monomio no igual

a cualquiera de los monomios líderes de los elementos de I. Por supuesto, $K[X] = N \otimes I$, cuando N es la envoltura lineal del conjunto normal de monomios. Un subconjunto G del ideal se llama su base de Groebner, si su conjunto F de monomios líderes tiene la propiedad que el monomio líder de cada elemento del ideal es divisible al menos por una palabra de F. A diferencia del caso no conmutativo, una base de Groebner mínima siempre es finita. El algoritmo de su construcción va junto con el mismo esquema, pero con algunas simplificaciones. Normalizando, como al principio, denota la sustitución de un elemento por uno proporcional, en el cual el coeficiente con la palabra líder equivale a uno. Si la palabra líder $\hat{u}$ de un elemento normalizado u es divisible por la palabra líder $\hat{v}$ de un elemento normalizado v: $\hat{u} = \hat{v}h$, entonces la reducción de u por v denota el reemplazo de u por el elemento obtenido después de normalizar $u - vh$. El papel de la composición f de dos elementos normalizados u y v es desempeñado por mínimo común múltiplo de las palabras líderes. Si $f = \hat{u}h$ y $f = \hat{v}g$, entonces el resultado de la composición se obtiene normalizando la diferencia $uh - vg$. Se observa que no es necesario considerar composiciones de elementos entre ellos mismos así como composiciones que son productos de palabras líderes, su resultado, en cualquier proporción se reduce a cero. Existen suficientemente muchos documentos dedicados al caso conmutativo – el primer conocimiento con el que es más confortable iniciar es un artículo de Buchberger [18] (Buchberger, 1983). Sin embargo, aquí se discute una de las más importantes aplicaciones de la base de Groebner en el álgebra conmutativa. El hecho es que hoy en día, el proceso de construcción de las bases de Groebner es una de las formas más universales de resolver sistemas no lineales de ecuaciones polinomiales, no únicamente manualmente, sino también por computadoras.

3.7 Grupos de Lie

Un grupo de Lie es un "grupo" el cual es también una "variedad". Los grupos surgen como una abstracción algebraica de la noción de simetría; un ejemplo importante es el grupo de rotaciones en el plano o espacio tridimensional. Las variedades, las cuales forman los objetos fundamentales en el campo de la geometría diferencial, generalizan los conceptos familiares de las curvas y superficies en el espacio tridimensional. En general, una variedad es un espacio que localmente se parece a un espacio euclidiano, pero cuyo carácter global podría ser totalmente distinto. La conjunción de estas dos al parecer ideas matemáticas dispares combina, y extiende significativamente, los métodos algebraicos de la teoría de grupos y el cálculo de multi-variable utilizado en la geometría analítica. Esta teoría resultante, particularmente las poderosas técnicas infinitesimales, puede entonces aplicarse a una amplia variedad de problemas físicos y matemáticos.

3.7.1 Variedades.

Una variedad n-dimensional es un espacio conexo, localmente compacto, con una base contable, cada punto del cual tiene una vecindad homeomorfa al espacio n euclidiano. Ejemplos son las curvas simples cerradas (variedad unidimensional) y la superficie de una esfera o de un toro (variedades bidimensionales). Un problema fundamental de la topología es (resuelto solo para $n = 1, 2$) clasificar las variedades n-dimensionales en tipos tales que las dos variedades sean homeomorfas si, y sólo si pertenecen al mismo tipo.

A Variedad suave es una variedad con una estructura adicional que permite hablar de diferenciabilidad o suavidad de funciones de valores reales; a estas variedades también se les conoce como variedades diferenciables. Con este planteamiento, puede definirse la noción de transformación suave entre variedades suaves y puede introducirse la diferencial de una transformación suave, la cual es una generalización de la derivada.

Los objetos tales como las ecuaciones diferenciales, los grupos de simetría, etc., están definidos sobre subconjuntos abiertos del espacio euclidiano $\mathbb{R}^m$. Las características geométricas subyacentes de estos objetos serán independientes de cualquier sistema de coordenadas particular sobre el subconjunto abierto el cual podría ser utilizado para definirlas, y se hace de gran importancia para liberarnos de la dependencia de coordenadas locales particulares, así que nuestros objetos serán esencialmente "libres de coordenadas". Más específicamente, si $U \subset \mathbb{R}^m$ es abierto y $\psi : U \to V$, donde $V \subset \mathbb{R}^m$ es abierto, es cualquier difehomorfismo, significando que ψ es una transformación diferenciable infinitamente con inversa diferenciable infinitamente, entonces los objetos definidos sobre U tendrán contrapartes equivalentes sobre V. Aunque la fórmula precisa para el objeto sobre U y su contraparte sobre V, en general, cambiará, las propiedades subyacentes esenciales permanecerán iguales. Una vez que nos hemos liberado de ésta dependencia en las coordenadas, es un pequeño paso a la definición general de una variedad suave. Desde este punto de vista, las variedades proveen las condiciones naturales para estudiar objetos que no dependen de coordenadas.

Definición 3.7.1. Una *variedad m-dimensional* es un conjunto M, junto con una colección contable de subconjuntos $U_\alpha \subset M$, llamados *cartas de coordenadas*, y funciones uno a uno $\chi_\alpha : U_\alpha \to V_\alpha$ en subconjuntos abiertos conectados $V_\alpha \subset \mathbb{R}^m$, llamados *transformación de coordenadas locales*, los cuales satisfacen las siguientes propiedades:

(a) Las cartas de coordenadas cubren M:

$$\bigcup_\alpha U_\alpha = M \ .$$

(b) Sobre el traslape de cualquier par de cartas de coordenadas $U_\alpha \cap U_\beta$ la transformación compuesta

$$\chi_\beta \circ \chi_\alpha^{-1} : \chi_\alpha(U_\alpha \cap U_\beta) \to \chi_\beta(U_\alpha \cap U_\beta)$$

es una función (infinitamente diferenciable) suave.

(c) Si $x \in U_\alpha$, $\tilde{x} \in U_\beta$ son puntos distintos de M, entonces existen subconjuntos abiertos $W \subset V_\alpha$, $\tilde{W} \subset V_\beta$, con $\chi_\alpha(x) \in W$, $\chi_\beta(\tilde{x}) \in \tilde{W}$, satisfaciendo

$$\chi_\alpha^{-1}(W) \cap \chi_\beta^{-1}(\tilde{W}) = \varnothing .$$

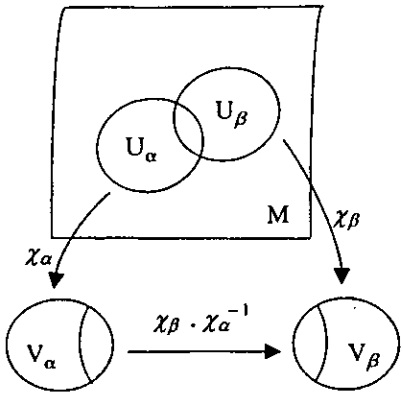


Figura 3.7.1 Cartas de coordenadas de una variedad

Las cartas de coordenadas $\chi_\alpha : U_\alpha \to V_\alpha$ dotan a la variedad M con la estructura de un espacio topológico. A saber, requerimos que cada subconjunto abierto $W_x \subset V_\alpha \subset \mathbb{R}^m$, $\chi_\alpha^{-1}(W_x)$ sea un subconjunto abierto de M. Estos conjuntos forman una *base* para la topología sobre M, así que $U \subset M$ es abierto si y sólo si para cada $x \in U$ hay una vecindad de x de la forma anterior contenida en U; así $x \in \chi_\alpha^{-1}(W) \subset U$ donde $\chi_\alpha : U_\alpha \to V_\alpha$ es una carta de coordenadas que contiene a x, y W es un subconjunto abierto de V_α. En términos de esta topología, el tercer requisito en la definición de una variedad es sólo un reacondicionamiento del axioma de separación de Hausdorff: si $x \neq \tilde{x}$ son puntos en M, entonces existen conjuntos abiertos U conteniendo x y $\tilde{U}$ conteniendo $\tilde{x}$ tales que $U \cap \tilde{U} = \varnothing$.

El grado de diferenciabilidad de las funciones traslapadas $\chi_\beta \circ \chi_\alpha^{-1}$ determina el grado de suavidad de la variedad M. Las *variedades suaves*, en las cuales las funciones que se traslapan son suaves, significan C^∞, difeomorfismos o subconjuntos abiertos de $\mathbb{R}^m$. Si requerimos que las funciones traslapadas $\chi_\beta \circ \chi_\alpha^{-1}$ sean funciones analíticas reales, entonces M se llama una *variedad analítica*. La mayoría de los ejemplos clásicos de variedades de hecho son analíticos. Alternativamente, podemos disminuir los requerimientos de diferenciabilidad y considerar *variedades-C^k*, en las cuales las funciones que se traslapan solo requieren tener derivadas continuas hasta de orden k.

3.7.2 Grupos de Lie.

El punto de inicio para el estudio de Sophus Lie [**40, 90**] de grupos continuos fue un estudio de ecuaciones diferenciales parciales. Sea

$$\frac{\partial x^i}{\partial \tau} = u^i(x)$$

la ecuación diferencial parcial para un proceso de flujo. Esta ecuación tiene la forma

$$x^i(\tau) = f^i(\tau; x_0).$$

Esta representa la posición de una partícula específica en el tiempo τ si su posición original en $\tau = 0$ es x_0,

$$f^i(\tau = 0, x_0) = x_0^i.$$

Para cada valor de τ se asocia una transformación del espacio de los puntos x en sí misma. Las transformaciones para todos los valores posibles de τ forman un grupo.

A primera vista, un grupo de Lie parece ser como una unión natural entre el concepto algebraico de un grupo y la noción de geometría diferencial de una variedad. Sin embargo, como se verá, ésta combinación de álgebra y cálculo conduce a poderosas técnicas para el estudio de simetría las cuales no están disponibles para los grupos finitos. Comenzamos recordando la definición de un grupo abstracto.

Definición 3.7.2. Un *grupo* es un conjunto G junto con una operación de grupo, generalmente llamada multiplicación, tal que para cualesquiera dos elementos g y h de G, el producto $g \cdot h$ es otra vez un elemento de G. Se requiere que la operación de grupo satisfaga los siguientes axiomas:

(1) *Asociatividad.* Si g, h y k son elementos de G, entonces

$$g \cdot (h \cdot k) = (g \cdot h) \cdot k.$$

(2) *Elemento identidad.* Existe un elemento distinguido e de G, llamado el elemento identidad, el cual tiene la propiedad de que

$$e \cdot g = g = g \cdot e$$

para toda g en G.

(3) *Inversos.* Para cada g en G hay un inverso, denotado g^{-1}, con la propiedad

$$g \cdot g^{-1} = e = g^{-1} \cdot g.$$

Antes de proceder a los grupos de Lie, discutimos un par de ejemplos elementales de grupos los cuales dan alguna idea de las características que distinguen los grupos de Lie del tipo más general de grupos.

Ejemplo 3.7.1. (a) Sea $G = \mathbb{Z}$, el conjunto de enteros, con la adición siendo la operación de grupo. Claramente la asociatividad se satisface, el elemento identidad es 0 y el "inverso" de un entero x es $-x$.

(b) Similarmente $G = \mathbb{R}$, el conjunto de los números reales, es también un grupo con la adición como la operación de grupo. Otra vez 0 es la identidad, y $-x$ la inversa del número real x. En los dos casos la operación de grupo es conmutativa: $g \cdot h = h \cdot g$ para $g, h \in G$. Tales grupos son llamados *abelianos*; ellos forman solamente una pequeña subclase de todo el rango de posibilidades para grupos.

(c) Sea $G = GL(n, \mathbb{Q})$, el conjunto de matrices invertibles $n \times n$ con entradas de números racionales. La operación de grupo está dada por la multiplicación de matriz. El elemento identidad es, por supuesto, la matriz de identidad I, la inversa de una matriz A es la ordinaria matriz inversa, la cual otra vez tiene entradas racionales.

(d) Similarmente, $GL(n, \mathbb{R})$, el conjunto de todas las matrices invertibles $n \times n$ con entradas reales es un grupo bajo la multiplicación de matrices, la identidad y la inversa siendo las mismas que en el ejemplo previo. Por brevedad, generalmente denotaremos el *grupo lineal general* $GL(n, \mathbb{R})$ por solo $GL(n)$.

La característica que distingue a un grupo de Lie es que también lleva la estructura de una variedad suave, así los elementos de grupo pueden ser continuamente variados. Así en cada par de arriba de ejemplos de grupos, el segundo caso es un grupo de Lie ya que también es una variedad suave. Para $\mathbb{R}$, la estructura de variedad es clara. Como para el grupo lineal general, se puede identificar con el subconjunto abierto

$$GL(n) = \{X : \det X \neq 0\}$$

del espacio $\mathbf{M}_{n \times n}$ de todas las matrices $n \times n$. Pero $\mathbf{M}$ es isomórfica a $\mathbb{R}^{n^2}$, las coordenadas son las entradas x_{ij} de X. Así $GL(n)$ es también una variedad n^2-dimensional. En ambos casos la operación de grupo es suave (de hecho analítica). Esto conduce a la definición general de un grupo de Lie.

Definición 3.7.3. Un *grupo de Lie de parámetro-r* es un grupo G el cual lleva también la estructura de una variedad suave r-dimensional en tal forma que ambas la operación de grupo

$$m : G \times G \to G, \qquad m(g, h) = g \cdot h, \qquad g, h \in G,$$

y la inversión

$$i : G \to G, \qquad i(g) = g^{-1}, \qquad g \in G,$$

son transformaciones suaves entre variedades.

Ejemplo 3.7.2. Aquí discutimos un par de ejemplos de grupos de Lie además de los dos ya presentados.

(a) Sea $G = \mathbb{R}^r$, con la obvia estructura de variedad, y sea la operación de grupo la adición vectorial $(x, y) \mapsto x + y$. El "inverso" de un vector x es el vector $-x$. Ambas operaciones son claramente suaves, así que $\mathbb{R}^r$ es un ejemplo de un grupo de Lie abeliano de parámetro-r.

(b) Sea $G = \mathrm{SO}(2)$, el grupo de rotaciones en el plano. En otras palabras

$$G = \left\{ \begin{pmatrix} \cos\theta & -\operatorname{sen}\theta \\ \operatorname{sen}\theta & \cos\theta \end{pmatrix} : 0 \le \theta < 2\pi \right\},$$

donde θ denota el ángulo de rotación. Observe que podemos identificar a G con el círculo unitario

$$S^1 = \{(\cos\theta, \operatorname{sen}\theta) : 0 \le \theta < 2\pi\}$$

es $\mathbb{R}^2$, la cual sirve para definir la estructura de variedad sobre SO(2). Si incluimos reflexiones obtenemos el grupo ortogonal

$$\mathrm{O}(2) = \{X \in \mathrm{GL}(2) : X^T X = \mathrm{I}\}.$$

Tiene la estructura de variedad de dos copias desconectadas de S^1.

Un *homomorfismo* de grupo de Lie es una transformación suave $\phi : G \to H$ entre dos grupos de Lie la cual respeta las operaciones de grupo:

$$\phi(g \cdot \tilde{g}) = \phi(g) \cdot \phi(\tilde{g}), \qquad g, \tilde{g} \in \mathrm{G}.$$

Si ϕ tiene una inversa suave, determina un *isomorfismo* entre G y H.

3.7.3 Corchetes de Lie.

La operación más importante sobre campos vectoriales es el corchete de Lie o conmutador. Este se define más fácilmente en términos de sus acciones como derivaciones de funciones. Específicamente, si $\mathbf{v}$ y $\mathbf{w}$ son campos vectoriales en M, entonces su *corchete de Lie* $[\mathbf{v}, \mathbf{w}]$ es el campo vectorial único que satisface

$$[\mathbf{v}, \mathbf{w}](f) = \mathbf{v}(\mathbf{w}(f)) - \mathbf{w}(\mathbf{v}(f)) \qquad 3.7.1$$

para todas las funciones suaves $f : M \to \mathbb{R}$. Es fácil verificar que $[\mathbf{v}, \mathbf{w}]$ es efectivamente un campo vectorial.

Proposición 3.7.1. *Los corchetes de Lie tienen las siguientes propiedades*:

(a) Bilinealidad

$$[c\,\mathbf{v} + c'\,\mathbf{v}', \mathbf{w}] = c[\mathbf{v}, \mathbf{w}] + c'[\mathbf{v}', \mathbf{w}],$$
$$[\mathbf{v}, c\,\mathbf{w} + c'\,\mathbf{w}'] = c[\mathbf{v}, \mathbf{w}] + c'[\mathbf{v}, \mathbf{w}'], \qquad 3.7.2$$

donde c, c' son constantes.

(b) Simetría torcida

$$[\mathbf{v}, \mathbf{w}] = -[\mathbf{w}, \mathbf{v}] . \qquad 3.7.3$$

(c) La identidad de Jacobi

$$[\mathbf{u}, [\mathbf{v}, \mathbf{w}]] + [\mathbf{w}, [\mathbf{u}, \mathbf{v}]] + [\mathbf{v}, [\mathbf{w}, \mathbf{u}]] = 0 . \qquad 3.7.4$$

La primera definición de los corchetes de Lie (3.7.1) asegura que es libre de coordenadas. Mas generalmente, si $F : M \to N$ es cualquier transformación suave, y v y w son campos vectoriales sobre M tal que $dF(\mathbf{v})$, $dF(\mathbf{w})$ son campos vectoriales F-relacionados a bien-definidos sobre N, entonces sus corchetes de Lie también son F-relacionados:

$$dF([\mathbf{v}, \mathbf{w}]) = [dF(\mathbf{v}), dF(\mathbf{w})] . \qquad 3.7.5$$

3.8 Álgebras de Lie

Si G es un grupo de Lie, entonces hay ciertos campos vectoriales distinguidos sobre G caracterizados por su invarianza (en un sentido de definición corta) bajo la multiplicación de grupo. Como se verá, estos campos vectoriales invariantes forman un espacio vectorial de dimensión finita, llamado el álgebra de Lie de G, el cual es en un sentido preciso el "generador infinitesimal" de G. De hecho casi toda la información en el grupo G está contenida en su álgebra de Lie. Esta observación fundamental es el pilar de la teoría de grupo de Lie; por ejemplo, nos habilita para reemplazar condiciones no lineales complicadas de invarianza bajo una acción de grupo por relativamente simples condiciones infinitesimales lineales. La potencia de éste método no puede ser sobreestimada-efectivamente casi toda la gama de aplicaciones de grupos de Lie a las ecuaciones diferenciales últimamente descansa en esta construcción.

Comenzamos con la figura global de grupo de Lie, direccionando la construcción análoga para grupos de Lie locales subsecuentemente. Sea G un grupo de Lie. Para cualquier elemento de grupo $g \in G$, la *transformación de multiplicación derecha* R_g: $G \to G$ definida por

$$R_g(h) = h \cdot g$$

es un difehomorfismo, con la inversa

$$R_{g^{-1}} = (R_g)^{-1}.$$

Un campo vectorial v sobre G se llama *invariante-derecho* si

$$dR_g(\mathbf{v}|_h) = \mathbf{v}|_{Rg(h)} = \mathbf{v}|_{hg}$$

para toda g y h en G. Observe que si v y w son invariante-derechos, también lo es cualquier combinación lineal $a\mathbf{v} + b\mathbf{w}$, $a, b, \in \mathbb{R}$; de aquí que el conjunto de todos los campos vectoriales invariantes-derechos forman un espacio vectorial.

Definición 3.8.1. El álgebra *de Lie* de un grupo de Lie G, tradicionalmente denotada por la correspondiente letra gótica $\mathfrak{g}$ minúscula, es el espacio vectorial de todos los campos vectoriales invariantes-derechos sobre G.

Observe que cualquier campo vectorial invariante-derecho está únicamente determinado por su valor en la identidad porque

$$\mathbf{v}|_g = dR_g(\mathbf{v}|_e)\,, \tag{3.8.1}$$

ya que $R_g(e) = g$. Contrariamente, cualquier vector tangente a G en e únicamente determina un campo vectorial invariante-derecho sobre G por la fórmula (3.8.1). Efectivamente,

$$dR_g(\mathbf{v}|_h) = dR_g(dR_h(\mathbf{v}|_e)) = d(R_g \circ R_h)(\mathbf{v}|_e) = dR_{hg}(\mathbf{v}|_e) = \mathbf{v}|_{hg}\,,$$

probando la invarianza-derecha de $\mathbf{v}$. Por lo tanto podemos identificar el álgebra de Lie $\mathfrak{g}$ de G con un espacio tangente a G en el elemento identidad

$$\mathfrak{g} \;\simeq\; TG|_e\,. \tag{3.8.2}$$

En particular, $\mathfrak{g}$ es un espacio vectorial de dimensión finita de las mismas dimensiones que el grupo de Lie subyacente.

En adición a su estructura de espacio vectorial, tal álgebra de Lie está además provista de una operación bilineal simétrica torcida, a saber el corchete de Lie. Efectivamente, si $\mathbf{v}$ y $\mathbf{w}$ son campos vectoriales invariantes-derechos sobre G, también es su corchete de Lie $[\mathbf{v}, \mathbf{w}]$, ya que por (3.7.5)

$$dR_g([\mathbf{v}, \mathbf{w}]) = [dR_g(\mathbf{v}), dR_g(\mathbf{w})] = [\mathbf{v}, \mathbf{w}]\,.$$

Esto motiva la definición general de un álgebra de Lie.

Definición 3.8.2. Un álgebra de Lie es un espacio vectorial $\mathfrak{g}$ junto con una operación bilineal

$$[\,\cdot,\cdot\,]\colon \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}\,,$$

llamado el *corchete de Lie* para $\mathfrak{g}$ que satisface los axiomas

(a) *Bilinealidad*

$$[c\mathbf{v} + c'\mathbf{v}', \mathbf{w}] = c[\mathbf{v}, \mathbf{w}] + c'[\mathbf{v}', \mathbf{w}]\,; \quad [\mathbf{v}, c\mathbf{w} + c'\mathbf{w}'] = c[\mathbf{v}, \mathbf{w}] + c'[\mathbf{v}, \mathbf{w}']\,,$$

para las constantes $c, c' \in \mathbb{R}$,

(b) *Simetría torcida* $\quad [\mathbf{v}, \mathbf{w}] = -[\mathbf{w}, \mathbf{v}]\,,$

(c) *La identidad de Jacobi* $\quad [\mathbf{u}, [\mathbf{v}, \mathbf{w}]] + [\mathbf{w}, [\mathbf{u}, \mathbf{v}]] + [\mathbf{v}, [\mathbf{w}, \mathbf{u}]] = 0\,,$

para toda $\mathbf{u}, \mathbf{v}, \mathbf{v}', \mathbf{w}, \mathbf{w}'$ en $\mathfrak{g}$.

♦

4

Introducción a los Grupos Cuánticos [19]

4.1 Introducción

Los grupos cuánticos surgieron primero en la literatura física, particularmente con el trabajo de L.D. Faddeev [**31, 32, 33, 34, 35**] y la escuela Leningrad, a partir del 'método de difusión inversa' (inverse scattering method), el cual había sido desarrollado para construir y resolver sistemas cuánticos 'integrables'. Ellos han experimentado gran interés en los pocos años transcurridos debido a sus conexiones inesperadas con tales, a primera vista, partes no relacionadas de las matemáticas como la construcción de invariantes de nudo y la teoría de representación de grupos algebraicos en la característica p.

En su forma original, los grupos cuánticos son álgebras asociativas cuyas relaciones que las definen se expresan en términos de una matriz de constantes (dependiendo del sistema integrable en consideración) llamada una matriz-R cuántica. Se encontró independientemente por V.G. Drinfel'd [**27**] y M. Jimbo [**43**] alrededor de 1985 que estas álgebras son álgebras de Hopf, las cuales en muchos casos, son deformaciones de 'álgebras envolventes universales' de álgebras de Lie. Un poco después, Yu.I. Manin [**72**] y S.L. Woronowicz [**100**] independientemente construyeron deformaciones no conmutativas del álgebra de funciones en los grupos $SL_2(\mathbf{C})$ y SU_2, respectivamente, y mostraron que muchos de los resultados clásicos acerca de los grupos algebraicos y topológicos admiten analogías en el caso no-conmutativo.

Así, aunque muchos de los artículos fundamentales sobre grupos cuánticos están escritos en el lenguaje de sistemas integrables, sus propiedades son accesibles por otras técnicas convencionales, tales como la teoría de grupos topológicos y algebraicos y las álgebras de Lie. La intención es presentar la teoría de grupos cuánticos desde este último punto de vista. De hecho, se concentra en el estudio de las 'álgebras de Lie' de grupos cuánticos, las cuales parecen ser el enfoque que se ha probado ser el más poderoso, particularmente en aplicaciones, pero también discute, en menor detalle, su relación con la 'topología y la geometría algebraica no conmutativa'.

Enseguida se describe qué es un grupo cuántico, comenzando por tratar de explicar el motivo para el uso del adjetivo 'cuántico'.

En mecánica clásica, el espacio de fase M de un sistema dinámico es una *variedad de Poisson* (Poisson Manifold). Esto significa que el espacio $\mathcal{F}(M)$ de funciones (diferenciables) complejo-

valuadas en M está provisto con un corchete de Lie $\{\ ,\ \} : \mathcal{F}(M) \times \mathcal{F}(M) \to \mathcal{F}(M)$ (satisfaciendo ciertas condiciones adicionales), llamado el corchete de Poisson. Las ecuaciones dinámicas que definen la evolución del tiempo del sistema son equivalentes a las ecuaciones

$$\frac{d}{dt} f(m(t)) = \{\, \mathcal{H}_{\mathrm{cl}}, f \}(m\,(t)\,)$$

para $f \in \mathcal{F}(M)$, donde $\mathcal{H}_{\mathrm{cl}}$ es una función fija en M llamada la (clásica) hamiltoniana, y $m(t) \in M$ es el 'estado' del sistema en el tiempo t. Por ejemplo, para una sola partícula moviéndose a lo largo de la línea real, M es el haz cotangente $T^*(\mathbb{R})$, y si q es la coordenada en $\mathbb{R}$ ('posición') y p la coordenada en la dirección de la fibra ('momentum'), el corchete de Poisson es

$$\{f_1, f_2\} = \frac{\partial f_1}{\partial p}\frac{\partial f_2}{\partial q} - \frac{\partial f_2}{\partial p}\frac{\partial f_1}{\partial q}\ .$$

En particular, el corchete de Poisson de las funciones de coordenadas es

$$\{p, q\} = 1\ . \tag{4.1.1}$$

En mecánica cuántica, el espacio M es sustituido por el conjunto de haces en un espacio de Hilbert complejo V, y el espacio $\mathcal{F}(M)$ de las funciones en M por el álgebra $\mathrm{Op}(V)$ de (no necesariamente limitados) operadores en V. La evolución del tiempo de un operador A está dado por

$$\frac{dA}{dt} = [\mathcal{H}_{\mathrm{qu}}, A]$$

para algún operador $\mathcal{H}_{\mathrm{qu}} \in \mathrm{Op}(V)$, llamado el (cuántico) hamiltoniano. Por ejemplo en el caso de una sola partícula moviéndose a lo largo de una línea real, V es el espacio $L^2(\mathbb{R})$ de funciones integrables-cuadráticas de q, y los operadores P y Q correspondiendo a las funciones de las coordenadas p y q están dados por

$$P = -\sqrt{-1}\, h\frac{\partial}{\partial q}\ , \qquad\qquad Q = \text{multiplicación por } q,$$

donde h es $1/2\pi$ veces la constante de Plank. Observe que

$$[P,Q] = -\sqrt{-1}\, h\, \mathrm{id}_V\ . \tag{4.1.2}$$

La cuestión es: cómo pasar de la descripción clásica a la descripción cuántica de un sistema. Este es el problema de la *cuantificación*. Idealmente, uno quisiera una transformación $\mathcal{Q}$ la cual asignara para cada función $f \in \mathcal{F}(M)$ un operador $\mathcal{Q}(f)$ sobre V. Además, ya que la evolución en el tiempo de la descripción clásica y la descripción cuántica se da tomando el corchete y el conmutador de Poisson con el hamiltoniano, respectivamente, $\mathcal{Q}$ debe satisfacer la relación

$$\mathcal{Q}\{f_1, f_2\} = \frac{[\mathcal{Q}(f_1), \mathcal{Q}(f_2)]}{-\sqrt{-1}h}$$

(la normalización viene de (4.1.1) y (4.1.2)). Desafortunadamente, se sabe que, aún para el caso más simple de una sola partícula moviéndose a lo largo de la línea real, no existe tal transformación $\mathcal{Q}$.

Existe, sin embargo, una formulación alterna del problema de cuantificación, introducida por J.E. Moyal en 1949 [79]. Esta comienza observando que la diferencia fundamental entre las descripciones clásicas y las cuánticas es que $\mathcal{F}(M)$ es un álgebra conmutativa, mientras que $\mathrm{Op}(V)$ es no conmutativa (cuando $\dim(V) > 1$). La idea de Moyal es tratar de reproducir los resultados de la mecánica cuántica reemplazando el producto usual en $\mathcal{F}(M)$ por un producto no-conmutativo $\bullet_h$, dependiendo de un parámetro h, tal que $\bullet_h$ se hace el producto usual cuando $h \to 0$, igual que la 'mecánica cuántica se vuelve mecánica clásica cuando la constante de Plank tiende a cero', y tal que

$$\lim_{h\to 0} \frac{f_1 \bullet_h f_2 - f_2 \bullet_h f_1}{h} = \{f_1, f_2\} . \tag{4.1.3}$$

Si pensamos en $\mathcal{F}(M)$ con el producto de Moyal $\bullet_h$ como un álgebra no conmutativa de funciones $\mathcal{F}_h(M)$, nos encontramos por nosotros mismos en el área de la geometría no conmutativa en el sentido de A. Connes. La filosofía aquí es que cualquier 'espacio' está determinado por el álgebra de las funciones en él (con el producto usual). Por ejemplo, cada variedad de álgebra afín sobre $\mathbf{C}$ está determinada (hasta isomorfismo) por el álgebra conmutativa de las funciones regulares en ella, mientras cada espacio topológico compacto está determinado por su álgebra-C* conmutativa de funciones continuas complejo-valuadas. Más precisamente, la categoría de 'espacios' en estos ejemplos es dual para la categoría de las álgebras correspondientes. De este modo, un álgebra no conmutativa debe ser vista como el espacio de funciones en un 'espacio no-conmutativo', y podemos decir que la construcción de Moyal proporciona una deformación del espacio clásico de fase M a la familia de espacios no conmutativos (o 'cuánticos') M_h tal que $\mathcal{F}_h(M)$ es el álgebra de las funciones en M_h.

La categoría de espacios cuánticos, entonces, puede definirse como la categoría dual a la categoría de las álgebras asociativas, aunque no necesariamente conmutativas. Para definir la noción de grupo cuántico, regresemos por un momento a la situación clásica. Si G es un grupo, la multiplicación $\mu : G \times G \to G$ de G induce un homomorfismo $\mu^* = \Delta : \mathcal{F}(G) \to \mathcal{F}(G \times G)$ de álgebras de funciones. Ahora, si definimos el álgebra $\mathcal{F}(G)$ y el producto tensor apropiadamente, $\mathcal{F}(G \times G)$ será isomórfica a $\mathcal{F}(G) \otimes \mathcal{F}(G)$ como un álgebra. Por ejemplo, si G es un grupo algebraico afín sobre $\mathbf{C}$, y $\mathcal{F}(G)$ es el álgebra de funciones regulares en G, el producto tensor algebraico ordinario lo será. Así, tenemos una comultiplicación $\Delta : \mathcal{F}(G) \to \mathcal{F}(G) \otimes \mathcal{F}(G)$. (La razón para esta terminología es que la multiplicación en $\mathcal{F}(G)$ puede verse como una transformación $\mathcal{F}(G) \otimes \mathcal{F}(G) \to \mathcal{F}(G)$.) Similarmente, la transformación inversa $\iota : G \to G$ induce una transformación $\iota^* = S : \mathcal{F}(G) \to \mathcal{F}(G)$, llamada la antípoda, y la evaluación en el

elemento identidad de G es un homomorfismo $\epsilon : \mathcal{F}(G) \to \mathbb{C}$, llamada la counidad. Las transformaciones Δ, S y ϵ satisfacen ciertas propiedades de compatibilidad las cuales reflejan las propiedades que definen la inversa y la asociatividad de la multiplicación en G, y se combinan para dar a $\mathcal{F}(G)$ la estructura de un *álgebra de Hopf*.

Podríamos por lo tanto *definir la categoría de grupos cuánticos para ser la categoría dual para la categoría de (no necesariamente conmutativa) álgebras de Hopf*. (Decimos 'podría' aquí, y en nuestra definición tentativa de espacio cuántico, porque, para asegurar que las categorías de espacios cuánticos y grupos cuánticos tengan propiedades razonables, sería necesario imponer algunas restricciones en la clase de álgebras las cuales son aceptables como 'álgebras cuantificadas de funciones'.) Como es práctica común en la literatura, a menudo abusaremos de la terminología refiriéndonos a un álgebra de Hopf por sí misma como un grupo cuántico.

Como lo sugiere la discusión precedente, una forma de tratar de construir ejemplos no clásicos de grupos cuánticos es buscar deformaciones, en la categoría de álgebras de Hopf, de álgebras clásicas de funciones $\mathcal{F}(G)$. Justamente como el corchete clásico de Poisson se puede recuperar como 'parte de primer orden' de la deformación de Moyal (ver (4.1.3)), así sale que la existencia de una deformación $\mathcal{F}_h(G)$ de $\mathcal{F}(G)$ automáticamente enriquece al grupo G a sí mismo con estructura extra, a saber aquella de un *grupo de Poisson-Lie*. Ésta es una estructura de Poisson sobre G la cual es compatible con la estructura de grupo en cierto sentido. Contrariamente para construir deformaciones de $\mathcal{F}(G)$, es natural comenzar describiendo las posibles estructuras del grupo de Poisson-Lie sobre G y entonces tratar de extender estas 'deformaciones de primer orden' a deformaciones completas. Este es el enfoque que se considera aquí. Los grupos de Poisson-Lie también son de interés en su propio derecho, por ellos se forma el agrupamiento natural para el estudio de sistemas integrables clásicos con simetría.

Existe otra álgebra de Hopf asociada a cualquier grupo de Lie G, el álgebra envolvente universal $U(\mathfrak{g})$ de su álgebra de Lie $\mathfrak{g}$. Esta es esencialmente el *dual* de $\mathcal{F}(G)$ en la categoría de las álgebras de Hopf. En general, el espacio vectorial dual A^* de cualquier álgebra de Hopf de dimensión finita A es también un álgebra de Hopf: la multiplicación $A^* \otimes A^* \to A^*$ es dual a la comultiplicación $\Delta : A \to A \otimes A$ de A, y la comultiplicación de A^* es dual a la multiplicación de A. Observe que A^* es conmutativa si y solo si A es coconmutativa, o sea, si y solo si $\Delta(A)$ está contenida en la parte simétrica de $A \otimes A$. Si, como es generalmente el caso en ejemplos de interés, A es de dimensión infinita, esta dualidad a menudo continúa dado que el dual y el producto tensor están definidos apropiadamente. Para una deformación $\mathcal{F}_h(G)$ de $\mathcal{F}(G)$ a través de las (no necesariamente conmutativas) álgebras de Hopf, por consiguiente, corresponde una deformación $U_h(\mathfrak{g})$ de $U(\mathfrak{g})$ a través de las (no necesariamente coconmutativas) álgebras de Hopf.

De hecho, solamente las deformaciones no coconmutativas de $U(\mathfrak{g})$ son de interés, ya que cualquier deformación de $U(\mathfrak{g})$ a través de las álgebras de Hopf coconmutativas es necesariamente de la forma $U(\mathfrak{g}_h)$ para alguna deformación de $\mathfrak{g}_h$ de $\mathfrak{g}$ a través de las álgebras dc Lie. Sin embargo, muchas álgebras de Lie interesantes tienen deformaciones no triviales. Este es el caso, por ejemplo, si $\mathfrak{g}$ es una (de dimensión finita) compleja semisimple álgebra de Lie, tal como el álgebra de Lie $sl_2(\mathbb{C})$ de matrices complejas de 2×2 de traza cero. Esto se sigue del hecho de que la condición de semisimplicidad está abierta, tal que cualquier pequeña deformación de $\mathfrak{g}$ aún será semisimple, mientras las álgebras de Lie semisimples están discretamente parametrizadas (por sus diagramas de Dynkin, por ejemplo).

El primer ejemplo de una deformación no coconmutativa de este tipo fue descubierto por P.P. Kulish y E.K. Sklyanin en 1981 [**61**] en el caso de $\mathfrak{g} = sl_2(\mathbb{C})$ (aunque la importancia de su estructura de Hopf no fue descubierta hasta más tarde). Observe que $sl_2(\mathbb{C})$ tiene una base

$$\overline{X}^+ = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \qquad \overline{X}^- = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \qquad \overline{H} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{4.1.4}$$

cuyos corchetes de Lie están dados por

$$[\overline{X}^+, \overline{X}^-] = \overline{H}, \qquad [\overline{H}, \overline{X}^\pm] = \pm 2\overline{X}^\pm. \tag{4.1.5a}$$

La comultiplicación está dada sobre estos elementos de base por

$$\Delta(\overline{H}) = \overline{H} \otimes 1 + 1 \otimes \overline{H}, \qquad \Delta(\overline{X}^\pm) = \overline{X}^\pm \otimes 1 + 1 \otimes \overline{X}^\pm, \tag{4.1.5b}$$

una asignación la cual se extiende únicamente a un homomorfismo de álgebra $\Delta : U(sl_2(\mathbb{C})) \to U(sl_2(\mathbb{C})) \otimes U(sl_2(\mathbb{C}))$. La deformación $U_h(sl_2(\mathbb{C}))$ está generada por los elementos H, $\overline{X}^\pm$, los cuales satisfacen las relaciones

$$X^+X^- - X^-X^+ = \frac{e^{hH} - e^{-hH}}{e^h - e^{-h}}, \qquad HX^\pm - X^\pm H = \pm 2X^\pm. \tag{4.1.6a}$$

Tiene una comultiplicación no coconmutativa dada sobre los generadores por

$$\Delta(H) = H \otimes 1 + 1 \otimes H,$$
$$\Delta(X^+) = X^+ \otimes e^{hH} + 1 \otimes X^+, \qquad \Delta(X^-) = X^- \otimes 1 + e^{-hH} \otimes X^-. \tag{4.1.6b}$$

Formalmente, al menos, está claro que (4.1.6a) y (4.1.6b) van por entre (4.1.5a) y (4.1.5b) cuando $h \to 0$. El álgebra de Hopf definida en (4.1.6a,b) es llamada 'el cuanto $sl_2(\mathbb{C})$'.

El álgebra de Hopf dual para: $U_h(sl_2(\mathbb{C}))$, el 'álgebra $F_h(SL_2(\mathbb{C}))$ de funciones sobre el cuanto $SL_2(\mathbb{C})$', fue descubierto por L.D. Fadeev y L.A. Takhtajan en 1985 [**35**]. Es el álgebra asociativa generada por los elementos a, b, c, d con las siguientes relaciones multiplicativas:

$$ab = e^{-h}ba, \quad ac = e^{-h}ca, \quad bd = e^{-h}db, \quad cd = e^{-h}dc, \tag{4.1.7}$$
$$bc = cb, \quad ad - da + (e^h - e^{-h})bc = 0, \tag{4.1.8}$$
$$ad - e^{-h}bc = 1, \tag{4.1.9}$$

y la comultiplicación

$$\Delta(a) = a \otimes a + b \otimes c\,, \qquad \Delta(b) = a \otimes b + b \otimes d\,,$$
$$\Delta(c) = c \otimes a + d \otimes c\,, \qquad \Delta(d) = c \otimes b + d \otimes d\,,$$

Observe que, cuando $h \to 0$, las relaciones (4.1.7), (4.1.8) y (4.1.9) solo dicen que la matriz

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

tiene entradas conmutantes y determinante uno. Así, $\mathcal{F}_h(SL_2(\mathbb{C}))$ es una deformación del álgebra de funciones sobre el grupo $SL_2(\mathbb{C})$ de la matriz 2×2 compleja de determinante uno.

Como se mencionó al principio de ésta introducción, la estructura de álgebra de $\mathcal{F}_h(G)$ puede ser descrita por una matriz de constantes, a saber

$$R = e^{-h/2}\begin{pmatrix} e^h & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & e^h - e^{-h} & 1 & 0 \\ 0 & 0 & 0 & e^h \end{pmatrix} \qquad 4.1.10$$

De hecho, si

$$T = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Las relaciones (4.1.7) y (4.1.8) equivalen a

$$(T \otimes 1)(1 \otimes T)R = R(1 \otimes T)(T \otimes 1)\,. \qquad (4.1.11)$$

Observe que $T \otimes 1$ y $1 \otimes T$ no conmutan, ya que las entradas a T no conmutan (si $h \neq 0$); vea también que R es lo más naturalmente vista como un elemento de $\mathrm{End}(\mathbb{C}^2 \otimes \mathbb{C}^2)$. Está en la forma (4.1.11) en la que los grupos cuánticos generalmente aparecen en la teoría de sistemas integrables.

Para el álgebra de Hopf dual $U_h(sl_2(\mathbb{C}))$, la matriz-R cuántica expresa, como sería de esperarse, la no coconmutatividad de la comultiplicación. A saber, sea $\Delta^{op}(x)$ el resultado de intercambiar el orden de los factores en $\Delta(x)$, para cualquier $x \in U_h(sl_2(\mathbb{C}))$. Se obtiene que hay un elemento invertible $\mathcal{R} \in U_h(sl_2(\mathbb{C})) \otimes U_h(sl_2(\mathbb{C}))$, llamada la 'matriz-R universal', tal que

$$\Delta^{op}(x) = \mathcal{R}\Delta(x)\mathcal{R}^{-1}$$

para toda $x \in U_h(sl_2(\mathbb{C}))$ (realmente, $\mathcal{R}$ es una suma infinita formal de elementos del producto tensor algebraico). La relación entre $\mathcal{R}$ y R es muy simple: el lector fácilmente verificará que, si se reemplaza $\overline{X}^{\pm}$ y $\overline{H}$ por $X^{\pm}$ y H en (4.1.4), se obtiene una representación de la matriz de $U_h(sl_2(\mathbb{C}))$; aplicando esta representación a $\mathcal{R}$ da la matriz R.

Los grupos cuánticos podrían haber permanecido como una curiosidad para la comunidad matemática por mucho, sin embargo, se encontraron sorprendentes conexiones con otras partes de las matemáticas, más notablemente la teoría de invariantes de enlaces y variedades-3, y la teoría de representación de las álgebras de Lie en la característica p.

El soporte depende de la relación clásica entre las trenzas y los enlaces. Recuerde que una *trenza sobre* m *hilos* es una colección de m cuerdas no intersectadas en $\mathbb{R}^3$ uniendo m puntos fijos en un plano a m puntos fijos en otro plano paralelo. Uniendo los puntos correspondientes en los dos planos en una forma estándar asocia a cualquier trenza un lazo (llamado su 'cerradura'), o sea, una colección de círculos no intersectantes en $\mathbb{R}^3$. Uniendo las trenzas final a final se hace el conjunto de clases isotópicas de trenzas dentro de un grupo $\mathcal{B}_m$. La relación con los grupos cuánticos surge porque hay un camino simple para asociar a cualquier matriz-R cuántica $R \in \mathrm{End}(V \otimes V)$ una representación ρ_m de $\mathcal{B}_m$ en $V^{\otimes m}$ para toda $m \geq 2$. Esto depende del hecho que R satisface la *ecuación de Yang-Baxter cuántica*

$$R_{12}R_{13}R_{23} = R_{23}R_{13}R_{12}\ ;$$

aquí, R_{12} significa $R \otimes \mathrm{id} \in \mathrm{End}(V^{\otimes 3})$, etc. Para obtener una invariante de lazos, uno necesita una familia de 'trazas' $tr_m : \mathrm{End}(V^{\otimes m}) \to \mathbf{C}$ tal que $tr_m(\rho_m(b)) = tr_n(\rho_n(b'))$ siempre que las cerraduras de las trenzas $b \in \mathcal{B}_m$ y $b' \in \mathcal{B}_n$ sean enlaces equivalentes. Gracias a un teorema clásico de A. Markov, se sabe precisamente que pares (b, b') tienen la última propiedad (y por esta razón, las tr_m son generalmente llamadas 'trazas de Markov'). Utilizando la matriz-R cuántica (4.1.10) y una traza de Markov acomodable, uno obtiene de éste modo el celebrado *polinomio de Jones*. De hecho, esta es esencialmente la construcción original de Jones [**44**], excepto que él obtuvo su matriz-R utilizando un 'álgebra de Hecke' en lugar de un grupo cuántico (pero debemos ver que las álgebras de Hecke deben ser probablemente consideradas como objetos 'cuánticos').

La aplicación a variedades-3 está basada en el bien conocido hecho de que cada variedad-3 conectada, orientada, compacta sin límite puede ser obtenida, hasta homeomorfismo, realizando cirugía en un enlace en la esfera de dimensión-3. Uno muestra que una combinación escogida inteligentemente de las invariantes cuánticas de este enlace depende solo de la variedad-3, y no de la elección del enlace junto con la cual se realiza la cirugía.

La aplicación de los grupos cuánticos a las representaciones de las álgebras de Lie en la característica p no es menos considerable. Hace uso de una cierta deformación 'estándar' $U_h(\mathfrak{g})$ de $U(\mathfrak{g})$, donde $\mathfrak{g}$ es cualquier álgebra de Lie semisimple compleja dimensional finita (y la cual se reduce, cuando $\mathfrak{g} = sl_2(\mathbf{C})$, para el álgebra encontrada por Kulish y Sklyanin [**61**]). Para describir la relación con la característica p, es conveniente reemplazar el parámetro de deformación h por $\epsilon = e^h$, y escribir $U_\epsilon(\mathfrak{g})$ para $U_h(\mathfrak{g})$. Entonces se da que la teoría de representación de $U_\epsilon(\mathfrak{g})$ depende crucialmente de ya sea que ϵ es una raíz de unidad o no. En el

último caso, la teoría es esencialmente la misma que la teoría de representación de g por sí misma (sobre $\mathbf{C}$), pero en la estructura se parece a la teoría de representación modular de g. Esto es más que una analogía: si ϵ es una raíz p-ava primitiva de la unidad, donde p es un primo, hay un homomorfismo de anillo desde $U_\epsilon(\mathfrak{g})$ hasta el álgebra envolvente $U_{\mathbf{F}p}(\mathfrak{g})$ de g sobre el campo $\mathbf{F}_p$ de elementos p (esto se obtiene esencialmente reemplazando ϵ por $1 \in \mathbf{F}_p$), y, con ciertas condiciones adicionales, para dar las representaciones de $U_{\mathbf{F}p}(\mathfrak{g})$. De este modo, ambas representaciones la modular de g y la característica teoría cero se reflejan en la teoría de representación de la familia de las álgebras de Hopf $U_\epsilon(\mathfrak{g})$ (sobre $\mathbf{C}$). Usando esta relación, un progreso sustancial se ha hecho hacia la solución de varias conjeturas de larga estancia en la teoría modular.

También debemos mencionar los papeles que juegan los grupos cuánticos en física, los cuales van bien ante sus orígenes en la teoría de dispersión inversa. Quizá la más interesante de éstas es la relación entre los grupos cuánticos y las teorías de campos cuánticos y conformes. La primera evidencia de esto fue la observación experimental de que las 'reglas de fusión' de ciertas teorías de campo conformes pueden reproducirse considerando la descomposición de productos tensores de representaciones de grupos cuánticos $U_\epsilon(\mathfrak{g})$ cuando ϵ es una raíz de la unidad. Llegó evidencia adicional desde un teorema notable de T. Kohno y Drinfel'd en la relación entre la ecuación de Knizhnik-Zamolodchikov (KZ) [59] y $U_h(\mathfrak{g})$. La ecuación de KZ es un sistema de ecuaciones diferenciales parciales de primer orden para una función de m variables complejas con valores en el producto tensor $V^{\otimes m}$, donde V es una representación de g. El sistema de KZ tiene singularidades regulares junto a los hiperplanos $z_i = z_j$ en $\mathbf{C}^m$, para toda $i \neq j$, y puede verse como una conexión en un paquete sobre el complemento $\mathcal{D}_m$ de estos hiperplanos, con la fibra $V^{\otimes m}$. Además, esta conexión tiene una propiedad de simetría la cual significa que existe una conexión inducida en un paquete sobre el espacio C_m de órbitas de la acción obvia del grupo simétrico Σ_m sobre $\mathcal{D}_m$. La propiedad crucial de la ecuación de KZ es que esta conexión es plana, lo cual implica que la monodromía de sus soluciones define una representación del grupo fundamental de C_m en $V^{\otimes m}$. El último grupo es exactamente la trenza del grupo $\mathcal{B}_m$ que ya discutimos antes, donde notamos que una representación de $\mathcal{B}_m$ sobre $V^{\otimes m}$ también podría ser obtenida utilizando la matriz-R cuántica. De acuerdo con Kohno y Drinfel'd, si utilizamos la matriz-R dada por la acción de la matriz-R universal de $U_h(\mathfrak{g})$ sobre $V \otimes V$, donde h es 'genérica', estas dos representaciones de $\mathcal{B}_m$ coinciden.

La importancia de la ecuación de KZ en la teoría del campo conforme es que se satisface por las 'funciones de punto-m' de la teoría. Así, el teorema de Kohno-Drinfel'd indica una conexión entre $U_h(\mathfrak{g})$ y la teoría de campo conforme. Esto ha sido recientemente extendido por D. Kazhdan y G. Lusztig [49], quienes han mostrado que la ecuación de KZ está íntimamente relacionada con la categoría de representaciones de las álgebras 'especializadas' $U_h(\mathfrak{g})$ cuando ϵ es una raíz de la unidad. Este resultado proporciona una 'explicación' para la coincidencia entre

las reglas de fusión que surgen en la teoría de campo conforme y aquellas que surgen desde los grupos cuánticos en las raíces de la unidad.

Estos y otros ejemplos en cuestión muestran que la teoría de los grupos cuánticos ocupa un lugar importante en la corriente de las matemáticas y la física matemática.

4.2 Grupos de Poisson-Lie y Biálgebras de Lie

Los *grupos de Poisson-Lie* son grupos de Lie G los cuales tienen estructuras de Poisson las cuales son compatibles con las operaciones de grupo en cierto sentido.

Los ejemplos más importantes de los grupos de Poisson-Lie, al menos para el estudio de los grupos cuánticos, se relacionan al caso en el cual G es un grupo compacto (o su complejidad).

Las versiones infinitesimales de los grupos de Poisson-Lie son llamadas *biálgebras de Lie*. Más precisamente, si G es cualquier grupo de Poisson-Lie, el dual $\mathfrak{g}^*$ de su álgebra de Lie $\mathfrak{g}$ resulta tener una estructura de álgebra de Lie natural, la cual es compatible con la de $\mathfrak{g}$ por sí misma, en un cierto sentido; juntos, se dice que estas forman una estructura de biálgebra de Lie en $\mathfrak{g}$. Contrariamente, si G es conectado y simple conectado, cada estructura de biálgebra de Lie sobre $\mathfrak{g}$ surge desde una única estructura de grupo de Poisson-Lie sobre G.

4.2.1 Variedades de Poisson.

A Definiciones. Sea M una variedad suave de dimensión finita m. Denotamos por $C^\infty(M)$ el álgebra de funciones suaves real valuadas sobre M.

Definición 4.2.1. Una estructura de Poisson *sobre M es una transformación bilineal-*$\mathbb{R}$

$$\{\,,\}_M : C^\infty(M) \times C^\infty(M) \to C^\infty(M),$$

llamada el corchete de Poisson, *el cual satisface las siguientes condiciones*:

$$\{f_1, f_2\} = -\{f_2, f_1\}, \tag{4.2.1}$$

$$\{f_1,\{f_2, f_3\}\} + \{f_2,\{f_3, f_1\}\} + \{f_3,\{f_1, f_2\}\} = 0, \tag{4.2.2}$$

$$\{f_1 f_2, f_3\} = f_1\{f_2, f_3\} + \{f_1, f_3\} f_2, \tag{4.2.3}$$

para toda. $f_1, f_2, f_3 \in C^\infty(M)$. *(Se omite el subindice en el corchete de Poisson siempre que no cause confusión.)*

La propiedad de simetría torcida (4.2.1) y la *identidad de Jacobi* (4.2.2) juntas significan que $\{\,,\}$ es un corchete de Lie sobre $C^\infty(M)$. La propiedad (4.2.3), la *identidad de Leibniz*, significa que, para toda $f \in C^\infty(M)$, la transformación $g \to \{g, f\}$ es una derivación de $C^\infty(M)$; de aquí que, hay un campo vectorial X_f sobre M tal que, $X_f(g) = \{g, f\}$ para toda g. (Un campo vectorial el cual sale de este modo es llamado un *campo vectorial hamiltoniano*.) En particular,

$\{g, f\}$ depende únicamente del diferencial de g y, en vista de la simetría torcida del corchete de Poisson, sobre el diferencial de f.

En coordenadas locales $(x_1, \ldots, x_m)$ sobre M, el corchete de Poisson puede escribirse en la forma

$$\{f,g\} = \sum_{i,j=1}^{m} w_{ij}(x)\frac{\partial f}{\partial x_i}\frac{\partial g}{\partial x_j} ,$$

donde las funciones definidas localmente w_{ij} son los coeficientes del bivector de Poisson

$$w_x = \sum_{i,j=1}^{m} w_{ij}(x)\frac{\partial}{\partial x_i}\otimes\frac{\partial}{\partial x_j} ,$$

ya que w es simétrica torcida,

$$w_{ij} = -w_{ji}$$

y la identidad de Jacobi es equivalente a

$$\sum_{r=1}^{m}(w_{ri}\frac{\partial w_{jk}}{\partial x_r}+w_{rj}\frac{\partial w_{ki}}{\partial x_r}+w_{rk}\frac{\partial w_{ij}}{\partial x_r}) = 0 .$$

Fue en esta forma que las estructuras de Poisson fueron primeramente introducidas por Lie (1888-93), aunque el caso especial en el cual $M = \mathbb{R}^{2n}$ con corchete

$$\{f,g\} = \sum_{i,j=1}^{n}(\frac{\partial f}{\partial x_i}\frac{\partial g}{\partial x_{i+n}}-\frac{\partial g}{\partial x_i}\frac{\partial f}{\partial x_{i+n}})$$

fue introducida por S. D. Poisson al inicio del siglo diecinueve.

4.2.2 Grupos de Poisson-Lie.

A Definiciones. Un grupo de Poisson-Lie es un grupo de Lie y una variedad de Poisson, siendo ambas estructuras compatibles en el siguiente sentido.

Definición 4.2.2. (a) *Un* grupo de Poisson-Lie *es un grupo G de Lie el cual tiene una estructura de Poisson* $\{\ ,\ \}$ *tal que la transformación de multiplicación* $\mu : G \times G \to G$ $(\mu(g_1, g_2) = g_1g_2)$ *de G es una transformación de Poisson (por supuesto, G × G es dado el producto de la estructura de Poisson).*

(b) *Un* homomorfismo $\Phi : G \to H$ de los grupos de Poisson-Lie *es un homomorfismo de los grupos de Lie el cual es también una transformación de Poisson.*

Por supuesto, la estructura de Poisson trivial sobre cualquier grupo de Lie es una estructura de Poisson-Lie. Para un ejemplo no-trivial, sea $\mathfrak{g}$ cualquier álgebra de Lie y proporcione su dual $\mathfrak{g}^*$

(estructura de Poisson-Lie). Entonces $\mathfrak{g}^*$, considerado como un grupo de Lie abeliano bajo la adición, es un grupo de Poisson-Lie.

4.2.3 Biálgebras de Lie.

Las analogías infinitesimales de los grupos de Poisson-Lie son llamadas biálgebras.

A La biálgebra de Lie de un grupo de Poisson-Lie. Es natural preguntar qué estructura sobre el álgebra de Lie $\mathfrak{g}$ de un grupo G de Lie está asociada a la estructura de un grupo de Poisson-Lie sobre G. De hecho, si G es un grupo de Poisson-Lie, hay una estructura del álgebra de Lie canónica sobre $\mathfrak{g}^*$: si $\xi_1, \xi_2 \in \mathfrak{g}^*$, escoja $f_1, f_2 \in C^\infty(G)$ con $(df_i)_e = \xi_i$, y establezca

$$[\xi_1, \xi_2]_{\mathfrak{g}^*} = (d\{f_1, f_2\})_e. \tag{4.2.4}$$

Asumiendo por un momento que $[\ ,\ ]_{\mathfrak{g}^*}$ está bien definido, es evidentemente simétrico torcido y satisface la identidad de Jacobi porque $\{\ ,\ \}$ tiene las mismas propiedades.

4.2.4 Deformación de Estructuras de Poisson y Cuantificación.

A Deformaciones de las álgebras de Poisson. Si A es un álgebra asociativa, una *deformación* de A es una familia de transformaciones asociativas bilineales

$$*_h : A \times A \to A,$$

dependiendo de un parámetro h, tal que $*_0$ es la multiplicación dada de A. Idealmente, quisiéramos que $*_h$ dependiera suavemente o analíticamente de h en algún sentido. Sin embargo, la mayoría de las álgebras A que interesa deformar son de dimensión infinita, en cuyo caso aún el significado de 'uniforme' o 'analítico' es problemático. Así, ignoraremos tales dificultades considerando solo deformaciones *formales*. Esto significa que tratamos a h como a un indeterminado y expande $*_h$ en una serie de potencias formal

$$a_1 *_h a_2 = \sum_{n=0}^{\infty} h^n \pi_n(a_1, a_2)$$

para ciertas transformaciones bilineales $\pi_n : A \times A \to A$. La asociatividad de $*_h$ es equivalente a

$$\sum_{r+s=n} \pi_r(\pi_s(a_1, a_2), a_3) = \sum_{r+s=n} \pi_r(a_1, \pi_s(a_2, a_3))$$

para toda $a_1, a_2, a_3 \in A$ y toda $n \geq 0$.

Ahora suponga que A es un *álgebra de Poisson* conmutativa, o sea, A es conmutativa como un álgebra, y, en adición a su estructura de álgebra asociativa, A está equipada con una transformación bilineal simétrica torcida $\{\ ,\ \} : A \times A \to A$, el corchete de Poisson, el cual satisface la identidad de Jacobi

$$\{a_1, \{a_2, a_3\}\} + \{a_2, \{a_3, a_1\}\} + \{a_3, \{a_1, a_2\}\} = 0$$

y la identidad de Leibniz

$$\{a_1a_2,a_3\} = a_1\{a_2,a_3\} + \{a_1,a_3\}a_2 .$$

Entonces decimos que $*_h$ es una *deformación del álgebra de Poisson*, o *cuantificación*, de A si

$$\frac{a_1 *_h a_2 - a_2 *_h a_1}{h} \equiv \{a_1,a_2\} \pmod h .$$

Esto tiene sentido porque

$$a_1 *_h a_2 \equiv a_1a_2 \pmod h$$

así que

$$a_1 *_h a_2 - a_2 *_h a_1 \equiv 0 \pmod h$$

porque A es conmutativa.

B Cuantificación de Weyl [98]. El caso de interés en el estudio de la cuantificación es $A = C^\infty(M)$, donde M es una variedad de Poisson. Considerando el caso más simple $M = \mathbb{R}^2$ con coordenadas p, q y la estructura (simpléctica) de Poisson

$$\{f_1,f_2\}(p,q) = \frac{\partial f_1}{\partial p}\frac{\partial f_2}{\partial q} - \frac{\partial f_2}{\partial p}\frac{\partial f_1}{\partial q} . \tag{4.2.5}$$

Podemos interpretar a M como el espacio de fase de una sola partícula moviéndose a lo largo de la línea real, con posición q y momento p.

C Cuantificación como deformación. Moyal **[79]** (1949) interpretó el procedimiento de la cuantificación de Weyl como una deformación del álgebra $A = C^\infty(\mathbb{R})$. De hecho, él calculó el producto $*_h$ sobre A tal que

$$\Phi(f_1 *_h f_2) = \Phi(f_1)\Phi(f_2).$$

4.3 Grupos de Poisson-Lie Colímites y la Ecuación de Yang-Baxter Clásica

Cómo ya se mencionó, la existencia de una cuantificación de un grupo de Lie G implica la existencia de una estructura de biálgebra de Lie sobre g. Recordemos que una estructura de biálgebra de Lie sobre g es una transformación lineal $\delta : g \to g \otimes g$, la cual es entre otras cosas, un cociclo. Entre los cociclos-1 de g con valores en $g \otimes g$ están los colímites-1, para los cuales $\delta(X) = [X, r]$ para algún $r \in g \otimes g$. Un elemento $r \in g \otimes g$ define una estructura de biálgebra de Lie en esta forma si y solo si la parte simétrica de r es un elemento invariante-g de $g \otimes g$ y la expresión

$$[[r, r]] = [r_{12}, r_{13}] + [r_{12}, r_{23}] + [r_{13}, r_{23}]$$

es un elemento invariante-g de $\mathfrak{g} \otimes \mathfrak{g} \otimes \mathfrak{g}$. La última condición se satisface, en particular, si $[[r, r]] = 0$: esta es la *ecuación de Yang-Baxter clásica* (EYBC). La EYBC primero apareció explícitamente en la literatura sobre sistemas hamiltonianos, pero es un caso especial del corchete de Schouten en geometría diferencial, introducido en 1940 **[91]**.

4.3.1 Biálgebras de Lie Colímites.

A Definiciones. Recordemos que una estructura sobre un álgebra de Lie es una transformación lineal $\delta: \mathfrak{g} \to \mathfrak{g} \otimes \mathfrak{g}$, su coconmutador, satisface las siguientes condiciones:

(a) δ es simétrica torcida,
(b) $\delta^*: \mathfrak{g}^* \otimes \mathfrak{g}^* \to \mathfrak{g}^*$ es un corchete de Lie sobre $\mathfrak{g}^*$,
(c) δ es un cociclo-1.

Desde ahora, escribiremos la acción de un elemento $X \in \mathfrak{g}$ sobre un elemento v de una representación V de $\mathfrak{g}$ como $X.v$, dado que la acción se entiende. Así, la condición del cociclo (c) es

$$\delta([X, Y]) = X.\delta(Y) - Y.\delta(X)$$

para toda $X, Y \in \mathfrak{g}$, con la acción adjunta en cada factor del producto tensor entendido $\mathfrak{g} \otimes \mathfrak{g}$.

Junto con los cociclos-1 de $\mathfrak{g}$ con valores en $\mathfrak{g} \otimes \mathfrak{g}$ están los colímites-1, para los cuales

$$\delta(X) = X.r \tag{4.3.1}$$

para algún $r \in \mathfrak{g} \otimes \mathfrak{g}$ y toda $X \in \mathfrak{g}$.

Definición 4.3.1. Una biálgebra de Lie colímite *es una biálgebra de Lie* $\mathfrak{g}$ *cuyo coconmutador es un colímite-1.*

B La ecuación de Yang-Baxter Clásica (EYBC) $[[r, r]] = 0$.

Una solución de la EYBC a menudo es llamada una *matriz-r clásica*, la terminología derivada del ejemplo $\mathfrak{g} = \mathrm{End}(V)$, donde V es un espacio vectorial, en cuyo caso $r \in \mathrm{End}(V \otimes V)$ puede verse como una matriz. Una estructura de biálgebra de Lie (colímite) la cual sale de una solución de la EYBC se dice que es cuasitriangular; si sale de una solución torcida de la EYBC, se dice que es triangular.

También se obtiene la ecuación de Yang Baxter clásica modificada (EYBCM)

$$[[r, r]] = -w\,. \tag{4.3.2}$$

Es claro que la EYBC es el caso limítrofe de la EYBCM cuando la forma bilineal $(\ ,\ )$ sobre $\mathfrak{g}$ 'tiende a infinito'.

Existe una relación muy estrecha entre las soluciones de la EYBC y de la EYBCM. La forma bilineal puede ser considerada como un elemento de $(\mathfrak{g} \otimes \mathfrak{g})^* \cong \mathfrak{g}^* \otimes \mathfrak{g}^*$, y por.cnde, usando el isomorfismo $\mathfrak{g} \cong \mathfrak{g}^*$, como un elemento de $t \in \mathfrak{g} \otimes \mathfrak{g}$ (t es llamado el *elemento de Casimir*). Entonces t es invariante-$\mathfrak{g}$ y simétrica, y un simple cálculo muestra que

$$[[t, t]] = w. \tag{4.3.3}$$

4.3.2 Grupos de Poisson-Lie Colímites.

En esta sección se describen las propiedades especiales de aquellos grupos de Poisson-Lie cuyas biálgebras de Lie tangentes son colímites.

A El corchete de Sklyanin. Si $\mathfrak{g}$ es una biálgebra de Lie colímite de dimensión finita, su coconmutador está dado, para $X \in \mathfrak{g}$, por

$$\delta(X) = (\mathrm{ad}_X \otimes 1 + 1 \otimes \mathrm{ad}_X)(r),$$

donde podemos y debemos asumir, sin perder generalidad, que $r \in \mathfrak{g} \otimes \mathfrak{g}$ es *simétrica torcida*. Si G es un grupo de Lie con álgebra de Lie $\mathfrak{g}$, para describir la estructura de Poisson sobre G asociada a δ debemos 'integrar' δ a un cociclo-1 de G con valores en $\mathfrak{g} \otimes \mathfrak{g}$. Pero esto es fácil: es trivial verificar que

$$w^R(g) = (\mathrm{Ad}_g \otimes \mathrm{Ad}_g)(r) - r$$

tiene la derivada correcta en el elemento identidad de G y la propiedad de cociclo

$$w^R(gg') = (\mathrm{Ad}_g \otimes \mathrm{Ad}_g)(w^R(g')) + w^R(g)$$

para $g, g' \in G$. Recordemos que el bivector de Poisson $w_g \in T_g(G) \otimes T_g(G)$ es el traslado derecho de $w^R(g)$ a g, y que el corchete de Poisson asociado está dado, para $f_1, f_2 \in \mathrm{C}^\infty(G)$, por

$$\{f_1, f_2\}(g) = \langle w_g, (df_1)_g \otimes (df_2)_g \rangle .$$

B Matrices-r y cociclos-2. El siguiente corolario nos da una interesante interpretación geométrica alternativa de la EYBC. Se aplica a los elementos $r \in \mathfrak{g} \otimes \mathfrak{g}$ los cuales son *no degenerados*, lo que significa que la transformación lineal asociada $\mathfrak{g}^* \to \mathfrak{g}$ es un isomorfismo de espacio vectorial.

Corolario 4.3.1. *Sea G un grupo de Lie de dimensión finita con álgebra de Lie $\mathfrak{g}$, y sea $r \in \mathfrak{g} \otimes \mathfrak{g}$ simétrico torcido y no degenerado. Entonces, r es una solución de la* EYBC *si y solo si el inverso del isomorfismo $\mathfrak{g}^* \to \mathfrak{g}$ inducido por r, considerado como una forma bilineal sobre $\mathfrak{g}$, es un cociclo-2.*

Recordemos que una forma bilineal w sobre un álgebra de Lie $\mathfrak{g}$ es un *cociclo-2* si

$$w([X, Y], Z) + w([Y, Z], X) + w([Z, X], Y) = 0$$

para toda $X, Y, Z \in \mathfrak{g}$.

4.4 Soluciones de la Ecuación de Yang-Baxter Clásica

En esta sección, se dan varios resultados los cuales parametrizan y, en algunos casos, explícitamente describen, todas las soluciones de la EYBC para ciertas álgebras de Lie $\mathfrak{g}$.

4.4.1 Soluciones Constantes de la EYBC.

A El espacio de parámetros de soluciones no torcidas. Comenzamos con la clasificación de las soluciones no simétricas torcidas de la EYBC con valores en $\mathfrak{g} \otimes \mathfrak{g}$, lo cual se debe a Belavin & Drinfel'd [6, 7] (1982, 1984). Esto es equivalente a resolver los sistemas de ecuaciones

$$r_{12} + r_{21} = t\,, \qquad [[r, r]] = 0 \tag{4.4.1}$$

donde t es el elemento de Casimir de $\mathfrak{g} \otimes \mathfrak{g}$ que corresponde a una forma bilineal invariante $(\,,\,)$ sobre $\mathfrak{g}$ (este puede ser cualquier múltiplo no cero de la forma bilineal estándar). Observe que, si σ es cualquier automorfismo de $\mathfrak{g}$, entonces $(\sigma \otimes \sigma)(r)$ satisface (4.4.1) si r lo hace. Estas dos soluciones de (4.4.1) son *equivalentes* si pueden obtenerse desde una a la otra aplicando un automorfismo de este modo.

Para describir el espacio de parámetros de equivalencia de clases de soluciones de (4.4.1), se da la siguiente definición. Fije un conjunto Π de raíces simples de $\mathfrak{g}$.

Definición 4.4.1. Una *cuádrupla* $(\Pi_1, \Pi_0, \tau, r^0)$, *donde* $\Pi_1, \Pi_0 \subset \Pi$, τ *es una biyección* $\Pi_1 \to \Pi_0$, *y* $r^0 \in \mathfrak{h} \otimes \mathfrak{h}$, *se dice que es* admisible *si satisface las siguientes condiciones*:

(i) $(\tau(\alpha), \tau(\beta)) = (\alpha, \beta)$ para toda $\alpha, \beta \in \Pi_1$ *(también usamos* $(\,,\,)$ *para denotar el producto interno sobre* $\mathfrak{h}^*$ *inducido por aquél sobre* $\mathfrak{h}$*)*;

(ii) *para cada* $\alpha \in \Pi_1$, *existe* $m \in \mathbb{N}$ *tal que* $\alpha, \tau(\alpha), \ldots, \tau^{m-1}(\alpha) \in \Pi_1$ *pero* $\tau^m(\alpha) \notin \Pi_1$;

(iii) $r^0_{12} + r^0_{21} = t_0$, *la componente de* t *en* $\mathfrak{h}^{\otimes 2} \subset \mathfrak{g}^{\otimes 2} = (\mathfrak{n}_- \oplus \mathfrak{h} \oplus \mathfrak{n}_+)^{\otimes 2}$;

(iv) $(\tau(\alpha) \otimes 1)(r^0) + (1 \otimes \alpha)(r^0) = 0$ para toda $\alpha \in \Pi_1$.

También la tripla (Π_1, Π_0, τ) *que satisface las condiciones (i) y (ii) es admisible.*

Para una tripla admisible (Π_1, Π_0, τ), las soluciones r^0 del sistema de ecuaciones lineales inhomogéneas (iii) y (iv) forman un espacio afín cuya dimensión se puede calcular como sigue. Ya que las raíces simples forman una base de $\mathfrak{h}^*$, r^0 se determina completamente por la matriz de los números complejos $(r^0_{\alpha\beta})$ definidos por

$$r^0_{\alpha\beta} = (\alpha \otimes \beta)(r^0) \qquad (\alpha, \beta \in \Pi)\,.$$

Las condiciones (iii) y (iv) son equivalentes a las siguientes ecuaciones:

$$r^0_{\alpha\beta} + r^0_{\beta\alpha} = (\alpha,\beta)\ , \qquad \text{si } \alpha,\beta \in \Pi\,, \tag{4.4.2}$$

$$r^0_{\tau(\alpha)\beta} + r^0_{\beta\alpha} = 0\ , \qquad \text{si } \alpha \in \Pi_1,\ \beta \in \Pi\,. \tag{4.4.3}$$

Ahora, cada raíz simple $\gamma \in \Pi$ puede ser expresada únicamente en la forma $\gamma = \tau^{-m}(\alpha)$ para algún entero $m \geq 0$ y algún $\alpha \in \Pi \setminus \Pi_1$, (si $\gamma \notin \Pi_1$, tome $m = 0$; si $\gamma \in \Pi_1$, utilice la condición (ii). Si $\delta = \tau^{-l}(\beta)$, con $l \geq 0$ y $\beta \in \Pi \setminus \Pi_1$, se sigue de (4.4.2) y (4.4.3) que

$$r^0_{\gamma\delta} - r^0_{\alpha\beta} = \begin{cases} (\tau^{-1}(\alpha) + \cdots + \tau^{l-m}(\alpha), \beta) & \text{si } m > l, \\ -(\alpha, \beta + \tau^{-1}(\beta) + \cdots + \tau^{m-l+1}(\beta)) & \text{si } m < l, \\ 0 & \text{si } m = l, \end{cases}$$

y que, contrariamente, si prescribimos $r^0_{\alpha\beta}$ para $\alpha, \beta \in \Pi \setminus \Pi_1$ arbitrariamente, sujeta solamente a (4.4.2), y entonces determinamos r^0 utilizando las fórmulas precedentes, entonces las condiciones (iii) y (iv) se satisfacen. Por tanto, para una tripla admisible dada (Π_1, Π_0, τ), la dimensión (compleja) del espacio de soluciones de (iii) y (iv) es $\frac{1}{2}k(k-1)$, donde $k = |\,\Pi \setminus \Pi_1\,|$.

El espacio de parámetros de soluciones de (4.4.1) es esencialmente el espacio de cuádruplas admisibles:

Teorema 4.4.1. *El espacio de parámetros de clases de equivalencia de soluciones de (4.4.1) es el cociente del espacio de cuádruplas admisibles por la acción natural del grupo de automorfismo del diagrama de Dynkin de* $\mathfrak{g}$. *Tiene una estratificación en finitamente muchos estratos, indexados por las órbitas del grupo de diagramas de automorfismos de* $\mathfrak{g}$ *en el conjunto de triplas admisibles. El estrato correspondiente a la tripla* (Π_1, Π_0, τ) es de dimensiones complejas $\frac{1}{2}k(k-1)$, *donde* $k = |\,\Pi \setminus \Pi_1\,|$.

4.4.2 Soluciones de la EYBC con Parámetros Espectrales.

A Clasificación de las soluciones. Existe una generalización de la ecuación de Yang-Baxter clásica, la cual es particularmente importante en aplicaciones físicas de la teoría, en la cual la matriz-r r es una función sobre $U \times U$, para algún conjunto U, con valores en $\mathfrak{g} \otimes \mathfrak{g}$, que satisfacen

$$[r_{12}(u_1,u_2), r_{13}(u_1,u_3)] + [r_{12}(u_1,u_2), r_{23}(u_2,u_3)] + [r_{13}(u_1,u_3), r_{23}(u_2,u_3)] = 0. \tag{4.4.4}$$

Esta es llamada generalmente la EYBC con *parámetros espectrales* u_1 y u_2. Pero se referirá simplemente como la EYBC.

En esta sección se consideran las soluciones de (4.4.4) para la cual $\mathfrak{g}$ es un álgebra de Lie simple compleja de dimensión finita. Se asume que

(a) U es un subconjunto abierto del plano complejo, y r es una función meromórfica sobre $U \times U$, y

(b) $r(u_1, u_2)$ es no degenerada para al menos un punto $(u_1, u_2) \in U \times U$.

Las soluciones *constantes* simétricas torcidas no pueden ser no degeneradas.

Observe que (4.4.4) y las condiciones (a) y (b) se preservan si reemplazamos r por una matriz-r *equivalente*, como se describe en la siguiente definición.

Definición 4.4.2. *Sean r y $\tilde{r}$ dos soluciones de la EYBC las cuales son meromórficas sobre conjuntos abiertos $U \times U$ y $\tilde{U} \times \tilde{U}$, respectivamente. Entonces, r y $\tilde{r}$ son equivalentes si existe una transformación holomórfica $\alpha : \tilde{U} \to \mathrm{Aut}(\mathfrak{g})$ y una función biholomórfica no constante $f : \tilde{U} \to U$ tal que*

$$\tilde{r}(\tilde{u}_1, \tilde{u}_2) = (\alpha(\tilde{u}_1) \otimes \alpha(\tilde{u}_2))(r(f(\tilde{u}_1), f(\tilde{u}_2))).$$

Observe que el grupo $\mathrm{Aut}(\mathfrak{g})$ de automorfismos de álgebra de Lie de $\mathfrak{g}$ es un grupo de Lie complejo, así que ésta definición tiene sentido.

Ya se han encontrado soluciones racionales

$$r(u_1, u_2) = \frac{t}{u_1 - u_2}, \tag{4.4.5}$$

$$r(u_1, u_2) = r_{\mathfrak{g}} + \left(\frac{u_1}{u_2 - u_1}\right) t, \tag{4.4.6}$$

donde $r_{\mathfrak{g}}$ es la matriz-r estándar para $\mathfrak{g}$.

Observe que la solución en (4.4.5) es una función de $u_1 - u_2$ únicamente, mientras que la de (4.4.6) depende únicamente de u_1 / u_2, y, de aquí que, haciendo el cambio de variable $u_1 \mapsto e^{u_1}$, $u_2 \mapsto e^{u_2}$, es equivalente a una solución la cual es una función de $u_1 - u_2$ únicamente.

B Soluciones elípticas. Sea $\{\gamma_1, \gamma_2\}$ una base de la retícula Γ y ponga χ_i y $\chi(\gamma_i)$. De éste modo, χ_1 y χ_2 son automorfismos de $\mathfrak{g}$ de orden finito conmutantes sin vector fijo común. Empleando la clasificación del automorfismo de orden finito de las álgebras de Lie simples complejas, se muestra que hay un isomorfismo de álgebras de Lie $\mathfrak{g} \cong sl_{n+1}(\mathbb{C})$ bajo el cual χ_1 y χ_2 son conjugadas por las matrices

$$Z=\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \epsilon & 0 & \cdots & 0 \\ 0 & 0 & \epsilon^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \epsilon^{n-1} \end{pmatrix} \quad \text{y} \quad X=\begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

respectivamente, donde $\epsilon = \exp[2\pi\sqrt{-1}/(n+1)]$. Se asume que $\chi_1 = Z$ y $\chi_2 = X$ desde ahora.

Observe que las matrices $X^j Z^k$ para $(j, k) \neq (0, 0)$ forman una base de $sl_{n+1}(\mathbb{C})$ consistente de eigenvectores simultáneos de χ_1 y χ_2, con eigenvalores ϵ^{-j} y ϵ^{-k}, respectivamente.

C Soluciones trigonométricas. Se tiene que la clasificación de las soluciones trigonométricas no degeneradas de la EYBC es muy similar a aquella de las soluciones constantes. Esto se puede explicar por las siguientes observaciones.

Observe primero que cada solución es de la forma

$$r(u) = \frac{t}{e^u - 1} + \tilde{r}(u) \,, \tag{4.4.7}$$

en donde $\tilde{r}(u)$, es holomórfica. Un cálculo directo muestra que, si $\tilde{r}$ es *constante* entonces r satisface la EYBC si y solo si $\tilde{r}$ satisface (4.4.1).

En segunda, las soluciones trigonométricas de la EYBC surgen al considerar las estructuras de biálgebra de Lie sobre el álgebra de Lie afín asociada a $\mathfrak{g}$. Es bien sabido que las estructuras de las álgebras de Lie afines se parecen mucho a las álgebras de Lie simples complejas de dimensión finita.

D Soluciones racionales. Cada solución racional no degenerada es equivalente a una de la forma

$$r(u_1, u_2) = \frac{t}{u_1 - u_2} + \tilde{r}(u_1, u_2) \,, \tag{4.4.8}$$

donde $\tilde{r}$ es un polinomio en u_1 y u_2 con valores en $\mathfrak{g} \otimes \mathfrak{g}$. Como en el caso trigonométrico, un cálculo muestra que si $\tilde{r}$ es constante, entonces (4.4.8) se cumple si y solo si $\tilde{r}$ es una solución simétrica torcida de la EYBC. El caso general no es mucho más complicado.

4.5 Álgebras de Hopf Cuasitriangulares

El espacio $\mathcal{F}(M)$ de funciones C^∞ sobre una variedad suave M forma un álgebra conmutativa bajo el punto de vista de la adición y la multiplicación. Si G es un grupo de Lie, la multiplicación y las transformaciones inversas de G proveen con estructura extra a $\mathcal{F}(G)$, llamada la transformación de *comultiplicación* $\mathcal{F}(G) \to \mathcal{F}(G) \otimes \mathcal{F}(G)$ (aquí se debe definir apropiadamente el producto tensor). Estas transformaciones satisfacen ciertas condiciones de compatibilidad que reflejan la asociatividad de la multiplicación en G y las propiedades que definen la inversa. La abstracción de estas propiedades conduce a la noción de un *álgebra de Hopf* (la cual no necesita ser conmutativa como un álgebra).

De hecho, a cualquier grupo de Lie G se le puede asociar una segunda álgebra de Hopf, el *álgebra envolvente universal* $U(\mathfrak{g})$ del álgebra de Lie $\mathfrak{g}$ de G. El álgebra envolvente universal no es conmutativa (a menos que G lo sea), pero siempre es coconmutativa, lo cual significa que la comultiplicación de $U(\mathfrak{g})$ toma valores en la parte simétrica de $U(\mathfrak{g}) \otimes U(\mathfrak{g})$.

Antes de la llegada de los grupos cuánticos, muy pocas álgebras de Hopf las cuales no son conmutativas ni coconmutativas eran conocidas (además de los productos tensores de las álgebras de funciones y las álgebras envolventes), pero la teoría de los grupos cuánticos proporciona una clase amplia de ejemplos naturales.

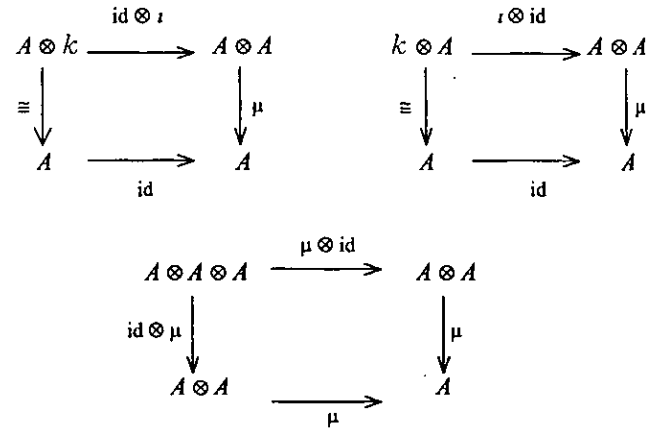
La transformación de la comultiplicación $A \to A \otimes A$ de un álgebra de Hopf arbitraria A permite definir el producto tensor $V \otimes W$ de las dos representaciones V y W de A. Si A es no conmutativa, $V \otimes W$ y $W \otimes V$ no necesariamente son isomórficos como representaciones de A, aunque son isomórficos para la clase especial de álgebras de Hopf *cuasitriangulares*. Tales álgebras de Hopf contienen un distinguido elemento invertible $\mathcal{R} \in A \otimes A$, llamado la *matriz-R universal*, desde la cual el isomorfismo $V \otimes W \to W \otimes V$ se construye. La mayoría de los grupos cuánticos que vamos a encontrar (o sus duales) son cuasitriangulares, y muchas de las aplicaciones de la teoría de grupos cuánticos hacen uso esencial de sus matrices-R universales. La propiedad crucial de $\mathcal{R}$ en estas aplicaciones es que satisface la *ecuación de Yang-Baxter cuántica*

$$\mathcal{R}_{12}\,\mathcal{R}_{13}\,\mathcal{R}_{23} = \mathcal{R}_{23}\,\mathcal{R}_{13}\,\mathcal{R}_{12}\,.$$

4.5.1 Álgebras de Hopf.

A Definiciones. Un álgebra con la unidad sobre un anillo conmutativo k (siempre suponiendo que tenga una unidad por sí misma) es un módulo-k A junto con una forma de multiplicar dos elementos $a_1, a_2 \in A$ para dar su producto $a_1.a_2 \in A$. La multiplicación debe ser bilineal sobre k y asociativa. El elemento unidad 1 de A tiene la propiedad $a.1 = 1.a = a$ para toda $a \in A$. Ahora reformularemos esto en términos de diagramas conmutativos.

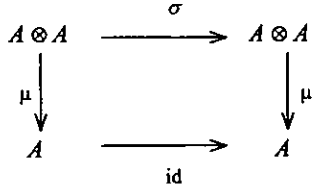
Definición 4.5.1. *Un* álgebra *sobre un anillo conmutativo k es un módulo-k A provisto con transformaciones módulo-k $\mu^A : A \underset{k}{\otimes} A \to A$, la* multiplicación, *y $\iota^A : A \to A$, la* unidad, *tal que los diagramas siguientes conmutan*:



(se omitió el superíndice sobre μ, ι y el subíndice sobre el producto tensor cuando no se causa confusión).

Por supuesto, en términos de la más familiar descripción de un álgebra dada primeramente, tenemos $\iota(\lambda) = \lambda 1$, $\mu(\alpha_1 \otimes \alpha_2) = \alpha_1.\alpha_2$. Los primeros dos diagramas expresan las propiedades del elemento unidad y el tercero la asociatividad de la multiplicación.

La condición de que un álgebra sea conmutativa puede similarmente ser expresada por la conmutatividad del diagrama



donde $\sigma : A \otimes A \to A \otimes A$ es la transformación lineal-k tal que $\sigma(\alpha_1 \otimes \alpha_2) = \alpha_2 \otimes \alpha_1$ para α_1, $\alpha_2 \in A$. En general, si se establece $\mu_{op} = \mu \circ \sigma$, entonces (A, ι, μ_{op}) es también un álgebra, llamada la *opuesta* de A y denotada por A_{op}. Observe que k por sí misma es un álgebra conmutativa sobre k.

Si A y B son álgebras sobre k, su producto tensor $A \underset{k}{\otimes} A$ es también un álgebra. La operación producto

$$(\alpha_1 \otimes b_1).(\alpha_2 \otimes b_2) = \alpha_1.\alpha_2 \otimes b_1.b_2, \qquad (\alpha_1, \alpha_2 \in A, \ \ b_1, b_2 \in B)$$

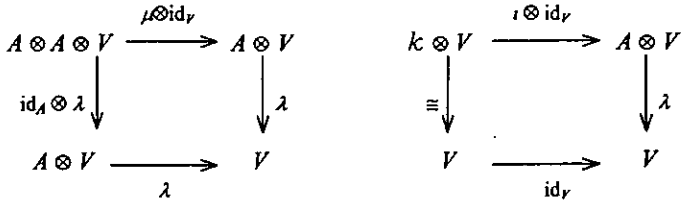
en $A \otimes B$ se puede expresar como la composición

$$(A\otimes B)\otimes(A\otimes B)\xrightarrow{\sigma_{23}} A\otimes A\otimes B\otimes B\xrightarrow{\mu^A\otimes\mu^B} A\otimes B\ ,$$

donde σ_{23} es σ aplicada al segundo y tercer factores del producto tensor, y la unidad de $A \otimes B$ es la compuesta.

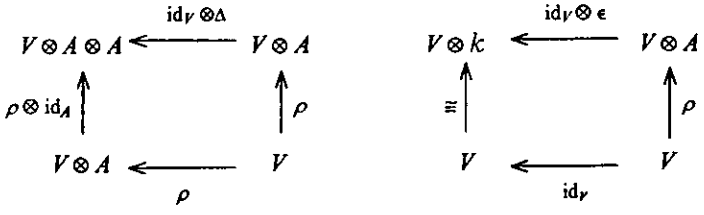
$$k\xrightarrow{\equiv} k\otimes k\xrightarrow{\iota^A\otimes\iota^B} A\otimes B\ .$$

B Representaciones de álgebras de Hopf. Si A es un álgebra sobre un anillo conmutativo k, un módulo-k V se llama un *módulo-A izquierdo* si existe una transformación de módulo-k $\lambda_V : A \otimes V \to V$ tal que los siguientes diagramas conmutan:



(se omite el subíndice si es inambiguo). Equivalentemente, $a \mapsto \lambda\ (a \otimes \cdot)$ debería ser un homomorfismo de álgebras desde A en los endomorfismos de V. Deberemos algunas veces escribir $\lambda(a \otimes v)$ como $a.v$ si la acción se entiende. Los módulos-A derechos son definidos similarmente.

Si A es una coálgebra sobre un anillo conmutativo k, un módulo-k V se llama un *comódulo-A derecho* si existe una transformación de módulo-k $\rho_V : V \to V \otimes A$ tal que los siguientes diagramas conmutan:



Si V es por sí misma un álgebra o una coálgebra, es natural requerir que éste módulo y las estructuras comódulo deban respetar la estructura extra sobre V. Asuma que A es una biálgebra. Un álgebra V es un álgebra *módulo-A izquierda* si es un módulo-A izquierdo y

$$a.(vw)=\sum_i (a_i.v)(a^i.w)\ ,\qquad a.1=\epsilon_A(a)1$$

para toda $a \in A$, $v, w \in V$, donde $\Delta_A(a) = \sum_i a_i \otimes a^i$.

Una coálgebra V es *una coálgebra módulo-A izquierda* si es un módulo-A izquierdo y

$$\Delta_V(a.v) = \sum_{i,j} a_i.v_j \otimes a^i.v^j, \qquad \epsilon_V(a.v) = \epsilon_A(a)\,\epsilon_V(v)$$

para toda $a \in A$, $v \in V$, donde $\Delta_V(v) = \sum_j v_j \otimes v^j$.

4.5.2 Álgebras de Hopf Cuasitriangulares.

Existen varios resultados en la literatura los cuales aseveran, bajo alguna hipótesis técnica adicional, que cada álgebra de Hopf coconmutativa es isomórfica al álgebra envolvente universal de un álgebra de Lie (y, dualmente, que cada álgebra de Hopf conmutativa es isomórfica al álgebra de funciones sobre de un grupo). La teoría de grupos cuánticos proporciona una gran familia de álgebras de Hopf las cuales no son conmutativas ni coconmutativas. Sin embargo, muchos de estos ejemplos (o sus duales) son 'casi conmutativos' en cierto sentido.

A Álgebras de Hopf casi coconmutativas. Un álgebra de Hopf es casi coconmutativa si su comultiplicación y su comultiplicación opuesta difieren por conjugación por un distinguido elemento invertible de $A \otimes A$.

Definición 4.5.2. *Un álgebra de Hopf A sobre un anillo conmutativo k se dice que es* casi coconmutativa *si existe un elemento invertible $\mathcal{R} \in A \otimes A$* tal que

$$\Delta^{op}(a) = \mathcal{R}\Delta(a)\,\mathcal{R}^{-1} \tag{4.5.1}$$

para toda $a \in A$.

B Álgebras de Hopf cuasitrangulares. Es claro que el elemento $\mathcal{R}$ en (4.5.1) no puede ser arbitrario ya que sabemos que A^{op} es un Álgebra de Hopf. De hecho,

$$(\Delta^{op} \otimes \mathrm{id})\Delta^{op}(a) = \mathcal{R}_{12}.(\Delta \otimes \mathrm{id})(\mathcal{R}).(\Delta \otimes \mathrm{id})\,\Delta(a).(\Delta \otimes \mathrm{id})\mathcal{R}^{-1}.\mathcal{R}_{12}^{-1},$$
$$(\mathrm{id} \otimes \Delta^{op})\Delta^{op}(a) = \mathcal{R}_{23}.(\mathrm{id} \otimes \Delta)(\mathcal{R}).(\mathrm{id} \otimes \Delta)\Delta(a).\,(\mathrm{id} \otimes \Delta)\mathcal{R}^{-1}.\mathcal{R}_{23}^{-1}.$$

De aquí que, una condición suficiente para la coasociatividad de Δ^{op} es

$$\mathcal{R}_{12}.(\Delta \otimes \mathrm{id})(\mathcal{R}) = \mathcal{R}_{23}.(\mathrm{id} \otimes \Delta)(\mathcal{R}). \tag{4.5.2}$$

Similarmente, una condición suficiente para las propiedades $(\epsilon \otimes \mathrm{id}) \circ \Delta^{op} = \mathrm{id}$ e $(\mathrm{id} \otimes \epsilon) \circ \Delta^{op} = \mathrm{id}$ de la counidad para mantenerse es que $(\epsilon \otimes \mathrm{id})(\mathcal{R}) = (\mathrm{id} \otimes \epsilon)(\mathcal{R}) = 1$.

Será conveniente hacer una afirmación ligeramente más fuerte.

Definición 4.5.3. *Una álgebra de Hopf casi coconmutativa $(A, \mathcal{R})$ se dice ser*

(i) colímite *si $\mathcal{R}$satisface (4.5.2), $\mathcal{R}_{21} = \mathcal{R}^{-1}$* y $(\epsilon \otimes \epsilon)(\mathcal{R}) = 1$;

(ii) cuasitriangular *si*

$$(\Delta \otimes \mathrm{id})(\mathcal{R}) = \mathcal{R}_{13}\mathcal{R}_{23}, \tag{4.5.3}$$
$$(\mathrm{id} \otimes \Delta)(\mathcal{R}) = \mathcal{R}_{13}\mathcal{R}_{12}; \tag{4.5.4}$$

(iii) triangular *si es cuasitringular, y, además, $\mathcal{R}_{21} = \mathcal{R}^{-1}$.*

Si A es cuasitriangular, el elemento $\mathcal{R}$ es llamado la matriz-R universal *de $(A, \mathcal{R})$.*

4.6 Representaciones y Categorías Cuasitensoras

Es bien sabido que las propiedades de la operación de tomar las sumas directas (de los espacios vectoriales, de las representaciones de grupo, . . .) pueden ser abstraídas en la noción de una categoría abeliana. Abstraer las propiedades de la operación producto tensor (sobre los espacios vectoriales, sobre las representaciones de grupo, . . .) conduce a la noción de una *categoría monoide*. A pesar de que la teoría básica de tales categorías fue desarrollada a principios de los 1960s por S. MacLane [**69**], el tema ha atraído la atención de una amplia audiencia recientemente, debido al descubrimiento de nuevos ejemplos interesantes. No sorprende que muchos de éstos se relacionan con los grupos cuánticos.

Existen varios resultados los cuales afirman, aproximadamente hablando, que una categoría monoidal con algunas propiedades adicionales 'es' la categoría de representaciones de algún grupo: esta es la *dualidad de Tannaka-Krein*. Tal vez el ejemplo más simple es la dualidad de Pontryagin para grupos abelianos localmente compactos. Para tales grupos G, todas las representaciones son sumas directas de las representaciones de dimensión-1, o *caracteres*. El conjunto de caracteres de G puede a sí mismo darse una estructura natural de un grupo $\hat{G}$ abeliano localmente compacto. El teorema de Pontryagin afirma que G puede ser recuperado completamente desde $\hat{G}$: de hecho, G es isomórfico al grupo de caracteres de $\hat{G}$.

En muchos ejemplos que ocurren naturalmente de categorías monoides, la operación producto tensor es conmutativa hasta isomorfismo, y los isomorfismos conmutativos son involutivos: entonces se tiene una *categoría tensora*. Por ejemplo, la categoría de representaciones de un grupo obviamente tiene esta propiedad. Más generalmente la categoría de representaciones de cualquier álgebra de Hopf triangular es una categoría tensora. Si disminuimos 'triangular' a 'cuasitriangular', el producto tensor aún es conmutativo, pero los isomorfismos conmutativos ya no son involutivos, en general. Se dice entonces que la categoría de representaciones es una categoría *cuasitensora* (o *monoide trenzado*).

En una categoría cuasitensora abeliana semisimple, el producto tensor está determinado por la descomposición en objetos simples del producto tensor de cualesquiera dos objetos simples, y por ende por una colección de enteros no negativos los cuales satisfacen ciertas condiciones las cuales reflejan las propiedades de conmutatividad y asociatividad del producto tensor. Encontramos datos combinatorios de esta clase en el estudio de teorías de campos conformes, donde eran llamadas *reglas de fusión*.

Los *enredos de cinta* son esencialmente colecciones de bandas de cintas no intersectadas en $\mathbb{R}^3$, ya sea cerradas o abiertas, y con anchura finita pero espesor cero. Los enredos de cinta pueden organizarse en una categoría cuasitensora, y las invariantes se obtienen construyendo un

functor desde ésta categoría a la categoría de representaciones de un tipo particular de álgebra de Hopf cuasitriangular. Si olvidamos el ancho de las bandas, y consideramos únicamente enredos de cinta sin finales libres, obtenemos un caso especial de invariantes de enlaces en $\mathbb{R}^3$. El celebrado *polinomio de Jones* puede ser construido de esta forma.

4.6.1 Categorías Monoides.

A Categorías abelianas. Como ya se mencionó, la noción de una categoría abeliana está diseñada para tomar las propiedades de las sumas directas de los espacios vectoriales, de las representaciones de grupo, etc. En general, una categoría C *admite sumas directas (finitas)* si, para cualquier conjunto finito de objetos $U_1, \ldots, U_n$ de C, existe un objeto U y morfismos $\pi_i : U \to U_i$ tales que, para cualquier objeto V y morfismos $f_i : V \to U_i$, existe un morfismo único $f : V \to U$ tal que $\pi_i \circ f = f_i$ para toda i. El objeto U entonces es único hasta isomorfismo; escribimos $U = \oplus_i U_i$ y $f = \oplus_i f_i$.

Una categoría la cual admite sumas directas se llama *aditiva* si, para cualesquiera objetos U y V, el conjunto $\mathrm{Hom}_C(U, V)$ de morfismos desde U hasta V es un grupo abeliano, y si las composiciones

$$\mathrm{Hom}_C(U, V) \times \mathrm{Hom}_C(V, W) \to \mathrm{Hom}_C(U, W) \tag{4.6.1}$$

son bi-aditivas.

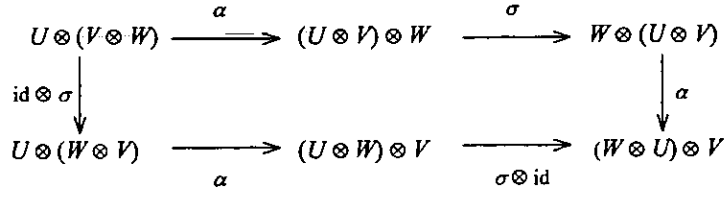
4.6.2 Categorías Cuasitensoras.

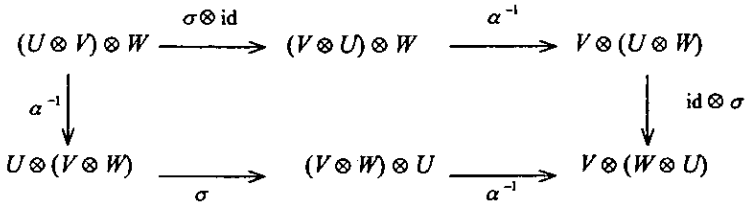
En muchas categorías monoides, tales como las categorías de espacios vectoriales y representaciones de grupo, el producto tensor es conmutativo hasta isomorfismo.

A Categorías tensoras.

Definición 4.6.1. Una categoría tensora *es una categoría monoide* C *la cual está provista con isomorfismos naturales* $\sigma_{U,V} : U \otimes V \to V \otimes U$, *para todos los objetos* U *y* V *de* C, *tales que*

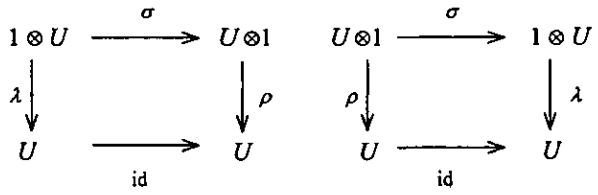
(i) los diagramas





conmutan para todos U, V, W;

(ii) los diagramas

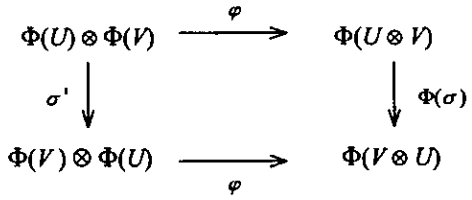


conmutan para toda U;

(iii) $\sigma_{V,U} \circ \sigma_{U,V} = id_{U \otimes V}$ *para toda U, V.*

Si C *es abeliana, se requiere que* σ *sea bi-aditiva.*

Un functor monoide $\Phi : C \to C'$ *entre categorías tensoras es un functor tensor si el diagrama*



conmuta para toda U, V.

B Categorías cuasitensoras.

Definición 4.6.2. *Una categoría cuasitensora es una categoría monoide* C *provista con isomorfismo functorial* $\sigma_{U,V} : U \otimes V \to V \otimes U$ *la cual satisface las condiciones (i) y (ii) de 4.6.1.*

4.7 Cuantificación de Biálgebras de Lie

Los ejemplos más importantes de grupos cuánticos son las deformaciones de las álgebras envolventes universales y de las álgebras de funciones sobre grupos.

Las obstrucciones para la existencia y unicidad de deformaciones de una clase dada de objetos matemáticos están descritas generalmente por alguna teoría de cohomología de los objetos en cuestión.

4.7.1 Deformaciones de Álgebras de Hopf.

A Definiciones

Definición 4.7.1. Una deformación *de un álgebra de Hopf* $(A, \iota, \mu, \epsilon, \Delta, S)$ *sobre un campo* k *es un álgebra de Hopf topológica* $(A_h, \iota_h, \mu_h, \epsilon_h, \Delta_h, S_h)$ *sobre el anillo* $k[[h]]$ *de las series de potencia formales en una* h *indeterminada sobre* k, *tal que*

(i) A_h *es isomórfica a* $A[[h]]$ *como un módulo-*$k[[h]]$;

(ii) $\mu_h \equiv \mu \pmod h$, $\Delta_h \equiv \Delta \pmod h$.

Dos deformaciones A_h y A'_h *son* equivalentes *si hay un isomorfismo* $f_h : A_h \to A'_h$ *de álgebras de Hopf sobre* $k[[h]]$ *la cual es la identidad (mod* h*).*

B Teoremas de rigor. Estaremos particularmente interesados en las deformaciones A_h del álgebra envolvente universal $U(\mathfrak{g})$ de un álgebra de Lie $\mathfrak{g}$, y del álgebra $\mathcal{F}(G)$ de funciones regulares sobre un grupo algebraico G. Para estas álgebras de Hopf A, $H^2(A, A)$ es (afortunadamente) en general no cero, así que son posibles las deformaciones no triviales.

Definición 4.7.2. *Una deformación de la álgebra de Hopf del álgebra envolvente universal* $U(\mathfrak{g})$ *de un álgebra de Lie* $\mathfrak{g}$ *se llama un álgebra* envolvente universal cuantificada, *o simplemente álgebra EUC.*

Una deformación del álgebra de Hopf del álgebra $\mathcal{F}(G)$ *de las funciones regulares sobre un grupo algebraico G se llama un álgebra* de función cuantificada, *o simplemente un álgebra FC.*

Teorema 4.7.8. *Sea G un grupo algebraico reductivo sobre un campo* k *de característica cero*, y *sea* $\mathfrak{g}$ *su álgebra de Lie. Sea* $\wedge^m(\mathfrak{g})^{\mathfrak{g}}$ *el subespacio de* $\wedge^m(\mathfrak{g})$ *que consiste de los elementos los cuales son invariables bajo la acción adjunta de* $\mathfrak{g}$.

(i) Para toda $m \geq 1$,

$$H^m(\mathcal{F}(G), \mathcal{F}(G)) \cong \wedge^m(\mathfrak{g}) \,/\, \wedge^m(\mathfrak{g})^{\mathfrak{g}}, \qquad H^m_{\text{coalg}}(\mathcal{F}(G), \mathcal{F}(G)) = 0.$$

Entonces, cada deformación de $\mathcal{F}(G)$ *es isomórfica a* $\mathcal{F}(G)[[h]]$ *como una coálgebra.*

(ii) *Si* $\mathfrak{g}$ *es semisimple,*

$$H^2(U(\mathfrak{g}), U(\mathfrak{g})) \cong \wedge^2(\mathfrak{g}), \quad H^3(U(\mathfrak{g}), U(\mathfrak{g})) \cong \wedge^3(\mathfrak{g})/\wedge^3(\mathfrak{g})^{\mathfrak{g}},$$
$$H^2_{\text{alg}}(U(\mathfrak{g}), U(\mathfrak{g})) = 0.$$

Entonces, cada deformación de $U(\mathfrak{g})$ *es isomórfica a* $U(\mathfrak{g})[[h]]$ *como un álgebra.*

4.7.2 Cuantificación.

A Álgebras de (Co-) Poisson-Hopf. Se ha argumentado que una 'cuantificación' de una variedad de Poisson M puede interpretarse como una deformación $\mathcal{F}_h(M)$ del álgebra de Poisson $\mathcal{F}(M)$ de las funciones C^∞ sobre M, tal que la 'parte de primer orden' del conmutador de dos elementos de $\mathcal{F}_h(M)$ es igual al corchete de Poisson de sus límites clásicos. Si G es un grupo de Poisson-Lie, entonces $\mathcal{F}(G)$ es también un álgebra de Hopf, y las dos estructuras son compatibles. De hecho, al nivel del grupo la condición de compatibilidad es

$$\{f_1 \circ m, f_2 \circ m\}_{G\times G} = \{f_1, f_2\}_G \circ m,$$

donde $f_1, f_2 \in \mathcal{F}(G)$, $m : G \times G \to G$ es la multiplicación en G y $\{\ ,\ \}_{G\times G}$ y $\{\ ,\ \}_G$ son los corchetes de Poisson de $G \times G$ y G. Ya que $f_i \circ m = \Delta(f_i)$, la ecuación precedente se hace

$$\{\Delta(f_1), \Delta(f_2)\}_{G\times G} = \Delta(\{f_1, f_2\}).$$

Esto sugiere la siguiente definición.

Definición 4.7.3. Un álgebra de Poisson *sobre un anillo conmutativo* k *es un álgebra conmutativa* A *sobre* k *provista con una transformación módulo-*k *simétrica torcida* $\{\ ,\ \}_A : A \otimes A \to A$, *el* corchete de Poisson, *tal que*:

(i) *la* identidad de Jacobi

$$\{a_1, \{a_2, a_3\}_A\}_A + \{a_3, \{a_1, a_2\}_A\}_A + \{a_2, \{a_3, a_1\}_A\}_A = 0$$

se mantiene para toda $a_1, a_2, a_3 \in A$;

(ii) *la* identidad de Leibniz

$$\{a_1a_2, a\}_A = \{a_1, a\}_A a_2 + a_1\{a_2, a\}_A$$

se mantiene para toda $a_1, a_2, a \in A$.

Un álgebra de Poisson-Hopf *sobre un anillo conmutativo* k *es un álgebra de Poisson* $(A, \{\ ,\ \}_A)$ *sobre* k, *la cual es también un álgebra de Hopf* $(A, \iota, \mu, \epsilon, \Delta, S)$ *sobre* k, *las dos estructuras siendo compatibles en el sentido que*

(iii) para *toda* $a_1, a_2 \in A$,

$$\{\Delta(a_1), \Delta(a_2)\}_{A\otimes A} = \Delta(\{a_1, a_2\}_A).$$

El corchete de Poisson $\{\ ,\ \}_{A\otimes A}$ sobre $A\otimes A$ está definido por

$$\{a_1\otimes a'_1\,,\,a_2\otimes a'_2\}_{A\otimes A}=\{a_1\,,\,a_2\}_A\otimes a'_1a'_2+a_1a_2\otimes\{a'_1\,,\,a'_2\}_A$$

Para discutir las estructuras correspondientes sobre álgebras envolventes universales, debemos dualizar 4.2.1.

Definición 4.7.4. Un álgebra de co-Poisson *sobre un anillo conmutativo k es una coálgebra conmutativa (A, ϵ, Δ) provista con una transformación módulo-k simétrica torcida $\delta: A\to A\otimes A$, el* co-corchete de Poisson, *que satisface las siguientes condiciones:*

(i) la compuesta

$$A\xrightarrow{\ \delta\ }A\otimes A\xrightarrow{\ \delta\otimes id\ }A\otimes A\otimes A\xrightarrow{\ \text{p.c.}\ }A\otimes A\otimes A$$

es cero, donde 'p.c.' significa la suma sobre permutaciones cíclicas de los factores en el producto tensor triple (esta es la identidad de co-Jacobi*);*

(ii) la identidad de co-Leibniz

$$(\Delta\otimes\mathrm{id})\,\delta=(\mathrm{id}\otimes\delta)\,\Delta+\sigma_{23}(\delta\otimes\mathrm{id})\,\Delta$$

se mantiene

Un álgebra de co-Poisson-Hopf es un álgebra de co-Poisson $(A, \epsilon, \Delta, \delta)$ la cual es también un álgebra de Hopf $(A, \iota, \mu, \epsilon, \Delta, S)$, las dos estructuras siendo compatibles en el sentido que

(iii) para toda $a_1, a_2 \in A$,

$$\delta(a_1a_2)=\delta(a_1)\Delta(a_2)+\Delta(a_1)\,\delta(a_2).$$

B Cuantificación.

Definición 4.7.5. *Sea A un álgebra de Poisson-Hopf conmutativa sobre un campo k de característica cero, y sea $\{\ ,\ \}$ su corchete de Poisson. Una* cuantificación *de A es una deformación del álgebra de Hopf A_h de A tal que*

$$\{x_1,x_2\}\equiv\frac{a_1a_2-a_2a_1}{h}(\mathrm{mod}\ h),$$

si $a_1, a_2 \in A_h$ se reduce a $x_1,\ x_2\ \in A$ (mod h).

Una cuantificación de un grupo de Poisson-Lie algebraico $(G,\{\ ,\ \})$ *es una cuantificación $\mathcal{F}_h(G)$ del álgebra $\mathcal{F}(G)$ de funciones regulares sobre G, considerada como un álgebra de Poisson, y $(G,\{\ ,\ \})$ se llama el* límite clásico *de $\mathcal{F}_h(G)$.*

◆

5

Un Enfoque Combinatorio de la Cuantificación de las Álgebras de Lie

En este capítulo se consideran las ideas nuevas en la cuantificación de las álgebras de Lie las cuales están siendo desarrolladas por V.K. Kharchenko [56]. Después se utilizará este método para investigar las cuantificaciones de las álgebras de Lie del tipo G_2.

5.1. Introducción

Las álgebras envolventes universales cuánticas aparecieron en los famosos artículos de Drinfeld [27] y Jimbo [43]. Desde entonces una gran cantidad de artículos y número de monografías se han dedicado a su investigación. Todas estas investigaciones se relacionan principalmente con una cuantificación particular de las álgebras de Lie de la serie clásica. Esto se explica primero por el hecho de que éstas álgebras de Lie tienen aplicaciones e interpretaciones visuales en especulaciones físicas, y luego por el hecho de que aún no se ha elaborado una noción general y comúnmente aceptada como estándar de un álgebra envolvente universal cuántica (ver una discusión detallada en [4, 68]). En el artículo de V.K. Kharchenko [56] se propone una solución combinatoria de este problema por medio del concepto de la operación (de Lie) cuántica [50, 53, 54]. En línea con la principal idea de tal enfoque, los elementos primitivos torcidos deben de jugar el mismo papel en las álgebras envolventes cuánticas de como lo hacen los elementos primitivos en el caso clásico. Por el criterio de Friederichs [23, 37, 67, 70, 76], los elementos primitivos forman el álgebra de Lie fundamental en el caso clásico. Por esta razón, se considera el espacio comprendido por los elementos primitivos torcidos y provistos con las operaciones cuánticas como una analogía cuántica de un álgebra de Lie.

5.2 Álgebras Envolventes Cuánticas

En esta sección se citan las nociones principales y se consideran algunos ejemplos, que en particular, muestran que las álgebras envolventes de Drinfeld-Jimbo así como sus modificaciones son álgebras envolventes cuánticas en nuestro sentir.

Recuerde que una variable x se llama una *variable cuántica* si un elemento g_x de un grupo abeliano fijo G y un caracter $\chi^x \in G^*$ se asocian con ella. Un polinomio no conmutativo en variables cuánticas, se llama una *operación cuántica* si todos sus valores, en todas las álgebras de Hopf, son primitivos torcidos dado que cada variable x tiene un valor $x = a$ tal que

$$\Delta(a) = a \otimes 1 + g_x \otimes a\,, \qquad g^{-1}ag = \chi^x(g)a\,, \qquad g \in G\,. \tag{5.2.1}$$

Sea $x_1, \ldots, x_n$ un conjunto de variables cuánticas. Para cada palabra u en $x_1, \ldots, x_n$ denotamos por g_u un elemento de G que se origina de u por el reemplazo de todas las x_i por g_{x_i}. En la misma forma, denotamos por χ^u un caracter que aparece de u por el reemplazo de todas las x_i con χ^{x_i}. Así sobre el álgebra libre $\mathbf{k}\langle x_1, \ldots, x_n\rangle$ está definida una graduación para el grupo $G \times G^*$, y para cada par de elementos homogéneos u, v fijamos las denotaciones $p_{uv} = \chi^u(g_v) = p(u,v)$.

La operación cuántica puede ser definida equivalentemente como un polinomio homogéneo $G \times 1$, el cual tiene solamente valores primitivos en todas las *álgebras de Hopf bigraduadas trenzadas*, dado que todas las variables cuánticas tienen valores homogéneos primitivos $g_a = g_x$, $\chi^a = \chi^x$ (ver **[50]**, secc. 1-4).

Recuerde que una *constitución* o multigrado de una palabra u es una secuencia de enteros no negativos $(m_1, m_2, \ldots, m_n)$ tal que u es de grado m_1 en x_1, $\mathrm{grad}_1(u) = m_1$; de grado m_2 en x_2, $\mathrm{grad}_2(u) = m_2$; y así sucesivamente. Ya que el grupo G es abeliano, todos los polinomios de constitución homogénea son homogéneos con respecto a la graduación. Definamos un conmutador torcido bilineal sobre el conjunto de polinomios no conmutativos homogéneos graduados por la fórmula

$$[u, v] = uv - p_{uv}vu\,. \tag{5.2.2}$$

Estos corchetes satisfacen las siguientes identidades diferenciales torcidas y de Jacobi:

$$[[u,v],w] = [u,[v,w]] + p_{wv}^{-1}[[u,w],v] + (p_{vw} - p_{wv}^{-1})[u,w]\cdot v\,; \tag{5.2.3}$$

$$[[u, v], w] = [u, [v, w]] + p_{vw}[[u, w], v] + p_{uv}(p_{vw}p_{wv} - 1)v \cdot [u, w]\,; \tag{5.2.4}$$

$$[u,v\cdot w] = [u,v]\cdot w + p_{uv}v\cdot[u,w]\,; \qquad [u\cdot v,w] = p_{vw}[u,w]\cdot v + u\cdot[v,w]\,; \tag{5.2.5}$$

donde por el punto se denota la multiplicación usual.

Para las pruebas de estas identidades se emplea la relación (5.2.2).

Prueba de (5.2.3):

$$[[u,v],w]=[u,[v,w]]+p_{wv}^{-1}[[u,w],v]+(p_{vw}-p_{wv}^{-1})[u,w]\cdot v\ ; \qquad (5.2.3)$$

$$\begin{aligned}[[u,\,v],\,w] &= [uv - p_{uv}vu,\,w]\\ &= uvw - p_{uv}vuw - p_{uv,w}(wuv - p_{uv}wvu)\\ &= uvw - p_{uv}vuw - p_{uv,w}wuv + p_{uv,w}p_{uv}wvu\\ &= uvw - p_{uv}vuw - p_{uw}p_{vw}wuv + p_{uw}p_{vw}p_{uv}wvu\ . \end{aligned} \qquad (5.2.3.a)$$

Desarrollando los términos que componen el lado derecho de (5.2.3):

$$[u,[v,w]]+p_{wv}^{-1}[[u,w],v]+(p_{vw}-p_{wv}^{-1})[u,w]\cdot v\ ;$$

$$\begin{aligned}[u,\,[v,\,w]] &= [u,\,vw - p_{vw}wv]\\ &= uvw - p_{vw}uwv - p_{u,vw}(vwu - p_{vw}wvu)\\ &= uvw - p_{vw}uwv - p_{uv}p_{uw}vwu + p_{uv}p_{uw}p_{vw}wvu\ , \end{aligned} \qquad (5.2.3.b)$$

$$\begin{aligned}p_{wv}^{-1}[[u,w],v] &= p_{wv}^{-1}([uw - p_{uw}wu,\,v])\\ &= p_{wv}^{-1}(uwv - p_{uw}wuv - p_{uw,v}(vuw - p_{uw}vwu))\\ &= p_{wv}^{-1}uwv - p_{wv}^{-1}p_{uw}wuv - p_{wv}^{-1}p_{uv}p_{wv}vuw +\\ &\quad p_{wv}^{-1}p_{uv}p_{wv}p_{uw}vwu\\ &= p_{wv}^{-1}uwv - p_{wv}^{-1}p_{uw}wuv - p_{uv}vuw + p_{uv}p_{uw}vwu\ , \end{aligned} \qquad (5.2.3.c)$$

$$\begin{aligned}(p_{vw}-p_{wv}^{-1})[u,\,w]\cdot v &= (p_{vw}-p_{wv}^{-1})(uw - p_{uw}wu)\cdot v\\ &= p_{vw}uwv - p_{vw}p_{uw}wuv - p_{wv}^{-1}uwv + p_{wv}^{-1}p_{uw}wuv\ . \end{aligned} \qquad (5.2.3.d)$$

Sumando los resultados parciales (5.2.3.b) + (5.2.3.c) + (5.2.3.d):

$$uvw - p_{vw}uwv - p_{uv}p_{uw}vwu + p_{uv}p_{uw}p_{vw}wvu + p_{wv}^{-1}uwv -$$

$$p_{wv}^{-1}p_{uw}wuv - p_{uv}vuw + p_{uv}p_{uw}vwu + p_{vw}uwv - p_{vw}p_{uw}wuv$$

$$- p_{wv}^{-1}uwv + p_{wv}^{-1}p_{uw}wuv =$$

$$uvw + p_{uv}p_{uw}p_{vw}wvu - p_{uv}vuw - p_{vw}p_{uw}wuv\ .$$

Los términos marcados con " ∘ " se simplifican a cero, e igualando este resultado con (5.2.3.a) se obtiene la igualdad, con lo cual la relación (5.2.3) queda comprobada.

Prueba de (5.2.4):

$$[[u, v], w] = [u, [v, w]] + p_{vw}[[u, w], v] + p_{uv}(p_{vw}p_{wv} - 1)v \cdot [u, w] . \quad (5.2.4)$$

El lado izquierdo de (5.2.4) es igual al de (5.2.3) por lo cual no se indica su desarrollo, pero se escribe nuevamente el resultado.

$$[[u, v], w] = uvw - p_{uv}vuw - p_{uw}p_{vw}wuv + p_{uw}p_{vw}p_{uv}wvu . \quad (5.2.4.a)$$

Desarrollando los términos que componen el lado derecho de (5.2.4):

$$[u, [v, w]] + p_{vw}[[u, w], v] + p_{uv}(p_{vw}p_{wv} - 1)v \cdot [u, w] .$$

$[u, [v, w]] =$ (equivale a 5.2.3.b)

$$= uvw - p_{vw}uwv - p_{uv}p_{uw}vwu + p_{uv}p_{uw}p_{vw}wvu , \quad (5.2.4.b)$$

$$p_{vw}[[u, w], v] = p_{vw}([uw - p_{uw}wu, v])$$
$$= p_{vw}(uwv - p_{uw}wuv - p_{uw,v}(vuw - p_{uw}vwu))$$
$$= p_{vw}uwv - p_{vw}p_{uw}wuv - p_{vw}p_{uv}p_{wv}vuw +$$
$$p_{vw}p_{uv}p_{wv}p_{uw}vwu , \quad (5.2.4.c)$$

$$p_{uv}(p_{vw}p_{wv} - 1)v \cdot [u, w] = p_{uv}(p_{vw}p_{wv} - 1)v \cdot (uw - p_{uw}wu)$$
$$= p_{uv}(p_{vw}p_{wv} - 1)(vuw - p_{uw}vwu)$$
$$= p_{uv}p_{vw}p_{wv}vuw - p_{uv}p_{vw}p_{wv}p_{uw}vwu -$$
$$p_{uv}vuw + p_{uv}p_{uw}vwu . \quad (5.2.4.d)$$

Sumando los resultados parciales (5.2.4.b) + (5.2.4.c) + (5.2.4.d):

$$\underset{}{uvw} - \underset{\circ}{p_{vw}uwv} - \underset{\circ\circ}{p_{uv}p_{uw}vwu} + \underset{}{p_{uv}p_{uw}p_{vw}wvu} + \underset{\circ}{p_{vw}uwv} -$$

$$\underset{\circ\circ\circ}{p_{vw}p_{uw}wuv} - \underset{\circ\circ\circ}{p_{vw}p_{uv}p_{wv}vuw} + \underset{\circ\circ\circ\circ}{p_{vw}p_{uv}p_{wv}p_{uw}vwu} +$$

$$\underset{\circ\circ\circ}{p_{uv}p_{vw}p_{wv}vuw} - \underset{\circ\circ\circ\circ}{p_{uv}p_{vw}p_{wv}p_{uw}vwu} - \underset{}{p_{uv}vuw} + \underset{\circ\circ}{p_{uv}p_{uw}vwu} =$$

$$uvw + p_{uv}p_{uw}p_{vw}wvu - p_{vw}p_{uw}wuv - p_{uv}vuw .$$

Los términos marcados con "$\circ$" se simplifican a cero, e igualando este resultado con (5.2.4.a) se obtiene la igualdad, con lo cual la relación (5.2.4) queda comprobada.

Prueba de (5.2.5):

(Sólo se prueba la primera parte, la segunda parte se deja al lector).

$$[u, v \cdot w] = [u, v] \cdot w + p_{uv} v \cdot [u, w] ;$$

$$\begin{aligned} [u, v \cdot w] &= uvw - p_{u,vw} vwu \\ &= uvw - p_{uv} p_{uw} vwu , \end{aligned} \tag{5.2.5.a}$$

$$\begin{aligned} [u, v] \cdot w &= (uv - p_{u,v} vu)w \\ &= uvw - p_{u,v} vuw , \end{aligned} \tag{5.2.5.b}$$

$$\begin{aligned} p_{uv} v \cdot [u, w] &= p_{uv} v(uw - p_{uw} wu) \\ &= p_{uv} vuw - p_{uv} p_{uw} vwu , \end{aligned} \tag{5.2.5.c}$$

Sumando los resultados parciales (5.2.5.b) + (5.2.5.c):

$$uvw - p_{u,v} vuw + p_{uv} vuw - p_{uv} p_{uw} vwu = uvw - p_{uv} p_{uw} vwu ,$$

e igualando este resultado con (5.2.5.a) se obtiene la igualdad, con lo cual la primera relación de (5.2.5) queda comprobada.

Es fácil ver que las siguientes *identidades restringidas* condicionales son válidas también

$$[u, v^n] = [\ldots[[u,v],v]\ldots,v] ; \qquad [v^n, u] = [v,[\ldots[v,u]\ldots]] , \tag{5.2.6}$$

ya que p_{vv} es una raíz t-ava primitiva, y $n = t$ o $n = tl^k$ en el caso de la característica $l > 0$.

Suponga que un álgebra de Lie g se define por los generadores $x_1, \ldots, x_n$ y las relaciones $f_i = 0$. Para convertir los generadores en variables cuánticas se les asocian elementos de $G \times G^*$ en una forma arbitraria. Sea $P = \|p_{ij}\|$, $p_{ij} = \chi^{x_i}(g_{x_j})$ la *matriz de cuantificación*.

Definición 5.2.1. Un *álgebra envolvente cuántica trenzada* es un álgebra de Hopf bigraduada trenzada $U_P^b(\mathfrak{g})$ definida por las variables $x_1, \ldots, x_n$ y las relaciones $f_i = 0$, donde la operación de Lie se reemplaza con (5.2.2), dado que en esta forma las f_i se convierten en las operaciones cuánticas f_i^*. El coproducto y las relaciones de conmutación en el producto tensor se definen por

$$\Delta(x_i) = x_i \otimes 1 + 1 \otimes x_i , \tag{5.2.7}$$

$$(x_i \otimes x_j)\cdot(x_k \otimes x_m) = (\chi^{x_k}(g_{x_j}))^{-1} x_i x_k \otimes x_j x_m . \tag{5.2.8}$$

Definición 5.2.2. Un *álgebra envolvente universal cuántica simple* de g es un álgebra $U_P(\mathfrak{g})$ que es isomórfica al álgebra de grupo torcida

$$U_P(\mathfrak{g}) = U_P^b(\mathfrak{g}) * G , \tag{5.2.9}$$

donde la acción de grupo y el coproducto se definen por

$$g^{-1}x_i g = \chi^{x_i}(g)x_i, \qquad \Delta(x_i) = x_i \otimes 1 + g_{x_i} \otimes x_i, \qquad \Delta(g) = g \otimes g. \tag{5.2.10}$$

Definición 5.2.3. Una *cuantificación con constantes* es una cuantificación simple donde adicionalmente algunos generadores x_i asociados al caracter trivial se reemplazan por las constantes $\alpha_i(1 - g_{x_i})$.

Las fórmulas (5.2.10) y (5.2.7) definen correctamente el coproducto ya que por definición de la operación cuántica $\Delta(f_i^*) = f_i^* \otimes 1 + g \otimes f_i^*$ en el caso de álgebras de Hopf ordinarias y $\Delta(f_i^*) = f_i^* \underline{\otimes} 1 + 1 \underline{\otimes} f_i^*$ en el caso trenzado.

Se tiene que observar que las cuantificaciones definidas dependen, esencialmente, de la representación combinatoria del álgebra de Lie. Por ejemplo, una relación adicional $[x_1, x_1] = 0$ no cambia el álgebra de Lie. Al mismo tiempo si $\chi^{x_1}(g_1) = -1$ entonces esta relación acepta la cuantificación y proporciona una relación no trivial para el álgebra envolvente cuántica, $2x_1^2 = 0$.

Ejemplo 5.2.1. Suponga que el álgebra de Lie está definida por un sistema de relaciones de constitución homogénea. Si los caracteres χ^i son tales que $p_{ij}p_{ji} = 1$ para toda i, j entonces el conmutador torcido por sí mismo es una operación cuántica. Por lo tanto, reemplazando la operación de Lie, todas las relaciones se hacen operaciones cuánticas también. Esto significa que el álgebra envolvente trenzada es el álgebra envolvente universal $U(\mathfrak{g}^{col})$ de la super-álgebra de Lie de color la cual se define por las mismas relaciones que el álgebra de Lie dada. La cuantificación simple aparece como el biproducto de Radford $U(\mathfrak{g}^{col}) \star \mathbf{k}[G]$ o, equivalentemente, como el álgebra envolvente-G universal de la super-álgebra de Lie de color $\mathfrak{g}^{col}$ (ver [**86**] o [**50**, Ejemplo 1.9]).

Ejemplo 5.2.2. Suponga que el álgebra de Lie $\mathfrak{g}$ se define por los generadores $x_1, \ldots, x_n$ y el sistema de las relaciones nil

$$x_j(\mathrm{ad}x_i)^{n_{ij}} = 0, \qquad 1 \le i \ne j \le n. \tag{5.2.11}$$

Generalmente se considera la matriz $A = \| a_{ij} \|$, $a_{ij} = 1 - n_{ij}$ en lugar de la matriz de grados (sin la diagonal principal) $\| n_{ij} \|$. El grafo de Coxeter $\Gamma(A)$ se asocia a cada tal matriz. Este grafo tiene los vértices $1, \ldots, n$, donde el vértice i está conectado por las aristas $a_{ij}a_{ji}$ con el vértice j.

Si $a_{ij} = 0$, entonces la relación $x_j \mathrm{ad}x_i = 0$ está en la lista (5.2.11), y la relación $x_i(\mathrm{ad}x_j)^{n_{ji}} = 0$ es una consecuencia de ello. El conmutador torcido $[x_j, x_i]$ es una operación cuántica si y solo si $p_{ij}p_{ji} = 1$. Bajo esta condición tenemos $[x_i, x_j] = -p_{ij}[x_j, x_i]$. Por lo tanto ambas, en el álgebra de Lie dada y en su cuantificación uno puede reemplazar la relación $x_i(\mathrm{ad}x_j)^{n_{ji}} = 0$ con $x_i \mathrm{ad}x_j = 0$. En otras palabras, sin pérdida de generalidad, podemos

suponer que $a_{ij} = 0 \leftrightarrow a_{ji} = 0$. Por el teorema de Gabber-Kac [38] tenemos que el álgebra $\mathfrak{g}$ es la componente homogénea positiva $\mathfrak{g}_1^+$ de un álgebra Kac-Moody $\mathfrak{g}_1$.

El teorema 6.1 de [50] describe las condiciones para que un polinomio homogéneo en dos variables, el cual es lineal en una de ellas, sea una operación cuántica. De este teorema se tiene el siguiente corolario y el siguiente teorema.

Corolario 5.2.1. *Si n_{ij} es un número simple o la unidad y en el primer caso p_{ii} no es una raiz n_{ij} - ava primitiva de la unidad, entonces la relación* (5.2.11) *acepta una cuantificación si y solo si* $p_{ij}p_{ji} = p_{ii}^{a_{ij}}$.

Teorema 5.2.1. *Para variables cuánticas x_1 y x_2, existe una operación de Lie cuántica lineal en x_1W de grado n en x_2 si y solo si, ya sea $p_{12}p_{21} = p_{22}^{1-n}$, o p_{22} es una raiz m-ava primitiva de la unidad, m|n, y $p_{12}^m p_{21}^m = 1$. Si una de estas condiciones se satisface, entonces todas las operaciones tienen la forma $W = \alpha[\ldots[[x_1x_2]x_1]\ldots x_2]$, $\alpha \in \mathbf{k}$, donde los corchetes están definidos por* (5.2.2).

El teorema 6.1 de [50] proporciona restricciones no esenciales en los parámetros no diagonales p_{ij} : si la matriz P define correctamente una cuantificación de (5.2.11) entonces para cada conjunto $Z = \{z_{ij} | z_{ij}z_{ji} = z_{ii} = 1\}$ la siguiente matriz lo hace también:

$$P_Z = \{p_{ij}z_{ij} | p_{ij} \in P, z_{ij} \in Z\}\ . \tag{5.2.12}$$

Ejemplo 5.2.3. Sea G generada libremente por $g_1, \ldots, g_n$ y sea A una matriz de Cartan generalizada simetrizada por $d_1, \ldots, d_n$, mientras que los caracteres están definidos por $p_{ij} = q^{-d_i a_{ij}}$. En este caso la cuantificación simple es el componente positivo del álgebra envolvente de Drinfeld-Jimbo junto con los elementos tipo grupo, $U_P(\mathfrak{g}) = U_q^+(\mathfrak{g}) * G$. Por medio de una deformación arbitraria (5.2.12) se puede definir un 'color' de $U_q^+(\mathfrak{g}) * G$.

El álgebra envolvente trenzada equivale a $U_q^+(\mathfrak{g})$ donde el coproducto y trenzado se definen correspondientemente por (5.2.7) y (5.2.8) con el coeficiente $q^{d_k a_{kj}}$. La fórmula (5.2.12) define correctamente su 'color' también.

Ejemplo 5.2.4. Si en el ejemplo de arriba completamos el conjunto de variables cuánticas por las nuevas $x_1^-, \ldots, x_n^-$; $z_1, \ldots, z_n$ tal que

$$\chi^{x^-} = (\chi^x)^{-1}, \quad g_{x^-} = g_x, \quad \chi^2 = \mathrm{id}, \quad g_{z_i} = g_i^2, \tag{5.2.13}$$

entonces por el teorema 6.1 de [50] las relaciones de Gabber-Kac (5.2.2), (5.2.3), y $[e_i, f_j] = \delta_{ij}h_i$ (ver [38], teorema 2) bajo la identificación $e_i = x_i$, $f_i = x_i^-$, $h_i = z_i$ aceptan la cuantificación con constantes $z_i = \varepsilon_i(1 - g_i^2)$. (Informalmente podemos considerar la cuantificación obtenida como una del álgebra Kac-Moody identificando a g_i con q^{h_i}, donde el

resto de las relaciones del álgebra de Kac-Moody, $[h_i, e_j] = a_{ij}e_i$, $[h_i, f_j] = -a_{ij}f_j$, es cuantificada a la acción G: $g_j^{-1}x_i^{\pm}g_j = q^{\mp d_{ij}a_{ij}}x_i^{\pm}$.) Esta cuantificación coincide con la de Drinfeld-Jimbo bajo una selección apropiada de x_i, x_i^- y ε_i dependiendo de la definición particular de $U_q(\mathfrak{g})$:

[65] $x_i = E_i$, $g_i = K_i$, $x_i^- = F_iK_i$, $p_{ij} = v^{-d_ia_{ij}}$, $\varepsilon_i = (v^{-d_i} - v^{d_i})^{-1}$;

[66] $x_i = E_i$, $g_i = \tilde{K}_i$, $x_i^- = F_i\tilde{K}_i$, $p_{i\mu} = v^{-\langle\mu,i'\rangle}$, $\varepsilon_i = (v_i^{-1} - v_i)^{-1}$;

[46] Δ_+ $x_i = e_i$, $g_i = t_i$, $x_i^- = t_if_i$, $p_{ij} = q_j^{-\langle h_j,\alpha_i\rangle}$, $\varepsilon_i = (q_i - q_i^3)^{-1}$;

[46] Δ_- $x_i = f_i$, $g_i = t_i$, $x_i^- = e_it_i$, $p_{ij} = q_j^{\langle h_j,\alpha_i\rangle}$, $\varepsilon_i = (q_i^{-1} - q_i)^{-1}$;

[78] $x_i = E_iK_i$, $g_i = K_i^2$, $x_i^- = F_iK_i$, $p_{ij} = q^{-2d_ia_{ij}}$, $\varepsilon_i = (1 - q^{4d_i})^{-1}$.

Por (5.2.13) los corchetes $[x_i, x_j^{-1}]$ son operaciones cuánticas sólo si $p_{ij} = p_{ji}$. Así en este caso los 'colores' (5.2.12) pueden ser solamente blanco-negro, $z_{ij} = \pm 1$.

En la analogía perfecta la cuantificación de Kang [45] de las álgebras de Kac-Moody generalizadas [16] es una cuantificación en el sentido del método combinatorial también.

5.3 Los Generadores de PBW (Poincaré-Birkhoff-Witt) y la Cristalización

El problema de construcción de la base para álgebras envolventes cuánticas se considera en esta subsección. Se indican dos métodos para la construcción de *generadores de PBW*. Uno de ellos modifica el proceso de construir la base de Hall-Shirshov reemplazando la operación de Lie con un conmutador torcido. El conjunto de generadores de PBW definido de este modo, los valores de *superletras duras*, juegan el mismo papel que la base del álgebra de Lie fundamental en el teorema de PBW. A primera vista parecería razonable considerar el módulo $k[G]$ generado por los valores de superletras duras como un álgebra de Lie cuántica. Sin embargo este módulo extremadamente importante está muy lejos de ser definido en forma única. Esencialmente depende del ordenamiento de los principales generadores, sus grados, y su casi nunca estable antípoda. También se observa el siguiente hecho importante. Nuestra definición de superletras duras no es constructiva y, por supuesto, no puede ser constructiva en general. El problema de construcción de base incluye el problema de la palabra para álgebras definidas por generadores y relaciones, mientras que este último no tiene solución algorítmica general (ver [9, 13]).

El segundo método está conectado con la idea de la cristalización de Kashiwara [46, 47] (ver también un desarrollo en [22, 62]). M. Kashiwara ha considerado el principal parámetro q del álgebra envolvente de Drinfeld-Jimbo como una temperatura de algún medio físico. Cuando la temperatura tiende a cero, el medio se cristaliza. Los generadores de PBW se deben cristalizar

también. En nuestro caso bajo este proceso ningun álgebra envolvente cuántica límite aparece, ya que las condiciones existentes normalmente incluyen igualdades de la forma $\prod p_{ij} = 1$ (ver [53]). No obstante si igualamos todos los parámetros de cuantificación a cero, las superletras duras formarían un nuevo conjunto de generadores de PBW para el álgebra envolvente universal cuántica dada. Para poner esto de otra forma, la base de PBW definida por las superletras acepta la cristalización de Kashiwara.

Los siguientes resultados proporcionan una base de PBW para las álgebras envolventes cuánticas.

Teorema 5.3.1. *Cada álgebra de Hopf de caracter H tiene un conjunto ordenado linealmente de elementos de constitución homogénea* $U = \{u_i \mid i \in I\}$ *tal que el conjunto de todos los productos* $gu_1^{n_1}u_2^{n_2}\ldots u_m^{n_m}$, *donde* $g \in G$, $u_1 < u_2 < \ldots < u_m$, $0 \le n_i < h(i)$ *forma una base de H. Aqui si* $p_{ii} = p_{u_i u_i}$ *no es una raiz de la unidad entonces* $h(i) = \infty$; *si* $p_{ii} = 1$ *entonces, ya sea* $h(i) = \infty$ *o* $h(i) = l$ *es la caracteristica del campo fundamental; si* p_{ii} *es una raiz primitiva t-ava de la unidad,* $t \neq 1$, *entonces* $h(i) = t$.

El conjunto *U* se refiere como a un conjunto de *generadores de PBW* de *H*. Este teorema se sigue fácilmente del teorema 2 de [52]. Recordemos algunas nociones necesarias.

Sea $a_1, \ldots, a_n$ un conjunto de generadores primitivos torcidos de *H*, y sean x_i las variables cuánticas asociadas. Considere el orden lexicográfico de todas las palabras en $x_1 > x_2 > \ldots > x_n$. Una palabra no vacía u se llama *estándar* si $uw > wu$ para cada descomposición $u = vw$ con v, w no vacías. Las siguientes propiedades son bien conocidas (ver, por ejemplo [21, 24, 64, 94, 95]).

1s. Una palabra u es estándar si y solo si es mayor que cada uno de sus finales.
2s. Cada palabra estándar comienza con una letra máxima que ella contiene.
3s. Cada palabra c tiene una representación única $c = u_1^{n_1}u_2^{n_2}\cdots u_k^{n_k}$, donde $u_1 < u_2 < \ldots < u_k$ son palabras estándares (el teorema de Lyndon).
4s. Si u, v son palabras estándares diferentes, y u^n contiene a v^k como una subpalabra $u^n = cv^k d$, entonces u por sí misma contiene a v^k como una subpalabra $u = bv^k e$.

Recuerde que una palabra *no asociativa* es una palabra donde los corchetes se arreglan de algún modo para mostrar como se aplica la multiplicación. Si $[u]$ denota una palabra no asociativa entonces por u denotamos una palabra asociativa obtenida de $[u]$ quitando los corchetes (por supuesto $[u]$ no es definida únicamente por u en general).

El conjunto de palabras *estándares no asociativas* se define como el conjunto más pequeño *SL* que contiene todas las variables x_i y satisface las siguientes propiedades.

1) Si $[u] = [[v][w]] \in SL$ entonces $[v], [w] \in SL$, y $v > w$ son estándares.
2) Si $[u] = [[[v_1][v_2]][w]] \in SL$ entonces $v_2 \leq w$.

Las siguientes declaraciones también son válidas.

5s. Cada palabra estándar tiene la única alineación de corchetes tal que la palabra no asociativa obtenida es estándar (teorema de Shirshov [94]).

6s. Los factores v, w de la descomposición no asociativa $[u] = [[v][w]]$ son las palabras estándares tales que $u = vw$ y v tiene la longitud mínima ([95]).

Definición 5.3.1. Una *superletra* es un polinomio que equivale a una palabra estándar no asociativa donde los corchetes significan (5.2.2). Una *superpalabra* es una palabra en superletras.

Por 5s cada palabra estándar u define la única superletra, que en lo subsecuente la denotaremos por $[u]$. Por ejemplo, las palabras $x_1x_2^2$, $x_1^2x_2$, $x_1x_2^3$, $x_1x_2x_1x_2^2$, $x_1x_2x_3^2x_2$, son estándares y ellas definen las siguientes superletras:

$$[x_1x_2^2] = [[x_1x_2]x_2], \quad [x_1^2x_2] = [x_1[x_1x_2]], \quad [x_1x_2^3] = [[[x_1x_2]x_2]x_2],$$

$$[x_1x_2x_1x_2^2] = [[x_1x_2][[x_1x_2]x_2]], \quad [x_1x_2x_3^2x_2] = [[x_1[[x_2x_3]x_3]]x_2].$$

En el teorema 5.2.1 tenemos $W = \alpha[x_1x_2^n]$. Si las variables son ordenadas en forma opuesta, $x_2 > x_1$, entonces $x_1x_2^n$ es una palabra no estándar, mientras que $x_2^nx_1$ si lo es, y uno puede ver que $[\ \ldots\ [[x_1x_2]x_2]\ \ldots\ x_2] = (-p_{12})^n p_{22}^{\frac{n(n-1)}{2}} [x_2^nx_1]$ dado que una de las condiciones existentes es válida (ver corolario 5.3.1 abajo). Por lo tanto las relaciones cuantificadas (5.2.11) se pueden escribir en una forma de igualdad a cero de algunas superletras:

$$[x_jx_i^{n_{ij}}] = 0, \quad [x_j^{n_{ji}}x_i] = 0, \quad j < i. \tag{5.3.1}$$

Sea D un grupo aditivo abeliano ordenado linealmente. Suponga que algunos grados-D positivos $d_1, \ldots, d_n \in D$ se asocian a $x_1, \ldots, x_n$. Definimos el grado de una palabra como $m_1d_1 + \ldots + m_nd_n$ donde $(m_1, \ldots, m_n)$ es la constitución de la palabra. El orden y el grado en las superletras se definen en la siguiente forma: $[u] > [v] \Leftrightarrow u > v$; $D([u]) = D(u)$.

Definición 5.3.2. Una superletra $[u]$ se llama *dura en H* dado que su valor en H no es una combinación lineal de valores de superletras del mismo grado en menos que $[u]$ superletras y G superpalabras de un grado menor.

Definición 5.3.3. Decimos que una *altura* de una superletra $[u]$ de grado d equivale a $h = h([u])$ si h es el número más pequeño tal que: primero p_{uu} es una raíz t-ava primitiva de unidad y ya sea $h = t$ o $h = tl^r$, donde l = car(k); y entonces el valor en H de $[u]^h$ es una combinación lineal de superpalabras de grado hd en menos que $[u]$ superletras y G superpalabras de un grado menor. Si no existe tal número entonces la altura es igual a infinito.

Claramente, si el álgebra H es D-homogénea entonces se pueden omitir las partes subrayadas de las definiciones de arriba.

Teorema 5.3.2 ([52], teorema 2). *El conjunto de todos los valores en H de todas las G-superpalabras W en las superletras duras* $[u_i]$,

$$W = g[u_1]^{n_1}[u_2]^{n_2}\cdots[u_m]^{n_m}, \tag{5.3.2}$$

donde $g \in G$, $u_1 < u_2 < \ldots < u_m$, $n_i < h([u_i])$ *es una base de H.*

Para encontrar el conjunto de los generadores de PBW es necesario primero incluir en U los valores de todas las superletras duras, después para cada superletra $[u]$ de una altura finita, $h([u]) = tl^k$, incluir los valores de $[u]^t, [u]^{tl}, \ldots, [u]^{tl^{(k-1)}}$, y enseguida para cada superletra de altura infinita, tal que p_{uu} es una raíz t-ava primitiva de la unidad, incluir el valor de $[u]^t$.

Obviamente el conjunto de los generadores de PBW juega el mismo papel que la base del álgebra de Lie lo hace en el teorema de PBW. No obstante el bimódulo $\mathbf{k}[G]$ generado por los generadores de PBW no se define en forma única. Depende del ordenamiento de los principales generadores, el grado D, y bajo la acción de la antípoda se transforma a un bimódulo diferente de generadores de PBW: $\mathbf{k}[G]S(U)$.

Otra forma de construir los generadores de PBW se deriva de la idea de la cristalización de M. Kashiwara [46, 47]. M. Kashiwara consideró el principal parámetro del álgebra envolvente de Drinfeld-Jimbo como la temperatura de algún medio físico. Cuando la temperatura tiende a cero el medio se cristaliza. Por este medio la base de 'cristal' debe aparecer. Si se reemplaza p_{ij} con cero entonces $[u, v]$ se hace un monomio uv, mientras que $[u]$ se hace un monomio u.

Lema 5.3.1 (Cristalización de Bases). *Bajo la cristalización de arriba el conjunto de los generadores de PBW construidos en el teorema* 5.3.2 *se hace otro conjunto de generadores de PBW.*

Prueba. Ver [52], corolario 1.

Lema 5.3.2 (Cristalización de superletras). *Una superletra* $[u]$ *es dura en H si y solo si el valor de* u *no es una combinación lineal de palabras menores del mismo grado y palabras G de un grado menor.*

Prueba. Ver [52], corolario 2.

Lema 5.3.3. *Sea B un conjunto de superletras que contienen* $x_1, \ldots, x_n$. *Si cada par* $[u], [v] \in B, u > v$ *satisface una de las siguientes condiciones*

1) $[[u][v]]$ *no es una palabra no asociativa estándar;*
2) *la superletra* $[[u][v]]$ *es no dura en H;*
3) $[[u][v]] \in B$,

entonces el conjunto B incluye todas las superletras duras en H.

Prueba. Sea $[w]$ una superletra dura de grado mínimo tal que $[w] \notin B$. Entonces $[w] = [[u][v]]$, $u > v$ donde $[u]$, $[v]$ son superletras duras. En efecto, si $[u]$ es no dura entonces por el lema 5.3.2 tenemos $u = \Sigma\alpha_i u_i + S$, donde $u_i < u$ y $D(u_i) = D(u)$, $D(S) < D(u)$. Tenemos $uv = \Sigma\alpha_i u_i v + Sv$, donde $u_i v < uv$. Por lo tanto por el lema 5.3.2, la superletra $[w] = [uv]$ no puede ser dura en H. Contradicción. Similarmente, si $[v]$ no es dura entonces $v = \Sigma\alpha_i v_i + S$, $v_i < v$, $D(v_i) = D(v)$, $D(S) < D(v)$. Por lo tanto $uv = \Sigma\alpha_i u v_i + uS$, $uv_i < uv$, y otra vez $[w]$ no puede ser dura.

Así, de acuerdo con la elección de $[w]$, tenemos $[u]$, $[v] \in B$. Ya que este par no satisface la condición 1) ni la 2), la condición 3), $[uv] \in B$, se mantiene. □

Lema 5.3.4. *Si* $T \in H$ *es un elemento primitivo torcido entonces*

$$T = \alpha[u]^h + \sum\alpha_i W_i + \sum\beta_j g_j W_j', \quad \alpha \neq 0, \tag{5.3.3}$$

donde $[u]$ *es una superletra dura,* W_i, *son superpalabras base en superletras menores que* $[u]$, $D(W_i) = hD([u])$, $D(W_j') < hD([u])$. *Aquí si* p_{uu} *no es una raíz de la unidad, entonces* $h = 1$; *si* p_{uu} *es una raíz t–ava primitiva de la unidad, entonces* $h = 1$, o $h = t$, o $h = tl^k$, *donde l es la característica del campo básico.*

Prueba. Considere una expansión de T en términos de la base (5.3.2)

$$T = \alpha g U + \sum_{i=1}^{k}\gamma_i g_i W_i + W', \quad \alpha \neq 0,$$

donde gU, $g_i W_i$ son elementos de base diferentes de máximo grado, y U es una de las palabras más grandes entre U, W_i con respecto al ordenamiento lexicográfico de las palabras en las superletras. En la expansión de base de tensores, el elemento $\Delta(T) - T \otimes 1 - g_t \otimes T$ tiene sólo un tensor de la forma $gU \otimes \ldots$ y este tensor es igual a $gU \otimes \alpha(g - 1)$. Por lo tanto $g = 1$ y se puede aplicar el lema 13 de [52]. □

Corolario 5.3.1. *Si una de las condiciones existentes en el teorema* 5.2.1 *se mantiene, entonces:*

$$[\ldots[[x_1x_2]x_2]\ldots x_2] = (-p_{12})^n p_{22}^{\frac{n(n-1)}{2}} [x_2[x_2\ldots[x_2x_1]\ldots]].$$

Prueba. Introduzcamos el orden opuesto, $x_2 > x_1$. Ya que $[\ldots[[x_1x_2]x_2]\ldots x_2]$ es una operación de Lie cuántica, tiene una representación (5.3.3) donde todos los sumandos tienen la

misma constitución, (1, n). Esto implica que $h = 1$, $u = x_2^n x_1$. Todas las palabras estándares de constitución menor que o igual a (1, n) son x_2, $x_2^k x_1$, $k \leq n$. Por definición de el orden lexicográfico $x_2 > x_2^n x_1$. Por lo tanto x_2 no ocurre en (5.3.3) como una superletra. Ya que cada sumando tiene grado 1 en x_1, la igualdad (5.3.3) se reduce a T = $\alpha [x_2^n x_1]$. Para encontrar α se pueden comparar los coeficientes en $x_2^n x_1$.

5.4 Sistemas de Relaciones de Groebner-Shirshov

En esta sección se lleva una forma de construir un sistema de relaciones de Groebner-Shirshov (ver definiciones en el capítulo 3 sección 3.5.3 de ésta tesis) para un álgebra envolvente cuántica. Este sistema se relaciona con los principales generadores primitivos torcidos, y, de acuerdo con el Lema del Diamante (ver [8, 9, 95]), determina la base de cristal. La utilidad del sistema de Groebner-Shirshov depende del hecho de que tal sistema no solamente define una base de un álgebra asociativa, sino que también proporciona un algoritmo de disminución simple para expansión de elementos sobre esta base (ver, por ejemplo [5]).

Sean $x_1, \ldots, x_n$ variables que tienen grados positivos $d_1, \ldots, d_n \in D$. Recuerde que un *ordenamiento de Hall* de palabras en $x_1, \ldots, x_n$ es un orden cuando las palabras son comparadas primeramente por el grado y después las palabras del mismo grado son comparadas por medio del orden lexicográfico. Considere el conjunto de relaciones

$$w_i = f_i , \quad i \in I, \tag{5.4.1}$$

donde w_i es una palabra y f_i es una combinación lineal de palabras menores de Hall. El sistema (5.4.1) se dice ser *cerrado bajo la composición* o un *sistema de relaciones de Groebner-Shirshov* si primero ninguna de las w_i contiene w_j, $i \neq j \in I$ como una subpalabra, y entonces para cada par de palabras w_k, w_j tal que alguna terminal no vacía de w_k coincide con un sobreconjunto de w_j, esto es $w_k = w_k' v$, $w_j = v w_j'$, la diferencia (una *composición*) $f_k w_j' - w_k' f_j$ puede reducirse a cero en el *álgebra libre* por medio de una secuencia de substituciones de un lado $w_i \to f_i$, $i \in I$.

Lema 5.4.1 (Lema del Diamante [8, 10, 95]). *Si el sistema* (5.4.1) *es cerrado bajo la composición, entonces las palabras que tienen ninguna* w_i *como subpalabras, forman una base del álgebra H definida por* (5.4.1).

Si ninguna de las palabras w_i tiene subpalabras w_j, $j \neq i$, entonces la declaración opuesta es válida también. En efecto, cualquier composición por medio de substituciones $w_i \to f_i$ puede ser reducida a una combinación lineal de palabras que no tiene subpalabras w_i. Ya que $f_i w_j' - w_j' f_j = (f_i - w_i)\, w_j' - w_i' (f_j - w_j)$, esta combinación lineal se iguala a cero en H. Por lo tanto todos los coeficientes tienen que ser cero.

Ya que el Lema de Cristalización de Bases proporciona la base que consiste de palabras, la nota de arriba da una forma para construir el sistema de relaciones de Groebner-Shirshov para cualquier álgebra envolvente cuántica.

Sea H un álgebra Hopf de caracter generada por semi-invariantes primitivas torcidas $a_1, \ldots, a_n$ (o un álgebra de Hopf bigraduada trenzada generada graduando los elementos primitivos homogéneos $a_1, \ldots, a_n$) y sean $x_1, \ldots, x_n$ las variables cuánticas relacionadas. Una superletra $[w]$ no dura en H es referida como una *mínima* si primero w no tiene subpalabras estándares propias que definan superletras no duras, y entonces w no tiene subpalabras u^h, donde $[u]$ es una superletra dura de la altura h.

Por el Lema de Cristalización de Superletras, para cada superletra $[w]$ mínima no dura en H se puede escribir una relación en H

$$w = \sum \alpha_i w_i + \sum \beta_j g_j w_j, \tag{5.4.2}$$

donde $w_j, w_i < w$ en el sentido de Hall, $D(w_i) = D(w)$, $D(w_j) < D(w)$. En la misma forma si $[u]$ es una superletra dura en H de una altura finita h entonces

$$u^h = \sum \alpha_i u_i + \sum \beta_j g_j u_j, \tag{5.4.3}$$

donde $u_j, u_i < u^h$ en el sentido de Hall, $D(u_i) = hD(u)$, $D(u_j) < hD(u)$. Las relaciones (5.2.1) y la operación de grupo proporcionan las relaciones

$$x_i g = \chi^{x_i}(g) g x_i, \qquad g_1 g_2 = g_3. \tag{5.4.4}$$

Teorema 5.4.1. *El conjunto de relaciones* (5.4.2), (5.4.3), *y* (5.4.4) *forma un sistema de Groebner-Shirshov que define a H. La base determinada por este sistema en el Lema del Diamante coincide con la base de cristal.*

Prueba. La propiedad 4s implica que ninguno de los lados izquierdos de (5.4.2), (5.4.3), (5.4.4) contiene otro como una subpalabra. Por lo tanto por el Lema de Cristalización de Bases es suficiente mostrar que el conjunto de todas las palabras c determinadas en el Lema del Diamante coincide con la base de cristal. Por 3s se tiene $c = u_1^{n_1} u_2^{n_2} \ldots u_k^{n_k}$, donde $u_1 < \ldots < u_k$ es una secuencia de palabras estándares. Cada palabra u_i, define una superletra dura $[u_i]$ ya que en el caso opuesto u_i y por lo tanto c, contiene una subpalabra w que define una superletra dura $[w]$ mínima. En la misma forma n_i no excede la altura de $[u_i]$. □

Lema 5.4.2. *En términos del lema* 5.3.3 *el conjunto de todas las superletras* $[[u][v]]$ *que satisfacen la condición* 2) *contiene todas las superletras no duras mínimas, sin generadores* x_i *no duros.*

Prueba. Si $[w]$ es una superletra mínima no dura $[w] = [[u][v]]$, donde $[u]$, $[v]$ son superletras duras. Por el Lema 5.3.3 tenemos $[u], [v] \in B$, mientras que $[[u][v]]$ no satisface ni 1) ni 3). □

5.5 Cuantificación de las Álgebras de Lie de Tipo G_2

En esta sección se consideran más a fondo las álgebras envolventes universales cuánticas de las álgebras nilpotentes de la serie G_2 definidas por las relaciones de Serre. Se citan primero listas de todas las superletras duras en la forma explícita, después el sistema de relaciones de Groebner-Shirshov, y enseguida los espacios $L(U_P(\mathfrak{g}))$ comprendidos por los elementos primitivos torcidos (o sea, las cuantificaciones del álgebra de Lie $\mathfrak{g}_P$ propia). En todos los casos las listas de superletras duras (al menos las mismas superletras duras) se hacen independientes de los parámetros de cuantificación. Esto significa, que los generadores de PBW resultan a partir de la base de Hall-Shirshov del álgebra de Lie fundamental reemplazando la operación de Lie con el conmutador torcido. Lo mismo es válido para el sistema de relaciones de Groebner-Shirshov. Observe que las bases de Hall-Shirshov, bajo el nombre *bases de Lyndon estándares*, para la serie de Lie clásica fueron construidas por P. Lalonde y A. Ram [**63**], mientras que los sistemas de Groebner-Shirshov de las relaciones de Lie fueron encontradas por L.A. Bokut' y A.A. Klein [**12**].

Los resultados obtenidos en esta sección son nuevos, pero las pruebas y formulaciones son analogías de los resultados de V.K. Kharchenko los cuales fueron obtenidos para las series infinitas A_n, B_n, C_n, D_n en [**56**].

En esta sección se aplican los resultados generales ya mencionados a las álgebras de Lie nilpotentes definidas por las relaciones de Serre (5.2.11) de tipo G_2. Sea $\mathfrak{g}$ el álgebra de Lie de este tipo.

5.5.1 Cuantificación de G_2^+.

Por el teorema de Serre el álgebra G_2^+ se define por dos generadores x_1, x_2 y dos relaciones

$$[[[[x_1x_2]x_2]x_2]x_2] = 0\,, \quad [x_1[x_1x_2]] = 0\,. \tag{5.5.1}$$

Si reemplazamos la operación de Lie con el conmutador torcido (5.2.2), estas relaciones toman la forma

$$x_1x_2^4 + a_1x_2x_1x_2^3 + a_2x_2^2x_1x_2^2 + a_3x_2^3x_1x_2 + a_4x_2^4x_1 = 0\,, \tag{5.5.2}$$

$$x_1^2x_2 + b_1x_1x_2x_1 + b_2x_2x_1^2 = 0\,, \tag{5.5.3}$$

donde

$$a_1 = -p_{12}p_{22}^{[4]}\,, \quad a_2 = p_{12}^2p_{22}p_{22}^{[3]}(p_{22}^2+1)\,, \quad a_3 = -p_{12}^3p_{22}^3p_{22}^{[4]}\,, \quad a_4 = p_{12}^4p_{22}^6\,; \tag{5.5.4}$$

$$b_1 = -p_{12}(1+p_{11})\,, \quad b_2 = p_{12}^2p_{11}\,, \tag{5.5.5}$$

y como siempre denotamos $p^{[n]} = 1 + p + \ldots + p^{n-1}$. De acuerdo al teorema 5.2.1 la primera de las relaciones (5.5.1) admite una cuantificación si y solo si, ya sea que $p_{12}p_{21} = p_{22}^{-3}$, o p_{22} es una raíz primitiva m–ava de la unidad, $m|4$, $m > 1$ con $(p_{12}p_{21})^m = 1$. La segunda lo hace si y solo si, ya sea $p_{12}p_{21} = p_{11}^{-1}$, o $p_{11} = -1$, $p_{12}p_{21} = \pm 1$. En particular si suponemos que

$$p_{11} \neq -1 , \quad \text{y} \quad p_{22}^{[4]} \neq 0 , \tag{5.5.6}$$

obtenemos la condición de existencia para la cuantificación

$$p_{12}p_{21}p_{22}^3 = 1 , \qquad p_{11} = p_{22}^3 . \tag{5.5.7}$$

Multipliquemos (5.5.2) desde la izquierda por x_1, mientras (5.5.3) desde la derecha por x_2^3. La diferencia de las relaciones obtenidas proporciona una nueva

$$(a_1 - b_1)x_1x_2x_1x_2^3 = -a_2x_1x_2^2x_1x_2^2 - a_3x_1x_2^3x_1x_2 - a_4x_1x_2^4x_1 + b_2x_2x_1^2x_2^3 . \tag{5.5.8}$$

Podemos multiplicar (5.5.2) desde la izquierda por $(a_1 - b_1)x_1x_2$, mientras (5.5.8) desde la derecha por x_2. Otra vez la diferencia lleva a una relación:

$$\begin{aligned}\{a_1(a_1 - b_1) - a_2\}x_1x_2^2x_1x_2^3 &= \{a_3 - a_2(a_1 - b_1)\}x_1x_2^3x_1x_2^2 \\ &+ \{a_4 - a_3(a_1 - b_1)x_1x_2^4x_1x_2 - a_4(a_1 - b_1)x_1x_2^5x_1 - b_2x_2x_1^2x_2^4 .\end{aligned} \tag{5.5.9}$$

En la misma forma, si multiplicamos (5.5.3) desde la derecha por $(a_1 - b_1)x_1x_2^3$, mientras (5.5.8) desde la izquierda por x_1, entonces la diferencia de las relaciones obtenidas después del reemplazo de todas las subpalabras $x_1^2x_2$ con $-b_1x_1x_2x_1 - b_2x_2x_1^2$ define una nueva relación

$$\begin{aligned}\{b_1(a_1 - b_1) + b_2\}x_1x_2x_1^2x_2^3 &= -a_2b_1x_1x_2x_1x_2x_1x_2^2 \\ &- a_3b_1x_1x_2x_1x_2^2x_1x_2 - a_4b_1x_1x_2x_1x_2^3x_1 + W ,\end{aligned} \tag{5.5.10}$$

donde W es una combinación lineal de palabras con la primera letra x_2. Podemos reducir esta relación además reemplazando la subpalabra $x_1^2x_2$ de acuerdo a (5.5.3):

$$\begin{aligned}(a_2b_1 - \{b_1(a_1 - b_1) + b_2\}b_1)x_1x_2x_1x_2x_1x_2^2 &= \{b_1(a_1 - b_1) + b_2\}b_2x_1x_2^2x_1^2x_2^2 \\ &- a_3b_1x_1x_2x_1x_2^2x_1x_2 - a_4b_1x_1x_2x_1x_2^3x_1 + W .\end{aligned} \tag{5.5.11}$$

Si las condiciones (5.5.6) son válidas entonces usando $p_{11} = p_{22}^3$ podemos fácilmente calcular

$$\begin{aligned}a_1 - b_1 &= -p_{12}p_{22}(1 + p_{22}), \qquad a_1(a_1 - b_1) - a_2 = p_{12}^3p_{22}, \\ b_1(a_1 - b_1) + b_2 &= p_{12}^2p_{22}p_{22}^{[5]}, \qquad a_2b_1 - \{b_1(a_1 - b_1) + b_2\}b_1 = -p_{12}^3p_{22}^3(1 + p_{22}^3).\end{aligned} \tag{5.5.12}$$

Ahora estamos listos para probar el teorema principal. Denotamos por $\mathcal{B}$ al conjunto de las siguientes superletras:

$$[A]=x_1\ ,\ [B]=[x_1x_2]\ ,\ [C]=[x_1x_2x_1x_2^2]\ ,\ [D]=[x_1x_2^2]\ ,\ [E]=[x_1x_2^3]\ ,\ [F]=x_2\ . \tag{5.5.13}$$

Si suponemos que $x_1 > x_2$, entonces evidentemente $A > B > C > D > E > F$.

Teorema 5.5.1. *Si $p_{11}^{[2]}$, $p_{22}^{[4]}$ son elementos no cero, entonces $\mathcal{B}$ es el conjunto de todas las superletras duras en $U_P(G_2^+)$. Si adicionalmente $p_{11} \neq 1$, entonces todas estas superletras tienen altura infinita, mientras que el álgebra $U_P(G_2^+)$ tiene solamente los siguientes elementos primitivos: ax_1, ax_2, $a - ag$, donde $g \in G$, $i = 1,2$, y también ax_i^n, dado que p_{ii} es una raíz primitiva t–ava de la unidad con $n = tl^k$, l es la característica del campo básico.*

Prueba. Mostremos primeramente que el conjunto $\mathcal{B}$ satisface todas las condiciones del Lema 5.3.3. Tenemos que mostrar que para cada par $[X]$, $[Y] \in \mathcal{B}$ tal que $X > Y$ la palabra no asociativa $[[X][Y]]$ pertenece ya sea a $\mathcal{B}$, o no es palabra estándar como palabra no asociativa, o define una superletra no dura en $U_P(G_2^+)$. Tenemos 15 posibilidades. En cuatro casos tenemos $[[A][F]] = [B]$, $[[B][D]] = [C]$, $[[B][F]] = [D]$, $[[D][F]] = [E]$. En cinco casos más: $[[A][B]]$, $[[A][C]]$, $[[A][D]]$, $[[A][E]]$, $[[E][F]]$, la palabra no asociativa es no dura por el lema 5.3.2 ya que la palabra asociativa obtenida omitiendo los corchetes contiene ya sea la subpalabra $x_1x_2^4$ o la subpalabra $x_1^2x_2$, y por lo tanto por las relaciones (5.5.2) y (5.5.3) son una combinación lineal de palabras menores en $U_P(G_2^+)$. En dos casos, $[[C][E]]$, y $[[C][F]]$, la palabra es no estándar como una palabra no asociativa. Queda por considerar los siguientes últimos cuatro casos.

a). $[[D][E]]$. La relación (5.5.9) muestra que la palabra DE en $U_P(G_2^+)$ es una combinación lineal de palabras menores. De este modo por el lema 5.3.2 la superletra $[[D][E]]$ es no dura.

b). $[[B][C]]$. Tenemos $BC = (x_1x_2)^3x_2$. La relación (5.5.11) muestra que la palabra BC es una combinación lineal de palabras menores en $U_P(G_2^+)$. Otra vez por el lema 5.3.2 la superletra $[[B][C]]$ es no dura.

c). $[[B][E]]$. En este caso $BE = x_1x_2x_1x_2^3$, y podemos usar la relación (5.5.8).

d). [[C][D]]. Multipliquemos la relación (5.5.11) desde la derecha por $(a_1 - b_1)x_2$, mientras la relación (5.5.8) desde la izquierda por $(a_2b_1 - \{b_1(a_1 - b_1) + b_2\}b_1)x_1x_2$. El término líder de la diferencia es igual a

$$\{a_3b_1(a_1 - b_1) - a_2(a_2b_1 - \{b_1(a_1 - b_1) + b_2\}b_1)\}\, x_1x_2x_1x_2^2x_1x_2^2$$

$$= -p_{12}^5p_{22}^5(1 + p_{22}^3)(1 + p_{22}^2)CD.$$

Por lo tanto CD es una combinación lineal de palabras menores también.

Así por el lema 5.3.3, el conjunto $\mathcal{B}$ contiene todas las superletras duras en $U_P(G_2^+)$. Queda por mostrar que todos los seis elementos de $\mathcal{B}$ definen superletras duras en $U_P(G_2^+)$ de altura infinita.

Por la versión homogénea de la definición 5.3.2 si una superletra $[X] \in \mathcal{B}$ es no dura entonces existe una representación en $U_P(G_2^+)$

$$[X] = \sum \alpha_i [Y_{i_1}]^{n_1} \cdot [Y_{i_2}]^{n_2} \cdots [Y_{i_k}]^{n_k}, \tag{5.5.14}$$

donde las $[Y_{i_s}]$ son superletras duras (y por lo tanto $[Y_{i_s}] \in \mathcal{B}$) con $Y_{i_s} < X$ para toda i_s. En este caso el grado de X en cada uno de los generadores, x_1, x_2, debería ser igual a este grado del lado derecho de (5.5.14). Es fácil ver, sin embargo, que ninguno de los multigrados de las superletras de $\mathcal{B}$, (1,0), (1,1), (2,3), (1,2), (1,3), (0,1), es una combinación lineal entera de multigrados de superletras menores de $\mathcal{B}$. Por lo tanto la suma en (5.5.14) es vacía, o, en otras palabras, cada superletra no dura de $\mathcal{B}$ es igual a cero en $U_P(G_2^+)$. Queda por observar que ninguna de las superletras $[B] - [E]$ es igual a cero en $U_P(G_2^+)$.

Todas estas superletras, excepto $[C]$, tienen grado en x_2 menor que 4, y grado en x_1 menor que 2, así que no son cero en el álgebra definida por las relaciones (5.5.2) y (5.5.3). La superletra $[C]$ tiene grado 3 en x_2, por lo tanto si es cero en $U_P(G_2^+)$, debería ser cero en el álgebra R definida por la única relación (5.5.3). Esta única relación es cerrada bajo la composición (sólo porque no hay composiciones en absoluto). Ya que la palabra $C = x_1x_2x_1x_2^2$ no tiene subpalabras $x_1^2x_2$ el Lema del Diamante implica $[C] \neq 0$ en R. Esta contradicción prueba la primera parte del teorema.

Por la versión homogénea de la definición 5.3.3 si una superletra $[X]$ tiene altura finita entonces existe una representación en $U_P(G_2^+)$

$$[X]^n = \sum \alpha_i [Y_{i_1}]^{n_1} \cdot [Y_{i_2}]^{n_2} \cdots [Y_{i_k}]^{n_k}, \tag{5.5.15}$$

donde $[Y_{i_s}] \in \mathcal{B}$, $Y_{i_s} < X$ para toda i_s, en cuyo caso el grado $n\text{grad}(X)$ de $[X]^n$ en cada una de las variables x_1, x_2 tiene que ser igual a la suma $n_1\text{grad}(Y_{i_1}) + n_2\text{grad}(Y_{i_2}) + \ldots + n_k\text{grad}(Y_{i_k})$. Observamos, sin embargo, que ninguno de los seis multigrados, $\text{grad}(A) = (1,0)$; $\text{grad}(B) = (1,1)$; $\text{grad}(C) = (2,3)$; $\text{grad}(D) = (1,2)$; $\text{grad}(E) = (1,3)$; $\text{grad}(F) = (0,1)$, satisface estas condiciones, o sea que un semigrupo generado por cada uno de estos multigrados no tiene intersección con un semigrupo generado por multigrados de superletras menores de $\mathcal{B}$. Este hecho significa que la suma en (5.5.14) es vacía, esto es que una superletra $[Y] \in \mathcal{B}$ tiene altura finita únicamente si tiene valor nilpotente, $[Y]^n = 0$, en $U_P(G_2^+)$.

Ahora suponga que $p_{11} \neq 1$. Mostraremos que ninguna de las $[Y]^n$, $[Y] \in \mathcal{B}$, $n > 1$ define un elemento primitivo torcido en $U_P(G_2^+)$ (por supuesto esto implicaría que $[Y]^n \neq 0$, en $U_P(G_2^+)$). Observe primero que el coproducto es homogéneo, esto es

$$\Delta([Y]^n = \sum Y_n^{(1)} \otimes Y_n^{(2)}, \tag{5.5.16}$$

donde la suma de multigrados de $Y_n^{(1)}$ y $Y_n^{(2)}$ es igual a la de $[Y]^n$.

Se denota por $S_{k,n}([Y])$ la suma de todos los términos en el lado izquierdo de (5.5.16) con $\text{grad}_{x_1}(Y_n^{(2)}) = k$, y $\text{grad}_{x_2}(Y_n^{(2)}) = 0$.

Tenemos:

$$S_{1,1}([B]) = (1-p_{12}p_{21})g_1x_2 \otimes x_1 = (1-p_{11}^{-1})g_1x_2 \otimes x_1 ,$$

$$S_{2,1}([C]) = (1-p_{12}p_{21})^2(1-p_{12}p_{22}p_{21})(p_{21}-p_{11}p_{12}^2p_{21}p_{22}^2p_{21}^2)g_1^2x_2^3 \otimes x_1^2$$

$$= (1-p_{11}^{-1})^2(1-p_{22}^{-2})p_{21}(1-p_{22}^{-1})g_1^2x_2^3 \otimes x_1^2 ,$$

$$S_{1,1}([D]) = (1-p_{12}p_{21})(1-p_{12}p_{22}p_{21})g_1x_2^2 \otimes x_1$$

$$= (1-p_{11}^{-1})(1-p_{22}^{-2})g_1x_2^2 \otimes x_1 ,$$

$$S_{1,1}([E]) = (1-p_{12}p_{21})(1-p_{12}p_{22}p_{21})(1-p_{12}p_{22}^2p_{21})g_1x_2^3 \otimes x_1$$

$$= (1-p_{11}^{-1})(1-p_{22}^{-2})(1-p_{22}^{-1})g_1x_2^3 \otimes x_1 ,$$

(para los cálculos uno puede usar, por ejemplo, la fórmula explícita (8) – (10) de [55]). Utilizando homogeneidad, tenemos $S_{n,n}([Y]) = S_{1,1}([Y])^n \neq 0$, dado que $Y \neq C$, y $S_{2n,n}(C) = S_{2,1}([C])^n \neq 0$. Por lo tanto ninguna de las $[Y]^n$ define elementos primitivos torcidos. En particular todos ellos son no cero y, consecuentemente, todas las superletras de $\mathcal{B}$ tienen altura infinita.

Finalmente si U es un elemento primitivo torcido homogéneo arbitrario que no pertenece a $\mathbf{k}G + \mathbf{k}x_1 + \mathbf{k}x_2$ entonces por el lema 5.3.4 existe una representación

$$U = \beta[Y]^n + \sum \alpha_i [Y_{i_1}]^{n_1} \cdot [Y_{i_2}]^{n_2} \cdots [Y_{i_k}]^{n_k} , \qquad (5.5.17)$$

donde todas las $[Y]$'s son superletras duras (por lo tanto son elementos de $\mathcal{B}$), y $Y_{i_m} < Y$. Hemos visto arriba que el multigrado de $[Y]^n$ nunca es una combinación lineal entera de multigrados de superletras menores de $\mathcal{B}$. Por lo tanto la suma en (5.5.17) es vacía, así que $U = \mathcal{B}[Y]^n$. Pero ya hemos probado arriba que ningún elemento de $[Y]^n$, $Y \in \mathcal{B}$ es primitivo torcido, excepto probablemente los casos $Y = x_1$, $Y = x_2$. Ahora es fácil ver que x_i^n es primitivo torcido si y solo si p_{ii} es la raíz de la unidad primitiva t-ava y $n = t$ o $n = tl^k$, donde l es la característica del campo básico $\mathbf{k}$ (ver por ejemplo argumentos en [50] teorema 5.1). El teorema está completamente probado.

La siguiente tabla define completamente la estructura algebraica de $U_P(G_2^+)$ en términos de los generadores de PBW, ya que para $X > Y$ tenemos $[X] \cdot [Y] = [[X][Y]] + p_{X,Y}[Y] \cdot [X]$, mientras que $[Y] \cdot [X]$ es un elemento de base.

$[[X][Y]]$	$[B]$	$[C]$	$[D]$	$[E]$	$[F]$
$[A]$	0	$\alpha_1[B]^3$	$\alpha_2[B]^2$	$\alpha_3[C] + \beta_3[D] \cdot [B] + \gamma_3[F] \cdot [B]^2$	$[B]$
$[B]$	–	0	$[C]$	$\alpha_4[D]^2 + \beta_4[F] \cdot [C]$	$[D]$
$[C]$	–	–	0	$\alpha_5[D]^3$	$\alpha_6[D]^2$
$[D]$	–	–	–	0	$[E]$
$[E]$	–	–	–	–	0

5.6 Conclusiones del Capítulo 5

Vemos que en el teorema G_2 la lista de superletras duras es independiente del parámetro p_{ij}. Este hecho significa que la base de Lalonde-Ram del álgebra de Lie fundamental (ver [62] figura 1) con el conmutador torcido en lugar de la operación de Lie coincide con el conjunto de todas las superletras duras.

♦

6

Construcción de Algoritmos Específicos

En esta sección se presenta el desarrollo de algunos algoritmos y programas de combinaciones de palabras que se relacionan con la cuantización de las álgebras de Lie, los cuales de alguna forma pueden ayudar a realizar desarrollos subsecuentes para el estudio de la materia y para la elaboración de programas computacionales más sofisticados.

Todos los programas se desarrollaron con el lenguaje C++ básico (sin el uso de clases) por las razones ya expresadas en las consideraciones preliminares del presente trabajo. Se empleó C++ Builder ver. 3.0 para edición y compilación de los programas por su facilidad de operación al nivel de los programas de este capítulo.

Los algoritmos han sido desarrollados siguiendo los axiomas matemáticos, y para la construcción de los programas se emplearon las instrucciones específicas contenidas en las referencias [2, 87]. Las consideraciones para la eficiencia de los algoritmos se dan en el Apéndice D de esta tesis.

6.1. Combinaciones de Palabras

Adicionalmente se consideran las siguientes *combinaciones de palabras*:

$x_1, x_2, \ldots, x_n, x_{n+1}, \ldots$ es un alfabeto.

$x_1x_2x_3$ es una palabra.

Ejemplos de palabras son:

x_1 ; x_2x_1 ; $x_2x_3x_1^2x_5x_1$.

Las palabras x_1x_2 y x_2x_1 son palabras diferentes.

También se puede considerar que $x_1 > x_2 > x_3 > \ldots$, y la ponderación de las palabras se da al igual que en un diccionario de lenguas (aunque, por comodidad, en los programas que se presentan aquí la ponderación o peso de las palabras es de acuerdo al valor numérico de su código ASCII, o sea que $a < b < c < \ldots < x < y < z$).

6.1.1. Palabras Estándares de Shirshov.

A Orden lexicográfico de las palabras (también ver sección 3.5.1 de esta tesis).

Sea $x_1, \ldots, x_n$ un conjunto de variables. Considere este conjunto como un alfabeto. En un conjunto de todas las palabras de este alfabeto, se define el orden lexicográfico tal que

$$x_1 > x_2 > \ldots > x_n .$$

Esto significa que dos palabras v y w se comparan moviéndose de izquierda a derecha hasta que se encuentra la primera letra distinta. Una de las palabras podría ser el inicio de la otra, en este caso se asume que la palabra más corta es mayor que la más larga (como sucede en los diccionarios). Por ejemplo, todas las palabras de longitud máxima de dos en dos variables, respetan el siguiente orden

$$x_1 > x_1^2 > x_1x_2 > x_2 > x_2x_1 > x_2^2 . \qquad 6.1.1$$

Este orden es estable bajo la multiplicación izquierda e inestable bajo la multiplicación derecha. No obstante, si $u > v$ y u no es el inicio de v, entonces la desigualdad se conserva bajo la multiplicación derecha (siguiente regla 4), aún multiplicando por diferentes palabras: $um > vt$.

Definición 6.1.1. Una palabra u se llama estándar (en el sentido de Shirshov) si, para cada representación $u = u_1u_2$, donde u_1 y u_2 son palabras no vacías, la desigualdad $u > u_2u_1$ se mantiene. Por ejemplo en la expresión $x_1 > x_1^2 > x_1^3 > \ldots > x_1^n \ldots$, existe solo una palabra estándar, que es x_1, y en la expresión 6.1.1 existen tres: x_1, x_1x_2, y x_2.

6.1.2. Reglas de orden:

1. Si $u > v$ y $v > w$ entonces $u > w$.
2. Para cualesquier palabras u y v sucede ser verdad que $u > v$ o $v > u$ o $u = v$.
3. Si $u > v$ entonces $wu > wv$.
4. Multiplicación por la derecha. Si $u > v$ y u **no es inicio de** v entonces $uw > vw$ para cualquier palabra w, y aún multiplicando por diferentes palabras: $um > vt$.

Ejemplo. Si se tienen las palabras; $u = x_1^2x_2$; $v = x_1^2x_2x_3$, entonces $x_1^2x_2 > x_1^2x_2x_3$ y si también $w = x_5^2$, ¿qué orden de palabra uw y vw será mayor?

Sustituyendo w en la desigualdad de la regla 4 se plantea la pregunta: $x_1^2x_2\,x_5^2$? $x_1^2x_2x_3\,x_5^2$.

Haciendo la comparación se observa que en la expresión de la izquierda los primeros dos términos son iguales a los de la derecha ($x_1^2x_2$), pero en el tercer término x_5 es menor que x_3, por lo tanto, la expresión del ejemplo se debe escribir como sigue: $x_1^2x_2\,x_5^2 < x_1^2x_2x_3\,x_5^2$, observe que u es inicio de v por lo que la regla 4 no se puede aplicar.

De la regla 4 se infiere que en este ejemplo no se puede realizar la multiplicación por la derecha sin que en ocasiones ocurra alguna alteración del orden lexicográfico. Δ

6.2 Palabras "Estándares"

El algoritmo para determinar palabras estándares se comprende mejor con la siguiente teoría.

Una palabra es estándar si para cada representación de una palabra $u = vw$ ocurre que $u > wv$ con v y w no vacías.

Por ejemplo: si $u = x_3\ x_1\ x_2$ se puede descomponer en v y w y formar wv

$v \quad w$

$wv = x_1\ x_2\ x_3$

$x_3\ x_1\ x_2 = = x_1\ x_2\ x_3$

$x_3 < x_1 \quad \Rightarrow u < wv$

o sea que: $x_3\ x_1\ x_2 < x_1\ x_2\ x_3 \therefore u$ no es estándar

Otro ejemplo, si $u = x_1 x_3 x_2$; se puede descomponer en *a)* $v = x_1$, $w = x_3 x_2$; y *b)* $v = x_1 x_3$, $w = x_2$;

a) $wv = x_3 x_2 x_1 \Rightarrow u > wv$

b) $wv = x_2 x_1 x_3 \Rightarrow u > wv$

$\therefore$ u si es estándar

Δ

Como una regla una palabra estándar siempre comienza con la primera letra que es la máxima de las que precede.

6.2.1 Algoritmo para determinar palabras estándares (figura 6.2.1).

Resumen: Este algoritmo se utiliza para determinar si una palabra dada es estándar o no lo es. La palabra original se descompone en dos representaciones y de acuerdo a la definición de palabra estándar se decide si es o no es estándar el proceso se repite (iterativamente) hasta haber formado todas las representaciones posibles. Si alguna de las representaciones no cumple la regla de palabras estándares el programa termina y muestra en pantalla el mensaje de palabra no estándar.

Nombre del programa: estancb.exe
Entradas: una palabra.
Salidas: determina y despliega un mensaje de palabra estándar o no estándar.

1.- Obtener la palabra a verificar (u) y determinar su longitud (cantidad de caracteres, para obtener un contador de representaciones posibles).

2.- Índice de subpalabras "v y w" j = 1;

3.- Se hacen copias de la palabra u en las variables tempv y tempw. Genera las palabras j-ésimas v y w a partir de tempv y tempw.

4.- Concatenar w y v j-ésimas para formar wv (w seguida de v) j-ésima.

5.- Índice de letras de las palabras "u y wv" i = 0;

6.- ¿El caracter i-ésimo de u es mayor que el caracter i-ésimo de wv?
Sí: indicador de palabra estándar x = 1. Continúa en el paso 10 (para la siguiente representación wv).
No: Continúa en el paso 7.

7.- ¿El caracter i-ésimo de u es igual que el carácter i-ésimo de wv?
Sí: indicador de palabra estándar x = 0 (es no estándar). Continúa en el paso 8.
No: indicador de palabra estándar x = 0. (definitivamente es no estándar puesto que $u < wv$). Continúa en el paso 10.

8.- Incrementa en uno el índice de caracteres i.

9.- ¿Se compararon todos los caracteres, es decir el índice i llegó al límite?
Sí: ir al paso 10.
No: continuar en el paso 6.

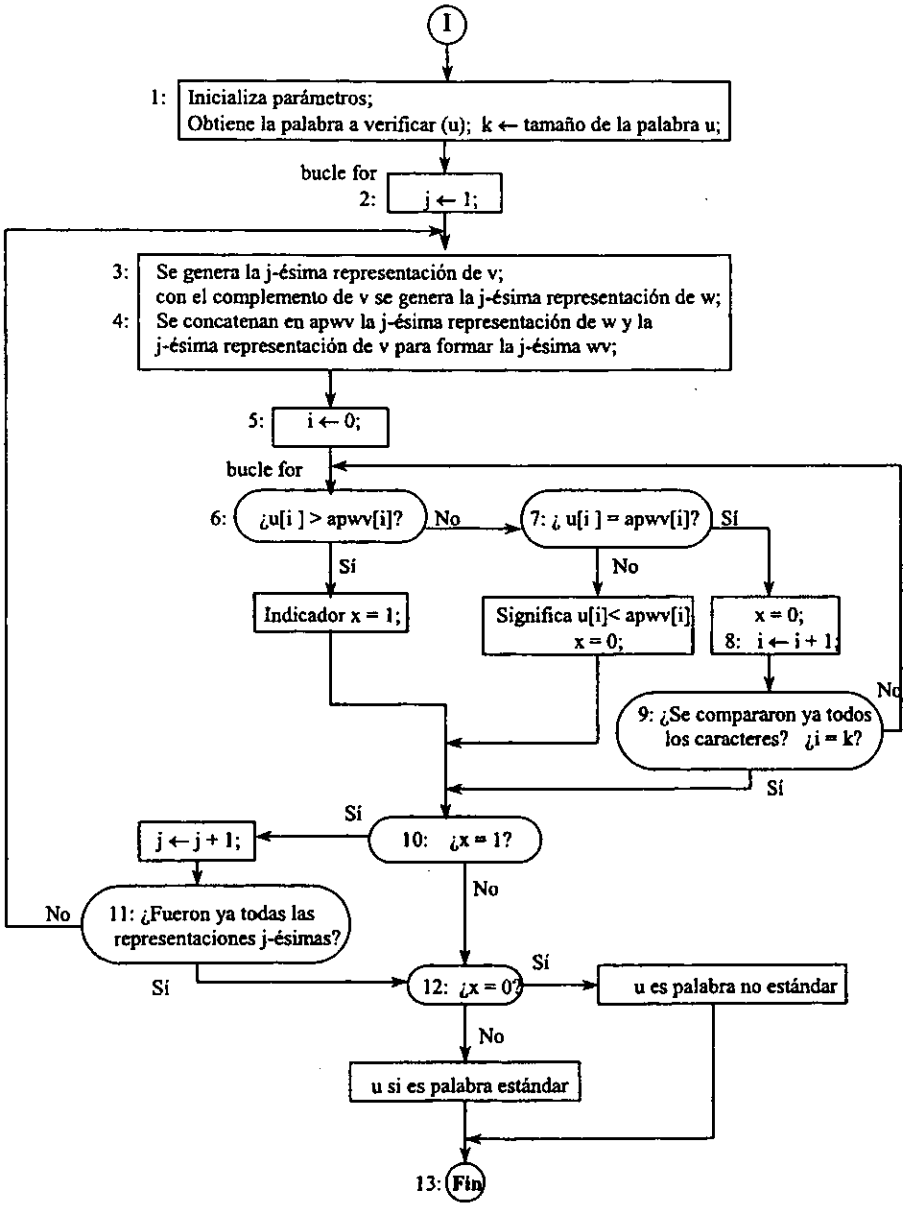
10.- ¿La palabra es estándar (o indicador x = 1)?
Sí: incrementa en uno el índice j de representaciones "wv"; ir al paso 11.
No: ir al paso 12.

11.- ¿Terminaron todas las posibles representaciones "wv" y comparaciones j-ésimas, es decir, j llegó a su límite?
Sí: ir al paso 12.
No: continuar en el paso 3.

12.- ¿Resultó palabra no estándar?
Sí: Exhibe resultado "palabra no estándar"; continúa en el paso 13.
No: Exhibe resultado "palabra estándar"; continúa en el paso 13.

13.- Fin.

Figura 6.2.1. Diagrama de flujo del programa estancb



Análisis de la eficiencia en tiempo.

Se considera que el peor caso es cuando el programa determina que una palabra es estándar, ya que se tienen que formar todas las posibles representaciones o combinaciones *wv*. El tiempo que requiere este algoritmo es del orden de n^2, y está determinado principalmente por la longitud de la cadena (n) en cuestión. Esto se determinó analizando y contando la cantidad de instrucciones elementales así como las iteraciones más importantes. Un desglose sería como el siguiente:

Tamaño de la palabra = $n = k$.

Instrucciones que no dependen de $n = c_i$; $0 \leq i \leq c_{max}$ (una constante).

En el caso peor $j = n - 1$ para la primera iteración **for**.

En la segunda instrucción **for** $i = n + 1$.

Para la tercera instrucción **for** $i = n$.

De acuerdo a la posición de los lazos iterativos dentro del algoritmo, la función obtenida aproximada es: $t = c_1 + c_2(n - 1)[c_3(n + 1) + c_4n + c_5]$, la cual se puede simplificar a una función de la forma $d_1n^2 + d_2n + d_3$.

Y por la regla del máximo $t(n) = cn^2$ (el tiempo es de orden cuadrático).

Eficiencia de espacio.

El espacio de memoria empleado es despreciable ya que la cantidad de variables usada es mínima. Se pueden manejar palabras hasta un máximo de 1000 caracteres porque se asignó memoria para arreglos de tamaño 1000, aunque este parámetro se podría aumentar hasta la capacidad de la pila de almacenamiento de memoria de la máquina y para valores muy grandes es mejor emplear asignación dinámica de memoria. Los caracteres en C++ ocupan 1 byte de espacio en memoria.

6.2.2 Ejemplos con el programa de palabras estándares.

Ejemplo 6.2.1.

```
Programa para determinar palabras estandar estancb.exe
Introduce la palabra a verificar considerando z mayor que a:
Palabra u = vw = xzy
Longitud de la palabra = 3
Representaciones wv posibles:
v = x   w = zy    wv = zyx
xzy < o = zyx
xzy es palabra no estandar
```

Ejemplo 6.2.2.
Palabra u = vw = zzywwwwwwxzyyzzywwwww
Longitud de la palabra = 21
Representaciones wv posibles:

v = z w = zywwwwwwxzyyzzywwwww	wv = zywwwwwwxzyyzzywwwwwz
v = zz w = ywwwwwwxzyyzzywwwww	wv = ywwwwwwxzyyzzywwwwwzz
v = zzy w = wwwwwwxzyyzzywwwww	wv = wwwwwwxzyyzzywwwwwzzy
v = zzyw w = wwwwwxzyyzzywwwww	wv = wwwwwxzyyzzywwwwwzzyw
v = zzyww w = wwwwxzyyzzywwwww	wv = wwwwxzyyzzywwwwwzzyww
v = zzywww w = wwwxzyyzzywwwww	wv = wwwxzyyzzywwwwwzzywww
v = zzywwww w = wwxzyyzzywwwww	wv = wwxzyyzzywwwwwzzywwww
v = zzywwwww w = wxzyyzzywwwww	wv = wxzyyzzywwwwwzzywwwww
v = zzywwwwww w = xzyyzzywwwww	wv = xzyyzzywwwwwzzywwwwww
v = zzywwwwwwx w = zyyzzywwwww	wv = zyyzzywwwwwzzywwwwwx
v = zzywwwwwwxz w = yyzzywwwww	wv = yyzzywwwwwzzywwwwwxz
v = zzywwwwwwxzy w = yzzywwwww	wv = yzzywwwwwzzywwwwwxzy
v = zzywwwwwwxzyy w = zzywwwww	wv = zzywwwwwzzywwwwwxzyy

zzywwwwwwxzyyzzywwwww < o = zzywwwwwzzywwwwwxzyy
zzywwwwwwxzyyzzywwwww es palabra no estandar

Ejemplo 6.2.3.
Palabra u = vw = zzzywwwwwzzyzwzzxxxzzzx
Longitud de la palabra = 23
Representaciones wv posibles:

v = z w = zzywwwwwzzyzwzzxxxzzzx	wv = zzywwwwwzzyzwzzxxxzzzxz
v = zz w = zywwwwwzzyzwzzxxxzzzx	wv = zywwwwwzzyzwzzxxxzzzxzz
v = zzz w = ywwwwwzzyzwzzxxxzzzx	wv = ywwwwwzzyzwzzxxxzzzxzzz
v = zzzy w = wwwwwzzyzwzzxxxzzzx	wv = wwwwwzzyzwzzxxxzzzxzzzy
v = zzzyw w = wwwwzzyzwzzxxxzzzx	wv = wwwwzzyzwzzxxxzzzxzzzyw
v = zzzyww w = wwwzzyzwzzxxxzzzx	wv = wwwzzyzwzzxxxzzzxzzzyww
v = zzzywww w = wwzzyzwzzxxxzzzx	wv = wwzzyzwzzxxxzzzxzzzywww
v = zzzywwww w = wzzyzwzzxxxzzzx	wv = wzzyzwzzxxxzzzxzzzywwww
v = zzzywwwww w = zzyzwzzxxxzzzx	wv = zzyzwzzxxxzzzxzzzywwwww
v = zzzywwwwwz w = zyzwzzxxxzzzx	wv = zyzwzzxxxzzzxzzzywwwwwz
v = zzzywwwwwzz w = yzwzzxxxzzzx	wv = yzwzzxxxzzzxzzzywwwwwzz
v = zzzywwwwwzzy w = zwzzxxxzzzx	wv = zwzzxxxzzzxzzzywwwwwzzy
v = zzzywwwwwzzyz w = wzzxxxzzzx	wv = wzzxxxzzzxzzzywwwwwzzyz
v = zzzywwwwwzzyzw w = zzxxxzzzx	wv = zzxxxzzzxzzzywwwwwzzyzw
v = zzzywwwwwzzyzwz w = zxxxzzzx	wv = zxxxzzzxzzzywwwwwzzyzwz
v = zzzywwwwwzzyzwzz w = xxxzzzx	wv = xxxzzzxzzzywwwwwzzyzwzz
v = zzzywwwwwzzyzwzzx w = xxzzzx	wv = xxzzzxzzzywwwwwzzyzwzzx
v = zzzywwwwwzzyzwzzxx w = xzzzx	wv = xzzzxzzzywwwwwzzyzwzzxx
v = zzzywwwwwzzyzwzzxxx w = zzzx	wv = zzzxzzzywwwwwzzyzwzzxxx
v = zzzywwwwwzzyzwzzxxxz w = zzx	wv = zzxzzzywwwwwzzyzwzzxxxz
v = zzzywwwwwzzyzwzzxxxzz w = zx	wv = zxzzzywwwwwzzyzwzzxxxzz
v = zzzywwwwwzzyzwzzxxxzzz w = x	wv = xzzzywwwwwzzyzwzzxxxzzz

zzzywwwwwzzyzwzzxxxzzzx > todas las representaciones posibles wv
por lo tanto zzzywwwwwzzyzwzzxxxzzzx si es palabra estandar

Ejemplo 6.2.4.

Palabra u = vw = xxxx
Longitud de la palabra = 4
Representaciones wv posibles:
v = x w = xxx wv = xxxx
xxxx < o = xxxx
xxxx es palabra no estandar

El programa estándar será de utilidad para desarrollar otros programas subsecuentes. También es útil para resolver problemas como el siguiente: considerando un conjunto de todas las palabras de las letras x_1, x_2, x_3 ¿cuál es el número de palabras estándares que tiene ≤ 4 letras?

6.3 Teorema de Lyndon

Cualquier palabra u tiene una representación de la forma $u = w_1^{n_1} w_2^{n_2} \ldots w_k^{n_k}$ (forma normal de Lyndon), donde $w_1 < w_2 < w_3 < \ldots < w_{k-1} < w_k$, y todas las palabras $w_1, \ldots, w_k$ son palabras estándares.

Por ejemplo $x_2x_1x_2x_3x_1x_1x_1 = x_2 \bullet x_1x_2x_3 \bullet x_1^3$ se observa que $x_2 < x_1x_2x_3 < x_1$
La palabra w_1 es el inicio de u y se tienen las propiedades:

1) w_1 es palabra estándar.
2) w_1 tiene una longitud que es el máximo posible.
3) $w_1 \leq w_2$.

6.3.1 Algoritmo para desarrollar una palabra en forma de Lyndon (figura 6.3.1).

Este algoritmo pone una palabra en forma de Lyndon separándola por puntos de acuerdo al teorema de Lyndon. Utiliza la función que determina si una palabra es estándar.

Nombre: LyndonCB.exe
Entradas: una palabra.
Salidas: la palabra en la forma de Lyndon.

1. - Inicializa tipos de datos.
2. - Introducir (por teclado) la palabra deseada.
3. - Determinar la longitud de la palabra.
4. - Copia el tamaño de la palabra a la variable cont.
5. - Llamar a la función (estand) que determina si la palabra es estándar; Obtiene la diferencia del total de caracteres "k" y el índice "j" de representaciones de la palabra.
6. - ¿La palabra es estándar?
   Sí: sigue en paso 13.
   No: continúa en paso 7.

7. - Al final de la representación "v" (apv) insertar "•" y un cero (para limitar esta palabra).

8. - Concatenar Ly con apv (j-ésima) para obtener forma de Lyndon.

9. - Determinar la longitud de la nueva palabra apw j-ésima (equivalente a w).

10. - ¿Falta sólo una letra?
Sí: concatenar Ly con apw y continúa en paso 12.
No: obtener nueva palabra "u" y continuar en paso 11.

11. - ¿Faltan más caracteres?
Sí: continúa en el paso 4.
No: continúa en el paso 12.

12. - Exhibe el resultado de la forma de Lyndon en la pantalla. Continúa en el paso 14.

13. - Concatena Ly, apv, apw; continúa en el paso 12.

14. - Fin del programa.

Análisis de la eficiencia en tiempo.

Se considera el peor caso cuando se realizan separaciones ("tipo Lyndon") en cada letra de la palabra en cuestión, lo cual es influido por la estructura de la palabra, ya sea que las subpalabras sean o no estándares. En este caso la función de palabras estándares podría ejecutarse hasta $n-1$ veces. En consecuencia el tiempo de este algoritmo será del orden de n^3, ya que la función para determinar palabras estándares está contenida dentro de un bucle while el cual se ejecuta $n-1$ veces.

Tamaño de la palabra = $n = k$.

Instrucciones que no dependen de $n = c_i$; $0 \le i \le c_{max}$ (una constante).

En el caso peor $k = n - 1$ para la iteración **while**.

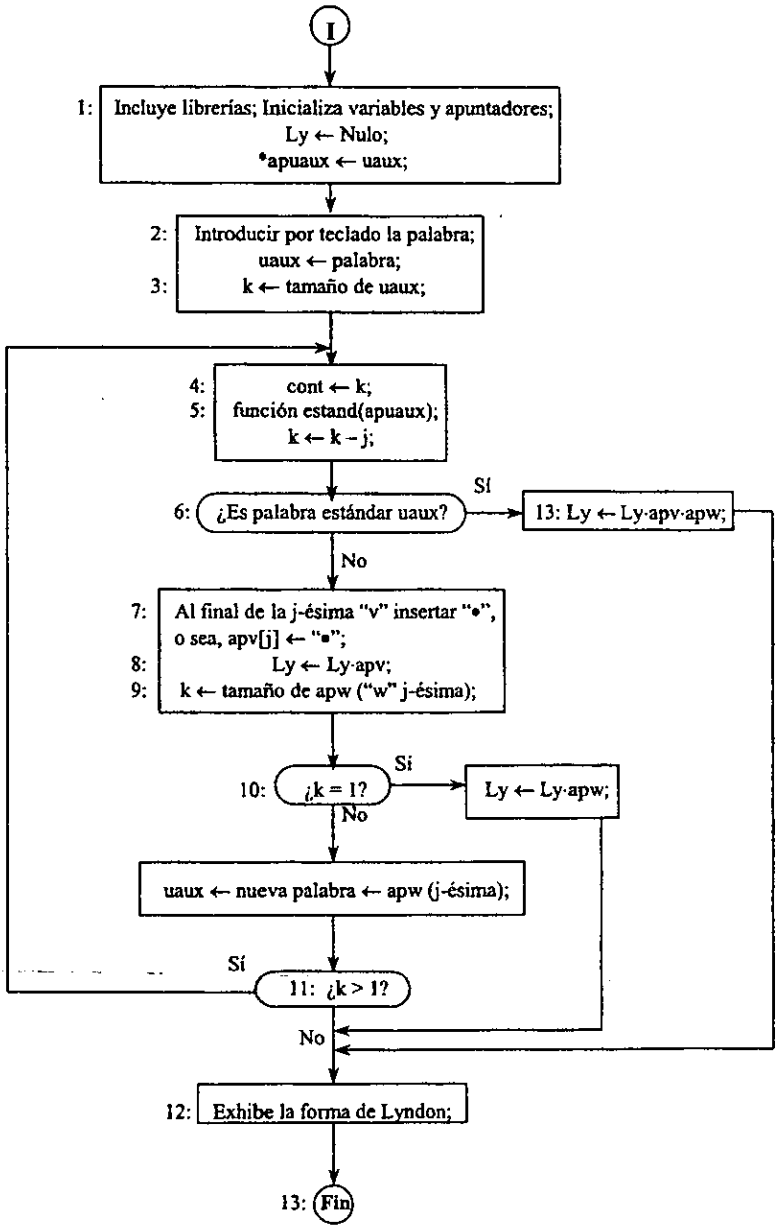
La función estand está dentro del bucle **while** y tiene un orden de cn^2

Entonces $t(n) = (n - 1)\, cn^2 = cn^3$ (orden cúbico).

Eficiencia de espacio.

El espacio de memoria empleado es despreciable, debido a que la cantidad de variables usada es mínima. Se pueden manejar palabras hasta un máximo de 500 caracteres, ya que se asignó memoria para 1000 caracteres por palabra. Al hacer la forma de Lyndon se podría introducir un caracter adicional entre cada dos caracteres de la palabra original para el peor caso. El tamaño de la palabra a manejar se podría aumentar hasta la mitad de la capacidad de la pila de almacenamiento de la máquina. En C++ los caracteres ocupan un espacio de memoria de 1 byte.

Figura 6.3.1. Diagrama de flujo del programa LyndonCB



6.3.2 Ejemplos del programa Lyndon.

Ejemplo 6.3.1.

Programa de Teorema de Lyndon, se ejecuta con: LyndonCB.exe
Introduzca la palabra a formar en Lyndon: lindon
Inicio
La palabra entrada es: lindon
Longitud de la palabra = 6
La forma de Lyndon es: li*nd*on

Ejemplo 6.3.2.

Introduzca la palabra a formar en Lyndon: bcbacc
La palabra entrada es: bcbacc
Longitud de la palabra = 6
La forma de Lyndon es: b*cba*c*c

Ejemplo 6.3.3.

Introduzca la palabra a formar en Lyndon: zazazaz
La palabra entrada es: zazazaz
Longitud de la palabra = 7
La forma de Lyndon es: za*za*za*z

Ejemplo 6.3.4.

Introduzca la palabra a formar en Lyndon: dcccbdbcba
La palabra entrada es: dcccbdbcba
Longitud de la palabra = 10
La forma de Lyndon es: dcccbdbcba

Ejemplo 6.3.5.

Introduzca la palabra a formar en Lyndon: zzzzzzzaz
Inicio
La palabra entrada es: zzzzzzzaz
Longitud de la palabra = 9
La forma de Lyndon es: zzzzzzza*z

6.4 Teorema de Shirshov, Palabras no Asociativas

Una *palabra no asociativa* es aquella donde "[,]" (los corchetes) se arreglan de algún modo para mostrar como se aplica la multiplicación. El conjunto de palabras no asociativas puede ser definido inductivamente mediante los siguientes axiomas:

(1) todas las letras son palabras no asociativas;
(2) si $[v]$ y $[w]$ son palabras no asociativas entonces $[[v][w]]$ es una palabra no asociativa;
(3) no existen otras palabras no asociativas.

Definición 6.4.1. Una palabra no asociativa $[u]$ se dice ser *estándar* (en el sentido de Shirshov) si:

(1) una palabra (asociativa) u obtenida de esta palabra quitando los corchetes es estándar;
(2) si $[u] = [[v][w]]$ entonces $[v]$ y $[w]$ son palabras no asociativas estándares;
(3) si $[u] = [[[v_1][v_2]][w]]$ entonces $v_2 \leq w$.

6.4.1 Teorema de Shirshov. Cada palabra estándar tiene una y solo una distribución de corchetes tal que la palabra no asociativa resultante es estándar.

Éste teorema, combinado con la definición inductiva de un conjunto de todas las palabras no asociativas, inmediatamente implica que cada palabra estándar (asociativa) u tiene una descomposición $u = vw$, donde $v > w$ y v y w son estándares. Aun, para la descomposición asociativa (distinta de la no asociativa), las palabras v y w no están definidas en forma única. Los factores v y w en la descomposición no asociativa $[u] = [[v][w]]$, observamos que, se pueden definir para ser palabras estándares tal que $u = vw$, donde v tiene una longitud mínima posible.

6.4.2 Algoritmo para obtener la representación en palabras estándares no asociativas (figura 6.4.1).

Este algoritmo es relativamente complejo por lo que a continuación se da una descripción de su operación basada en el siguiente ejemplo (considere **z > a**):

zywzys (palabra estándar) **6.4.1**

z (el primer elemento de izquierda a derecha) es estándar; **ywzys** es no estándar, por lo cual se toma **y**; entonces **zy** es estándar; **wzys** es no estándar, se toma **w**, entonces **zyw** es estándar; **zys** es estándar también. Se agregan los corchetes correspondientes **[[zyw][zys]]**. Observe que **zyw > zys**. Se toma **zyw** y se procesa de igual forma obteniendo **[z[yw]]** la cual está ya en su mínima expresión (observe que **yw < zys** de acuerdo con (3) de la definición 6.4.1), falta desarrollar **[zys]** con el mismo procedimiento quedando como resultado final:

[[z[yw]][z[ys]]] **6.4.2**

Resumen: Se verifica si la palabra insertada es estándar. Siendo estándar continúa el proceso, en caso contrario el programa se termina. Se toma el primer elemento de la palabra original. Se pregunta si el resto de la palabra (segunda subpalabra) es estándar. Si es estándar la primera subpalabra se almacena en una variable y la segunda subpalabra en otra variable. Si la segunda subpalabra es no estándar se quita su primer elemento y se concatena éste a la primera subpalabra continuando este proceso hasta que la segunda subpalabra es estándar. En este momento se tendrán dos subpalabras las cuales deberán ser estándares. Ahora se toma la primera subpalabra como palabra original y el proceso se repite hasta llegar a la mínima expresión (con una o dos letras) de la primera subpalabra; También es aquí en donde se colocan los corchetes de las expresiones mínimas. En cada obtención de dos subpalabras estándares se hace un registro de los corchetes que se deben obtener tanto de izquierdos como de derechos de la expresión general, y cuando el programa pasa a tomar la primera subpalabra deja pendiente por procesar a la segunda subpalabra. Las subpalabras pendientes se registran en un contador de palabras pendientes de procesar para su posterior proceso como si fuera palabra original. Cada vez que se toma una subpalabra pendiente para su proceso se actualizan los indicadores y apuntadores.

Nombre: eaenacb.exe
Entradas: una palabra en forma estándar asociativa.
Salidas: la palabra entrada representada en forma estándar no asociativa

1.- Inicializa variables, contadores, y apuntadores. Se ingresa la palabra en la variable cad, se determina su longitud con longpal y se hacen copias en las variables cadaux y cav.

2.- La palabra cad es estándar?
Sí: va al paso 3. No: termina el programa.

3.- ¿La longitud de la palabra cad es mayor que 2?
Sí: continúa en el paso 4. No: Termina programa.

4.- Determina la longitud de la palabra de la variable cad (longcad); Inicia un bucle **do**.

5.- ¿La longitud de la palabra en cad es mayor de 2?
Sí: continúa en el paso 6. No: continúa en el paso 20.

6.- Incrementa los contadores de corchetes izquierdos (contcori) y derechos(contcord) .

7.- Inicializa índice j = 1 e índice de palabras i = 0.

8.- Hace una separación de la palabra original tomando inicialmente un caracter para cau (primera parte) y el resto para cav (segunda parte).

9.- Determina la longitud de cav y esta se copia a las variables apsbp[cp + 1] y palau.

10.- Inicia bucle. ¿La palabra palau es estándar?
Sí: Continúa en el paso 11. No: continúa en el paso 13.

11.- Copia la parte de la variable cau en la variable apsbp[cp] y se determina su longitud.

12.- La parte en la variable cav se copia en otra variable apsbp[cp+1]. Se copia cav a la variable cad y se determina su longitud. Continúa en el paso 18.

13.- La segunda parte de la palabra (cav) no es estándar por lo que el primer elemento de ésta se quita y se concatena a la primera parte (cau).

14.- Guarda la variable cau en la variable apsbp[cp]; incrementa el índice i en uno, j = 1 y se copia cav en cad.

15.- Nuevamente se obtiene la segunda subpalabra en cav y se determina su longitud con longcav; se copia cav en la variable apsbp[cp+1] que le corresponde; se copia cav a palau.

16.- ¿Es la segunda parte (cav) de la palabra menor que dos caracteres (longcav < 2)?
Sí: continúa en el paso 17. No: continúa en paso 18.

17.- Copia cav a su localidad correspondiente apsbp[cp+1].

18.- Regresa al paso 10 mientras la subpalabra palau sea no estándar y aún existen dos o más caracteres en la segunda parte de la palabra (en cav), de otro modo continúa en el paso 19.

19.- Copia la segunda subpalabra de apsbp[cp+1] a la variable apend[pend] (indicando que está pendiente a ser procesada); incrementa en uno los contadores de palabras pendientes pe y pend; copia la primera subpalabra de apsbp[cp] en la variable cad, y se determina su longitud. Continúa en el paso 38.

20.- ¿La longitud de la primera subpalabra (cad) es igual a 1?
Sí: continúa en paso 21. No: continúa en paso 27.

21.- Agrega corchetes izquierdos según cantidad en contcori a la cadena general.

22.- Contador de corchetes izquierdos contcori = 0.

23.- Agrega a la cadena general (cadgral) la primera parte obtenida de la división de la palabra contenida en apsbp[cp].

24.- Concatena a cadgral la cantidad de corchetes derechos indicados en concord[pe].

25.- Inicializa contcord[pe] = 0, decrementa contador pe en uno, incrementa cp (contador de palabras) en uno, disminuye el contador pend (pendientes).

26.- Copia la subpalabra pendiente de procesar en apend[pend] a apsbp[cp] y a cad; determina la longitud de cad. Continúa en el paso 38.

27.- Copia la palabra a verificar en cad y se llama la función estand.

28.- ¿ Es estándar la subpalabra cad?
Sí: continúa en el paso 29. No: sontinúa en el paso 33.

29.- Agrega corchetes izquierdos según lo indique contcori.

30.- Reinicializa contcori = 0 y se agrega un corchete izquierdo más; concatena la primera subpalabra apsbp[cp] (indicada por cp contador de palabras) a la cadena general (cadgral); agrega un corchete derecho a la cadena general.

31.- Agrega tantos corchetes derechos según lo indique contcord[pe].

32.- Inicializa contcord[pe] = 0; disminuye en uno a pe; incrementa en uno a cp y disminuye en uno a pend; copia apend[pend] en apsbp[cp] y en cad y se determina su longitud; continúa en el paso 38.

33.- Agrega corchetes izquierdos de acuerdo a contcori.

34.- Inicializa contcori = 0; copia apsbp[cp] en cadaux; separa el primer caracter de la palabra en apsbp[cp].

35.- Incrementa en uno el contador de palabras cp.

36.- apsbp[cp] toma el siguiente caracter de la palabra contenida en cadaux; incrementa en uno el contador de palabras cp.

37.- Copia la palabra de apend[pend] en apsbp[cp] y en cad; determina la longitud de cad.

38.- ¿Hay aún palabras pendientes (pend>=0)?
Si: continúa en el paso 5. No: continúa en el paso 39.

39.- Exhibe las subpalabras no asociativas que forman la palabra original

40.- Exhibe la palabra resultante en la forma estándar no asociativa.

41.- Fin de programa.

Análisis de la eficiencia en tiempo.

El peor caso se tendría cuando cada una de las letras que forman la palabra estándar asociativa tuvieran que ser separadas todas por corchetes. El tiempo de este algoritmo será del orden de n^4, lo cual se deriva de los bucles en el mismo algoritmo.

Tamaño de la palabra = $n = k$.

Instrucciones que no dependen de $n = c_i$; $0 \leq i \leq c_{max}$ (una constante).

Primer bucle **do** se ejecuta $n - 2$ veces.

Existen bucles **for** internos que se ejecutan $n - 1$ veces.

La función estand es del orden n^2 y está dentro del primer bucle **do**.

El segundo bucle **do** es interno al primer bucle **do** y se ejecuta $n - 2$ veces y contiene además a la función estand de orden n^2.

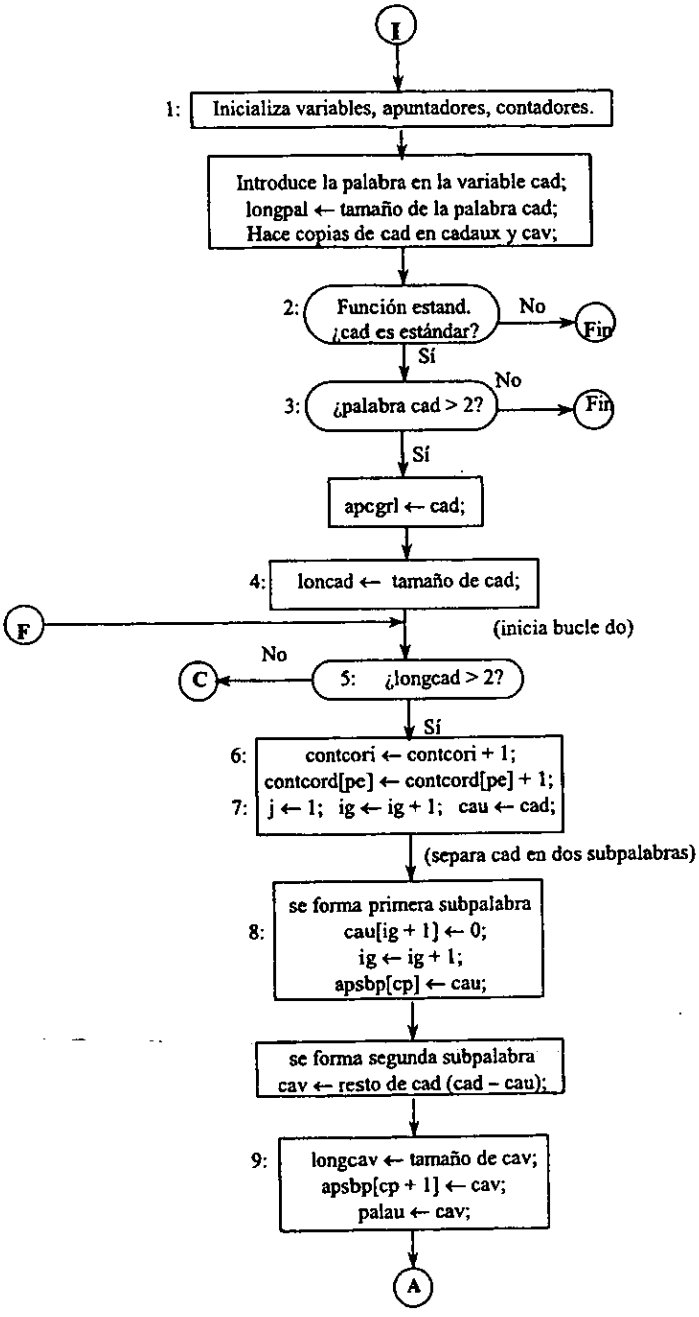
Entonces $t(n) = c_{max}(n-2)\{n^2 + n-1 + (n-2)[n^2 + (n-1)]\}$.

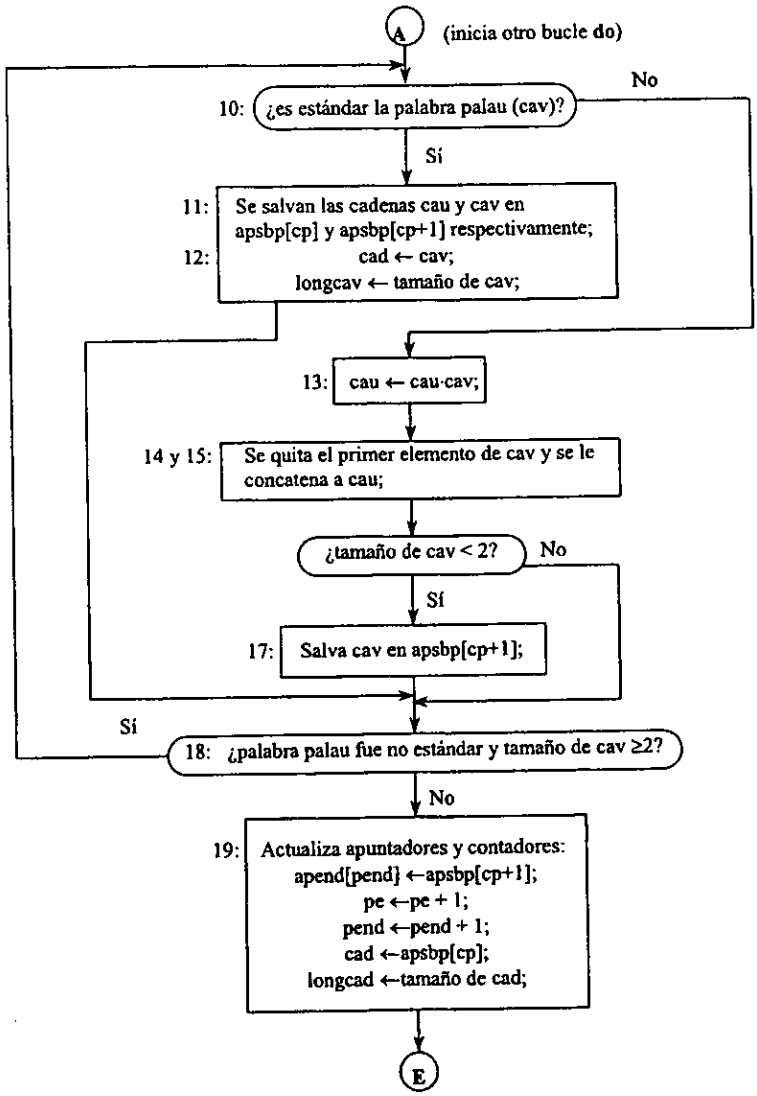
El orden es por lo tanto n^4.

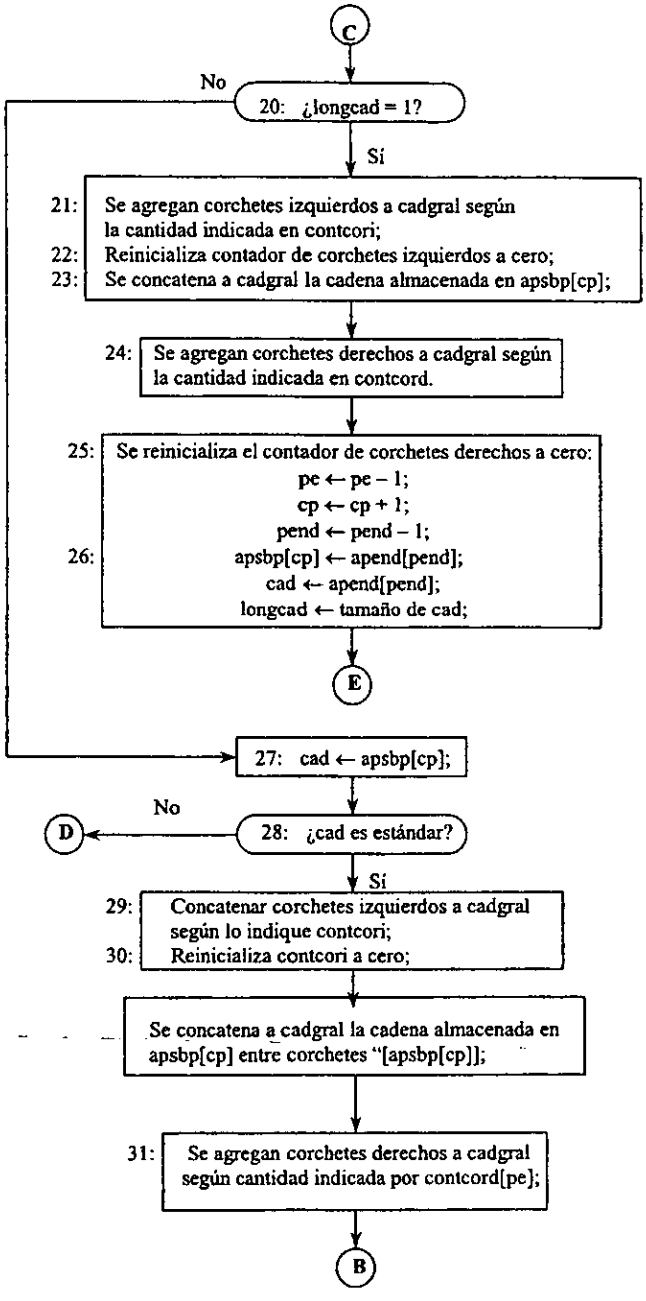
Eficiencia de espacio.

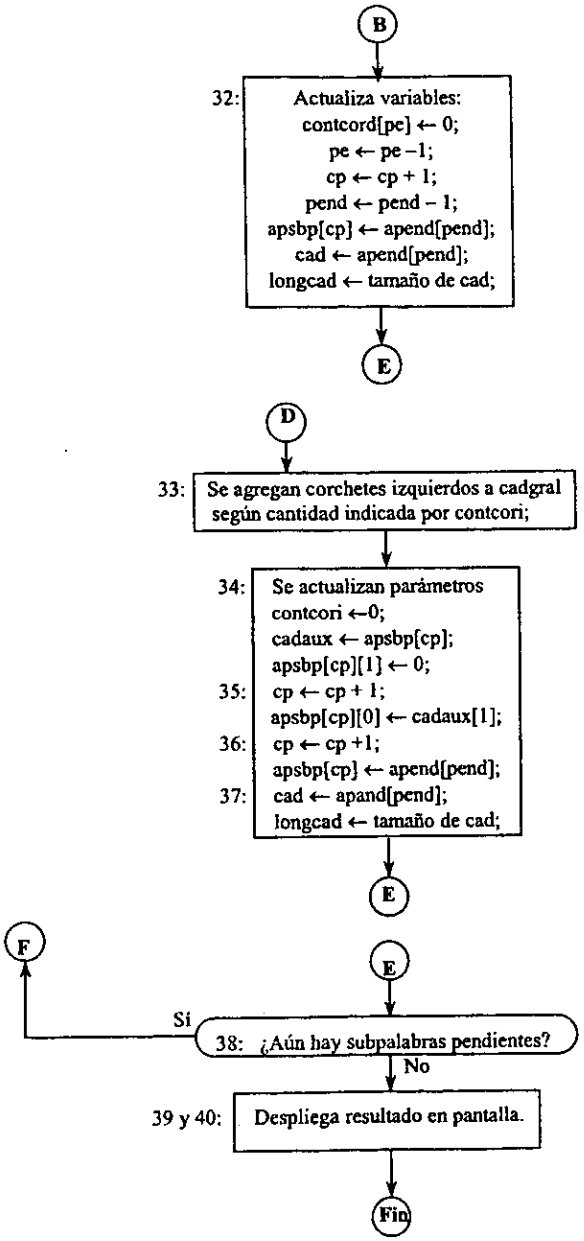
El espacio de memoria empleado es despreciable, aunque una simple palabra de 3 letras genera un mínimo de 6 corchetes. Únicamente se debe tener cuidado con el tamaño del resultado en la cadena general, ya que esta aumentará su tamaño. El tamaño de la palabra que se introduce por teclado que se puede manejar se reduce hasta en un tercio de la capacidad asignada a los arreglos en el programa. En este caso se hicieron asignaciones de memoria de 1000 bytes lo cual nos permite manejar palabras con un tamaño máximo aproximado de 350 caracteres, en caso de tener que manejar palabras mayores, sólo se tendría que cambiar el valor de la asignación de memoria a un valor más alto sin que esto represente problema alguno.

Figura 6.4.1. Diagrama de Flujo del Programa eaenacb.exe




A
(inicia otro bucle do)
No
10: ¿es estándar la palabra palau (cav)?
Sí
11: Se salvan las cadenas cau y cav en apsbp[cp] y apsbp[cp+1] respectivamente;
12: cad ← cav;
longcav ← tamaño de cav;
13: cau ← cau·cav;
14 y 15: Se quita el primer elemento de cav y se le concatena a cau;
¿tamaño de cav < 2?
No
Sí
17: Salva cav en apsbp[cp+1];
Sí
18: ¿palabra palau fue no estándar y tamaño de cav ≥2?
No
19: Actualiza apuntadores y contadores:
apend[pend] ←apsbp[cp+1];
pe ←pe + 1;
pend ←pend + 1;
cad ←apsbp[cp];
longcad ←tamaño de cad;
E


C
No
20: ¿longcad = 1?
Sí
21: Se agregan corchetes izquierdos a cadgral según
la cantidad indicada en contcori;
22: Reinicializa contador de corchetes izquierdos a cero;
23: Se concatena a cadgral la cadena almacenada en apsbp[cp];
24: Se agregan corchetes derechos a cadgral según
la cantidad indicada en contcord.
25: Se reinicializa el contador de corchetes derechos a cero:
pe ← pe – 1;
cp ← cp + 1;
pend ← pend – 1;
26: apsbp[cp] ← apend[pend];
cad ← apend[pend];
longcad ← tamaño de cad;
E
27: cad ← apsbp[cp];
No
D
28: ¿cad es estándar?
Sí
29: Concatenar corchetes izquierdos a cadgral
según lo indique contcori;
30: Reinicializa contcori a cero;
Se concatena a cadgral la cadena almacenada en
apsbp[cp] entre corchetes "[apsbp[cp]];
31: Se agregan corchetes derechos a cadgral
según cantidad indicada por contcord[pe];
B


B
32:
Actualiza variables:
contcord[pe] ← 0;
pe ← pe −1;
cp ← cp + 1;
pend ← pend − 1;
apsbp[cp] ← apend[pend];
cad ← apend[pend];
longcad ← tamaño de cad;
E
D
33:
Se agregan corchetes izquierdos a cadgral
según cantidad indicada por contcori;
34:
Se actualizan parámetros
contcori ←0;
cadaux ← apsbp[cp];
apsbp[cp][1] ← 0;
35:
cp ← cp + 1;
apsbp[cp][0] ← cadaux[1];
36:
cp ← cp +1;
apsbp[cp] ← apend[pend];
37:
cad ← apand[pend];
longcad ← tamaño de cad;
E
F
E
Sí
38:
¿Aún hay subpalabras pendientes?
No
39 y 40:
Despliega resultado en pantalla.
Fin

6.4.3 Ejemplos del teorema de Shirshov.

Ejemplo 6.4.1

PROGRAMA eaenacb.exe
PROGRAMA DE TEOREMA DE SHIRSHOV PARA REPRESENTAR PALABRAS ESTANDAR NO ASOCIATIVAS
Este programa representa una palabra estandar asociativa dada
en forma estandar no asociativa
Inicio
Introduzca la palabra
zywzys
Resultado, palabra estandar no asociativa: [[z[yw]][z[ys]]]
FIN DE PROGRAMA

Ejemplo 6.4.2

Inicio
Introduzca la palabra
zyyy
Resultado, palabra estandar no asociativa: [[[zy]y]y]

Ejemplo 6.4.3

Inicio
Introduzca la palabra
azzz
Error: palabra no estandar. Termina programa

Ejemplo 6.4.4

Inicio
Introduzca la palabra
zzzzyzzzyzzzzw
Resultado, palabra estandar no asociativa: [[[z[z[z[zy]]]][z[z[zy]]]][z[z[z[zw]]]]]

Ejemplo 6.4.5

Inicio
Introduzca la palabra
955555951
Resultado, palabra estandar no asociativa: [[[[[[95]5]5]5]5][9[51]]]

6.5 Desarrollos cuánticos.

6.5.1 Conmutador cuántico.

Sean $x_1, x_2, \ldots, x_n$ *variables cuánticas*, y dada una matriz de cuantización P = $\| p_{ij} \|$ se establece:

$$[x_i, x_j] = x_i x_j - p_{ij} x_j x_i$$

Si u, v son palabras: $u = x_{i_1} x_{i_2} \ldots x_{i_k}$; $v = x_{j_1} x_{j_2} \ldots x_{j_m}$,

entonces $p_{u,v} = p_{i_1 j_1} \cdot p_{i_1 j_2} \cdot p_{i_1 j_3} \cdot \ldots p_{i_1 j_m} \cdot p_{i_2 j_1} \cdot p_{i_2 j_2} \ldots p_{i_2 j_m} \cdot p_{i_3 j_1} \cdot p_{i_3 j_2} \ldots$

Por ejemplo: $p_{x_1 x_2, x_2 x_3} = p_{12} p_{13} p_{22} p_{23}$.

Lema 6.5.1 $p_{u,vw} = p_{u,v} \cdot p_{u,w}$; $p_{uv,w} = p_{u,w} \cdot p_{v,w}$.

En general el conmutador cuántico es:

$$[u, v] = uv - p_{u,v} vu$$

6.5.2 Criterio de Specht-Wever como ejemplo de aplicación.

a) Operador de Specht-Wever

$$\sigma(x_{i_1} x_{i_2} \ldots x_{i_k}) = [\ldots[[x_{i_1} x_{i_2}] x_{i3}] \ldots] x_{i_k}].$$

Por ejemplo:

$$\sigma(x_1 x_2) = [x_1, x_2] = x_1 x_2 - p_{12} x_2 x_1.$$

$$\sigma(x_1 x_2 x_3) = [[x_1, x_2], x_3] = (x_1 x_2 - p_{12} x_2 x_1) x_3 - p_{13} p_{23} x_3 (x_1 x_2 - p_{12} x_2 x_1)$$
$$= x_1 x_2 x_3 - p_{12} x_2 x_1 x_3 - p_{13} p_{23} x_3 x_1 x_2 + p_{13} p_{23} p_{12} x_3 x_2 x_1.$$

b) Condición de Specht-Wever.

Si un polinomio f tiene grado n, entonces la condición Spetch-Wever tiene la forma

$$\sigma(f) = n \cdot f.$$

Se plantean ejercicios como el siguiente: supongamos que $p_{12} = p_{21} = 1$, probar que $\sigma([x_1 x_2]) = 2[x_1 x_2]$.

6.5.3 Planteamiento del problema algorítmico.

Dada una expresión como por ejemplo:

[d,[[d,c],[d,[b,a]]]], 6.5.1

desarrollar los corchetes empleando la teoría del párrafo 6.5.1 de arriba.

Solución: desarrollar los primeros corchetes más internos y sustituir el resultado en la cadena original, o sea para el ejemplo: se desarrolla primero

$$[d,c] = dc - P(dc)cd \qquad 6.5.2$$

y sustituyendo en 6.5.2 la nueva cadena queda:

$$[d,[dc - P(dc)cd,[d,[b,a]]]] \qquad 6.5.3$$

en seguida se toma el siguiente nuevo conjunto de corchetes más internos [b,a] y se repite el proceso hasta terminar de desarrollar todos los corchetes.

6.5.4 Algoritmo de desarrollos cuánticos (figura 6.5.1).

Este algoritmo hace el desarrollo de corchetes cuánticos. La expresión que se inserte debe tener los corchetes y comas completos y las palabras deben estar en letras minúsculas. Los parámetros se manejan en forma simbólica. El programa usa tres funciones: estand, oplie7B, y ordenaB.

Resumen: Se inserta la cadena según se indica en el párrafo anterior; el programa busca un juego de corchetes abiertos y cerrados más internos y desarrolla la expresión dentro de estos, eliminando los corchetes después del desarrollo; repite el procedimiento hasta agotar todos los corchetes. Las comas ayudan a hacer la división de una palabra dentro de los corchetes en dos subpalabras $[u, v] = uv - p_{u,v}vu$.

Nombre del programa descua5b.exe
Entradas: una palabra con corchetes y comas.
Salidas: desarrollo cuántico de la expresión entrada.

Algoritmo completo:

1.- Inicio. Declara variables y apuntadores y limpia la pantalla.

2.- Exhibe en pantalla el mensaje del programa.

3.- Pregunta si se quiere utilizar una cadena ya dispuesta como de demostración.
Sí: continúa en el paso 4.
No: Solicita ingresar la cadena completa a ser procesada. Continúa en el paso 4.

4.- La cadena insertada se guarda en la variable cad y se determina su longitud con tamcad.

5.- Verifica corchetes abiertos y cerrados, en caso de desigualdad u omisión se avisa y se termina el programa.

6.- Obtiene la cadena sin corchetes ni comas, o sea únicamente caracteres alfabéticos; verifica si esta es palabra estándar. En caso de que no sea palabra estándar, el programa termina.

7.- Obtiene la cantidad y los diferentes generadores que intervienen en la expresión así como la longitud de la cadena sin corchetes.

8.- Se retoma la cadena original. Busca el primer corchete abierto.

9.- ¿Existen corchetes abiertos?
Sí: continúa en paso 10. No: continúa en el paso 51 (ya no hay mas desarrollos).

10.- Inicializa índices i = 0, k = 0, j = 0.

11.- ¿El índice i es menor o igual que el tamaño de la cadena (¿se revisaron todos los caracteres?)?
Sí: continúa en el paso 12. No: continúa en el paso 49.

12.- Inicializa apuntador de cadena k = 0.

13.- ¿El caracter es un corchete abierto?
Sí: i = i + 1. Continúa en el paso 14. No: i = i + 1. Continúa en el paso 48 a buscar otro corchete "[".

14.- Se genera la primera parte de la palabra (hasta donde se encontró el corchete abierto), se guarda en cadini y la delimita con un nulo.

15.- ¿Es el siguiente un caracter alfabético (en cadaux[i])?
Sí: continúa en el paso 16. No: continúa en el paso 48 a buscar otro corchete "[".

16.- cau[k] = cadaux[i] inicio de primera parte de palabra (u). Variable control de bucle x = 0.

17.- Inicia un bucle. i = i +1, k = k + 1. cau[k] = cadaux[i] (actualiza índices y palabra cau).

18.- ¿El siguiente caracter cadaux[i] es una coma? (se debe encontrar una coma).
Sí: variable x = 1. Continúa en el paso 26. No: continúa en el paso 19.

19.- ¿Es una coma el caracter cadaux[i]?
Sí: x=1. Continúa en el paso 26. No: continúa en el paso 20.

20.- ¿cadaux[i] es caracter alfabético (minúsculas)?
Sí: continúa en el paso 25. No: continúa en el paso 21

21.- ¿cadaux[i] es un signo "+"?
Sí: continúa en el paso 25. No: continúa en el paso 22.

22.- ¿cadaux[i] es un signo "-"?
Sí: continúa en el paso 25. No: continúa en el paso 23.

23.- ¿cadaux[i] es un caracter "P" o "(" o ")"?
Sí: continúa en el paso 25. No: continúa en el paso 24.

24.- ¿cadaux[i] es un signo "*"?
Sí: continúa en el paso 25. No: continúa en el paso 26.

25.- ¿Es el índice i menor que tamcad?
Sí: continúa en el paso 17. No: continúa en el paso 26.

26.- ¿El control de bucle x es cero?
Sí: continúa en el paso 48. No: continúa en el paso 27.

27.- Delimita la primera parte (u) de la palabra con cau[k] = 0 .

28.- Copia cau a un apuntador de la misma apcau.

29.- Inicializa control de bucle x = 0; actualiza índice (de caracteres de la cadena) i = i + 1.

30.- ¿cadaux[i] es un caracter alfabético?
Sí: continúa en el paso 31. No: continúa en el paso 48.

31.- Inicializa índice k = 0 y asigna el caracter inicial a cav (cav[k] = cadaux[i]) la cual es la segunda parte (v) de la palabra a desarrollar.

32.- Inicia bucle; actualiza índices i = i + 1; k = k + 1; nueva asignación cav[k] = cadaux[i].

33.- Busca un corchete de cierre, ¿cadaux[i] = "]"?
Sí: x = 1 y continúa en el paso 40. No: continúa en el paso 34.

34.- Busca más caracteres alfabéticos. ¿cadaux[i] es un caracter alfabético?
Sí: continúa en el paso 39. No: continúa en el paso 35.

35.- ¿cadaux[i] es un signo "+"?
Sí: continúa en el paso 39. No: continúa en el paso 36.

36.- ¿cadaux[i] es un signo "-"?
Sí: continúa en el paso 39. No: continúa en el paso 37.

37.- ¿cadaux[i] es un caracter "P" o "(" o ")"?
Si: continúa en el paso 39. No: continúa en el paso 38.

38.- ¿cadaux[i] es un signo "*"?
Sí: continúa en el paso 39. No: x = 0; continúa en el paso 40.

39.- ¿El índice i es menor que tamcad?
Sí: continúa en el paso 32. No: continúa en el paso 40.

40:- ¿El controlador de bucle x es cero?
Si: continúa en el paso 48. No: continúa en el paso 41.

41.- Delimita la segunda parte (v) de la palabra con cav[k] = 0:

42.- Copia cav a un apuntador de la misma apcav.

43.- Inicializa controlador de bucle x = 0; se genera la parte restante de la cadena y se guarda en cadfin.

44.- Ya que se tienen las dos divisiones [u, v] de la palabra, se llama a la función que realiza la operación de Lie (oplie) o desarrollo de los corchetes para ese juego de palabras. Regresa el resultado en la variable caw.

45.- Copia el resultado a un apuntador (caw→cades).

46.- Une las partes cadini, cades y cadfin para formar nuevamente la cadena completa pero con el desarrollo de corchetes previamente realizado. Guarda esta composición en cad y se determina su nueva longitud con tamcad; Se copia cad en cadaux y cades.

47.- Prepara índice de parámetros para el siguiente desarrollo ipar = ipar + 1.

48.- Continúa en el paso 11.

49.- Se determina si aún existen corchetes "[".

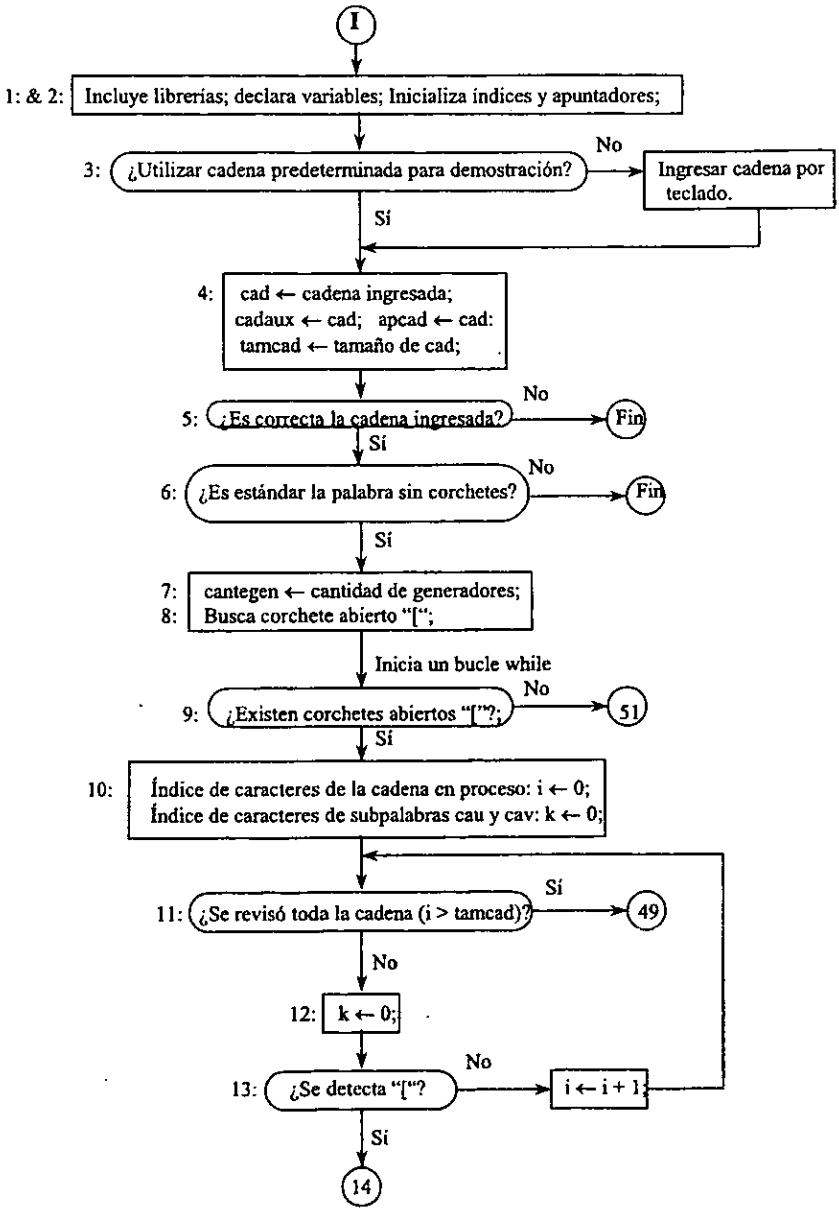
50.- Continúa en el paso 9.

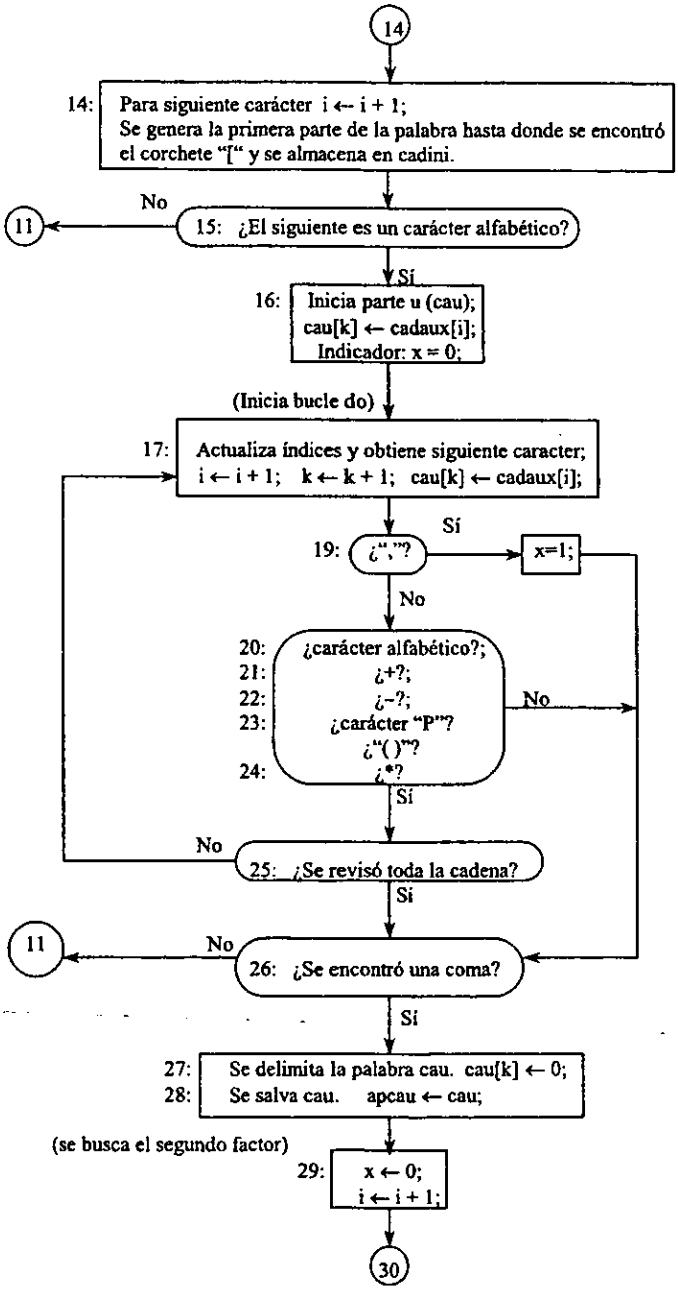
51.- Llama a la función ordena, la cual ordena la cadena resultante escribiendo primero los parámetros de cuantización cuando los hay.

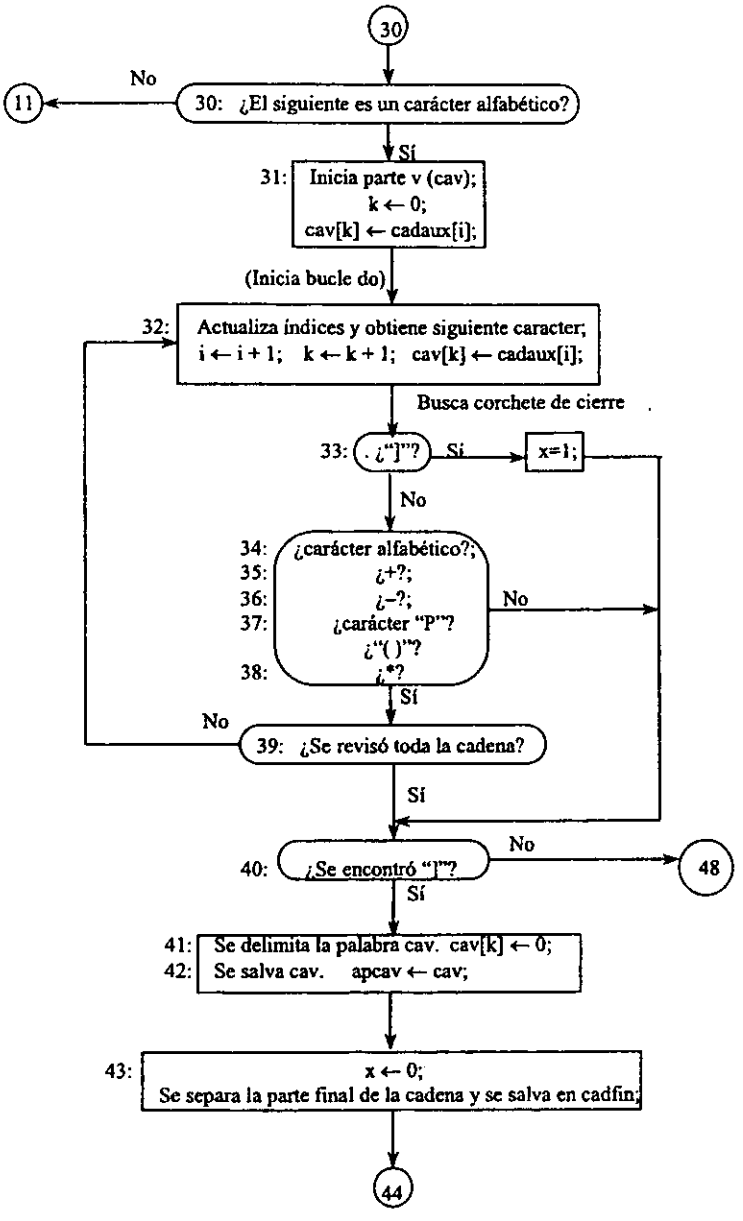
52.- Se exhibe en la pantalla la cadena resultante desarrollada.

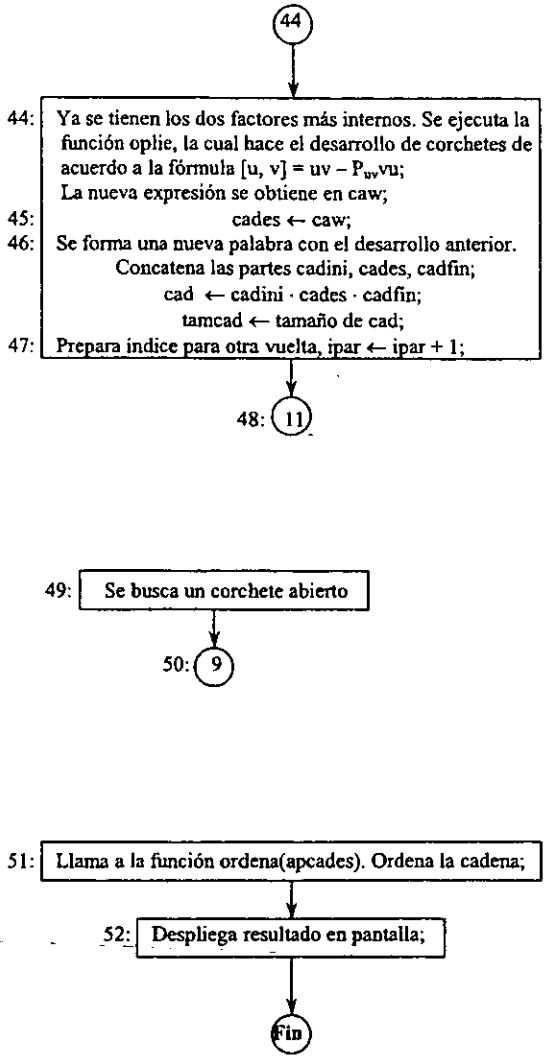
53.- Fin del programa.

Figura 6.5.1. Diagrama de Flujo del Programa descua5b.exe




14
14: Para siguiente carácter i ← i + 1;
Se genera la primera parte de la palabra hasta donde se encontró el corchete "[" y se almacena en cadini.
No
11
15: ¿El siguiente es un carácter alfabético?
Sí
16: Inicia parte u (cau);
cau[k] ← cadaux[i];
Indicador: x = 0;
(Inicia bucle do)
17: Actualiza índices y obtiene siguiente caracter;
i ← i + 1; k ← k + 1; cau[k] ← cadaux[i];
Sí
19: ¿","?
x=1;
No
20: ¿carácter alfabético?;
21: ¿+?;
22: ¿-?;
23: ¿carácter "P"?
¿"( )"?
24: ¿*?
No
Sí
No
25: ¿Se revisó toda la cadena?
Si
11
No
26: ¿Se encontró una coma?
Sí
27: Se delimita la palabra cau. cau[k] ← 0;
28: Se salva cau. apcau ← cau;
(se busca el segundo factor)
29: x ← 0;
i ← i + 1;
30


30
No
11
30: ¿El siguiente es un carácter alfabético?
Sí
31: Inicia parte v (cav);
k ← 0;
cav[k] ← cadaux[i];
(Inicia bucle do)
32: Actualiza índices y obtiene siguiente caracter;
i ← i + 1; k ← k + 1; cav[k] ← cadaux[i];
Busca corchete de cierre
33: ¿"]"? Sí x=1;
No
34: ¿carácter alfabético?;
35: ¿+?;
36: ¿-?;
37: ¿carácter "P"?
¿"( )"?
38: ¿*?
No
Sí
No
39: ¿Se revisó toda la cadena?
Sí
40: ¿Se encontró "]"?
No
48
Sí
41: Se delimita la palabra cav. cav[k] ← 0;
42: Se salva cav. apcav ← cav;
43: x ← 0;
Se separa la parte final de la cadena y se salva en cadfin;
44


44
44: Ya se tienen los dos factores más internos. Se ejecuta la función oplie, la cual hace el desarrollo de corchetes de acuerdo a la fórmula [u, v] = uv – P_{uv}vu;
La nueva expresión se obtiene en caw;
45: cades ← caw;
46: Se forma una nueva palabra con el desarrollo anterior.
Concatena las partes cadini, cades, cadfin;
cad ← cadini · cades · cadfin;
tamcad ← tamaño de cad;
47: Prepara índice para otra vuelta, ipar ← ipar + 1;
48: 11
49: Se busca un corchete abierto
50: 9
51: Llama a la función ordena(apcades). Ordena la cadena;
52: Despliega resultado en pantalla;
Fin

Análisis de tiempo de ejecución y espacio de memoria:

Este algoritmo tiene una eficiencia en tiempo determinada por la expansión de los corchetes de Lie y por la estructura misma de la expresión de que se trate. Debido a la complejidad del análisis, este tiempo se ha determinado observando los ejemplos prácticos, mismos que se presentan adelante. Se obtuvo una relación de tiempo de orden exponencial $t(n) = c(e^n)$, donde n es la cantidad de generadores en total. Esto era de esperarse ya que en cada par de factores desarrollados, la cadena crece a un ritmo exponencial. En los bucles además de los generadores se tienen corchetes y comas y se han considerado como una función de n.

El espacio de memoria que se requiere está determinado por la cadena final obtenida principalmente, y por las variables que contienen subpalabras en proceso. Por ejemplo observando el ejemplo 6.5.4 vemos que, a partir de 6 generadores, se ha generado una cantidad de 1200 caracteres aproximadamente y corresponden mas o menos a la expresión $s(n) = (e^{n+1})$, por lo tanto se debe reservar una cantidad relativamente grande de memoria, o bien hacer uso de almacenamiento dinámico en disco. En este programa se han reservado 10000 bytes (10kbytes) para almacenamiento de la cadena final, y como se sabe cada caracter ocupa 1 byte de memoria. También se encontró que no se pueden manejar más de 7 generadores con este diseño de programa. Para reducir la cantidad de caracteres generados en forma sustancial, se tendría que diseñar un programa que realizara las operaciones aritméticas con los parámetros lo cual requeriría alimentar los datos numéricos de los mismos.

Observe que en este caso, así como esta diseñado el programa, n no podrá ser mayor de 7 generadores en total, si se quiere manejar valores mayores lo más recomendable es modificar el programa para utilizar asignación de memoria dinámica.

Algoritmo de la función oplie5B (figura 6.5.2).

Esta función es llamada para realizar la operación cuántica $[u, v] = uv - p_{u,v}vu$.

Entradas: índice de parámetros, primera ("u") y segunda ("v") partes de la palabra dentro de un juego de corchetes pasadas por referencia.
Salidas: la operación cuántica ya desarrollada. Regresa el resultado en la variable caw.

1.- Inicio: declaración de las variables, parámetros de la función y apuntadores e índices.

2.- Obtiene copias de las partes (u, v) de la operación a desarrollar y determina sus longitudes copiando éstas en variables auxiliares; adicionalmente copia las partes ("u, v") nuevamente en otras variables auxiliares.

3.- Separa los caracteres alfabéticos para la parte "u" y para la parte "v" y determina su longitud.

4.- Inicializa nuevas variables chuvaux = 0; gen = 0; k = 0;

5.- Genera parámetros P tomando como base los generadores en "u" y en "v". Obtiene las combinaciones de los parámetros P(u,v) por ejemplo: P(abac,cbe) = p(a,c)p(a,b)p(a,e)p(b,c)p(b,b)p(b,e)p(a,c)p(a,b)p(a,e)p(c,c)p(c,b)p(c,e).

6.- Determina los términos que forman "u" y "v" y la cantidad de signos "+" y "–", primero para "u" y luego para "v". Esto se hace revisando la cadena en proceso caracter por caracter. Los signos se van guardando en la variable signu[cansigu]. Los otros términos se guardan en la variable teru[genu][j]. La variable genu se incrementa cada que se detecta un signo, y j se incrementa con cada caracter que no es signo. Se tienen variables similares para la parte "v".

7.- Calcula la cantidad total de signos.

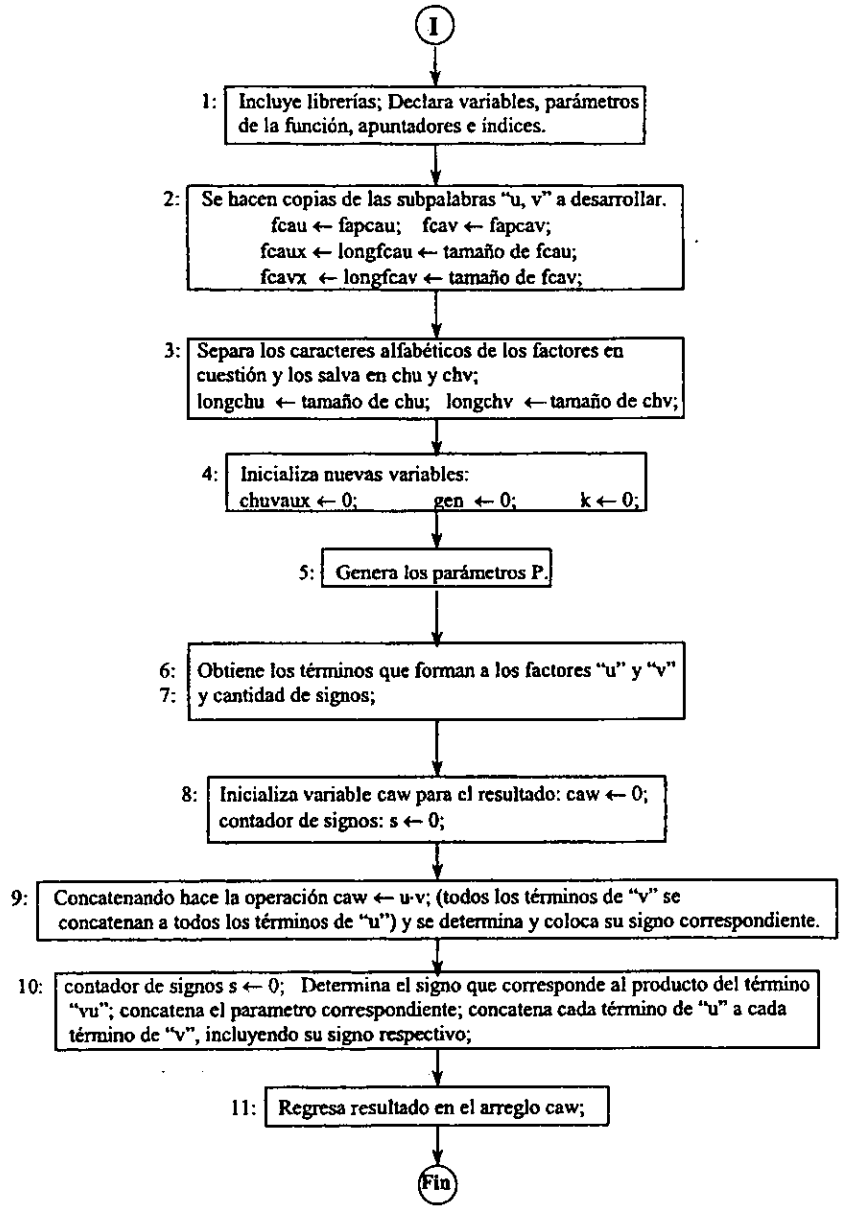
8.- Inicializa la variable que contendrá el resultado de esta función caw[0] = 0 y un contador de signos s = 0. Primero se conformará la parte uv del desarrollo cuántico.

9.- Realiza las operaciones "u·v". Toma un término de "u" y lo concatena a caw, en seguida, toma un término de "v" y lo concatena a caw, también se determina el signo del siguiente producto y es colocado en la cadena caw. Sigue tomando términos de "v" hasta llegar al valor indicado por genv. Toma el siguiente término de "u" y repite el proceso anterior hasta haber incluido todos los términos de "u" y de "v".

10.- Inicializa contador de signo s = 0; determina el signo que corresponde al producto del término "v" por el término "u", lo concatena a la cadena caw y después concatena a caw primero los términos de los parámetros, después concatena los términos "v" y por último los términos "u". Contador de signos s = s + 1. Éste paso se repite hasta concatenar a caw todos los signos y términos en cuestión.

11.- Fin de la función oplie5B. Regresa el resultado en el arreglo caw.

Figura 6.5.2. Diagrama de Flujo de la Función oplie5B.

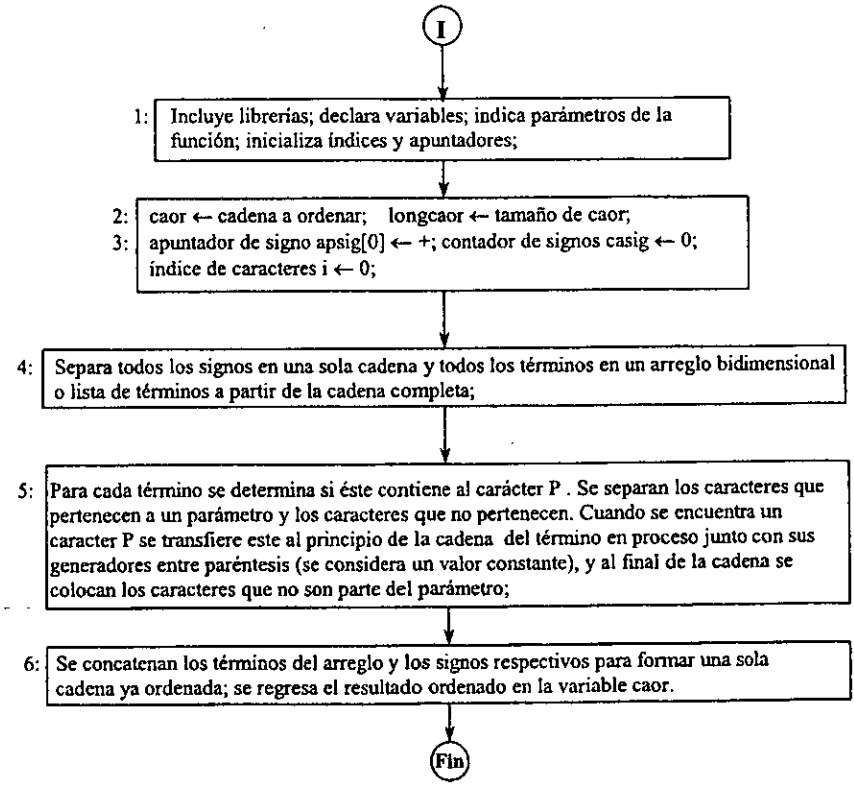


Algoritmo de la función orden5B (figura 6.5.3).

Esta función ordena la cadena resultante del desarrollo cuántico. Cuando los hay, primero los parámetros obtenidos y después los elementos (o variables cuánticas) que no son parte de un parámetro.

Nombre: ordena5B.
Entradas: la cadena resultante del desarrollo cuántico sin ordenar.
Salida: la cadena ya ordenada con los parámetros en primera instancia.

Figura 6.5.3. Diagrama de Flujo de la Función orden5B.



6.5.5 Ejemplos de desarrollos cuánticos.

Ejemplo 6.5.1.

Programa DESCUA (DESarrollo CUAntico) descua5B.exe
Programa para desarrollar expresiones dentro de corchetes
Cadena para demostracion: [d,[[d,c],[d,[b,a]]]]
Desea utilizar la expresion de prueba? si = s, no = n; n
Introduzca la expresion completa con corchetes y comas: [[[z,y],z],z]
La expresion sin corchetes es: zyzz
La palabra zyzz no es estandar.

Ejemplo 6.5.2.

Cadena para demostracion: [d,[[d,c],[d,[b,a]]]]
Desea utilizar la expresion de demostracion? si = s, no = n; n
Introduzca la expresion completa con corchetes y comas: [z,y]
La expresion sin corchetes es: zy
Existen 2 generadores
Cadena a procesar: [z,y] Presione una tecla para continuar
CADENA OBTENIDA:
zy-P(zy)yz
FIN DE PROGRAMA, presione una tecla para salir

Ejemplo 6.5.3.

Introduzca la expresion completa con corchetes y comas: [[z,y],[[z,y],y]]
Existen 2 generadores
Cadena a procesar: [[z,y],[[z,y],y]] Presione una tecla para continuar
CADENA OBTENIDA:
zyzyy-P(zy)zyyzy-P(zy)P(yy)zyyzy+P(zy)P(yy)P(zy)zyyyz-P(zy)yzzyy+P(zy)P(zy)yzyzy
+P(zy)P(zy)P(yy)yzyzy-P(zy)P(zy)P(yy)P(zy)yzyyz-P(zz)P(zy)P(zy)P(yz)P(yy)P(yy)zy
yzy+P(zz)P(zy)P(zy)P(yz)P(yy)P(yy)P(zy)zyyyz+P(zz)P(zy)P(zy)P(yz)P(yy)P(yy)P(zy)
yzyzy-P(zz)P(zy)P(zy)P(yz)P(yy)P(yy)P(zy)P(zy)yzyyz+P(zz)P(zy)P(zy)P(yz)P(yy)P(y
y)P(zy)P(yy)yzyzy-P(zz)P(zy)P(zy)P(yz)P(yy)P(yy)P(zy)P(yy)P(zy)yzyyz-P(zz)P(zy)P
(zy)P(yz)P(yy)P(yy)P(zy)P(yy)P(zy)yyzzy+P(zz)P(zy)P(zy)P(yz)P(yy)P(yy)P(zy)P(yy)
P(zy)P(zy)yyzyz

Ejemplo 6.5.4.

Cadena para demostracion: [d,[[d,c],[d,[b,a]]]]
Desea utilizar la expresion de demostracion? si = s, no = n; s
La expresion sin corchetes es: ddcdba
Existen 4 generadores
Cadena a procesar: [d,[[d,c],[d,[b,a]]]] Presione una tecla para continuar
CADENA OBTENIDA:
ddcdba-P(ba)ddcdab-P(db)P(da)ddcbad+P(db)P(da)P(ba)ddcabd-P(dc)dcddba+P(dc)P(ba)
dcddab+P(dc)P(db)P(da)dcdbad-P(dc)P(db)P(da)P(ba)dcdabd-P(dd)P(db)P(da)P(cd)P(cb
)P(ca)ddbadc+P(dd)P(db)P(da)P(cd)P(cb)P(ca)P(dc)ddbacd+P(dd)P(db)P(da)P(cd)P(cb)
P(ca)P(ba)ddabdc-P(dd)P(db)P(da)P(cd)P(cb)P(ca)P(ba)P(dc)ddabcd+P(dd)P(db)P(da)P
(cd)P(cb)P(ca)P(db)P(da)dbaddc-P(dd)P(db)P(da)P(cd)P(cb)P(ca)P(db)P(da)P(dc)dbad
cd-P(dd)P(db)P(da)P(cd)P(cb)P(ca)P(db)P(da)P(ba)dabddc+P(dd)P(db)P(da)P(cd)P(cb)
P(ca)P(db)P(da)P(ba)P(dc)dabdcd-P(dd)P(dc)P(dd)P(db)P(da)dcdbad+P(dd)P(dc)P(dd)P
(db)P(da)P(ba)dcdabd+P(dd)P(dc)P(dd)P(db)P(da)P(db)P(da)dcbadd-P(dd)P(dc)P(dd)P(
db)P(da)P(db)P(da)P(ba)dcabdd+P(dd)P(dc)P(dd)P(db)P(da)P(dc)cddbad-P(dd)P(dc)P(d
d)P(db)P(da)P(dc)P(ba)cddabd-P(dd)P(dc)P(dd)P(db)P(da)P(dc)P(db)P(da)cdbadd+P(dd
)P(dc)P(dd)P(db)P(da)P(dc)P(db)P(da)P(ba)cdabdd+P(dd)P(dc)P(dd)P(db)P(da)P(dd)P(
db)P(da)P(cd)P(cb)P(ca)dbadcd-P(dd)P(dc)P(dd)P(db)P(da)P(dd)P(db)P(da)P(cd)P(cb)
P(ca)P(dc)dbacdd-P(dd)P(dc)P(dd)P(db)P(da)P(dd)P(db)P(da)P(cd)P(cb)P(ca)P(ba)dab
dcd+P(dd)P(dc)P(dd)P(db)P(da)P(dd)P(db)P(da)P(cd)P(cb)P(ca)P(ba)P(dc)dabcdd-P(dd
)P(dc)P(dd)P(db)P(da)P(dd)P(db)P(da)P(cd)P(cb)P(ca)P(db)P(da)baddcd+P(dd)P(dc)P(
dd)P(db)P(da)P(dd)P(db)P(da)P(cd)P(cb)P(ca)P(db)P(da)P(dc)badcdd+P(dd)P(dc)P(dd)
P(db)P(da)P(dd)P(db)P(da)P(cd)P(cb)P(ca)P(db)P(da)P(ba)abddcd-P(dd)P(dc)P(dd)P(d
b)P(da)P(dd)P(db)P(da)P(cd)P(cb)P(ca)P(db)P(da)P(ba)P(dc)abdcdd

◆

Conclusiones

Mediante la utilización de las herramientas computacionales actuales es posible desarrollar programas y algoritmos para aplicaciones en diversas partes de la ciencia. Esta obra ejemplifica tal aseveración en el campo de las álgebras de Lie cuánticas. Como se puede observar, en este trabajo se ha tratado de exponer, en forma estructurada, los conocimientos y las bases teóricas importantes que se relacionan con la problemática de los algoritmos en la teoría de las álgebras de Lie cuánticas. Se consideran tanto problemas y teoremas algebraicos fundamentales, así como problemas algorítmicos y desarrollos de programas específicos para tratar algunos teoremas y proposiciones de las matemáticas. De manera especial se realizaron planteamientos relacionados con las combinaciones de las palabras, las cuales intervienen en la cuantificación de las álgebras de Lie.

Siendo campos tan amplios el de los grupos cuánticos y el de las álgebras de Lie, solo se han dado los conceptos introductorios que consideré como los más importantes. Como resultado importante se presenta la cuantificación del álgebra de tipo $G2$ en cuyos procedimientos se pueden emplear los programas construidos en esta misma tesis. Se expone el método de V.K. Kharchenko de una solución combinatoria para la cuantificación de las álgebras de Lie, por medio del concepto de la operación de Lie cuántica y considerando el espacio comprendido por los elementos primitivos torcidos provistos con las operaciones cuánticas como una analogía cuántica de un álgebra de Lie. Las cuantificaciones definidas dependen, esencialmente, de la representación combinatoria particular del álgebra de Lie.

Cuando se trabaja con un álgebra definida por generadores y relaciones definidas, se está trabajando con el lenguaje de las palabras (elementos de un álgebra libre). Cabe resaltar que mediante el ordenamiento de Hall, el conjunto de todas las palabras se puede tener totalmente ordenado gracias a lo cual, se pueden realizar definiciones inductivas y comprobaciones, y también se pueden desarrollar algoritmos de manera muy versátil. Como una aplicación inmediata en el capítulo seis se construyeron algunos programas que son de utilidad para el manejo de combinaciones de palabras, por ejemplo, con el programa "eaenacb" se pueden obtener las superletras "[u]" a partir de las palabras estándares "u"; mediante el programa "estancb" se determina si una palabra dada es estándar o no lo es; mediante el programa "lyncb" se logra representar una palabra de acuerdo al teorema de Lyndon; y como una introducción más hacia el manejo de axiomas de grupos cuánticos se presenta el programa "descua5B" para el desarrollo en corchetes cuánticos de una palabra estándar y aplicando el conmutador (de Lie)

cuántico. Estos programas se siguen de los axiomas matemáticos tratados en los capítulos previos.

Observe como los teoremas matemáticos ya establecidos ayudan a la construcción de algoritmos, y estos a su vez ayudan a demostrar y a comprender mejor a los teoremas. También, los algoritmos desarrollados con una base sólida y con el empleo de la computadora, pueden ser empleados para hacer otros desarrollos o ejercicios de una manera mucho más rápida de como se realizaría en forma manual. Por ejemplo se puede verificar rápidamente la condición de Specht-Weber con el programa aquí construido de desarrollo de corchetes cuánticos, o dadas algunas expresiones con corchetes del teorema de Shirshov de palabras estándares no asociativas, se determina más fácilmente cual es más larga o más corta, etc.

En cuanto a los problemas que se presentan, se observa que actualmente existen pocos algoritmos para computadora ya desarrollados en el campo de las álgebras de Lie cuánticas para poder efectuar procesos con combinaciones de palabras, sin embargo, estudiando los teoremas matemáticos que se relacionados entre sí, podemos hacer desarrollos importantes.

Por lo que respecta a la problemática de la construcción de los programas específicos, se observa, por un lado, que en su desarrollo se deben tener presentes los diferentes teoremas matemáticos relacionados entre sí, y por el lado computacional, se deben tener en cuenta los factores como son: la cantidad de memoria total que será ocupada una vez hecho el desarrollo, la memoria en bytes que será ocupada por cada variable que interviene en el proceso, y la relación precisa entre las mismas variables. Un problema importante es la representación simbólica la cual será propuesta por el programador y sin perder de vista el objeto que representa. También, resulta difícil hacer una programación plenamente estructurada debido a la estrecha relación de los parámetros y variables que se manejan.

Vale la pena mencionar que, parte de este trabajo se ha presentado en el taller de estructuras cuánticas de la FES-Cuautitlán, y también se ha derivado una publicación en el idioma español (Álgebras envolventes universales cuánticas del álgebra de Lie nilpotente del tipo G_2, V. K. Kharchenko - V. González), ya que como se ha observado, prácticamente la mayor parte de las referencias de los temas tratados se encuentra en el idioma inglés.

♦

Apéndice A

Algunos Conceptos Básicos de Álgebra.

En este apéndice se presentan algunos conceptos básicos de álgebra los cuales son de importancia central en el desarrollo de algoritmos y sistemas para computación matemática simbólica.

Grupos, anillos, ideales y campos

Grupo. Un grupo G es:

(a) un conjunto no vacío $g_1, g_2, \ldots, g_n \in G$ junto con
(b) una operación, llamada multiplicación de grupo ($\circ$) tal que

A1 $g_i \in G, g_j \in G \Rightarrow g_i \circ g_j \in G$ cerradura.

A2 $g_i \circ (g_j \circ g_k) = (g_i \circ g_j) \circ g_k$ asociatividad.

A3 $g_1 \circ g_i = g_i = g_i \circ g_1$ para toda g_i existencia de identidad.

A4 $g_k \circ g_l = g_l \circ g_k = g_1$ inversa única $g_l \circ = g_k^{-1}$.

El grupo G se dice ser un *semigrupo* si satisface solamente las primeras dos condiciones, o sea, posee una operación producto asociativa, pero no tiene un elemento identidad.

Un grupo se llama *abeliano* (o conmutativo) si la operación producto conmuta:

A5 $g_i \circ g_j = g_j \circ g_i$ para toda $g_i, g_j \in G$ conmutatividad.

Por ejemplo el conjunto de los números reales, excluyendo 0, forma un grupo bajo la operación de multiplicación. La operación de identidad es 1. Pero con la operación de adición, se forma un grupo con el elemento identidad 0.

Otro ejemplo, el conjunto de matrices no singulares reales $n \times n$ forman un grupo llamado $Gl(n, r)$. El subconjunto de estas matrices con determinante +1 forma un (sub)grupo llamado $Sl(n, r)$. La colección de matrices unitarias $n \times n$ $U(n)$ también forma un grupo bajo la multiplicación de matrices.

Anillo. Un *anillo* (R; +, $\circ$) es un conjunto no vacío R cerrado bajo dos operaciones binarias "+" y "$\circ$" tal que (R; +) es un grupo abeliano (se mantienen los axiomas de grupo A2 – A5 con respecto a la +), "$\circ$" es asociativa y tiene una identidad (se conservan los axiomas A2 y A3 con respecto a "$\circ$"), y el cual satisface el axioma: ($\forall\, a, b, c \in$ R)

A6 $a \circ (b + c) = (a \circ b) + (a \circ c)$, y
$(a + b) \circ c = (a \circ c) + (b \circ c)$ distributividad.

Un *anillo conmutativo* es un anillo en el cual "$\circ$" es conmutativa (se mantiene el axioma A5 de grupos con respecto a "$\circ$"). Un *dominio integral* es un anillo conmutativo el cual satisface el axioma adicional:

A7 $a \circ b = a \circ c$ y $a \neq 0 \Rightarrow b = c$
para toda $a, b, c \in$ R ley de cancelación.

Se observa que para anillos normalmente denotamos el elemento identidad con respecto a + por 0, el elemento identidad con respecto a $\circ$ por 1, y el inverso de a con respecto a + por $-a$.

Ideal

Un subconjunto $I \subseteq R$ es un *ideal* si satisface las siguientes dos condiciones:

1. I es un subgrupo aditivo del grupo aditivo de R:

$$(\forall\, a, b \in I)[a - b \in I]\,.$$

2. $RI \subseteq I$; I es cerrado bajo la multiplicación con elementos de anillo:

$$(\forall\, a, \in R)(\forall\, b \in I)[ab \in I]\,.$$

Campo. Un campo F es

(a) un conjunto de elementos $f_0, f_1, f_2, \ldots,$
junto con dos operaciones:

+ llamada adición
$\circ$ llamada multiplicación escalar

tal que los siguientes postulados se mantienen:

Postulado A. F es un grupo abeliano bajo "+" (se cumplen los axiomas A2 – A5 respecto a "+"), con f_0 como la identidad.

Postulado B

B1. $f_i \circ f_j \in F$ cerradura.

B2. $f_i \circ (f_j \circ f_k) = (f_i \circ f_j) \circ f_k$ asociatividad.

B3. $f_i \circ 1 = 1 \circ f_i = f_i$ identidad.

B4. $f_i \circ f_i^{-1} = 1 = f_i^{-1} \circ f_i, f_i \neq f_0$ inverso, excepto para f_0.

B5. $f_i \circ (f_j + f_k) = f_i \circ f_j + f_i \circ f_k$

$(f_i + f_j) \circ f_k = f_i \circ f_k + f_j \circ f_k$ ley distributiva.

decimos que el campo es conmutativo si se obedece el siguiente postulado B-6

B6. $f_i \circ f_j = f_j \circ f_i$ conmutatividad.

Un campo es un anillo conmutativo en el cual cada elemento no cero tiene un inverso multiplicativo.

El conjunto de los enteros (positivos, negativos, y cero) forma un dominio integral y se denota por **Z**. Los ejemplos más familiares de campos son los números racionales **Q**, los números reales **R**, y los números complejos **C**.

♦

Apéndice B

Glosario Corto de Nociones Seleccionadas de la Teoría de Grupos Clásicos

En este glosario se recuerdan algunas definiciones de la teoría de los grupos y las álgebras de Lie.

Módulo A. Un grupo abeliano M con la ley de composición escrita aditivamente $\gamma = \alpha + \beta$, $\alpha, \beta, \gamma \in M$ se llama *módulo A izquierdo* con respecto a un álgebra A si una multiplicación de M por elementos de A desde la izquierda esta definido, con las propiedades

$$\begin{aligned} &a(\alpha+\beta) = a\alpha + a\beta, \\ &(a+b)\alpha = a\alpha + b\alpha, \\ &(ab)\alpha = a(b\alpha), \qquad a, b \in A; \quad \alpha, \beta \in M. \end{aligned}$$

Módulo A derecho y *bimódulo A* se definen muy análogamente.

Observe que usualmente definimos un módulo con respecto a un anillo el cual no es necesariamente un álgebra.

Ideal y álgebra cociente

Una subálgebra $I \subset A$ de un álgebra A se llama el *ideal izquierdo (derecho, de los dos lados)* en A si para cualquier $u \in I$ y cualquier $a \in A$, $au \in I$ (correspondientemente, $ua \in I$, ambas $au \in I$, $ua \in I$).

Cualquier ideal de los dos lados $i \subset A$ define la descomposición del álgebra A en clases de equivalencia (*coconjuntos*) con la relación de equivalencia:

$$a \sim b, \quad \text{si y solo si } a - b \in I; \qquad a, b \in A.$$

En el conjunto de los coconjuntos uno puede definir la multiplicación natural

$$(a + I)(b + I) = (ab + I),$$

o sea el producto de los coconjuntos de los elementos a y b es igual al coconjunto de ab.

Esta álgebra de coconjuntos se llama *álgebra cociente.*

Isomorfismo, automorfismo, homomorfismo, endomorfismo

Sean X y X' dos conjuntos con algunas relaciones entre los elementos de cada conjunto.

Por ejemplo, X y X' pueden ser grupos (de Lie), y las relaciones correspondientes pueden ser multiplicaciones de grupo: $ab = c$ para $a, b, c \in X$ y $a'b' = c'$ para $a', b', c' \in X'$. Otro ejemplo es los conjuntos ordenados con desigualdades definidas $a > b$, $a, b \in X$ y $a' > b'$, $a', b' \in X'$.

Sea la existencia de una transformación uno a uno $\rho: X \longrightarrow X'$ preservando las relaciones entre los elementos de X, X', o sea si alguna relación es completada para $a, b \in X$ entonces la relación correspondiente es completada para $\rho(a)$, $\rho(b) \in X'$ y viceversa. En este caso los conjuntos X y X' se llaman *isomórficos*: $X \cong X'$, y la transformación ρ se llama isomorfismo.

En particular, si los conjuntos coinciden $X = X'$, una transformación ρ uno a uno, preservando algunas relaciones, se llama *automorfismo.*

Si cada elemento $a \in X$ se transforma en una imagen única, un solo elemento $a' \in X'$, pero invertido en general no es verdadero (por ejemplo a' puede ser la imagen de varios elementos de X o no ser la imagen de cualesquiera elementos de X) y la transformación conserva las relaciones de estructura en X y X', entonces la transformación se llama *homomorfismo*.

La transformación homomórfica de un conjunto en si mismo se llama *endomorfismo*.

Representaciones: exacta, irreducible, reducible, completamente reducible, adjunta

Una representación de un álgebra A (grupo G) es un homomorfismo de A (o G) en un álgebra (grupo) de transformaciones lineales de algún espacio vectorial V.

Una representación es término *exacta* si el homomorfismo es un isomorfismo.

Un subespacio $V_1 \subset V$ de una espacio V de representación se llama *subespacio invariante* con respecto a un álgebra A (grupo G) si $T\upsilon \in V_1$ para toda $\upsilon \in V_1$ y toda $T \in A$ (o $T \in G$).

Una representación se llama *irreducible* si el espacio de representación V no tiene subespacios invariantes (excepto el espacio completo V y el espacio cero $\{0\}$). De otro modo la representación se llama *reducible*.

Una representación se llama *completamente reducible* o *descomponible* si todas las transformaciones lineales de la representación se pueden presentar en la forma de matrices de block diagonal, cada block actuando en el subespacio invariante correspondiente. De otro modo la representación se llama *idescomponible*.

La representación de un grupo de Lie G (álgebra de Lie L) en el espacio vectorial del álgebra de Lie L por si misma se llama *representación adjunta* y las correspondientes transformaciones se denotan por Ad_g, $g \in G$ (ad_X, $X \in L$). En el caso del álgebra de Lie, la representación adjunta se define por el conmutador en L

$$\mathrm{ad}_X Y = [X, Y], \qquad X, Y \in L.$$

Sistemas de raíces de un álgebra de Lie simple, raíces simples y positivas

Sea L un álgebra de Lie semisimple con la subálgebra de Cartan $H \subset L$ y α una función lineal en H. Si el subespacio lineal $L^\alpha \subset L$ definido por la condición

$$L^\alpha := \{Y \in L \mid [X, Y] = \alpha(X)Y \quad \forall X \in H\},$$

existe (o sea $L^\alpha \neq 0$), la función α se llama *la raíz* de L, y L^α se llama *el subespacio de la raíz*. El sistema de raíces no cero se denota por Δ. Realmente todos los subespacios son de dimensión uno, tal que L^α es *vector raíz*.

La subálgebra de Cartan y los vectores raíces dan la muy conveniente base para una arbitraria álgebra de Lie semisimple L y proporciona su clasificación. En particular, la *base de Cartan-Weyl* de un álgebra de Lie semisimple L consiste de una base $\{H_i\}$ de la subálgebra de Cartan H y vectores raíces $E_\alpha \in L^\alpha$. En esta base, para cualquier $\alpha, \beta \in \Delta$ las RC (relaciones de conmutación) definidas tiene la forma

$$[H_i, E_\alpha] = \alpha(H_i)E_\alpha, \qquad H_i \in H,$$

$$[E_\alpha, E_\beta] = \begin{cases} 0 & \text{si } \alpha+\beta \neq 0 \quad \text{y} \quad \alpha+\beta \notin \Delta, \\ H\alpha & \text{si } \alpha+\beta = 0, \\ N_{\alpha,\beta} E_{\alpha+\beta} & \text{si } \alpha+\beta \in \Delta, \end{cases}$$

donde las constantes $N_{\alpha,\beta}$ satisfacen la identidad $N_{\alpha,\beta} = -N_{-\beta,-\alpha}$.

Así el problema de la clasificación se reduce al estudio de posibles conjuntos de las constantes $N_{\alpha,\beta}$.

Una raíz α se llama *positiva* si la primera coordenada de la correspondiente H_α es positiva. El subsistema de raíces positivas se denota por Δ_+.

Una raíz positiva se llama *simple* si no se puede expresar como suma de otras dos raíces positivas.

Existe una correspondencia uno a uno entre las raíces $\alpha \in \Delta$ y elementos $H_\alpha \in H$ de la subálgebra de Cartan definida por la igualdad

$$\langle X, H_\alpha \rangle = \alpha \langle X \rangle, \qquad \forall X \in H.$$

Aquí $\langle \cdot, \cdot \rangle$ denota la *forma de Killing* (producto escalar) sobre L

$$\langle X, Y \rangle = \mathrm{Tr}(\mathrm{ad}_X \mathrm{ad}_Y), \qquad \forall X, Y \in L.$$

Generalmente, el producto escalar $\langle H_\alpha, H_\beta \rangle$ se denota simplemente como (α, β).

Álgebras de Lie solubles, nilpotentes, semisimples y simples; subálgebras de Cartan y Borel

Un álgebra de Lie L se llama *soluble* si la definición recursiva

$$L^{(0)} := L, \; L^{(1)} := [L^{(0)}, L^{(0)}], \ldots, L^{(n+1)} := [L^{(n)}, L^{(n)}], \ldots \qquad n = 0, 1, 2, \ldots$$

produce la subálgebra cero después de un número finito de pasos, o sea $L^{(n)} = 0$ para algún $n < \infty$.

Un álgebra de Lie se llama *nilpotente* si la definición recursiva

$$L_{(0)} := L, \; L_{(1)} := [L_{(0)}, L], \ldots, L_{(n+1)} := [L_{(n)}, L_{(n)}], \ldots \qquad n = 0, 1, 2, \ldots$$

produce la subálgebra cero después de un número finito de pasos, o sea $L_{(n)} = 0$ para algún $n < \infty$.

Un álgebra de Lie L es *semisimple* si no tiene ideales abelianos no cero.

Un álgebra de Lie L es *simple* si no tiene ideales (excepto el ideal cero y la misma L).

La subálgebra de Cartan H de un álgebra de Lie semisimple L es la máxima subálgebra en L con la representación adjunta completamente reducible.

La subálgebra Boreal $\mathcal{B}$ de un álgebra de Lie simple L es la máxima subálgebra soluble en L.

Operadores tensoriales. Sea $g \to M(g)$ una representación de matriz de un grupo G en un espacio vectorial de dimensión finita V y $g \to Ug$ es una representación unitaria de G en un espacio de Hilbert $\mathcal{H}$. Un conjunto $\{T^\alpha\}_{\alpha=1}^{\dim V}$ de operadores en $\mathcal{H}$ se llama *operador tensorial* si

$$U_g^{-1} T^\alpha U_g = M_b^\alpha(g) T^b .$$

Álgebra envolvente universal

Un álgebra envolvente universal de Lie L es un álgebra cociente

$$\mathcal{U}(L) := A_L/J_{[\cdot,\cdot]},$$

donde A_L es un álgebra (libre) asociativa generada por toda X_i L ($i = 1, \ldots,$ dim L), $J_{[\cdot,\cdot]}$ es el ideal de dos lados generado por los elementos de la forma $XY - YX - [X, Y]$, $\forall X, Y \in L$.

Objetos Básicos y Construcciones

Un álgebra A es un espacio vectorial con una ley de multiplicación que satisface las condiciones de linealidad y distributividad.

Por ejemplo

$(\alpha a + \beta b)c = \alpha ac + \beta bc$ para $\alpha, \beta \in K; a, b, c \in A$.

También recordemos que un mapeo o transformación $d : A \to A$ se llama *derivación* sobre un álgebra A (no necesariamente asociativa) si se satisfacen las siguientes condiciones:

$d(xy) = (dx)y + x(dy)$.

◆

Apéndice C

Listados de los Programas

C.1 Listado del programa para determinar palabras estándares "estancb".

```
// DETERMINA SI UNA PALABRA ES ESTANDAR O NO.
// Nombre: estancb.exe
// Entradas: una palabra.
// Salidas: si la palabra es estandar x = 1, si es palabra no estandar x = 0.
#pragma hdrstop
#include <condefs.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <iostream.h>
//---------------------------------------------------------------------------
#pragma argsused

int main(int argc, char **argv)
{
    char o[100] = "Programa para determinar palabras estandares estancb.exe";
    char tempv[1000], tempw[1000]="", apv[1000]="", apw[1000]="",apwv[1000]="", u[1000];
    int i, j, k, x;
    cout << endl << o << endl << endl;          // Despliega en la pantalla el mensaje contenido en la variable "o".
    cout << "Introduce la palabra a verificar considerando z mayor que a:"<<endl;
    cout << "Palabra u = vw = ";
    cin >> u;                                                    // Se guarda en u la palabra ingresada.
    cout << endl;
    k = strlen(u);                                               // Se obtiene la longitud de la palabra.
    cout << "Longitud de la palabra = " << k << endl;
    cout << "Representaciones wv posibles:" << endl;
    for(j=1;j<k;j++)                // j: apuntador desde la primera hasta la penultima letra de la palabra original.
      {
        for(i=0;i<=k;i++)                                        // Se hacen copias de la palabra original.
          {
            tempv[i]=u[i];
            tempw[i]=u[i];
          }
        tempv[j]=0;                                          // Obtencion de una parte de la representacion v.
        for(i=0;i<k;i++)
          { apv[i]=tempv[i]; }
        cout << endl << "v = " << apv << " ";                          // Se muestra la parte v en pantalla.
        for(i=0;i<k;i++)                                  // Obtencion de la otra parte de la representacion w.
          {
            apw[i]=tempw[j+i];
            apwv[i]=tempw[j+i];
          }
        cout << "   w = " << apw << " ";                          // Se muestra la parte w en pantalla.
        strcat(apwv,apv);                                                // Concatenacion de w y v.
        cout << "    wv = " << apwv ;                          // Se muestra la combinacion wv en pantalla.
        x=0;
```

```
        for(i=0;i<k;i++)                        // Se compara letra a letra la palabra u con la combinacion wv.
           {
            if(u[i]>apwv[i])
              {                  // Como la letra (i) de la palabra u es mayor que la letra (i) de la combinacion wv,
               x=1;     break;                          // entonces indicador de palabra x = 1 (si es estandar).
              }
            else
               if(u[i]==apwv[i])
                 {
                  x = 0;          // En este caso las letras comparadas son iguales, entonces indicador de palabra
                  continue;       // x = 0 (no es estandar) y se procede a comparar mas caracteres.
                 }
                else{
                     x=0;                              // El caracter (i) de u es menor que el caracter (i) de wv
                     break;                            // por lo cual la palabra es no estandar.
                    }
           }
        if(x==1)
           continue;                // Si la palabra es estandar sigue comparando u con mas representaciones wv.
        else
           break;                                      // No se cumplen las condiciones de palabra estandar.
     }                                                                          // Termina bucle for.
   cout << endl<<endl;
   if(x==0)
     {
       cout << u <<" < o = " << apwv << endl << endl;          // Muestra resultado de palabra no estandar.
       cout << u << " es palabra no estandar"<< endl;
     }
   else
      {                                                                  // Muestra resultado de palabra si estandar.
       cout << u <<" > todas las representaciones posibles wv " << endl << endl;
       cout << "por lo tanto " << u << " si es palabra estandar"<< endl;
      }
   cout << endl << "Presione la tecla enter para reiniciar" << endl;
   getch( );
   return 0;
}                                                                          // Fin de programa.
```

C.2 Listado del programa del teorema de Lyndon "LyndonCB".

```
// PROGRAMA PARA PONER UNA PALABRA EN FORMA LYNDON.
// Nombre: LyndonCB.exe
// Entradas: Una palabra cualesquiera.
// Salidas: La palabra entrada representada en forma del teorema de Lyndon.
#pragma hdrstop
#include <condefs.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <conio.h>
#include <stdlib.h>
#include <iostream.h>
#include "estand.h"
//---------------------------------------------------------------------------
USEUNIT("estand.cpp");
```

```
//---------------------------------------------------------------------------
#pragma argsused
int estand (char*);                    // Se requiere de una funcion para determinar si la palabra es estandar o no.
int main(int argc, char **argv)
{
    extern int x;                                          // Variable indicadora de palabra estandar/no estandar.
    extern char u[1000];                                   // Variable para ingresar la palabra a manejar.
    extern int i, j, k, cont;                  // i, j son indices apuntadores a la palabra, k y cont son contadores
                                               // de separaciones de la palabra.
    char uaux[1000];                                       // Variable para ingresar la palabra.
    char *apuaux = uaux;                                   // Apuntador de la palabra completa.
    char m[100] = "Programa de Teorema de Lyndon, se ejecuta con: LyndonCB.exe";
    extern char tempv[1000], tempw[1000], apv[1000], apw[1000], apwv[1000];   // Variables temporales.
    char l[80], Ly[1000] = "";                             // Ly guarda la forma de Lyndon.
    clrscr( );
    cout << m << endl;                                     // Exhibe mensaje de la variable m en la pantalla.
    cout << "Introduzca la palabra a formar en Lyndon: ";
    gets (uaux);
    k = strlen(uaux);                                      // Se obtiene la longitud de la palabra.
    cout << endl << "Inicio" << endl;
    cout << "La palabra entrada es: " << uaux << endl;
    cout << "Longitud de la palabra = " << k << endl;
    while(k>1)
        {
          cont=k;                                          // Longitud de la palabra en la variable cont.
          estand(apuaux);          // Funcion que determina si una palabras es estandar o no. Resultados en apw.
          k=k-j;                                  // j = separador, k = longitud de la palabra, u = palabra actual.
          if(x==0)                                                                  // Si es palabra no estandar.
             {                                                                      // (Palabra no estandar apwv).
             apv[j]='*';
             apv[j+1]='\0';
             strcat(Ly,apv);                                 // Concatenar apv a Ly. La forma de Lyndon es Ly.
             k=strlen(apw);                                  // Longitud de la nueva palabra.
             if(k!=1)                                        // ¿Falta mas de una letra?
                 {
                   for(i=0;i<cont;i++)                       // Forma nueva palabra de longitud k.
                      {u[i]=apw[i];}                         // Reasignacion de la nueva palabra.
                   strcpy(uaux,u);
                 }                                           // La siguiente forma es u.
             else                                            // k es igual a uno (solo falta una letra).
                   strcat(Ly,apw);                           // Agregar w (la ultima letra) al final.
           }                                                 // Fin de condicion verdadera.
          else                                               // La palabra u si es estandar.
             {
              strcat(Ly,apv);                                // Concatena parte v a Ly.
              strcat(Ly,apw);                                // Concatena parte w a Ly.
             }
        }                                                    // Fin de while.
    cout << endl;
    cout << "La forma de Lyndon es: " << Ly ;                // Se exhibe en la pantalla el resultado.
    cout << endl << endl;
    cout << "FIN DE PROGRAMA" << endl;
    cout << endl << "Presione enter para salir..." << endl;
    gets(l);
    return 0;
}                                                            // Fin de programa principal.
```

C.3 Listado del programa del teorema de Shirshov.

```
// ESTE PROGRAMA DESARROLLA PALAbRAS ESTANDARES ASOCIATIVAS A ESTANDARES
// Nombre: eaenacb.exe
// Entradas: una palabra estandar
// Salidas: palabra en forma estandar no asociativa
#pragma hdrstop
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <iostream.h>
//---------------------------------------------------------------------
USEUNIT("estandar.cpp");
//---------------------------------------------------------------------
#pragma argsused
int estand(char*);
int main(int argc, char **argv)
{
    extern int ex;
    char cad[1000], cau[1000], cav[1000], su[1000];
    int longpal;                                          // longpal es longitud de palabra.
    int longcav, longcad;
    int i, j, cp = 0, pend=0, pe=0;
    char apsbp[100][1000];                                // APuntador a SuBPalabra.
    char cadgral[1000]="";
    char apcgrl[100], cadaux[1000];
    char apend[100][1000];
    int contcord[500];
    int contcori=0;
    clrscr( );
    cout << endl;
    cout << "PROGRAMA eaenacb.exe" << endl;
    cout << "PROGRAMA DE TEOREMA DE SHIRSHOV PARA REPRESENTAR PALABRAS" << endl;
    cout << " ESTANDAR NO ASOCIATIVAS" << endl;
    cout << "Este programa representa una palabra estandar asociativa dada" <<endl;
    cout << "en forma estandar no asociativa" << endl;
    cout << "Inicio" << endl;
    cout << "Introduzca la palabra"<<endl;
    cin >> cad;                                           // Entra la cadena.
    longpal = strlen(cad);                                // Determina su longitud.
    cout << "La palabra insertada es: " << cad << endl;
    cout << "Longitud de la palabra = " << longpal << endl <<endl;
    strcpy(cadaux,cad);
    cp=0;                                                 // cp: Contador de Palabras.
    strcpy(cav,cad);
    estand(cad );
    if(ex==0)
        {
          cout << "Error: palabra no estandar. Termina programa" <<endl;
          getch();
          exit(1);
        }
```

```
if(!longpal>2)
    {
     cout << "Error palabra muy corta" << endl;
     getch();
     exit(1);
    }
strcpy(apcgrl,cad);
longcad=strlen(cad);
do
    {
     if (longcad>2)
         {                                                    // Inicia separacion de cadena cad.
          contcori++;
          contcord[pe]++;
          j=1;
          i=0;                                                // Indice para palabras u.
          strcpy(cau,cad);                                    // Copia la variable original a dos subpalabras.
          cau[i+1]=0;                                         // Inicia tomando el primer caracter.
          i++;
          strcpy(apsbp[cp],cau);                              // Apuntador al primer caracter de la palabra.
          for(int i=0;i<longpal;i++)                          // Obtiene segunda parte (subpalabra 2).
              {cav[i]=cad[j+i];}
          longcav=strlen(cav);                                // Determina la longitud de cav.
          strcpy(apsbp[cp+1],cav);                            // Guarda cav en la localidad siguiente a apsbp.
          strcpy(su,cav);
          do                                                  // Verifica segunda parte de la palabra.
             {
             estand(su);                                      // Verifica si u es estandar.
             if(ex == 1)
                 {                  // cav es subpalabra estandar. Se deben agregar parentesis correspondientes.
                 strcpy(apsbp[cp],cau);                       // Apuntador a la primera subpalabra.
                 strcpy(apsbp[cp+1],cav);                     // Apuntador a la segunda subpalabra.
                 strcpy(cad,cav);
                 longcav=strlen(cav);
                 }
             else{                                            // cav es palabra no estandar.
                  strcat(cau,cav);          // Se debe quitar el primer elemento de cav y concatenarlo con cau.
                  cau[i+1]=0;                                 // Procede a tomar siguiente caracter. Separa elemento.
                  strcpy(apsbp[cp],cau);                      // Actualiza apuntador de subpalabras.
                  i++;
                  j=1;
                  strcpy(cad,cav);
                  for(int i=0;i<longcav;i++)
                      {cav[i]=cad[j+i];}                      // Obtiene segunda parte (subpalabra 2).
                  longcav=strlen(cav);
                  strcpy(apsbp[cp+1],cav);                    // Actualiza apuntador de subpalabras.
                  strcpy(su,cav);
                  if(longcav<2)
                    {strcpy(apsbp[cp+1],cav);}                // Resulta un solo caracter el cual es estandar.
                 }
             }while(ex==0 && longcav>=2);                     // Fin de while (se sigue haciendo mientras ex = 0).
                                                              // Se obtuvieron las siguientes subpalabras: [[cau],[cav]].
          strcpy(apend[pend],apsbp[cp+1]);                    // Subpalabra pendiente "<<apsbp[cp+1]. pend inicial=0.
          pe++;
          pend++;
          strcpy(cad,apsbp[cp]);
```

```
        longcad=strlen(cad);
    }
else
    {
     if(longcad==1)                                    // ¿Se tiene una subpalabra de solo un caracter?
        {
         for(int i=0;i<contcori;i++)                   // Se agregan corchetes "[" segun lo indique contcori.
             {strcat(cadgral,"[");}
         contcori=0;
         strcat(cadgral,apsbp[cp]);                    // Se concatena la subpalabra apsbp[cp] a la cadena general.
         for(int i=0;i<contcord[pe];i++)               // Se agregan corchetes "]" segun lo indique contcord.
             {strcat(cadgral,"]");}
         contcord[pe]=0;
         pe--;
         cp++;
         pend--;
         strcpy(apsbp[cp],apend[pend]);                // Obtiene la siguiente subpalabra pendiente para procesarla.
         strcpy(cad,apend[pend]);
         longcad=strlen(cad);                                        // Se determina su longitud.
        }
      else
        {                                                                  // longcad==2.
         strcpy(cad,apsbp[cp]);                                  // Palabra a verificar debe estar en cad.
         estand(cad );                                                     // ¿cad es estandar?
         if(ex==1)
           {
            for(int i=0;i<contcori;i++)                // Se agregan corchetes "[" segun lo indique contcori.
                {
                 strcat(cadgral,"[");
                }
            contcori=0;
            strcat(cadgral,"[");  // Se agrega un corchete "[" adicional para la subpalabra estandar minima.
            strcat(cadgral,apsbp[cp]);          // Se concatena la subpalabra estandar minima a la cadena gral.
            strcat(cadgral,"]");                                   // Se agrega un corchete "]" de cierre.
            for(int i=0;i<contcord[pe];i++)            // Se agregan corchetes "]" segun lo indique contcord.
                {strcat(cadgral,"]");}
            contcord[pe]=0;
            pe--;
            cp++;
            pend--;
            strcpy(apsbp[cp],apend[pend]);  // Obtiene la siguiente subpalabra pendiente para procesarla.
            strcpy(cad,apend[pend]);                                        // Toma segunda parte.
            longcad=strlen(cad);                                 // Obtiene su longitud y se va a separar.
           }
         else
            {
             for(int i=0;i<contcori;i++)               // Se agregan corchetes "[" segun lo indique contcori.
                 {strcat(cadgral,"[");}
             contcori=0;
             strcpy(cadaux,apsbp[cp]);
             apsbp[cp][1]= '0';
             cp++;
             apsbp[cp][0]=cadaux[1];
             cp++;
             strcpy(apsbp[cp],apend[pend]);
             strcpy(cad,apend[pend]);
```

```
                    longcad=strlen(cad);
                  }
              }
          }
    }while (pend>=0);                                          // ¿Aun hay palabras pendientes?
    cout << endl;
    cout << "Subpalabras que forman la expresion estandar no asociativa:";
    cout << endl << endl;
    for(int tot=0;tot<cp;tot++)                                // Lista las subpalabras estandares minimas.
        {
          cout << "subpalabra [" << tot << "] = " << apsbp[tot] << endl;
        }
  cout << "Resultado, palabra estandar no asociativa: " << cadgral << endl;
                                  // Exhibe el resultado en la pantalla la palabra estandar no asociativa obtenida.
  cout<<"FIN DE PROGRAMA "<<endl<<"Presione enter para salir"<<endl;
  getch( );
  return 0;
}
```

C.4 Listado del programa de desarrollo de corchetes cuánticos.

```
// PROGRAMA DE DESARROLLO DE CORCHETES CUANTICOS "descua5B".
// El programa desarrolla palabras estandares con corchetes y comas.
// El resultado se obtiene en forma simbolica con caracteres alfabeticos.
// Nombre: descua5B.exe
// Entradas: una palabra con corchetes (cuanticos) y comas.
// Salidas: la palabra desarrollada en forma cuantica con parametros simbolicos.
#pragma hdrstop
#include <condefs.h>
#include<iostream.h>
#include<conio.h>
#include<ctype.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
//---------------------------------------------------------------------------
USEUNIT("oplie5B.cpp");                                        // Funcion para realizar la operacion de Lie.
USEUNIT("orden5B.cpp");                                        // Funcion para ordenar la cadena obtenida.
USEUNIT("estandar.cpp");                                  // Funcion para determinar si la palabra es o no estandar.
//---------------------------------------------------------------------------
const int cardec = 96;
const int parmax = 3;
int ipar=0;
#pragma argsused
char oplie(int, char*,char*);
char ordena(char*);
int estand(char*);
int main(int argc, char **argv)
{
  extern char caw[10000];
  extern char caor[10000];
  extern int ex;
  char cadvic[] = "[d,[[d,c],[d,[b,a]]]]";
  char cad[10000] = "", cadaux[10000], cades[10000], cadini[200] = "", cadfin[200] = "";
  char* apcad = cadvic, *apcades = cades, *apcin = cadini, *apcfin = cadfin;
```

```
char cora[] = "[", corc[] = "]";                                    // Corchetes abiertos y cerrados.
char cau[10000], cav[10000];
char *apcau = cau, *apcav = cav;                                    // Se hizo esta asignacion para inicializar apuntadores.
long int pdw[100], parametro[100][100], par[100][100];
int maxw;
char* p;
char* apcora,* apcorc;
char* pardef, abc[100] = "a";
int param [100];
long int longcad,tamcad;
int x, y, marc, i, j, k;;
int cca = 0, ccc = 0;                                               // Contadores de corchetes abiertos y cerrados.
char* str = cadvic;
char* tkn = "[], ";
char* ptr = str;
char val,*apval;
clrscr( );
cout << endl;
cout << "Programa DESCUA2 (DESarrollo CUAntico) descua5B.exe" << endl << endl;
cout << "Programa para desarrollar expresiones dentro de corchetes" << endl;
cout << endl << endl;
cout << "Cadena para demostracion: " << cadvic << endl << endl;
cout << "Desea utilizar la expresion de demostracion? si = s, no = n; " << endl;
cin >> val;
if(val =='n'){
  cout << "Introduzca la expresion completa con corchetes y comas: " ;
  cin >> cad;
  }
else strcpy(cad,apcad);                                             // Pone en cad la cadena de prueba.
tamcad = strlen(cad);                                               // Longitud de la cadena completa.
strcpy(cadaux,cad);
strcpy(apcad,cad);
p = strstr(cad,cora);          // Contar numero igual de corchetes abiertos y cerrados. Localiza corchetes abiertos.
while (p)                        // Determina cuantos corchetes abiertos. p = primera posicion de corchete abierto.
  {
  cca++;                                                            // Contador de Corchetes Abiertos.
  p = strstr(++p,cora);                                             // Busca siguiente corchete abierto.
  }
if (cca == 0){                                                      // Cantidad de corchetes abiertos = cca.
  cout << "No existen corchetes abiertos, presione una tecla para salir" << endl;
  getch( );
  exit (1);                                                         // Termina si no hay corchetes abiertos.
  }
p = strstr(cad,corc);                                               // Localiza primer corchete cerrado.
while (p)
  {
  ccc++;
  p = strstr(++p,corc);                                             // Busca siguientes corchetes cerrados.
  }
if (ccc == 0){                                                      // Cantidad de corchetes cerrados = ccc.
  cout << "No existen corchetes cerrados, presione una tecla para salir" << endl;
  getch( );
  exit (1);
  }
if (cca != ccc){
  cout << "La cantidad de corchetes abiertos no es igual a la cantidad de los" << endl;
```

```
cout << "corchetes cerrados" << endl;
cout << "Presione una tecla para salir" << endl;
getch( );
exit (1);
}
ptr = strtok(cad, tkn);                          // Obtiene la expresion sin corchetes (elimina corchetes).
strcpy(apcad,ptr);
while (ptr)
 {                                                                  // primer argumento = caracter 0.
 ptr= strtok(0, tkn);
 if (ptr != 0){
   strcat(apcad,ptr);
   }
 }
cout << "La expresion sin corchetes es: " << apcad << endl;                  // Expresion sin corchetes.
longcad = strlen(apcad);                          // Longitud de la cadena sin corchetes = longcad.
 estand(apcad);                                                   // Verifica si la palabra es estandar.
if(ex!=1){
  cout << "La palabra " << apcad << " no es estandar." << endl;
  cout << "Presione una tecla" << endl;
  getch( );
  exit(1);
  }
int cantgen=0;                                                    // Obtiene la cantidad de generadores.
for(int i=0;i<longcad;i++){
  for(int j=i+1;j<=longcad;j++){
    if(apcad[i] != apcad[j])
      x=1;
    else {x=0;
        break;}
    }
  cantgen=cantgen+x;
  }
cout << endl << "Existen " << cantgen << " generadores" << endl;
strcpy(cad,cadaux);
cout << endl << "Cadena a procesar: " << cad << "  Presione una tecla para continuar" << endl;
getch( );
p = strchr(cadaux,'[');                                           // Busca primer corchete abierto.
while (p)                                                          // Mientras existan corchetes ...
{
 i=0, k=0, j=0;                          // i: apuntador de caracteres de la cadena. j: renglones, k: columnas.
 while (i <= tamcad)                                              // Bucle para encontrar expresiones internas.
  {                                                  // Mientras el indice sea menor a la long. de la cadena.
  k = 0;                                                          // Inicializa apuntador de cadena u, cau.
  if (cadaux[i] != '['){                                          // Inicia busqueda de corchete "[" para inicio.
    i++;
    continue;}              // No es "[", busca otro para iniciar busqueda. despues de apuntar al siguiente caracter.
  else i++;                                                       // Si es "[", ahora busca un caracter.
  for (int cain = 0; cain < i-1; cain++){                         // Genera parte anterior a u.
     cadini[cain] = cadaux[cain];
     }
  cadini[i-1]=0;                                                  // Delimita la cadena anterior a u con nulo.
  if (!('a' <= cadaux[i] && cadaux[i] <= 'z')){          // Busca un caracter alfabetico. Revision de cadena inicial.
    continue;}                                  // Sigue un caracter no alfabetico. Regresa a iniciar nueva busqueda.
  cau[k] = cadaux[i];                                             // Si es caracter, inicio de palabra parte u = cau.
  x=0;
```

```
do                                                          // Busca una coma u otro caracter.
 {                                                          // Debe de salir al encontrar la coma.
 i++; k++;
 cau[k]=cadaux[i];                                          // Puede encontrar otros caracteres, pero
 if (cadaux[i] == ','){                                     // debe encontrar una coma.
   x=1;
   break;}                                                  // Si es coma sale del bucle do y sigue.
 else if ('a' <= cadaux[i] && cadaux[i] <= 'z')
      continue;
    else if (cadaux[i] == '+')
         continue;
       else if(cadaux[i] == '-')
            continue;
          else if(cadaux[i] == 'P' ||cadaux[i] == '(' || cadaux[i] == ')')
               continue;
             else if(cadaux[i] == '*'){
                  continue;}
               else {
                  cout << endl << "no pudo iniciar" << endl;
                  cout << "presione una tecla para terminar" << endl;
                  getch( );
                  break;}                                   // Sale del bucle do.
 }while (i<tamcad);                                         // Fin de do para encontrar cau.
if (x==0)
  continue;                                 // Instruc. de while. Regresa a iniciar nueva busqueda de inicio.
cau[k] = 0;                                 // Delimita palabra u, sustituye la coma por nulo.
strcpy(apcau,cau);                          // Copia la parte u al apuntador del mismo.
x=0;    i++;                                // Va al siguiente caracter a buscar inicio de v.
if(!('a' <= cadaux[i] && cadaux[i] <= 'z'))  // Busca un caracter alfabetico de inicio de v.
 continue;                                  // Si es no caracter regresa a seguir otra busqueda.
k=0;
cav[k] = cadaux[i];                                         // Asigna caracter inicial a cav.
do                                                          // Busca un corchete "]" o mas caracteres de v.
 {
 i++; k++;
 cav[k]=cadaux[i];                                          // Primer caracter de v.
 if (cadaux[i] == ']'){                                     // Busca corchete cerrado.
   x=1;
   break;}                                                  // Termina de encontrar la composicion minima.
 else if('a' <= cadaux[i] && cadaux[i] <= 'z')              // No es "]", busca mas caracteres de v.
      continue;
    else if(cadaux[i] == '+')
         continue;
       else if(cadaux[i] == '-')
            continue;
          else if(cadaux[i] == 'P' ||cadaux[i] == '(' || cadaux[i] == ')')
               continue;
             else if(cadaux[i] == '*')
              continue;
             else { x=0;
                  cout << endl << "no encontro corchete ]" << endl;
                  cout << endl << "Presione una tecla para seguir" << endl;
                  getch( );
                  break;}
 }while (i<tamcad);                                         // Fin del bucle do para cav.
```

```
  if (x==0)
    continue;                                          // Regresa a iniciar nueva busqueda de inicio.
  cav[k] = 0;                                          // Parte v = cav. Delimita palabra v, sustituye "]" por cero.
  strcpy(apcav,cav);                                   // Ahora ya se tiene cau y cav.
  x=0;
  for (int cafi = i+1; cafi <= tamcad; cafi++){        // Genera parte posterior a v.
     cadfin[x] = cadaux[cafi];
     x++;
     }
  cadfin[x]=0;                                         // Asegura delimitar cadfin.
  oplie(ipar, apcau, apcav);          // Desarrolla corchetes regresa resultado en caw. Entradas cau y cav e ipar.
  strcpy(cades,caw);                                   // Copia resultado.
  strcat(cadini,cades);          // Agrega la cadena obtenida a la cadena de inicio (concatena a la parte anterior a u).
  strcat(cadini,cadfin);                               // Une la parte restante.
  strcpy(cad,cadini);                                  // Obtiene nueva cadena en cad.
  tamcad = strlen(cad);                  // Determina la longitud de la nueva cadena completa.
  strcpy(cadaux,cad);
  strcpy(cades,cad);
  ipar++;                                  // Prepara indice de parametros para siguiente vuelta.
  }                                                    // Fin de segundo while (i < = tamcad).
  p = strchr(cadaux,'[');                  // En caso de haber mas coloca en p un parentesis abierto.
 }                                                     // Fin de primer while cuando p = 0.
 ordena(apcades);                                      // Se ordena la cadena.
 cout << endl << "CADENA OBTENIDA: " << endl << endl;
 cout << caor << endl ;
 cout << endl << "FIN DE PROGRAMA,  presione una tecla para salir" << endl;
 getch( );
 return 0;
}
```

Listado de la función oplie5B.

```
// Entradas: indice de parametros, primera (u) y segunda (v) partes de la palabra dentro de un juego de corchetes
// pasadas por referencia.
// Salidas: la operacion cuantica ya desarrollada. Regresa el resultado en la variable caw.
#include<iostream.h>
#include<conio.h>
#include<ctype.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
char caw[10000]="";
char oplie(int contpar, char* fapcau, char* fapcav)
{
  const int cardec = 96; const int parmaxf = 3;
  char fcau[10000], fcav[10000], fcauv[10000], fcavu[10000];
  char* apfcau=fcau,* apfcav=fcav, *apcaw = caw;
  int puv = 12, longfcau, longfcav, fcaux, fcavx, maxu, maxv, max;
  int pdu[100], pdv[100], p[100][100], par[100][100];
  char caci[2000], chu[10000], chv[10000], chuv[200][200], chpar[200][200];
  char *fapchu=chu, *fapchv=chv;
  char chuaux[10000], chvaux[10000], chuvaux[10000];
  char *apuvaux = chuvaux;
  int longchu, longchv, genu, genv;
  char teru[200][200],terv[200][200];
  char signu[200]="", signv[200]="", signo[200];
```

```
char* apsignu=signu, *apsignv=signv;
int cansigu=0, cansigv=0;                                          // Cantidad de signos.
strcpy(fcau,fapcau);                                               // Parte u de la palabra.
longfcau = strlen(fcau);                                           // Determina la longitud de la parte u.
fcaux = longfcau;
strcpy(fcav,fapcav);                                               // Parte v de la palabra.
longfcav = strlen(fcav);                                           // Determina la longitud de la parte v.
fcavx = longfcav;
strcpy(chu,fcau);                                                  // La expresion a desarrollar es: fcau · fcav.
strcpy(chv,fcav);
for(int i=0;i<longfcau;i++){                                       // Revisa la cadena fcau hasta encontrar un signo aritmetico
    if('a'<=chu[i]&&chu[i]<='z')                                   // es decir, toma los primeros caracteres alfabeticos.
        continue;
    else{chu[i]=0;                                                 // En chu quedan los caracteres.
     break;}
 }
longchu=strlen(chu);                                               // Longitud de la cadena u de caracteres.
for(int i=0;i<longfcav;i++){                                       // Revisa la cadena fcav hasta encontrar un signo aritmetico.
    if('a'<=chv[i]&&chv[i]<='z')
        continue;
    else{chv[i]=0;                                                 // Longitud de la cadena v de caracteres.
         break;}
 }
longchv=strlen(chv);                                               // Longitud de los caracteres de v.
strcpy(chuaux,chu);
strcpy(chvaux,chv);
                                  // Obtiene las combinaciones de los parametros p(u,v) por ejemplo: p(abac,cbe)=
                                  // p(a,c)p(a,b)p(a,e)p(b,c)p(b,b)p(b,e)p(a,c)p(a,b)p(a,e)p(c,c)p(c,b)p(c,e)
strcpy(chuvaux, "0");                                              // Inicializa chuvax a ceros.
long int gen = 0;
int k = 0;
for (int i = 0; i < longchu; i++){          // Genera los parametros en base a una combinacion de los los generadores.
   for (int j = 0; j < longchv; j++){
        strcpy(chuv[gen], apuvaux);                 // apuvaux apunta a chuvaux . Inicializa chuv[gen] a ceros.
        chuv[gen][k]='P';                                          // Inicia colocando el caracter P.
        strcat(chuv[gen],"(");                                     // Agrega un parentesis "(".
        fapchu[i+1]=0;                                             // Delimita para tener un solo caracter.
        strcat(chuv[gen], &fapchu[i]);              // Agrega caracter alfabetico de chu (o sea de la parte u).
        // strcat(chuv[gen], ",");
        fapchv[j+1]=0;
        strcat(chuv[gen], &fapchv[j]);              // Agrega caracter alfabetico de chv (o sea de la parte v).
        strcat(chuv[gen], ")");                     // Agrega caracter alfabetico de chu (o sea de la parte u).
        strcpy(chu,chuaux);                                        // Restaura chu.
        strcpy(chv,chvaux);                                        // Restaura chv.
        gen++;                                     // Actualiza para el siguiente parametro a formar.
       }
   }
for (int i = 0; i < longchu*longchv; i++){
   strcat(chpar[contpar],chuv[i]);
  }                                                // Los parametros resultantes quedan en chpar[contpar].
apsignu[0]='+';                  / Determina los terminos que forman u y v y la cantidad de signos "+" y "-".
cansigu=1;
int i=0;
int j=0;
genu=0;
```

```
while(i<longfcau)
 {
  if(fcau[i] == '-' || fcau[i] == '+')          // Detecta si hay un signo en la parte u y lo guarda en la variable signu.
    {signu[cansigu] = fcau[i];
    cansigu++;
    genu++;
    j = 0;}
  else{                                                    // Pasa a formar parte de un termino de u.
       teru[genu][j] = fcau[i];
       j++;
       teru[genu][j]= 0;                          // En teru qudan los terminos correspondientes a las partes u.
    }
  i++;                                                     // Actualiza indice para el siguiente caracter.
 }
apsignv[0]='+';
cansigv=1;
i=0;
j=0;
genv=0;
while(i<longfcav)
 {
  if (fcav[i] == '-' || fcav[i] == '+'){                   // Detecta los signos de la parte v.
    signv[cansigv] = fcav[i];
    cansigv++;
    genv++;
    j = 0;}
  else{
       terv[genv][j] = fcav[i];                  // En terv quedan los terminos de las partes correspondientes a v.
       j++;
       terv[genv][j] = '\0';
    }
  i++;
 }
caw[0]=0;                              // Inicializa la variable caw para la preparacion de resultado final, primero uv.
int s=0;                                                   // Variable de signo.
for(int i = 0; i<= genu ; i++){                            // Formara la parte correspondiente a uv.
   for(int j=0; j<=genv; j++){
       strcat(caw,teru[i]);                                          // Agrega termino u teru[i].
       strcat(caw,terv[j]);                                          // Agrega termino v terv[i].
       if(i==genu && j==genv)                                // Se determina el ultimo factor uv.
          break;                        // Despues del ultimo factor uv el signo que sigue se procesa en forma diferente.
       else{                                                 // Obtiene el signo de acuerdo al signo de los factores u y v.
            if(j<genv){
              if(signu[i]==signv[j+1])
                  signo[s]='+';
              else signo[s]='-';
             }
            else{if(signu[i+1]==signv[0])
                   signo[s]='+';
                 else signo[s]='-';
                }
            strncat(caw, &signo[s],1);                         // Agrega el signo correspondiente.
            s++;
           }
      }
  }
```

```
 s=0;                                                    // Procede a obtener el resultado Pvu.
 for(int j = 0; j<= genv ; j++){                         // Indice de terminos v.
   for (int i=0; i<=genu; i++){                          // Indice de terminos u.
      if(signv[j]==signu[i]){                            // Determina el signo de acuerdo al producto vu.
          signo[s] = '-';
         }
      else signo[s] = '+';
      strncat(caw, &signo[s],1);                         // Agrega el signo correspondiente.
      strcat(caw,chpar[contpar]);                        // Agrega la parte correspondiente a los parametros.
      strcat(caw,terv[j]);                               // Agrega la parte correspondiente a la parte v.
      strcat(caw,teru[i]);                               // Agrega la parte correspondiente a la parte u.
      s++;                                               // Incrementa el indice de signos.
     }
   }
 return (char)caw;                                       // El resultado ya desarrollado es regresado en la variable caw.
}                                                        // Fin de la funcion oplie5B.
```

Listado de la función ordena5B.

// Esta funcion ordena la cadena resultante del desarrollo cuantico. Cuando los hay, primero los parametros obtenidos y despues los elementos que no son parte de un parametro.

```
// Entradas: la cadena resultante del desarrollo cuantico sin ordenar.
// Salida: la cadena ya ordenada con los parametros en primera instancia.
#include<iostream.h>
#include<conio.h>
#include<ctype.h>
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
char caor[10000]="";
char ordena(char *orcad)
{
  char cadcar[100]="", cadpar[500]="", cadpos[500]= "";
  char tertem[200][500], tercad[200][500];
  int gene;
  char sig[200];
  char *apsig=sig;
  int casig=0;
  int longcaor, longter, i, j, x, k;
  strcpy(caor,orcad);                                    // Copia la cadena a ordenar.
  longcaor = strlen(caor);                               // Determina la longitud de la cadena.
  apsig[0]='+';                  // Determina los terminos que forman la cadena y la cantidad de signos "+" y "-".
  casig=0;                                               // Cantidad de signos.
  i=0;
  j=0;
  gene=0;
  while(i<longcaor)                                      // Verifica cada uno de los caracteres de la cadena original.
   {
    if(caor[i] == '-' || caor[i] == '+')                 // Si el caracter es signo se asigna a la variable sig.
      {sig[casig] = caor[i];
      casig++;
      gene++;
      j = 0;
      }
```

```
  else{                                          // Si el caracter no es signo se agrega al termino en curso.
       tercad[gene][j] = caor[i];
       j++;
       tercad[gene][j]= 0;                       // Asegura que el ultimo caracter sea nulo.
      }
  i++;
 }                                               // Fin de while.
for (i=0;i<=gene;i++){                           // Para cada termino.
   int l=0;
   longter = strlen(tercad[i]);                  // Determina la longitud del termino a ser ordenado en tercad[i].
   for(j=0;j<=longter;j++)
      cadcar[j]=0;                               // Inicializa cadcar.
   j=0;
   do                                            // Para cada elemento.
    {
     for(int aux=j ;aux<=longter; aux++)         // Revisa cada caracter de un termino.
         tertem[i][aux] = 0;                     // Inicializa termino tertem[i].
     if(tercad[i][j]=='P'){                      // ¿El caracter en curso es P?
         strcpy(cadpar,"");                      // Si: inicializa la cadena de parametros.
         for(k=0;k<=4;k++){                      // Parte del parametro.
              cadpar[k]=tercad[i][j];            // "(", "caracter", "caracter", ")".
              j++;                               // Incrementa indice de caracteres.
             }
         x=0;
         strcpy(cadpos,"");
         for(int cafi=j; cafi<= longter; cafi++){   // El resto de un termino (despues del parametro)
              cadpos[x]=tercad[i][cafi];           // se forma en la variable cafi.
              x++;
             }
         strcat(tertem[i],cadpar);           // Se agrega la cadena que corresponde al parametro en proceso.
         strcat(tertem[i],cadcar);           // Se agrega la cadena que tiene unicamente caracteres generadores.
         strcat(tertem[i],cadpos);           // Se agrega la parte final del termino en proceso.
         strcpy(tercad[i],tertem[i]);        // Se copia tertem[i] para dejarla libre a tercad[i].
         tercad[i][longter]=0;               // Delimita el termino.
         j=j-l;
         strcpy(cadcar,"");                  // Inicializa para nuevo caracter.
         l=0;
         }                                   // Fin de primera parte de if.
     else{                                   // El termino es no P.
           cadcar[l]=tercad[i][j];           // El caracter en curso se agrega a la cadena que no es de parametro.
           l++;   j++;
           cadcar[l] = 0;                    // Se delimita la cadena que no es de parametro.
         }                                   // Fin de segundo for.
    }while(j<longter);                       // Despues de while se obtiene el termino tercad[i].
  }                                          // Continua procesandose este bucle mientras aun haya terminos. Fin de primer for.
 caor[0]=0;                                  // Inicializa caor para formar la cadena completa resultante.
 for (i=0; i<=gene ;i++){
      strcat(caor, tercad[i]);               // Se agrega el termino tercad[i].
      strncat(caor, &sig[i],1);              // Se agrega el termino signo " i " (&sig[i]).
     }
       return (char)caor;                    // La funcion regresa el resultado de la cadena ordenada en la variable caor.
}
```

Apéndice D

Algoritmia Elemental.

Introducción

Los algoritmos deben de funcionar correctamente en todos los casos del problema que afirman resolver.

Existen distintas formas de seleccionar el algoritmo más eficiente para resolver un problema cuando están disponibles varias técnicas que compiten entre sí. Resulta crucial cómo cambia la eficiencia del algoritmo a medida que los casos del problema se vuelven mayores y por lo tanto (normalmente) más difíciles de resolver. También existe la distinción entre eficiencia media de un algoritmo cuando se utiliza en muchos casos de un problema y su eficiencia en el peor caso posible. La estimación pesimista, del peor caso posible, suele ser adecuada cuando tenemos que estar seguros de resolver un problema en una cantidad limitada de tiempo.

La línea de ataque para analizar algoritmos consiste en *contar el número de operaciones elementales* que efectúa el algoritmo. A medida que aumenta el tamaño de los operandos las operaciones se vuelven más lentas.

Problemas y ejemplares

La mayoría de los problemas tienen una colección infinita de ejemplares (es decir, casos diferentes que se pueden presentar).

Un algoritmo debe de funcionar correctamente en todos los ejemplares o casos del problema que manifiesta resolver. Para mostrar que un algoritmo es incorrecto solamente es necesario encontrar un ejemplar del problema para el cual no sea capaz de encontrar una respuesta correcta. Del mismo modo que es posible demostrar que un teorema no es válido encontrando un único contraejemplo, un algoritmo puede rechazarse tomando como base un único resultado incorrecto. Cuando especificamos un problema, es importante definir su dominio de definición, esto es el conjunto de casos que deben considerarse.

Todo dispositivo de cálculo real tiene un límite que afecta al tamaño de los casos que puede manejar, bien porque las instancias de entrada son demasiado grandes, o porque nos quedamos sin espacio. Distintas máquinas tienen distintos límites.

La eficiencia de los algoritmos

Cuando se tiene que resolver un problema, es posible que estén disponibles varios algoritmos adecuados. Esto plantea la pregunta de cómo decidir entre varios algoritmos cuál es preferible. Si

solamente tenemos que resolver uno o dos casos pequeños de un problema más bien sencillo, quizá no nos importe demasiado qué algoritmo utilizaremos: en este caso podríamos decidirnos a seleccionar sencillamente el que sea más fácil de programar, o uno para el cual ya exista un programa, sin preocuparnos por sus propiedades teóricas. Sin embargo, si tenemos que resolver muchos casos, o si el problema es difícil, quizá tengamos que seleccionar de una manera más cuidadosa.

El enfoque *empírico* (o *a posteriori*) para seleccionar un algoritmo consiste en programar las técnicas competidoras e ir probándolas en distintos casos con ayuda de una computadora. El enfoque *teórico*(o *a priori*) consiste en determinar matemáticamente la cantidad de recursos necesarios para cada uno de los algoritmos *como función del tamaño de los casos considerados*. Los recursos que más nos interesan son el tiempo de computación y el espacio de almacenamiento, siendo el primero normalmente el más importante.

El *tamaño* de un ejemplar se corresponde formalmente con el número de bits que se necesitan para representar el ejemplar en una computadora, utilizando algún esquema de codificación precisamente definido y razonablemente compacto. Sin embargo, para hacer más claros nuestros análisis, lo normal será que seamos menos formales, y utilizaremos la palabra "tamaño" para indicar cualquier entero que mida de alguna forma el número de componentes de un ejemplar. Por ejemplo si hablamos de cadenas de caracteres, mediremos normalmente el tamaño de un ejemplar por la longitud de la cadena de que se trate.

También resulta posible analizar los algoritmos utilizando un enfoque *híbrido*, en el cual la forma de la función que describe la eficiencia del algoritmo se determina teóricamente, y entonces se determinan empíricamente aquellos parámetros numéricos que sean específicos para un cierto programa y para una cierta máquina.

Si deseamos medir la cantidad de espacio que utiliza un algoritmo en función del tamaño de los ejemplares, está a nuestra disposición una unidad natural, a saber, el *bit*. Independientemente de la máquina que se esté utilizando, la noción de un bit de almacenamiento está bien definida. Si, por otra parte, tal como suele suceder, deseamos medir la eficiencia de un algoritmo, en función del tiempo que se necesita para llegar a una respuesta, entonces no existe una opción tan evidente. Está claro que no se puede pensar en expresar esta eficiencia, digamos, en segundos, puesto que no se dispone de una computadora estándar a la cual se pudieran referir todas las medidas.

Una respuesta a este problema está dada por el *principio de invarianza*, que afirma que dos construcciones distintas de un mismo algoritmo no diferirán en su eficiencia en más de alguna constante multiplicativa. Si esta constante fuera, por ejemplo cinco, entonces sabemos que si la primera construcción requiere un segundo para resolver casos de un cierto tamaño, entonces la

segunda construcción (quizá en una máquina distinta, o escrita en un lenguaje de programación distinto) no requerirá más de cinco segundos para resolver los mismos casos.

Diremos que un algoritmo para algún problema requiere un tiempo del orden de $t(n)$ para una función dada t, si existe una constante positiva c y una construcción del algoritmo capaz de resolver todos los casos de tamaño n en un tiempo que no sea superior a $ct(n)$ segundos. El uso de segundos en esta definición es evidentemente arbitrario.

Hay ciertos órdenes que se producen con tanta frecuencia que merece la pena darles un nombre. Por ejemplo, supongamos que el tiempo necesario para que un algoritmo resuelva un caso del tamaño n nunca es más que cn segundos, en donde c es una constante adecuada. Diremos entonces que el algoritmo requiere un tiempo en el orden de n, o más simplemente que requiere un tiempo *lineal*. En este caso también hablamos de un *algoritmo lineal*. Si un algoritmo nunca necesita más de cn^2 segundos para resolver un caso de tamaño n, entonces diremos que requiere un tiempo en el orden de n^2, o bien que requiere un tiempo cuadrático, y le llamaremos *algoritmo cuadrático*. De manera similar un algoritmo es *cúbico, polinómico* o *exponencial* si requiere un tiempo en el orden de n^3, n^k o c^n, respectivamente, en donde k y c son constantes adecuadas.

Operación elemental

Una operación elemental es aquella cuyo tiempo de ejecución se puede acotar superiormente por una constante que solamente dependerá de la construcción particular usada: de la máquina, del lenguaje de programación, etc. De esta manera la constante no depende ni del tamaño, ni de los parámetros del ejemplar que se esté considerando. Sólo el número de operaciones elementales importará en el análisis, y no el tiempo exacto requerido por cada una de ellas, por lo que diremos que las operaciones elementales se pueden ejecutar a coste unitario.

◆

Referencias

1. ANICK, D. 'On the homology of associative algebras'. *Trans. Am. Math.Soc. 296*, No. 2 (1986), 641-659. Zbl. 598.16028.

2. ARNUSH, C. 'Aprendiendo Borland C++ 5 en 21 días'. Prentice Hall Hispanoaméricana, S.A. México, 1997.

3. BAUMSLAG, B. & CHANDLER, B. 'Teoría de Grupos', *Departamento de Matemática New York University*, McGraw-Hill, 1972.

4. BAUTISTA, C. 'A Poinkare-Birkhoff-Witt theorem for generalized Lie color algebras', *Journal of Mathematical Physics* 39 N7(1998) 3829-3843.

5. BEIDAR, K.I., MARTINDALE III, W.S. and MIKHALEV, A.V. 'Rings with Generalized Identities' (*Pure and Applied Mathematics* 196, Marcel Dekker, New York-Basel-Hong Kong, 1996).

6. BELAVIN, A.A. & DRINFEL'D, V.G. (1982) 'Solutions of the classical Yang-Baxter equation for simple Lie algebras', *Funct. Anal. Appl.* **16**, 159-180.

7. BELAVIN, A.A. & DRINFEL'D, V.G. (1984b) 'Classical Yang-Baxter (*sic*) equation for simple Lie algebras', *Funct. Anal. Appl.* **17**, 220-221.

8. BERGMAN, G.M. 'The diamond lemma for ring theory', *Adv in Math. 29* No. 2, (1978) 178-218. Zbl. 326.16019

9. BOKUT', L.A. 'Unsolvability of the word problem and subalgebras of finitely presented Lie algebras', *Izv.Akad.Nauk. Ser. Mat.* 36 N6(1972) 1173-1219.

10. BOKUT', L.A. 'Imbeddings into simple associative algebras', *Algebra and Logic* 15 N2(1976) 117-142.

11. BOKUT', L.A., L'VOV, I.V., KHARCHENKO, V.K. (1988): Non-commutative rings. Itogi Nauki Tekh., Ser. Sovrem. Probl. Mat., Fundam. Napravleniya *18*, 5-116. English transl.: Encycl. Math. Sci. *18*, Berlin-Heidelberg-New York, Springer-Verlag, 1-106, 1991. Zbl.725.16003.

12. BOKUT', L.A. and KLEIN, A.A. 'Serre relations and Groebner-Shirshov bases for simple Lie algebras I, II', *International Journal of Algebra and Computation* 6 N4(1996 389-412).

13. BOKUT', L.A. and KUKIN, G.P. 'Algorithmic and Combinatorial Algebra' (*Mathematics and Its Applications* v. 255, Kluwer Academic Publishers, Dordrecht-Boston-London, 1994).

14. BOKUT', L.A. and MALCOMSON, P. 'Groebner bases for quantum enveloping algebras', *Israel Journal of Mathematics* 96(1996) 97-113.

15. BOONE, W. 'The word problem'. *Ann. Math.*, 1959, v. 70, N2, pp. 16-32.

16. BORCHERDS, R. 'Generalized Kac-Moody algebras', *Journal of Algebra*, 11(1988) 501-512.

17. BRASSARD, G., BRATLEY, P. 'Fundamentos de algoritmia'. Prentice Hall, 1997.

18. BUCHBERGER, B. (1983): Grobner Bases: An algorithmic method in polynomial ideal theory. CAMP Bull. *290*, No. 83. See also: Math. Appl., D. Reidel Publ. Co. *16*, 184-232 (1985). Zbl. 587.13009.

19. CHAICHIAN, M., DEMICHEV, A. 'Introduction to Quantum Groups', World Scientific, 1996.

20. CHARI, V., PRESSLEY, A. 'A Guide to Quantum Groups', Cambridge University Press,1994.

21. CHEN, K.T., FOX, R.H. and LYNDON, R.C. 'Free differential calculus IV, the quotients groups of the lower central series', *Ann. of Math.* 68(1958) 81-95.

22. CLIFT, G. 'Crystal bases and Young tableaux', *Journal of Algebra* 202 N1(1998) 10-35.

23. COHN, P.M. 'Sur le critère de Friederichs pour les commutateur dans une algèbre associative libre', *C. r. Acad. Sci. Paris* 239 N13(1954) 743-745.

24. COHN, P.M. 'Universal Algebra' (Harper and Row, New-York, 1965).

25. COX, D., LITTLE, J., SHEA, D. O'. 'Ideals, Varieties, and Algorithms', *An Introduction to Computational Algebraic Geometry and Conmutative Algebra*, Springer-Verlag, 1991.

26. DAVIS, M., PUTNAM, H. and ROBINSON, J. 'The decision problem for exponential Diophantie equations'. *Ann. Math.*, 1961, 74, p. 425-436.

27. DRINFELD, V.G. 'Hopf algebras and the Yang-Baxter equation', *Soviet Math. Dokl.*, 32(1985) 254-258.

28. DRINFEL'D, V.G. in *Proc. of the International Congress of Mathematicians (Berkley, 1986)* (American Mathematical Society, 1987), p.798.

29. DURDEVICH, M. 'Generalized Braided Quantum Groups'. *1st Journal. Math* 98. 329-348 (1997).

30. Enciclopedia McGraw-Hill de Ciencia y Tecnología. Tercera Edición, 1998.

31. FADDEEV, L. (1980) 'Quantum completely integrable models in field theory', *Soviet Scientific Reviews Sect C* 1, 107-55, Harwood Academic Publishers, Chur, Switzerland.

32. FADDEEV, L. D. (1984) 'Integrable models in (1 + 1)-dimensional quantum field theory', in *Recent Advances in Field Theory and Statistical Mechanics, Les Houches, Session XXXIX, 1982*, J.-B. Zuber & R. Stora (eds), pp. 561-608, North-Holland, Amsterdam.

33. FADDEEV, L., SKLYANIN, E. K. & TAKHTADJAN, L.A. (1979) 'The quantum inverse problem I', *Theoret. Math. Phys.* **40**, 194-220 (Russian).

34. FADDEEV, L. D. & TAKHTADJAN, L. (1979) 'The quantum inverse scattering method of the inverse problem and the Heisenberg XYZ model', *Russian Math. Surveys* **34** (5), 11-68.

35. FADDEEV, L. & TAKHTADJAN, L. (1987) 'Hamiltonian Methods in the Theory of Solitons' Springer, Berlin

36. FADDEEV, L.D., RESHETIKHIN N.YU. and TAKHTADJAN, L.A. 'Algebra i Analiz 1' (1989) 178 (Transl.: *Leningrad Math. J.* 1 (1990) 193). AMANE, 'A Poincarè-Birkhoff-Witt theorem for quantized universal enveloping algebras of type *AN*', *Publ.RIMS. Kyoto Univ.* 25(1989) 503-520.

37. FRIEDRICHS, K.O. 'Mathematical aspects of the quantum theory of fields. V', *Communications in Pure and Applied Mathematics* 6(1953) 1-72.

38. GABBER, O. and KAC, V. 'On defining relations of certain infinite-dimensional Lie algebras', *Bulletin (New series) of the American Mathematical Society* 5, N2(1981) 185-189.

39. GEDDES, K. O., CZAPOR, S. R., LABAHN, G. 'Algorithms for Computer Algebra' Kluwer Academic Publishers, 1992, 581 p.

40. GILMORE, R. (University of South Florida), 'Lie Groups, Lie Algebras, and Some of Their Applications', John Wiley & Sons, 1974.

41. GÖDEL, K. 'Über formal unentscheidbare Sätze der Principia Mathematica und verwander Systeme', *1.- Montash. Math. Phys.*, 1931, 38, S. 173-198.

42. HIGMAN, G. 'Subgroups of finitely presented groups'. *Proc. Roy. Soc.*, 1961, v. 262, N 1311, pp. 455-457.

43. JIMBO, M. 'A q-difference analogue of $U(\mathfrak{g})$ and the Yang-Baxter equation', *Lett. Math. Phis.* 10(1985) 63-69.

44. JONES, V.F.R. (1987) 'Hecke algebra representations of braid groups and link polinomials', *Ann of Math.* **126**, 335-388.

45. KANG, S.-J. 'Quantum deformations of generalized Kac-Moody algebras and their modules', *Journal of Algebra*, 175(1995) 1041-1066.

46. KASHIWARA, M. 'Crystallizing the q-analogue of universal enveloping algebras', *Comm. Math. Phis.* 133(1990) 249-260.

47. KASHIWARA, M. 'On crystal bases of the q-analog of universal enveloping algebras', *Duke Mathematical Journal* 63 N2(1991) 465-516.

48. KASSEL, C. 'Quantum Groups', 1995 Springer-Verlag New York, Inc.

49. KAZHDAN, D. & LUSTZIG, G. (1991) 'Affine Lie algebras and quantum groups', *Duke Math. J. Internat. Math. Res. Notices* **2**, 21-9.

50. KHARCHENKO, V.K. 'An algebra of skew primitive elements', *Algebra and Logic* 37 N2(1998), 181-224, English translation 101-126. QA/0006077.

51. KHARCHENKO, V. K., KELLER, J. & RODRÍGUEZ-ROMO, S. 'Actions of GLq(2,C) on C(1, 3) and its four dimensional representations', *Communications in Algebra*, 27(4), 1843-1878 (1999).

52. KHARCHENKO, V.K. 'A quantum analogue of the Poinkare-Birkhoff-Witt theorem', *Algebra and Logic* 38 N4(1999) 476-507; English translation 259-276.

53. KHARCHENKO, V.K. 'An existence condition for multilinear quantum operations', *Journal of Algebra* 217(1999) 188-228.

54. KHARCHENKO, V.K. 'Character Hopf algebras and quantizations of Lie algebras', *Doklady Mathematics* 60 N3(1999) 328-329.

55. KHARCHENKO, V.K. 'Skew primitive elements in Hopf algebras and related identities', *Journal of Algebra*, 238(2001), 534-559.

56. KHARCHENKO, V.K. 'A combinatorial aproach to quantification of Lie algebras', *MSRI preprint No. 2000-005.*

57. KHARCHENKO, V.K. 'Problemas Fundamentales de Teoria de Algoritmos' (1998). No publicado.

58. KLIMIK, A. & SCHMÜDGEN, K. 'Quantum groups and their representations', Springer, 1997.

59. KNIZHNIK, V. & ZAMOLODCHIKOV, A. (1984) 'Current algebra and Wess-Zumino model in two dimensions', *Nucl. Phys.* **B 247**, 83-103.

60. KNUTH, D. E. 'The Art of Computer Programming' Ed. 2. *Vol.* 1. *Fundamental Algorithms*. Addison-Wesley, 1973.

61. KULISH, P. & SKLYNANIN, E.K. (1982b) 'Quantum spectral transform method'. Recent developments, in *Integrable Quantum Field Theories, Tvarminne, 1981*, J. Hietarinta & C. Montonen (eds), Lecture Notes in Physics 151, pp. 61-119, Springer, Berlin.

62. KUNIVBA, A., MISRA, K.C., OKADO, M., TAKAGI, T. and UCHIYAMA, J. 'Crystals for Demazure modules of classical affine Lie algebras', *Journal of Algebra* 208(1998) 185-215.

63. LALONDE, M. and RAM, A. 'Standard Lyndon bases of Lie algebras and enveloping algebras', *Trans. Amer. Math. Soc.* 347 N5(1995) 1821-1830.

64. LOTHAIRE, M. 'Combinatorics on words', (Encyclopedia of Mathematics and its Applications 17, Addison-Wesley Publ. Co. 1983).

65. LUSZTIG, G. 'Quantum groups at roots of 1', *Geometria Dedicada* 35, N1-3(1990) 89-113.

66. LUSZTIG, G. 'Introduction to Quantum Groups' (*Progress in Mathematics* 10, Birkhauser Boston, 1993).

67. LYNDON, R.C. 'A theorem of Friedrichs', *Michigan Mathematical Journal* 3, N1(1955-1956) 27-29.

68. LYUBASHENKO, V. and SUDBERY, A. 'Generalized Lie algebras of type *An*', *Journal of Mathematical Physics* 39, N6(1998) 3487-3504.

69. MACLANE, S. (1971) 'Categories for the Working Mathematician', *Graduate Texts in Mathematics* 5, Springer, Berlin.

70. MAGNUS, W. 'On the exponential solution of differential equations for a linear operator', *Communications in Pure and Applied Mathematics* 7(1954) 649-673.

71. MANIN, YU.I., 'Quantum groups and non-commutative geometry'. (Center des Researchers Mathématiques, Montréal, 1988).

72. MANIN, YU.I. (1989) 'Multiparameter quantum deformation of the general linear supergroup', *Comm. Math. Phys.* **123**, 163-75.

73. MANTON, N.S. *Nucl. Phys.* **B158** (1979) 141.

74. MARKOV, A.A. 'Algorithm theory'. *Transactions of the Steklov Institute*, v. 42, Moscow, 1954.

75. MATIJASEVIC, JU. V. 'Enumerable sets are Diophantie ones'. DAN SSSR, 1970, 191, N2, c. 279-282.

76. MILNOR, J.W. and MOORE, J.C. 'On the structure of Hopf algebras', *Annals of Math.* 81(1965) 211-264.

77. MISHRA, B. 'Algorithmic Algebra', N.Y. Springer, 1993

78. MONTGOMERY, S. 'Hopf Algebras and Their Actions on Rings' (CBMS 82, AMS, Providence, 1993).

79. MOYAL, J.E. (1949) 'Quantum mechanics as a statistical theory', *Proc. Camb. Phil. Soc.* 45, 99-124.

80. NOVIKOV, P.S. 'The words equality problem for group theory is algorithmically unsolvable'. *Transactions of the Steclov Mathematical Institute*, 44. Moscow, 1955, pp. 1-144.

81. OLVER, P.J. 'Applications of Lie Groups to Differential Equations', Second Edition, Springer-Verlag, 1986, 1993.

82. OZIEWICZ, Z. 'Clifford Hopf gebra for two-dimensional space'. Math.QA/0011263 (2000). Submitted to Miscellanea Algebraica, Akedemia Swietokrzyska, Kielce, Poland.

83. PÉREZ-LÓPEZ, C. 'Matemática Informatizada con MATLAB'. Ed. Ra-ma. 1996.

84. POST, E.L. 'Finite combinatory processes-formulation 1'. *J. Symbolic Logic*, 1936, pp. 103-105.

85. POST, E.L. 'Recursive unsolvability of a problem of Thue'. *J. Symbolic Logic*, 1947, v.12, N 1, pp. 1-11.

86. RADFORD, D.E. 'The structure of Hopf algebras with projection', *Journal of Algebra* 92(1985) 322-347.

87. REISDORPH, K. 'Aprendiendo Borland C++ Builder 3 en 21 días'. Prentice Hall. México, 1999.

88. RODRÍGUEZ-ROMO, S. 'Communications in Algebra', 27(9),4307-4325 (1999). *Actions of the Dipper-Donkin quantization* GL_2 *on the Clifford algebra* C(1,3).

89. RODRÍGUEZ-ROMO, S. 'Dipper-Donkin algebra as global symmetry of quantum chains'. *J. Phys. A: Math. Gen.* 32 (1999), 7017-7030.

90. SAGLE, A.A., WALDE, R.E. 'Introduction to Lie groups and Lie algebras', Academic Press, 1973.

91. SCHOUTEN, J.A. (1940) 'Über differentialkomitanten zweier kontravarianter Grösen', *Nederl. Akad. Wetensch. Proc. Ser. A* **43**, 449-452.

92. TCEITIN, G.S. 'Associative system with unsolvable the equivalence problem'. *Transactions od the Steclov Institute*, v. 52, Moscow, 1958, pp. 172-189.

93. TURING, A.M. 'On computable numbers with an applications to the Entscheidungsproblem'. *Proc. London Math. Soc.*(2), 1937, 42, pp. 230-265. *Correction.- Proc. London Math. Soc.*(2), 1947, 43, pp. 544-546.

94. SHIRSHOV, A.I. 'On free Lie rings', *Matem. Sbornic* 45(87) N2(1958) 113-122.

95. SHIRSHOV, A.I. 'Some algorithmic problems for Lie algebras', *Sibirskii Math. Journal* 3 N2(1962) 292-296.

96. UFNAROVSKIJ, V.A. (1980): 'Poincaré series of graded algebras'. *Mat. Zametki* 27, No. 1, 21-32. English transl.: Math. Notes *27*, 12-18 (1980). Zbl. 432.16018

97. UFNAROVSKIJ, V.A. 'Encyclopedia of Mathematical Sciences', vol. 57, Algebra VI, *Combinatorial and Asymptotic Methods of Algebra.* 1994.

98. WEYL, H. (1927) 'Quantenmechanik und Gruppentheorie', *Z. Phys.* **46**, 1-46

99. WIRTH, N. 'Algoritmos y estructuras de datos'. Prentice Hall Hispanoamericana. México 1987.

100. WORONOWICZ, S.L. (1987b) Twisted *SU*(2) group. 'An example of a non-commutative differential calculus', *Publ. Res. Inst. Math. Sci.* **23**, 117-81.

101. WORONOWICZ, S.L. *Comm. Math. Phys.* **111** (1987) 613.

102. WORONOWICZ, S.L. *Invent. Math.* **93** (1988) 35.

103. YAMANE, 'A Poincarè-Birkhoff-Witt theorem for quantized universal enveloping algebras of type *AN*', *Publ.RIMS. Kyoto Univ.* 25(1989) 503-520.

104. YAP, CH.K. 'Fundamental Problems of Algorithmic Algebra'. New York. Oxford University Press, Inc. 2000.

♦