UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIAS FÍSICAS

Rompimientos maximales no semisimples de E_6 y sus aplicaciones para modelos con Z'

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIAS (FÍSICA)

PRESENTA:
Enrique Escalante Notario

Tutor Pricipal
Dr. Saúl Noé Ramos Sánchez
Instituto de Física

Miembros del Comité Tutor
Dr. José David Vergara Oliver
Instituto de Ciencias Nucleares

Dr. José Antonio García Zenteno
Instituto de Ciencias Nucleares

Ciudad Univesitaria, CDMX Noviembre 2018

Esta investigación fue realizada con el apoyo del Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) de la UNAM IN100217 “Altas energías y cuerdas tras el hallazgo del Higgs.” Este trabajo fue parcialmente apoyado por el proyecto CONACyT F-252167.

Dedicado a todos los profesores que siguen enseñando con tizas de colores.

Índice general

Introducción

Sin duda, los conceptos de grupo y campos han sido apropiadamente integrados en teorías físicas tan exitosas como la teoría cuántica de campos. Esta teoría representa el corazón del llamado *Modelo Estándar de Partículas Elementales* (SM, por sus siglas en inglés), el cual predice y reproduce todas las observaciones de interacciones subnucleares con una precisión sin precedentes.

La acción del SM es invariante globalmente bajo el grupo de transformaciones de la relatividad de Einstein, el grupo de Poincaré. Adicionalmente, es invariante bajo un conjunto de transformaciones locales o de norma que son correctamente caracterizadas por el grupo $\mathrm{SU}(3)\times\mathrm{SU}(2)\times\mathrm{U}(1)$. En estos términos, las partículas del SM se transforman en representaciones irreducibles de dicho grupo.

En la Tabla 1 se muestran las representaciones de $\mathrm{SU}(3)\times\mathrm{SU}(2)\times\mathrm{U}(1)$ en las que se transforman ambas quiralidades de las tres generaciones de leptones (ν^i, e^i) y quarks (u^i, d^i), y el bosón de Higgs. Adicionalmente, se muestran las representaciones formadas por el campo vectorial B_μ asociado al generador de hipercarga $\mathrm{U}(1)$, por los bosones débiles W^a_μ asociados a $\mathrm{SU}(2)$, y por los gluones g^b_μ asociados a $\mathrm{SU}(3)$.

Partícula	$\mathrm{SU}(3)$	$\mathrm{SU}(2)$	$\mathrm{U}(1)$
(ν_L, e_L)	$\mathbf{1}$	$\mathbf{2}$	$-1/2$
e^c_L	$\mathbf{1}$	$\mathbf{1}$	1
ν^c_L	$\mathbf{1}$	$\mathbf{1}$	0
(u_L, d_L)	$\mathbf{3}$	$\mathbf{2}$	$1/6$
u^c_L	$\bar{\mathbf{3}}$	$\mathbf{1}$	$-2/3$
d^c_L	$\bar{\mathbf{3}}$	$\mathbf{1}$	$1/3$
Higgs H	$\mathbf{1}$	$\mathbf{2}$	$-1/2$
B_μ	$\mathbf{1}$	$\mathbf{1}$	0
$W^{1,2,3}_\mu$	$\mathbf{1}$	$\mathbf{3}$	0
$g^{1,\ldots,8}_\mu$	$\mathbf{8}$	$\mathbf{1}$	0

Tabla 1: Representaciones de las partículas elementales del SM bajo el grupo de norma $\mathrm{SU}(3)\times\mathrm{SU}(2)\times\mathrm{U}(1)$. Se deben considera dos copias más para las tres familias observadas de quarks y leptones.

Las tres componentes del grupo de simetrías norma del SM son interpretadas físicamente como las responsables del comportamiento de tres de las fuerzas fundamentales halladas en la naturaleza: la fuerza fuerte ($\mathrm{SU}(3)$), la fuerza débil ($\mathrm{SU}(2)$) y el electromagnetismo ($\mathrm{U}(1)$). [1] Cada fuerza fundamental posee una cantidad que determina la intensidad de las interacciones regidas por cada fuerza, de forma similar a como ocurre con la "constante" de estructura del electromagnetismo, α_{em}. Estas cantidades podemos llamarlas análogamente *constantes de estruc-*

[1] En realidad, esta identificación es más compleja, ya que la fuerza débil y el electromagnetismo surgen del rompimiento espontáneo del grupo $\mathrm{SU}(2)\times\mathrm{U}(1)$ mediante el mecanismo de Higgs, en el que $\mathrm{SU}(2)$ deja de ser una simetría del universo observable.

tura y denotarlas como $\alpha_3, \alpha_2, \alpha_1$ respectivamente para $\mathrm{SU}(3)\times\mathrm{SU}(2)\times\mathrm{U}(1)$. Un resultado interesante es que, como en el electromagnetismo, estas constantes de estructura adoptan distintos valores al ser medidas a diferentes escalas energéticas. Más interesante aún es que sus valores tienden a coincidir cuando son medidas a escalas tan grandes como 10^{15} GeV.

Tras la confirmación de la existencia del campo de Higgs H [1] reconocida con el premio Nobel 2013, existen pocas interrogantes directas para el SM. Sin embargo, la (muy probable) existencia de materia oscura, la escala de las masas de los neutrinos y la misteriosa tendencia de las constantes de acoplamiento de unificarse a altas energías, podrían ser interpretados como signos de una teoría de gran unificación (GUT, por sus siglas en inglés) subyacente, en la que nuevamente la estructura de grupos juega un papel relevante.

Las GUT están basadas en grupos de simetrías que contiene al grupo de norma del SM. Por ejemplo, el grupo $\mathrm{SU}(5)$ propuesto por Georgi y Glashow [2] coincide en *rango* con el del SM, pero sus generadores incluyen a los generadores de $\mathrm{SU}(3)\times\mathrm{SU}(2)\times\mathrm{U}(1)$ como un subconjunto. Similarmente, el grupo $\mathrm{SO}(10)$, propuesto por Fritzsch y Minkowski [3], también contiene como subgrupo al grupo del SM, e incluso a $\mathrm{SU}(5)$. Otros grupos de GUT incluyen al grupo de Pati y Salam $\mathrm{SU}(4)\times\mathrm{SU}(2)\times\mathrm{SU}(2)$ [4], el grupo de trinificación $\mathrm{SU}(3)^3$ [5], $\mathrm{SU}(6)$ [6], entre otros.

Recientemente, los grupos excepcionales E_6, E_7 y E_8 han sido el centro de actividad científica, debido entre otras cosas a su potencial de proveer predicciones de física nueva y a su surgimiento natural en teorías de supercuerdas. Además, forman parte de la curiosa cadena de GUT

$$\mathrm{SU}(3)\times\mathrm{SU}(2)\times\mathrm{U}(1)\subset\mathrm{SU}(5)\subset\mathrm{SO}(10)\subset E_6\subset E_7\subset E_8\,.$$

El grupo excepcional E_6 ha sido considerado como un grupo de unificación por más de 20 años [7, 8]. La representación fundamental permite la posibilidad de que una generación entera de quarks y leptones (incluyendo un neutrino derecho) y dos dobletes de Higgs estén unidos en una sola representación.

A pesar de su potencial, el grupo E_6 sólo ha sido explorado en un par de trabajos, como los de Anderson [9] y Korh [10]. En comparación con sus subgrupos $\mathrm{SU}(5)$ y $\mathrm{SO}(10)$ [3, 11, 12], nos podemos incluso a atrever a conjeturar que el interés en E_6 ha sido prácticamente nulo, debido quizá a que ese grupo incluye simetrías de norma asociadas a fuerzas aún no detectadas en teorías de campos dotadas con E_6 como grupo de norma. Pero son precisamente estas fuerzas una de las motivaciones del presente trabajo.

El paradigma que conduce a la construcción de modelos de partículas a partir de grupos que contienen las simetrías del SM es conocido como *top-down*. En estos escenarios, se supone que los grupos mayores revelan la física que es válida a escalas de energía superiores a las medidas experimentalmente, por lo que las simetrías correspondientes son rotas (probablemente, de manera espontánea) a altas energías y dan lugar a la física de bajas energías que es observada.

Sin embargo, existe otro enfoque en el cual el SM es una teoría de campo que debe ser extendida para responder las preguntas motivadas por los resultados experimentales que difieren de los resultados predichos por el SM. A este enfoque se le conoce como *bottom-up*. A partir de ambos enfoques, se obtienen modelos que predicen física más allá del SM.

Un prolífico ejemplo de la física nueva que se puede encontrar a partir de la inclusión de simetrías $\mathrm{U}(1)$ adicionales [13–17]. Estas simetrías se invocan para explicar la estabilidad del protón o del vacío correspondiente al potencial de Higgs, o bien, para prevenir la aparición de acoplamientos que violen la simetría discreta CP de manera violenta o para explicar la presencia de algunas señales (ligeramente) anómalas en los colisionadores de partículas, algunas de las cuales han desaparecido en mediciones subsecuentes (como la tristemente célebre señal de 750 GeV). Esto motiva un estudio sistemático del origen de las simetrías $\mathrm{U}(1)$ en, por ejemplo, teorías de gran unificación definidas por grupos como $\mathrm{SO}(10)$, E_6, E_7, E_8, y otras propuestas.

Para explorar esas simetrías $\mathrm{U}(1)$ de manera sistemática como resultado del rompimiento (espontáneo o no) de los grupos de norma más comunes en la literatura, en este trabajo se presenta un nuevo software denominado

`dynkin`. `dynkin` permite trabajar con las propiedades de grupos asociados a álgebras del tipo "A", "D" y "E". `dynkin` es capaz de obtener información sobre el álgebra, incluyendo: la dimensión, el rango, la matriz de Cartan, el diagrama de Dynkin, raíces positivas y raíces simples. Además, a partir del peso máximo de una representación en particular, `dynkin` puede construir el sistema completo de pesos y sus principales características. Adicionalmente, y justamente con la finalidad de descubrir las simetrías $\mathrm{U}(1)$ que pueden surgir de un grupo dado, `dynkin` permite calcular los rompimientos de un álgebra dada, a partir de una regla de rompimiento, aportando especialmente las cargas $\mathrm{U}(1)$ de las representaciones resultantes bajo el subgrupo remanente.

Con el objetivo de explorar la efectividad de nuestro programa `dynkin` y su aplicación en el estudio de las simetrías $\mathrm{U}(1)$ en la física de partículas, este trabajo se enfocó en los rompimientos de E_6 al grupo de norma del SM que incluyen dos simetrías de norma Abelianas extra. Dado que las representaciones **27** y **78** de E_6 contienen los fermiones de materia y los bosones de norma del SM, respectivamente, se analizaron con `dynkin` las consecuencias para la distribución de las cargas Abelianas en las reglas de ramificación de cada una de las representaciones resultantes. Como veremos, el uso de nuestro software conduce a una clasificación de las simetrías $\mathrm{U}(1)'$, a veces también llamadas Z' porque así se conocen los bosones de norma asociados a estas simetrías, provenientes de E_6.

Para concluir esta introducción, describimos la estructura de esta tesis. En el capítulo 1, estudiamos diversos aspectos de teoría de grupos que serán empleados muy frecuentemente en todo el trabajo. En el capítulo 2, discutimos los aspectos fundamentales del SM, sus problemas y las extensiones del SM. En el capítulo siguiente, se da un panorama general de la construcción de `dynkin`, junto con un manual de uso tomando como referencia al álgebra $\mathrm{SU}(5)$. En el último capítulo, se analizan todos los rompimientos maximales no semisimples de E_6 al SM, y se hace el rompimiento para las representaciones irreducibles (*irreps*) **27** y **78** con ayuda de `dynkin`. Debido a que existe una libertad en la elección de la hipercarga se encuentran diversas asignaciones junto con las respectivas cargas adicionales correspondientes a cada asignación.

Capítulo 1

Simetrías

La palabra simetría proviene del latín *symmetrĭa*, que a su vez viene del griego $\sigma\nu\mu\mu\varepsilon\tau\rho\acute{\iota}\alpha$ que se traduce como "con medida". En física, se entiende como cualquier rasgo de un sistema físico que queda invariante ante una transformación. Un ejemplo sencillo es tomar un cuadrado y girarlo 90°, el resultado será el mismo sistema antes de rotar. Decimos entonces que el cuadrado tiene una simetría bajo ese tipo de rotaciones.

La noción de simetría es el corazón de la física teórica actual, un ejemplo es el *Modelo Estándar de las Partículas Elementales* (SM, por sus siglas en inglés), que está formado de una teoría de norma donde las simetrías corresponden a redundancias internas de las partículas elementales. El conjunto de transformaciones de una teoría de norma forman típicamente un *grupo de Lie*, que recibe el nombre de **grupo de norma**.

Por otro lado, existen las *Teorías de Gran Unificación* cuyo principio es que el grupo de norma del SM es un subgrupo de un grupo de Lie "más grande" y puede obtenerse por rompimientos espontáneos de simetría.

Por lo anterior, necesitamos desarrollar un lenguaje matemático para describir las simetrías de norma. Asimismo, es requerido desarrollar un algoritmo computacional para hacer las rupturas ante mencionadas.

1.1. Grupos de Lie y álgebras de Lie

Una de las estructuras algebraicas más sencillas es el **grupo**. Este sólo necesita de un conjunto y una regla de composición sobre los elementos del conjunto, la cual tiene ciertas propiedades [18, 19].

DEFINICIÓN **1.1.1.** *Se llama **grupo** G a un conjunto de elementos $g_1, g_2, \cdots$, (cuyo número puede ser finito, infinito contable o continuo) dotado con una regla de composición $*$ que satisface:*

- *Ser cerrada:* $g_1 * g_2 \in G$.
- *Asociativa:* $(g_1 * g_2) * g_3 = g_1 * (g_2 * g_3)$.
- *Existe el elemento identidad:* $e \in G$, *tal que* $e * g = g * e = g, \forall g \in G$.
- *Existe el elemento inverso:* g^{-1} *para todo* $g \in G$, *tal que* $g * g^{-1} = g^{-1} * g = e$.

En adelante, se omitirá el símbolo de la regla de composición $g_1 * g_2 = g_1 g_2$, por simplicidad en la notación.

Si la regla además es conmutativa $g_1\, g_2 = g_2\, g_1, \forall g_1, g_2 \in G$, diremos que el grupo es **abeliano**. En caso contrario, diremos que el grupo es **no abeliano**.

Si un subconjunto de los elementos del grupo por sí mismo cumple con la definición de grupo con la misma ley de composición, es un **subgrupo**.

DEFINICIÓN **1.1.2.** *Si H es un subgrupo de G, tal que*

$$g \in G, h \in H \text{ si } h' = ghg^{-1} \in H,$$

*entonces H es un **subgrupo invariante**.*

Un grupo es **simple** si no tiene subgrupos invariantes, y **semisimple** si no tiene subgrupos invariantes abelianos.

El estudio de los grupos finitos inició con los trabajos de Galois sobre soluciones a ecuaciones algebraicas. Por otro lado, Sophus Lie (1873) observó que las simetrías de una ecuación diferencial daban lugar a grupos con parámetros continuos. Estas ideas dieron origen a lo que hoy conocemos como **teoría de Lie**, con aportes de Weyl, Cartan, Chevalley, Killing, Serre, Harish-Chandra y otros.

En los primeros trabajos de Lie, la idea subyacente era construir una teoría de "grupos continuos" que complementará la teoría de grupos discretos. El objetivo era desarrollar una teoría capaz de unificar el estudio de las simetrías en el área de ecuaciones diferenciales ordinarias. Si bien el estudio continuó su desarrollo en otra dirección, la teoría de Lie juega un papel central en las matemáticas contemporáneas.

Los grupos y conjuntos con los que Lie trabajó, en general, no eran grupos de Lie en realidad, dado que la estructura de grupo estaba definida sólo cerca de la identidad. Fue Weyl (1924) quién por primera vez estudió sistemáticamente grupos definidos globalmente. Los aportes fundamentales que realizó Lie, fueron asociar a cada grupo de transformaciones continuas un álgebra de Lie y definir una aplicación del álgebra de Lie al grupo de Lie por medio de grupos monoparamétricos.

DEFINICIÓN **1.1.3.** *Sea un conjunto G cuyos elementos $R(a) = R(a_1, a_2, \cdots, a_n)$ dependen de n parámetros reales continuos y satisfacen:*

- *Cerradura: si $R(c) = R(a)R(b)$, entonces $c = f(a, b)$ es una función analítica y real.*[1]
- *Asociatividad: sea $R(a)\,(R(b)R(c)) = (R(a)R(b))\,R(c)$ implica que $f(a, f(b, c)) = f(f(a, b), c)$.*
- *Elemento neutro: sea $R(a_0)$ tal que $R(a)R(a_0) = R(a_0)R(a) = R(a)$, entonces $f(a_0, a) = f(a, a_0) = a$;*
- *Elemento inverso: sea $R^{-1}(a')$, $R^{-1}(a')R(a) = R(a)R^{-1}(a') = R(a_0)$ implica $f(a', a) = f(a, a') = a_0$.*

*Se trata de un **grupo de Lie n-dimensional**, cuyos elementos tienen estructura de variedad diferenciable.*

Los grupos de Lie que nos interesan en física son aquellos que son **compactos**, es decir, para los que la variedad asociada es compacta [8].

Los elementos de un grupo de Lie compacto se pueden definir cerca de la identidad, por medio de una serie de Taylor a primer orden en los *generadores del álgebra* T_i, a esta aproximación la llamaremos *transformación infinitesimal* [20].

Si hacemos una secuencia de transformaciones infinitesimales, se obtiene una transformación finita. Esta corresponde a la **exponencial** de una combinación lineal de los generadores. Si los generadores son hermíticos, la exponencial resulta ser un operador **unitario**.

$$G \ni R(\beta) = e^{-i\beta \cdot T} \equiv \sum_{k=0}^{\infty} \frac{(-i\beta \cdot T)^k}{k!}, \tag{1.1}$$

donde $\beta = (\beta_1, \cdots, \beta_n)$ son n parámetros continuos reales.

Desde un punto de vista geométrico, los generadores son campos vectoriales sobre la variedad, que definen direcciones particulares cerca del elemento identidad de la variedad. Si un grupo de Lie puede interpretarse en

[1] Si cambiamos la condición de ser analítica a sólo continua, obtendremos un grupo continuo.

física como un grupo de transformaciones sobre una variedad diferenciable el álgebra de Lie físicamente puede concebirse como un conjunto de transformaciones infinitesimales.

DEFINICIÓN **1.1.4.** *Sea un grupo de Lie G y* $\mathfrak{g}$ *su espacio tangente en la identidad. Dado un conjunto de elementos* $T_i \in \mathfrak{g}\,(i = 1, \cdots, n)$*, llamados generadores con una regla de composición*

$$[T_i, T_j] = \sum_k i c_{ij}^k T_k, \tag{1.2}$$

donde c_{ij}^k *se denominan constantes de estructura. Este conjunto define el* ***álgebra de Lie*** $\mathfrak{g}$*. El* ***producto de Lie*** $[\cdot, \cdot]$*, tiene las siguientes propiedades:*

- *Es bilineal.*
- *Antisimétrico.*
- *Cumple la identidad de Jacobi.*

La antisimetría del producto implica que las constantes de estructura satisfacen $c_{ij}^k = -c_{ji}^k$. Si elegimos una *representación hermítica* para los generadores, las constantes de estructura resultan ser números reales.

En particular, el elemento identidad es $R(0) = I$ y el elemento inverso de $R(\beta)$ es $R^{-1}(\beta) = R(-\beta) = R^\dagger(\beta)$. Si $|\beta| \ll 1$ podemos quedarnos solo con el término lineal,

$$R(\beta) \approx I - i\beta \cdot T\,. \tag{1.3}$$

Exigiendo que la multiplicación del grupo sea cerrada a segundo orden, recuperamos el producto de Lie. Por lo tanto, es posible conocer todas las propiedades del grupo (cerca de la identidad) si conocemos las propiedades del álgebra de Lie, ya que podemos reconstruir el grupo por exponenciación de los generadores.

Sea un subconjunto $\mathfrak{h}$ de los elementos del álgebra $\mathfrak{g}$, tal que $\mathfrak{h}$ es un álgebra de Lie con el mismo producto de Lie, entonces $\mathfrak{h}$ es una **subálgebra** de $\mathfrak{g}$.

DEFINICIÓN **1.1.5.** *Sean* $X \in \mathfrak{g}$ *y* $Y \in \mathfrak{h}$*, si*

$$[X, Y] \in \mathfrak{h}$$

entonces $\mathfrak{h}$ *es un* ***ideal****.*

Si $\mathfrak{h}$ es distinto de $\mathfrak{g}$ y $\{I\}$ el ideal es **propio**.

Un tipo muy importante de álgebras de Lie son aquellas que no tiene ideales propios y se denominan **simples**. Si el álgebra no contiene ideales abelianos es **semisimple**, y podrá ser escrita como la suma directa de álgebras simples.

Una manera de estudiar álgebras de Lie es por medio de invariantes aritméticos, que resultan ser números que caracterizan a un álgebra. El primero de estos invariantes es el **rango** del álgebra de Lie que es el número máximo de generadores que forma un subálgebra abeliana; el número total de generadores linealmente independientes es la **dimensión** del álgebra.

DEFINICIÓN **1.1.6.** *Una* ***representación de un álgebra de Lie*** $\mathfrak{g}$ *sobre un espacio vectorial* $l-$*dimensional* V *es un homomorfismo de álgebras de Lie:*

$$\varphi : \mathfrak{g} \to \mathit{End}(V)$$

. Donde $\mathit{End}(V)$ *es el conjunto de endomorfismos de* V*.*

Eligiendo una base para el espacio vectorial. La representación será un subconjunto de $GL(n)$, por eso se le suele llamar representación lineal del álgebra. Una representación particular es la *representación adjunta* que se define de la siguiente manera:

DEFINICIÓN **1.1.7.** *Sea $T_i \in \mathfrak{g}$, definimos el endomorfismo*

$$\begin{aligned} &ad_{T_i} : \mathfrak{g} \to \mathfrak{g} \\ &ad_{T_i}(T_j) = [T_i, T_j]\,. \end{aligned}$$

*Llamaremos **representación adjunta** de $\mathfrak{g}$ al homomorfismo*

$$\begin{aligned} ad : \mathfrak{g} &\to End(\mathfrak{g}) \\ T_i &\mapsto ad_{T_i}\,. \end{aligned}$$

De la definición anterior, obtienen los elementos de matriz para la representación adjunta en términos de las constantes de estructura

$$[ad_{T_i}]^k_j = -ic^k_{ij} \equiv (L_i)^k_j \tag{1.4}$$

y usando la identidad de Jacobi, los elementos $(L_i)^k_j$ satisfacen el álgebra de Lie. Las propiedades de un álgebra de Lie queda completamente determinadas por las constantes de estructura.

Ya que los generadores del álgebra de Lie forman un espacio vectorial con elementos $|T_i\rangle$, podemos dotar a este espacio de un producto escalar a través de la representación adjunta de la siguiente manera:

$$\langle T_i | T_j \rangle = \text{tr}\left(L_i^\dagger L_j\right), \tag{1.5}$$

La ec. 1.5 es conocida como *forma de Killing*. Para álgebras de Lie simples y semisimples la forma de Killing es no singular; por lo tanto, podremos diagonalizar el producto como

$$\langle T_i | T_j \rangle = T^{-1}\delta_{ij}, \tag{1.6}$$

donde T es el llamado *índice de Dynkin* que es un número real positivo que depende de la condición de normalización y del álgebra.

1.2. Teoría de estructura para álgebras de Lie

Bajo esta teoría, los generadores de un álgebra de Lie $\mathfrak{g}$ con dimensión n y rango r se puede dividir en dos conjuntos:

- El **subálgebra de Cartan** que es la máxima subálgebra abeliana de G que contiene un número r de elementos hermíticos ($H_i = H_i^\dagger$), tales que

$$[H_i, H_j] = 0, \quad i, j = 1, \cdots, r; \tag{1.7}$$

- y los $n - r$ **generadores restantes** E_α que satisfacen una ecuación de eigenvalores de la forma

$$[H_i, E_\alpha] = \alpha_i E_\alpha, \quad i = 1, \cdots, r;\ \alpha = 1, \cdots, n - r. \tag{1.8}$$

Los números reales α_i en 1.8 son las constantes de estructura en esta nueva base llamada **base de Cartan-Weyl**. Esto permite hacer una asociación unívoca entre los generadores E_α y un vector $\alpha = (\alpha_1, \cdots, \alpha_r)$ que pertenece a un espacio euclidiano r-dimensional. El vector α se llama **raíz** y al espacio euclidiano **espacio de raíces**.

Tomemos como ejemplo el álgebra $\mathfrak{su}_2$ que tiene 3 generadores J_i con $i = 1, 2, 3$, que satisfacen

$$[J_i, J_j] = i\varepsilon_{ij}^k J_k, \tag{1.9}$$

donde ε_{ij}^k es el símbolo de Levi-Civita. Definiendo

$$J_\pm = J_1 \pm iJ_2 \tag{1.10}$$

y sustituyendo en 1.9, el álgebra se convierte en

$$[J_3, J_\pm] = \pm J_\pm, \tag{1.11}$$
$$[J_+, J_-] = 2J_3 \tag{1.12}$$

y los productos restantes iguales a cero. Hemos cambiado las constantes de estructura, pero mantuvimos la forma del álgebra. En este caso la subálgebra de Cartan sólo tiene un elemento que es J_3.

Es importante notar, que si bien se cumple que $[J_+, J_+] = 0$, éste no puede ser un elemento de la subálgebra de Cartan, ya no satisface la ecuación de eigenvalores $[J_+, J_3] \not\propto J_3$. Por lo tanto, los elemento de la subálgebra de Cartan deben de satisfacer las ecs. 1.7 y 1.8 al mismo tiempo. Sin embargo, siempre es posible hacer una cambio de coordenadas para llegar a una base de Cartan-Weyl, donde se satisfacen las ecuaciones antes mencionadas. Cabe mencionar que la base de Cartan-Weyl no es única, ya que también depende de la elección de sistema de referencia en la variedad.

Las ecs. 1.7 y 1.8 no definen completamente el álgebra de Lie original; falta la relación $[E_\alpha, E_\beta]$ con α, β arbitrarios. Para ello recordemos que en mecánica cuántica, el papel de observables físicos está dado a los operadores hermíticos diagonalizables. Sea un estado $|\lambda\rangle$ en un espacio de Hilbert, éste está caracterizado por los eigenvalores obtenidos de la aplicación de los observables sobre el estado en cuestión. Además los eigenestados de los observables forma un base del espacio de Hilbert.

Si se supone que los elementos de álgebra de Lie son operadores sobre un espacio del Hilbert, la acción de los generadores sobre los estados del espacio de Hilbert, nos darán una **representación** de los elementos del álgebra. Ya que los elementos de la subálgebra de Cartan son diagonalizables y hermíticos, toman el papel de observables

$$H_i |\lambda\rangle = \lambda_i |\lambda\rangle , \quad i = 1, \cdots, r. \tag{1.13}$$

Los eigenvalores obtenidos forman un vector $\lambda = (\lambda_1, \cdots, \lambda_r)$ en un espacio euclidiano r-dimensional que llamaremos **espacio de pesos**. El vector λ asociado al eigenestado $|\lambda\rangle$ es llamado **peso**. Tomando como base ortonormal de espacio de Hilbert a los estados $|\lambda\rangle$, falta determinar la acción de los elementos fuera de la subálgebra de Cartan sobre los estados $|\lambda\rangle$,

$$H_i(E_\alpha |\lambda\rangle) = E_\alpha H_i |\lambda\rangle + \alpha_i E_\alpha |\lambda\rangle = (\lambda_i + \alpha_i) E_\alpha |\lambda\rangle . \tag{1.14}$$

Suponiendo que $E_\alpha |\lambda\rangle$ sigue estando en el espacio de Hilbert y como los generadores H_i extraen la información del estado, podemos afirmar que el estado $E_\alpha |\lambda\rangle$ tiene asociado el peso $\lambda + \alpha$. Por lo tanto,

$$E_\alpha |\lambda\rangle = N_{\lambda,\alpha} |\lambda + \alpha\rangle , \tag{1.15}$$

donde $N_{\lambda,\alpha}$ es una constante de proporcionalidad compleja, que está determinada por la condición de normalización y una fase.

Por otro lado, usando la ec. 1.8 obtenemos lo siguiente:

$$\left[H_i, E_\alpha^\dagger\right] = -\alpha_i E_\alpha^\dagger; \tag{1.16}$$

$$[H_i, E_{-\alpha}] = -\alpha_i E_{-\alpha}; \tag{1.17}$$

ya que la relación entre E_α y α es unívoca, llegamos a que

$$E_\alpha^\dagger = E_{-\alpha}. \tag{1.18}$$

De lo anterior, se concluye que el operador E_α "mueve" el peso de λ a $\lambda + \alpha$ y el operador $E_\alpha^\dagger$ en la dirección $-\alpha$ el espacio de peso. Por lo tanto, se deberia interpretar a las raíces como direcciones particulares en el espacio de pesos.

La representación adjunta definida en 1.4 es particularmente importante, ya que está dada por las constantes de estructura del álgebra. De análisis anterior, sabemos que las componentes de las raíces son las constantes de estructura del álgebra en la base de Cartan-Weyl. Por lo tanto, existe un relación entre las raíces y la representación adjunta.

Aplicando los generadores del álgebra de Lie sobre los estados $|T_i\rangle$, la forma en que actúan los generadores sobre estados es la siguiente:

$$T_i \left|T_j\right\rangle = \left|[T_i, T_j]\right\rangle, \tag{1.19}$$

esta relación es conocida como **acción adjunta** y es consecuencia de lo siguiente:

$$T_i \left|T_j\right\rangle = \sum_k \left|T_k\right\rangle \left\langle T_k\right| T_i \left|T_j\right\rangle = \sum_k \left|T_k\right\rangle \left[L_i\right]_{kj} = \sum_k -ic_{kj}^i \left|T_k\right\rangle = \sum_k ic_{ij}^k \left|T_k\right\rangle = \left|\sum_k ic_{ij}^k T_k\right\rangle = \left|[T_i, T_j]\right\rangle, \tag{1.20}$$

donde se ha usado la representación adjunta $\left[L^i\right]_{jk} = -ic_{jk}^i$ y que los estados $|T_i\rangle$ forma un base.

En la base Cartan-Weyl, un conjunto de r estados denotados por $|H_i\rangle$ y $n-r$ estados $|E_\alpha\rangle$. Sobre estos estados apliquemos los elementos de la subálgebra de Cartan.

$$H_i \left|H_j\right\rangle = 0, \tag{1.21}$$

$$H_i \left|E_\alpha\right\rangle = \alpha_i \left|E_\alpha\right\rangle, \tag{1.22}$$

que son equivalentes a las ecs. 1.7 y 1.8. Comparándolas se afirmar que: *las raíces son un caso particular de pesos*. Por lo tanto, los estados $|H_i\rangle$ corresponden a los generadores de la subálgebra de Cartan y tienen pesos nulos, mientras que los estados $|E_\alpha\rangle$ tienen un peso correspondiente igual a α. Además los estados son ortonormales bajo el producto 1.6.

Aplicando las ideas anteriores a $E_\alpha \left|E_{-\alpha}\right\rangle$, éste tiene un peso igual a $\alpha - \alpha$. Por lo tanto, el estado debe ser un combinación lineal de los estados $|H_i\rangle$. Usando la ec. 1.20

$$E_\alpha \left|E_\alpha\right\rangle = \left|[E_\alpha, E_{-\alpha}]\right\rangle = \left|\alpha^i H_i\right\rangle, \tag{1.23}$$

donde α^i son componentes del vector dual a la raíz α en el espacio de raíz, con el producto escalar definido en el espacio de raíces como

$$\alpha \cdot \beta = \alpha_i \beta^i = \alpha^j \beta_j = \alpha_i h^{ij} \beta_j. \tag{1.24}$$

que relaciona a α^i y α_i. Este producto escalar es definido positivo. Si la longitud de las raíces está normalizada de tal manera que

$$\sum_{\alpha \neq 0} \alpha_i \alpha_j = \delta_{ij} \quad \text{ó} \quad \sum_{\alpha \neq 0} \alpha \cdot \alpha = r = \text{rango}, \tag{1.25}$$

entonces $h^{ij} = \delta^{ij}$ y podemos identificar a α^i con α_i: $\alpha^i = \alpha_i$.

En general, el estado $E_\alpha |E_\beta\rangle$ debe tener un peso asociado igual a $\alpha + \beta$. Como el número de raíces es finito, $E_\alpha |\beta\rangle$ es distinto de cero, si y sólo si, $\alpha + \beta$ es raíz. Resumiendo

$$[E_\alpha, E_\beta] = \begin{cases} 0 & \alpha + \beta \quad \text{no es raíz} \\ N_{\alpha,\beta} E_{\alpha+\beta} & \alpha + \beta \quad \text{es raíz} \\ \alpha \cdot H & \alpha + \beta = 0 \end{cases} \tag{1.26}$$

Para calcular los coeficientes $N_{\alpha,\beta}$, primero se debe aplicar la identidad de Jacobi a los generadores $E_\alpha, E_\beta, E_\gamma$

$$[[E_\alpha, E_\beta], E_\gamma] + [[E_\beta, E_\gamma], E_\alpha] + [[E_\gamma, E_\alpha], E_\beta] = 0. \tag{1.27}$$

De ésta se deriva lo siguiente, si $\alpha+\beta+\gamma = 0$, entonces $\alpha N_{\beta,\gamma} + \beta N_{\gamma,\alpha} + \gamma N_{\alpha,\beta} = 0$ y $N_{\beta,\gamma} = N_{\gamma,\alpha} = N_{\alpha,\beta}$.

Se debe encontrar una relación recursiva que involucre a estos coeficientes. Para hacer esto se introduce a β en una secuencia de raíces que se obtiene sumando α, como se muestra en la Fig. 1.1. Esta secuencia es

$$\beta - m\alpha \quad \beta - (m-1)\alpha \quad \cdots \quad \beta \quad \beta + \alpha \quad \cdots \quad \beta + n\alpha \tag{1.28}$$

todas son raíces, excepto

$$\beta - (m-1)\alpha, \tag{1.29}$$
$$\beta + (n+1)\alpha. \tag{1.30}$$

Aplicando la identidad de Jacobi a las raíces $\alpha, \beta + k\alpha, -\alpha$ se obtiene la relación de recursión.

$$N^2_{\alpha,\beta+(k-1)\alpha} = N^2_{\alpha,\beta+k\alpha} + \alpha \cdot (\beta + k\alpha) \tag{1.31}$$

Donde hemos usado la identidad de las constantes $N_{\alpha,\beta}$, $N_{-\alpha,-\beta} = -N_{-\beta,-\alpha} = -N^*_{\alpha,\beta}$.

Las raíces definen direcciones particulares en el espacio de raíces. Si iniciamos con la raíz β y sumamos n veces la raíz α, debe pasar que $\beta + (n+1)\alpha$ no sea raíz. De manera análoga, debe suceder que $\beta - (m+1)\alpha$ no sea raíz, esto se muestra en la Fig. 1.1.

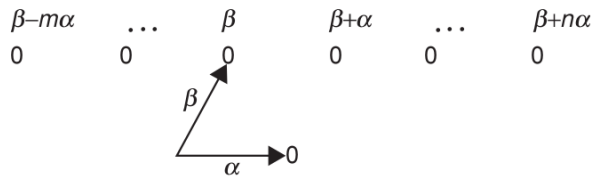


Figura 1.1: Esta cadena de raíces nos muestra como calcular los coeficientes $N_{\alpha,\beta}$

Esta relación de recursión satisface las condiciones

$$N^2_{\alpha,\beta+n\alpha} = 0, \tag{1.32}$$
$$N^2_{-\alpha,\beta-m\alpha} = 0. \tag{1.33}$$

Luego la condición inicial $N_{\alpha,\beta+n\alpha} = 0$ conducen a

$$N^2_{\alpha,\beta+(k-1)\alpha} = (n-k+1)\left(\alpha\cdot\beta + \frac{1}{2}(n+k)\alpha\cdot\alpha\right) \tag{1.34}$$

y la condición $N^2_{-\alpha,\beta-m\alpha} = N^2_{\alpha,\beta-(m+1)\alpha} = 0$ implica que

$$N^2_{\alpha,\beta-(m+1)\alpha} = (n+m+1)\left(\alpha\cdot\beta + \frac{1}{2}(n-m)\alpha\cdot\alpha\right) = 0 \tag{1.35}$$

De la expresión anterior, se extrae la siguiente información:

- $N^2_{\alpha,\beta+k\alpha} = (n-k)(m+k+1)(\alpha\cdot\alpha)/2 \geq 0$. Notemos que $\alpha\cdot\beta > 0$ cuando $m-n > 0$ y $\alpha\cdot\beta < 0$ cuando $m-n < 0$.
- El producto escalar que se encuentra entre paréntesis en la ec. 2.9

$$-n \leq \frac{2\alpha\cdot\beta}{\alpha\cdot\alpha} = -n+m \leq m. \tag{1.36}$$

  donde m, n son entero no negativos. La expresión

$$2\frac{\alpha\cdot\beta}{\alpha\cdot\alpha} = -(n-m) \tag{1.37}$$

  es conocida como **fórmula maestra**.

- Si α y β son raíces, entonces

$$\beta' = \beta - 2\frac{\beta\cdot\alpha}{\alpha\cdot\alpha}\alpha \tag{1.38}$$

  es también raíz. La raíz es obtenida por reflejar β en el hiperplano ortogonal a α.

Recapitulando se sabe que

$$\begin{aligned}
&[H_i, H_j] = 0, && \\
&[H_i, E_\alpha] = \alpha_i E_\alpha, && \\
&[E_\alpha, E_\beta] = \alpha^i H_i, && \alpha+\beta = 0, \\
&\qquad = N_{\alpha,\beta}E_{\alpha+\beta} && \alpha+\beta \neq 0 \quad \text{una raíz}, \\
&\qquad = 0 && \alpha+\beta \quad \text{no es una raíz}.
\end{aligned} \tag{1.39}$$

Los coeficientes $N_{\alpha,\beta}$ están determinados en términos de enteros no negativos m, n por

$$N^2_{\alpha,\beta+k\alpha} = (n-k)(m+k+1)(\alpha\cdot\alpha)/2 \tag{1.40}$$

donde $\beta + k\alpha$ es una raíz sólo cuando $k = -m, \cdots, n$. Las raíces están normalizadas por

$$\sum_{\alpha\neq 0} \alpha\cdot\alpha = r = \text{rango} \tag{1.41}$$

En la deducción de las constantes de estructura $N_{\alpha,\beta}$ se obtuvo que

$$\frac{2(\alpha\cdot\beta)}{\alpha\cdot\alpha} \quad \text{es un entero,} \tag{1.42}$$

$$\beta' = \beta - \frac{2(\alpha\cdot\beta)}{\alpha\cdot\alpha}\alpha \quad \text{es una raíz.} \tag{1.43}$$

La raíz β' es obtenida por reflejar la raíz β es el hiperplano ortogonal a α. Estas dos observaciones son todo lo que se necesita para construir el espacio de raíces de cualquier rango.

Escribiendo

$$\begin{aligned} 2(\alpha\cdot\beta)/(\alpha\cdot\alpha) &= l \\ 2(\alpha\cdot\beta)/(\beta\cdot\beta) &= l', \end{aligned} \tag{1.44}$$

donde l y l' son enteros, entonces por la desigualdad de Schwarz

$$0 \leq \cos^2(\theta_{\alpha\beta}) = \left(\frac{\alpha\cdot\beta}{\alpha\cdot\alpha}\right)\left(\frac{\alpha\cdot\beta}{\beta\cdot\beta}\right) = \frac{l'l}{4} \leq 1. \tag{1.45}$$

Como l y l son enteros su producto es un entero entre 0 y 4. Por lo tanto, los posibles ángulos son:

l'l	$\theta_{\alpha\beta}$
0	$\pi/2$
1	$2\pi/3,\ \pi/3$
2	$3\pi/4, \pi/4$
3	$5\pi/6, \pi/6$
4	$\pi, 0$

Tabla 1.1: Ángulos permitidos

La última línea no es interesante ya que sabemos que las raíces siempre están en pares, es decir, si α es raíz, entonces $-\alpha$ también lo es.

Al conjunto completo de raíces, se denomina **sistema de raíces**. El sistema de raíces es invariante ante un conjunto de reflexiones con respecto a hiperplanos ortogonales a las raíces, el cual es un subconjunto de las isometrías del espacio de raíces. Dicho conjunto de reflexiones tiene la estructura de grupo y se llama **grupo de Weyl**, el cual siempre es discreto.

Es necesario introducir una base más útil para trabajar con las representaciones del álgebra. Una mejor base es el conjunto de las propias raíces, ya que la aplicación de los generadores tiene una descripción más simple en esta base.

En cualquier base de Cartan-Weyl, las componentes de las raíces están fijas. Una raíz se denomina **positiva**, si su primera componente no cero es positiva. Aunque esta clasificación depende de la elección de la base, eventualmente, se demostrará que los resultados obtenidos no dependen de esta elección.

Es fácil probar que no todas las raíces positivas son linealmente independientes. Sin embargo, existen r raíces positivas que no se pueden escribir como la suma de dos raíces positivas. A estas raíces las llamaremos **simples**.

En la Fig. 1.2 están representadas las raíces asociadas al álgebra $\mathfrak{su}_3$, donde $\{e_1, e_2\}$ es la base canónica del espacio euclidiano 2-dimensional. Las raíces que se encuentran apuntando a la derecha son las raíces positivas. Las raíces simples se pueden elegir como , α_1 y α_2.

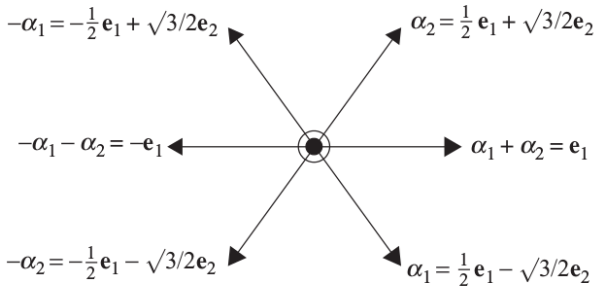


Figura 1.2: Sistema de raíces de $\mathfrak{su}(3)$, tomado de [20]

Las propiedades de las raíces simples son:

1. Si α y β son raíces simples diferentes, entonces $\alpha - \beta$ no es raíz.

2. Debido a que $\alpha - \beta$ no es raíz
$$E_{-\alpha}\left|E_\beta\right\rangle = E_{-\beta}\left|E_\alpha\right\rangle = 0. \tag{1.46}$$
En las ecs.1.37 y 1.44, haciendo $m = m' = 0$ lo que implica $l = -n$ y $l' = -n'$, se obtiene
$$cos\theta_{\alpha\beta} = -\frac{\sqrt{nn'}}{2}, \quad \frac{\beta^2}{\alpha^2} = \frac{n}{n'}. \tag{1.47}$$
Conociendo los enteros n y n' para cada raíz simple es equivalente a conocer los ángulos entre las raíces simples, y su longitud relativa.

3. El ángulo entre cualquier par de raíces simples satisface
$$\frac{\pi}{2} \leq \theta < \pi. \tag{1.48}$$
La primera desigualdad se sigue de 1.47 porque el coseno es menor o igual que cero. La segunda desigualdad se obtiene porque todas las raíces simples son postivivas.

4. Cualquier raíz positiva se puede escribir como la suma de raíces simples con coeficientes no negativos enteros.

5. Las raíces simples forman un base completa, ya que su número es igual al rango del álgebra.

Finalmente usando la Tabla. 1.1 y las propiedades anteriores llegamos a que:

- Los ángulos posibles entre dos raíces simples son: $\pi/2$, $2\pi/3$, $3\pi/4$ y $5\pi/6$.
- Y sólo existen dos longitudes distintas de raíces, ya que $\beta^2/\alpha^2 = 1, 2, 3$.

1.3. Diagramas de Dynkin

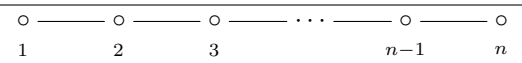
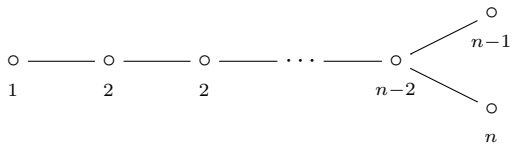
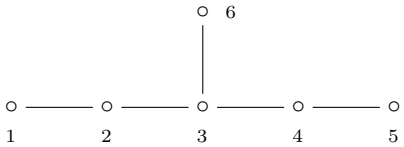
En la sección anterior, se encontraron restricciones para los posibles sistemas de raíces de álgebras de Lie simples. Por lo tanto, podemos hacer una clasificación de las álgebras a partir de la clasificación del sistema de raíces. Los primeros trabajos en esta área fueron hecho por Killing (1888-1890) y fueron formalizados por Cartan (1894). Cartan demostró que existen 4 series infinitas denotadas por las letras "A_n", "B_n", "C_n" y "D_n" y las álgebras en cada familia se diferencia por su rango. Además existen 5 álgebras excepcionales: "F_4", "G_2", "E_6", "E_7" y "E_8".

Dynkin encontró una manera diagramática para capturar toda la información sobre el sistema de raíces y, por lo tanto, del álgebra [21]. Estos son los llamados **Diagramas de Dynkin**, los cuales son grafos bidimensionales que se construyen respetando las siguientes reglas:

1. Las raíces simples son denotadas por puntos blancos si sólo existe una longitud para las raíces. Si existen dos longitudes diferentes, las raíces más grandes se denotan con puntos blancos y las pequeñas con puntos negros.

2. El ángulo entre dos raíces simples es denotado por líneas que conectan sus puntos correspondientes: ninguna línea significan un ángulo de $\pi/2$, una línea corresponde a un ángulo de $2\pi/3$, dos líneas es para un ángulo de $3\pi/4$ y finalmente un ángulo de $5\pi/6$ corresponde a tres líneas. El número de líneas entre dos puntos de que nos da información sobre el cociente de las longitudes de las raíces: dos raíces conectadas por 3 líneas corresponden a un factor de $\sqrt{3}$, dos líneas a $\sqrt{2}$, una línea a 1 y cuando no existe ninguna linea no existe constricción alguna.

Los diagramas de Dynkin contienen toda la información sobre el álgebra de Lie correspondiente. Por ejemplo, todas las álgebras de Lie simples tiene un diagrama de Dynkin **conexo**, mientras las álgebras semisimples tienen diagramas desconectados. Los diagramas para todas las álgebras simples se muestran en la Tabla. 1.2. A partir de los diagramas se puede reconstruir el sistema de raíces, ya que toda raíz es una combinación lineal de raíces simples.

Álgebra	Grupo	Rango	Diagrama de Dynkin
$A_n = \mathfrak{su}_{n+1}$	$SU(n+1)$	$^\forall n$	
$B_n = \mathfrak{so}_{2n+1}$	$SO(2n+1)$	$^\forall n$	
$C_n = \mathfrak{usp}_{2n}$	$USp(2n)$	$^\forall n$	
$D_n = \mathfrak{so}_{2n}$	$SO(2n)$	$^\forall n$	
E_6	E_6	6	

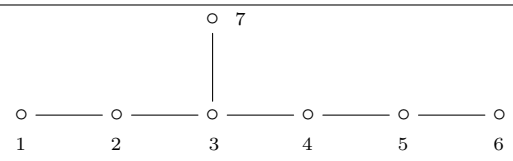
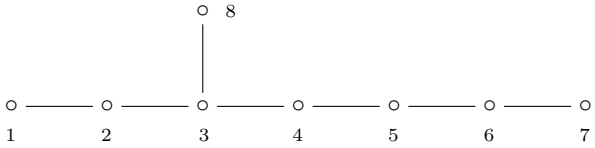
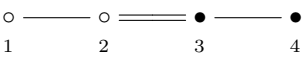
Álgebra	Grupo	Rango	Diagrama de Dynkin
E_7	E_7	7	
E_8	E_8	8	
F_4	F_4	4	
G_2	G_2	2	

Tabla 1.2: Diagramas de Dynkin

Nos centraremos en las álgebras que solo tienen una longitud, es decir, en las familias A, D y E. Se sabe que las raíces simples son una base no ortogonal del espacio de raíces, se construye una matriz cuadrada $r \times r$ que codifique esta no ortogonalidad [8].

DEFINICIÓN **1.3.1.** *Usando el producto euclidiano asociado al espacio de raíces, se define las componentes A_{ij} de una matriz cuadrada $r \times r$*

$$A_{ij} = 2\frac{\alpha_i \cdot \alpha_j}{\alpha_j \cdot \alpha_j}. \tag{1.49}$$

*donde α_i es una raíz simple. A esta matriz es llamada **matriz de Cartan**.*

Las matrices de Cartan para las familias A, D y E se muestran en la Tabla 1.3

Álgebra	Grupo	Rango	Matriz de Cartan
$A_n = \mathfrak{su}_{n+1}$	$SU(n+1)$	$^{\forall}n$	$\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2 & -1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 2 & -1 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix}$

Álgebra	Grupo	Rango	Matriz de Cartan
$D_n = \mathfrak{so}_{2n}$	$SO(2n)$	$^{\forall}n$	$\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2 & -1 & -1 \\ 0 & 0 & 0 & \cdots & -1 & 2 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 0 & 2 \end{pmatrix}$
E_6	E_6	6	$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & -1 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & 0 \\ 0 & 0 & -1 & 0 & 0 & 2 \end{pmatrix}$
E_7	E_7	7	$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 2 \end{pmatrix}$
E_8	E_8	8	$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$

Tabla 1.3: Matrices de Cartan

Se resaltan las siguientes propiedades:

- De la definición de la matriz de Cartan junto con la condición 1.47, podemos afirmar que:
  - $A_{ii} = 2$,
  - $A_{ij} = \{0, -1, -2, -3\}$ si $i \neq j$.
- Se pueden construir las matrices de Cartan, a partir del diagrama de Dynkin correspondiente. Debido a que cada raíz simple corresponde a un punto en el diagrama de Dynkin y que el producto de dos raíces está

codificado en el número de líneas que conectan a los puntos. Además, cada raíz simple corresponde a una fila (columna) de la matriz de Cartan.

Las raíces simples son un base de espacio de raíces, podemos usar el sistema de raíces simples como una base para cualquier espacio de pesos. Un elemento Λ de dicho espacio es una combinación lineal de las raíces simples de la forma

$$\Lambda = \sum_i \bar{\lambda}^i \frac{2}{\alpha_i^2}\alpha_i = \sum_i \bar{\lambda}^i \hat{\alpha}_i, \quad \text{donde} \quad \hat{\alpha}_i = \frac{2\alpha_i}{\alpha_i^2}. \tag{1.50}$$

Para las álgebras A, D y E, el factor $2/\alpha_i^2$ es 1, debido a que las raíces simples más largas están normalizadas a $\sqrt{2}$. Las componentes $\bar{\lambda}^i$, forman un vector r-dimensional $\bar{\lambda} = (\bar{\lambda}^1, \bar{\lambda}^2, \cdots, \bar{\lambda}^r)$, que está asociado al peso Λ.

La **base dual** $\{\hat{\mu}^i\}$ asociada a la base $\{\hat{\alpha}_i\}$, también es un base del espacio de pesos. Por lo que cualquier elemento se puede escribir como una combinación lineal de sus elementos

$$\Lambda = \sum_j a_j \hat{\mu}^j. \tag{1.51}$$

Esta base se denomina **base de Dynkin** y a las componentes a_i, **etiquetas de Dynkin**. Para hacer el cambio de base, las componentes a_i se escriben

$$a_i = \frac{2\Lambda \cdot \alpha_i}{\alpha_i^2} = \sum_j \bar{\lambda}_j \frac{2\alpha_j \cdot \alpha_i}{\alpha_j^2}\frac{2}{\alpha_i^2} = \sum_j \bar{\lambda}_j \frac{2}{\alpha_j^2} A_{ji}. \tag{1.52}$$

Una característica importante de la base de Dynkin es que todas las componentes a_i son números enteros para cualquier peso o raíz. El producto euclidiano que posee cualquier espacio de pesos, se escribe en estas bases como

$$\begin{aligned} \Lambda \cdot \Lambda' &= \sum_{ij} \bar{\lambda}^i \frac{2}{\alpha_i^2}(\alpha_i \cdot \alpha_j)\frac{2}{\alpha_j^2}\bar{\lambda'}^j = \sum_{ij} \bar{\lambda}_i \frac{2}{\alpha_i^2} A_{ij} \bar{\lambda'}_j \\ &= \sum_j a_j \bar{\lambda'}^j = \sum_i a'_i \bar{\lambda}^i = \sum_{ij} a_i G_{ij} a_j, \end{aligned} \tag{1.53}$$

donde G_{ij} define un **tensor métrico**

$$G_{ij} = (A^{-1})_{ij}\frac{\alpha_j \cdot \alpha_j}{2}. \tag{1.54}$$

Por lo tanto, el tensor métrico cumple la función de matriz de cambio de bases

$$\bar{\lambda}^i = G^{ij} a_j.$$

Los tensores métricos de las álgebras A, D y E están en la Tabla 1.4

Álgebra	Tensor Métrico
A_n	$G_{ij} = \frac{1}{n+1} \begin{pmatrix} 1\cdot n & 1\cdot(n-1) & 1\cdot(n-2) & \cdots & 1\cdot 2 & 1\cdot 1 \\ 1\cdot(n-1) & 2\cdot(n-1) & 2\cdot(n-2) & \cdots & 2\cdot 2 & 2\cdot 1 \\ 1\cdot(n-2) & 2\cdot(n-2) & 3\cdot(n-2) & \cdots & 3\cdot 2 & 3\cdot 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1\cdot 2 & 2\cdot 2 & 3\cdot 2 & \cdots & (n-1)\cdot 2 & (n-1)\cdot 1 \\ 1\cdot 1 & 2\cdot 1 & 3\cdot 1 & \cdots & (n-1)\cdot 1 & n\cdot 1 \end{pmatrix}$
D_n	$G_{ij} = \frac{1}{2} \begin{pmatrix} 2 & 2 & 2 & \cdots & 2 & 1 & 1 \\ 2 & 4 & 4 & \cdots & 4 & 2 & 2 \\ 2 & 4 & 6 & \cdots & 6 & 3 & 3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 2 & 4 & 6 & \cdots & 2(n-2) & n-2 & n-2 \\ 1 & 2 & 3 & \cdots & n-2 & n/2 & (n-2)/2 \\ 1 & 2 & 3 & \cdots & n-2 & (n-2)/2 & n/2 \end{pmatrix}$
E_6	$G_{ij} = \frac{1}{3} \begin{pmatrix} 4 & 5 & 6 & 4 & 2 & 3 \\ 5 & 10 & 12 & 8 & 4 & 6 \\ 6 & 12 & 18 & 12 & 6 & 9 \\ 4 & 8 & 12 & 10 & 5 & 6 \\ 2 & 4 & 6 & 5 & 4 & 3 \\ 3 & 6 & 9 & 6 & 3 & 6 \end{pmatrix}$
E_7	$G_{ij} = \frac{1}{2} \begin{pmatrix} 4 & 6 & 8 & 6 & 4 & 2 & 4 \\ 6 & 12 & 16 & 12 & 8 & 4 & 8 \\ 8 & 16 & 24 & 18 & 12 & 6 & 12 \\ 6 & 12 & 18 & 15 & 10 & 5 & 9 \\ 4 & 8 & 12 & 10 & 8 & 4 & 6 \\ 2 & 4 & 6 & 5 & 4 & 3 & 3 \\ 4 & 8 & 12 & 9 & 6 & 3 & 7 \end{pmatrix}$
E_8	$G_{ij} = \begin{pmatrix} 4 & 7 & 10 & 8 & 6 & 4 & 2 & 5 \\ 7 & 14 & 20 & 16 & 12 & 8 & 4 & 10 \\ 10 & 20 & 30 & 24 & 18 & 12 & 6 & 15 \\ 8 & 16 & 24 & 20 & 15 & 10 & 5 & 12 \\ 6 & 12 & 18 & 15 & 12 & 8 & 4 & 9 \\ 4 & 8 & 12 & 10 & 8 & 6 & 3 & 6 \\ 2 & 4 & 6 & 5 & 4 & 3 & 2 & 3 \\ 5 & 10 & 15 & 12 & 9 & 6 & 3 & 8 \end{pmatrix}$

Tabla 1.4: Tensores métricos

La raíz simple α_i tiene componentes A_{ij} en la base de Dynkin, por lo que podemos conocer el conjunto de

raíces simples a partir de la matriz de Cartan. Para derivar el sistema completo de raíces partiendo de las raíces simples, es necesario conocer las combinaciones lineales de raíces simples que resultan ser raíces. Para esto se tienen las siguientes restricciones

1. En una base de Cartan-Weyl, el número de raíces nulas es igual al rango del álgebra, es decir, la degeneración de la raíz nula coincide con el rango del álgebra. Mientras que las raíces no nulas son no degeneradas.

2. Si α es raíz, $-\alpha$ también lo es. Las raíces que se pueden formar por combinaciones lineales con coeficientes no negativos de raíces simples, son raíces positivas en alguna base. Por lo tanto, solo necesitamos la mitad de las raíces no nulas.

3. Existe una **raíz máxima** de la cual se obtienen todas las raíces por sustracción de raíces simples.

1.4. Representaciones

El uso de grupos de Lie implica introducir una **representación** n-dimensional del grupo. En Física, y en particular en Teoría Cuántica de Campos, se utiliza frecuentemente las representaciones *fundamental*, *anti-fundamental* y *adjunta* [22–24]. En esta sección, entenderemos qué significan estos adjetivos y por qué estas representaciones tiene un papel sobresaliente en la construcción de teoría de norma.

Para poder hacer cálculos explícitos es necesario introducir un representación, que es un mapeo de un objeto matemático a otro con la misma estructura algebraica, este mapeo preserva las operaciones en dichos objetos. La definición formal para una representación de un álgebra de Lie es

DEFINICIÓN **1.4.1.** *Una **representación** de un álgebra de Lie $\mathfrak{g}$ sobre un espacio vectorial $n-$dimensional V es un homomorfismo de álgebras de Lie:*

$$\varphi : \mathfrak{g} \to End(V),$$

donde End(V) es el conjunto de endomorfismos de V.

Eligiendo una base para el espacio vectorial V, obtenemos otro homomorfismo del álgebra $\mathfrak{g}$ al álgebra del $GL(n, \mathbb{C})$, definido como

DEFINICIÓN **1.4.2.** *Una **representación lineal** de álgebra de Lie es un homomorfismo*

$$\begin{aligned} \varphi : \mathfrak{g} &\to GL(n, \mathbb{C}) \\ \varphi(T_i) &= R_i \end{aligned}$$

tal que

$$[R_i, R_j] = ic_{ij}^k R_k \tag{1.55}$$

donde el producto de Lie del álgebra, es sustituido por el conmutador habitual de matrices.

En la Fig. 1.3 se muestran las posibles representaciones que son:

1. **Cero**: todas las constantes de estructura desaparecen y el álgebra resultante es conmutativa.

2. **Nilpotente**: en este caso, la representación solo tiene elementos en la parte triangular superior.

3. **Soluble**: los elementos no nulos están sobre y por encima de la diagonal.

4. **Reducible**: en este caso, existe un sector que tiene elementos nulos. Esta representación está asociada a un álgebra de Lie no semisimple.

5. **Completamente reducible**: en esta representación es diagonalizable a bloques y representa un álgebra de Lie semisimple.

6. **Irreducible** (irrep): esta representación no tiene un sector definido con elementos nulos, y representa un álgebra de Lie simple.

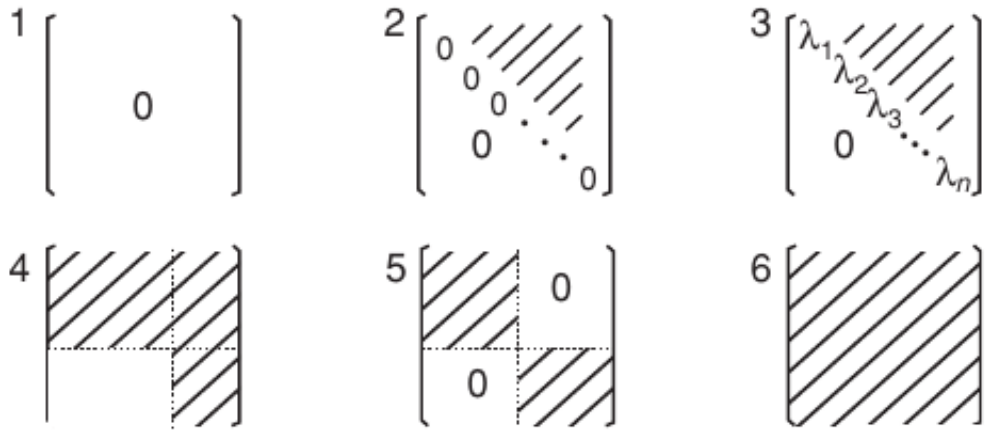


Figura 1.3: Estructura de las representaciones para diferentes tipos de álgebras de Lie

Si la representación es inyectiva, se dirá que es **fiel**. En general, una representación **reducible** se puede escribir como la suma directa de irreps.

A partir de las constantes de estructura c_{ij}^k, se pueden construir matrices cuyos elementos sean $(L_i)_j^k = (c^k)_{ij}$. Estas matrices satisfacen las relaciones de conmutación del álgebra, como consecuencia de la identidad de Jacobi, y forman una irrep del álgebra llamada **representación adjunta** cuya dimensión es igual a la dimensión del álgebra.

En Mecánica Cuántica, la descripción de las simetrías está caracterizada por la acción de los generadores de un grupo de simetría sobre los elementos de un espacio de Hilbert que describen a los estados de un sistema físico. Por lo tanto, todos los estados en el espacio de Hilbert que estén conectados por los generadores E_α forman una base de una irrep. De la sección anterior, este conjunto está relacionado con un sistema de pesos, se puede afirmar que *conociendo los sistemas de pesos, conocemos las irreps de un álgebra de Lie*.

La teoría de representaciones de álgebras de Lie simples se resume elegantemente en términos de diagramas de Dynkin, ya que estos contienen todas las raíces simples con las cuales se puede reconstruir el sistema de raíces, de la misma manera, se puede reconstruir el sistema de pesos para una irrep en particular.

Sea un espacio de Hilbert n-dimensional donde los generadores del grupo G están representados fielmente por matrices $n \times n$. Estas matrices obedecen el álgebra de Lie $\mathfrak{g}$ asociado al grupo G. Además, los elementos del espacio de Hilbert $|\lambda\rangle$ pueden ser representados por un vector r-dimensional λ, este vector es un punto en el espacio de raíces debido a que las raíces simples son una base de cualquier espacio de pesos. En este esquema, los elementos de la subálgebra de Cartan son los ejes en el espacio de raíces.

Para iniciar la construcción de una irrep, nótese que:

1. Existe un vector de **peso máximo** $|\Lambda\rangle$ en el espacio de raíces asociado a la irrep, similar a la raíz máxima.

2. Las componentes de los pesos en la base de Dynkin siempre son números enteros.

3. La aplicación del operador E_α sobre el estado $|\lambda\rangle$ en el espacio de Hilbert, es equivalente la traslación $\lambda \to \lambda + \alpha$ en el espacio de raíces.

Dynkin demostró el siguiente teorema:

TEOREMA **1.4.1.** *Cada irrep de un álgebra de Lie tiene asociado un conjunto de enteros no negativos, y este resulta ser el peso máximo de una y solo una irrep.*

Sea el peso máximo $\Lambda = (a_1, a_2, \cdots, a_r)$ en la base de Dynkin, entonces la i-ésima raíz simple[2] debe ser **sustraída** de Λ a_i veces siempre y cuando a_i sea mayor que 0. Esto genera un conjunto de pesos, a los cuales se les deberá volver a aplicar la misma regla a cada uno. Este proceso se detiene cuando el conjunto tiene un solo elemento y las componentes de éste son todas negativas. A este elemento se conoce como **peso mínimo**. Existe la posibilidad de degeneración en los pesos, la cual debe ser contabilizada.

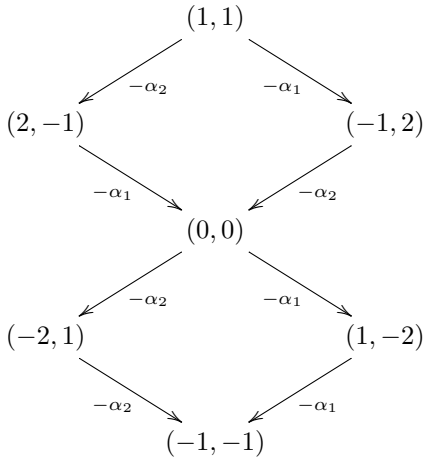
Por ejemplo, para $\mathfrak{su}_3$, el diagrama de Dynkin es $\underset{1}{\circ}\!\!-\!\!\!-\!\!\underset{2}{\circ}$ y su matriz de Cartan es

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \tag{1.56}$$

Si el peso máximo es $\Lambda = (1, 0)$, aplicado el algoritmo se obtiene $(-1, 1)$. Volviendo a aplicar llegamos a $(0, -1)$, y aquí se detiene el proceso. Esta es la representación fundamental de $\mathfrak{su}(3)$, que tiene una dimensión igual a 3.

$$(1,0) \xrightarrow[-\alpha_1]{} (-1,1) \xrightarrow[-\alpha_2]{} (0,-1)$$

Asimismo, podemos calcular la representación adjunta de $\mathfrak{su}(3)$, que tiene como peso máximo $(1, 1)$, que resulta ser la raíz máxima.



Para caracterizar una irrep se tienen invariantes aritméticos, tales como: la dimensión, la degeneración de cada uno de los pesos, la altura, el índice y la clase de conjugación. Para calcular la dimensión, se utiliza la **fórmula de**

[2]Corresponde a la $i-$ésima fila de la matriz de Cartan.

Weyl:

$$\text{Dim}(\Lambda) = \prod_{\alpha \in \Delta^+} \frac{(\Lambda + \delta, \alpha)}{(\delta, \alpha)}, \tag{1.57}$$

donde $\delta = (1, 1, \cdots, 1)$ en la base de Dynkin, Λ es el peso máximo de la irrep, y Δ^+ es el conjunto de raíces positivas.

Para calcular la degeneración $n_{\Lambda'}$ del peso Λ' en la irrep de Λ, usamos la **fórmula recursiva de Freudental**

$$\left[(\Lambda + \delta) \cdot (\Lambda + \delta) - (\Lambda' + \delta) \cdot (\Lambda' + \delta)\right] n_{\Lambda'} = 2 \sum_{\substack{\alpha \in \Delta^+ \\ k \in \mathbb{Z}^+}} n_{(\Lambda' + k\alpha)\cdot\alpha} (\Lambda' + k\alpha) \cdot (\alpha), \tag{1.58}$$

Por ejemplo, aplicando la ec. 1.57 a la representación adjunta de $\mathfrak{su}(3)$ se obtiene que Dim$((1, 1))$ = 8. Usando la ec. 1.58 en el peso $(0, 0)$ de la representación adjunta de $\mathfrak{su}(3)$ se obtiene 2.

Las irreps de un álgebra se clasifican en complejas y autoconjugadas. Una representación **compleja** no posee una forma bilineal invariante, mientras que una **autoconjugada** sí. Además, las autoconjugadas se clasifican en reales (ortogonales) y pseudoreales (simplécticas); una representación **real** tiene una forma bilineal simétrica invariante, mientras que una representación **pseudoreal** tiene una forma bilineal antisimétrica invariante.

Se define la **conjugación de una representación** usando las etiquetas de Dynkin como se muestra en la Tabla 1.5

$\mathfrak{g}$:	Conjugación
A_n :	$\overline{(a_1, a_2, \cdots, a_n)} = (a_n, \cdots, a_2, a_1)$.
D_{2n+1} :	$\overline{(a_1, a_2, \cdots, a_{2n-1}, a_{2n}, a_{2n+1})} = (a_1, a_2, \cdots, a_{2n-1}, a_{2n+1}, a_{2n})$.
E_6 :	$\overline{(a_1, a_2, a_3, a_4, a_5, a_6)} = (a_5, a_4, a_3, a_2, a_1, a_6)$.
Álgebras restantes	$\overline{(a_1, a_2, \cdots, a_n)} = (a_1, a_2, \cdots, a_n)$.

Tabla 1.5: Conjugaciones

La condición de **autoconjugación** de una representación es $\overline{(a_1, a_2, \cdots, a_n)} = (a_1, a_2, \cdots, a_n)$, de otro modo será compleja.

Lo que implica que todas las álgebras excepto A_n, D_{2n+1}, y E_6, solo tienen representaciones autoconjugadas. Para discernir si una representación autoconjugada es real ó pseudoreal, definimos el **nivel** de un peso en una irrep, como el número de raíces simples que se tienen que sustraer del peso máximo para llegar a dicho peso, este número es único. Se define la **altura** de una irrep $\Lambda = (a_1, a_2, \cdots, a_n)$ como el máximo nivel en la irrep, y se calcula

$$T(\Lambda) = \sum_i R_i a_i \tag{1.59}$$

donde $R_i = 2\sum (A^{-1})_{ij}$

En una representación autoconjugada los pesos que tiene el mismo nivel k, son los negativos de los pesos que están en el nivel $T(\Lambda) - k$. Por lo tanto, si Λ es autoconjugada y $T(\Lambda)$ es par, la irrep Λ será **real**; si $T(\Lambda)$ es impar, la irrep será **pseudoreal**.

Existe la posibilidad que varias irreps de un álgebra tengan la misma dimensión, un ejemplo claro son las irreps conjugadas. Se introduce un nuevo invariante llamado **índice** que se define como

$$l(\Lambda) = \frac{\text{Dim}(\Lambda)}{\text{Dim}(Adj)} C(\Lambda) \tag{1.60}$$

donde $C(\Lambda) = (\Lambda, \Lambda + 2\delta)$[3], es el eigenvalor del operador invariante de Casimir de segundo orden. Existe la posibilidad, que las irreps tengan la misma dimensión y el mismo indice, éstas deben estar relacionadas por conjugación, por lo tanto, solo difieren en su **clase de conjugación**.

Las clases de congruencia para las álgebras estudiadas se muestran en la Tabla 1.6

$\mathfrak{g}$:	Clase de conjugación
A_n :	$C_c(R) := a_1 + 2a_2 + 3a_3 + \cdots \pmod{n+1}$
D_n :	$C_{c1}(R) := a_{n-1} + a_n \pmod 2$
	$C_{c2}(R) := \begin{cases} 2a_1 + 2a_3 + \cdots + 2a_{n-2} + (n-2)a_{n-1} + na_n \pmod 4 & \text{for } n = 2k+1, \\ 2a_1 + 2a_3 + \cdots + 2a_{n-3} + (n-2)a_{n-1} + na_n \pmod 4 & \text{for } n = 2k \end{cases}$
E_6 :	$C_c(R) := a_1 - a_2 + a_4 - a_5 \pmod 3$
E_7 :	$C_c(R) := a_4 + a_6 + a_7 \pmod 2$
E_8 :	$C_c(R) := 0$

Tabla 1.6: Clases de conjugación

Con estos invariantes es posible construir un sistema de nomenclatura para las irreps. Si dos irreps tienen la misma dimensión y el mismo índice, deben estar relacionadas por conjugación y colocaremos un barra sobre la dimensión (que será el nombre de la irrep, y se escribirá en negritas) si su clase de conjugación es la mayor.

Por ejemplo, las irreps de $\mathfrak{su}_3$: $(1,0)$ y $(0,1)$ tienen una dimensión igual a 3 y un indice igual a 1, calculando la clase de conjugación para ambas irreps, se obtiene

$$C_c((1,0)) = 1,$$
$$C_c((0,1)) = 2.$$

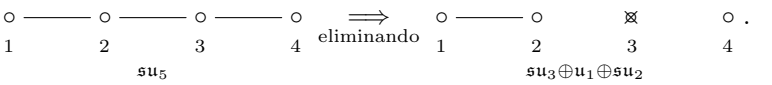
Por lo tanto, $(1,0) = \mathbf{3}$ y $(0,1) = \bar{\mathbf{3}}$.

1.5. Subálgebra maximales y reglas de rompimiento

Estamos interesados encontrar las subálgebras **maximales** (aquellas que no están contenidas en otras). Estas subálgebras caen en dos clases: regulares y especiales; las primeras a su vez se dividen en semisimples y no semisimples. En esta sección describiremos el algoritmo para encontrar estas tres clases de subálgebras.

[3] Este corresponde al producto 1.24

Una subálgebra **no semisimple** es una subálgebra semisimple multiplicada por un factor $U(1)$. Para obtener este tipo de subálgebra, se remueve un punto del diagrama de Dynkin. El resultado es dos o más diagramas de Dynkin no conexos, que simbolizan subálgebras semisimples y el punto eliminado, es decir, la raíz simple es el generador de $U(1)$. Por ejemplo, la subálgebra no semisimple $\mathfrak{su}_3 \oplus \mathfrak{su}_2 \oplus \mathfrak{u}_1$ que se puede obtener de $\mathfrak{su}_5$ por la eliminación del tercer punto del diagrama de Dynkin.



En la base de Dynkin cada componente de un peso esta asociada a una raíz simple, eliminar la $i-$ésima raíz en el diagrama corresponde a extraer el dígito asociado a dicha raíz. La carga $U(1)$ asociada a un peso λ se obtiene por el producto euclidiano de la $i-$ésima fila del tensor métrico de el álgebra original dado en la Tabla 1.4 y el peso. Existe la libertad de normalizar el resultado, en general, es un número racional, nosotros no impondremos esta normalización. Una vez hecha la ruptura para cada peso del sistema de pesos, se deben agrupar para encontrar sistemas de pesos asociados a las irreps de la subálgebra.

Por ejemplo, considerando el rompimiento $\mathfrak{su}(5) \to \mathfrak{su}(3) \oplus \mathfrak{su}(2) \oplus U(1)$, donde la matriz de Cartan y el tensor métrico de $\mathfrak{su}(5)$ son:

$$A_{ij}(A4) = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix} \tag{1.61}$$

(1.62)

$$G_{ij}(A4) = \frac{1}{5}\begin{pmatrix} 4 & 3 & 2 & 1 \\ 3 & 6 & 4 & 2 \\ 2 & 4 & 6 & 3 \\ 1 & 2 & 3 & 4 \end{pmatrix} \tag{1.63}$$

Tomando la irrep $\mathbf{5} = (1.0.0.1)$ de $\mathfrak{su}(5)$, el sistema de pesos junto con su respectiva ruptura es

$$\begin{array}{rcl} \mathfrak{su}(5) & \to & \mathfrak{su}(3) \oplus \mathfrak{su}(2) \oplus U(1) \\ (1,0,0,0) & \to & ((1,0),(0),2/5) \\ (-1,1,0,0) & \to & ((-1,1),(0),2/5) \\ (0,-1,1,0) & \to & ((0,-1),(0),2/5) \\ (0,0,-1,1) & \to & ((0,0),(1),-3/5) \\ (0,0,0,-1) & \to & ((0,0),(-1),-3/5) \end{array} \tag{1.64}$$

Se observa que las rupturas de los primeros tres pesos están asociadas a la irrep $\mathbf{3}$ de $\mathfrak{su}3$, $\mathbf{1}$ de $\mathfrak{su}(2)$ y con una carga igual a $2/5$. Mientras las últimas dos rupturas están asociadas a la irrep $\mathbf{1}$ de $\mathfrak{su}(3)$, $\mathbf{2}$ de $\mathfrak{su}(2)$ y una carga de $-3/5$. Resumiendo

$$\mathbf{5} = (\mathbf{3},\mathbf{1})_{2/5} + (\mathbf{1},\mathbf{2})_{-3/5} \tag{1.65}$$

Notemos que no hay restricción para quitar dos o más puntos del diagrama de Dynkin, lo que daría como resultado el mismo número de factores $U(1)$ que raíces eliminadas. Cada carga $U(1)$ se calcula separadamente.

Para obtener una subálgebra **semisimple**, debemos eliminar un punto del *diagrama extendido de Dynkin*. Los diagramas extendidos se construyen añadiendo el negativo de la raíz máxima α_x. El conjunto resultante de raíces

ahora es linealmente dependiente, pero si se elimina una raíz se recupera la independencia lineal y se obtiene un sistema de raíces simples valido para la subálgebra, él cual en general es semisimple.

Los diagramas extendidos de Dynkin para las álgebras están en la Tabla 1.7

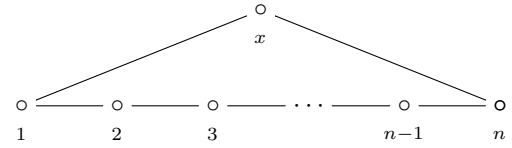
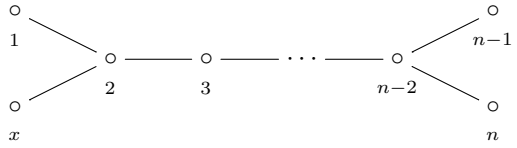
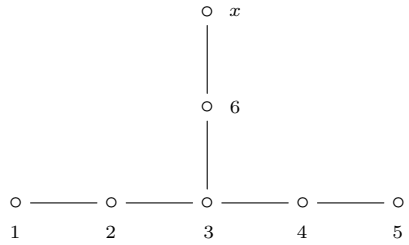
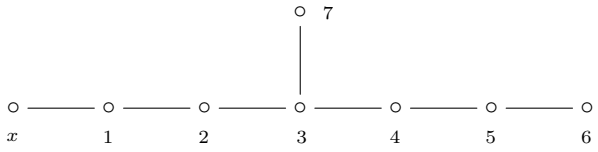
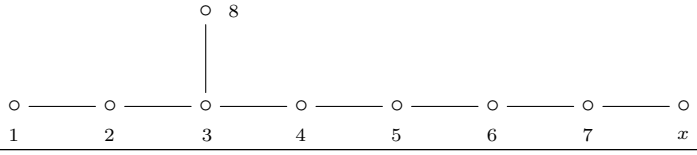
Algebra	Grupo	Rango	Diagrama extendido
A_n	$SU(n+1)$	$^{\forall}n$	
D_n	$SO(2n)$	$^{\forall}n$	
E_6	E_6	6	
E_7	E_7	7	
E_8	E_8	8	

Tabla 1.7: Diagramas extendidos de Dynkin

Para hacer las rupturas, debemos eliminar la componente de Dynkin del peso λ, asociada a la raíz eliminada y sustituirla por el producto euclidiano (α_x, λ).

Las subálgebras **especiales** no se pueden obtener del sistema de raíces. La inclusión de una subálgebra especial no sigue un patrón general, por lo que se debe hacer para cada álgebra-subálgebra.

Por lo anterior, se nota que este proceso será más complicado cuando los rompimientos sean más complejos, la dimensión de las irreps sea mayor y el álgebra sea más compleja. Como solución se creó un sorfware escrito en Python3, para hacer estos cálculos de manera automática.

Capítulo 2

El Modelo Estándar de Partículas Elementales y más allá

Durante el proceso de entendimiento de la estructura interna de la materia, se fueron modificando los bloques fundamentales de los cuales está constituida. Empezando con el concepto de átomo, como elemento indivisible y fundamental del cual está construido todo y que interacciona mediante fuerzas eléctricas y magnéticas con otros átomos. Hasta llegar al concepto de quarks y leptones que interacciones por medio de fuerzas fundamentales principalmente, denominadas electromagnéticas, nucleares débiles y nucleares fuertes.

Detrás de estas teorías se encuentra el concepto de simetría, las partículas poseen un característica denominada *grado de libertad*, que es un parámetro físico independiente que es necesario para caracterizar el estado físico de un sistema. Si los grados de libertad internos, es decir, que no dependen de las coordenadas espacio-temporales, son redundantes se produce una teoría de norma durante el proceso de cuantización [25]. Los grupos que caracterizan esta redundancia se les conoce como grupos de norma. En general, los grupos de norma son grupos de Lie como los que se estudiaron en el capítulo anterior.

El SM está formado por dos teorías de norma denominadas *Modelo Electrodébil* y *Cromodinámica Cuántica*, las cuales tienen asociadas los grupos $\mathrm{SU}(2)\times\mathrm{U}(1)$ y $\mathrm{SU}(3)$ respectivamente.

En este capítulo se dará un panorama de estos temas, junto con sus problemas y posibles soluciones, en particular, *Supersimetría* y *Teorías de Gran Unificación*.

2.1. Campos y simetrías

En la actualidad, los elementos básicos para la construcción de teorías de física fundamental son: simetrías y campos. Una simetría de un sistema es la cualidad de ser invariante ante un transformación. Estas transformaciones pueden ser continuas (como una rotación) o discretas (como una reflexión), y generalmente, tienen una estructura de grupo. Si las transformaciones no dependen de cada punto del espacio-tiempo son globales; en el caso contrario, son locales.

Las **simetrías** que tienen mayor relevancia son:

- **Simetría de Poincaré**: está relacionada con las transformaciones de coordenadas del espacio-tiempo y su efecto en funciones de dichas coordenadas. Estas transformaciones pueden ser: rotaciones en el espacio tridimensional, *boosts* y traslaciones en el espacio-tiempo. Las dos primeras forman un grupo de Lie, lla-

mado **grupo de Lorentz** que junto al grupo de traslaciones constituyen el grupo completo de isometrías del espacio-tiempo de Minkowski llamado **grupo de Poincaré**. Esta simetría es global.

- **Simetría interna**: está asociada a las transformaciones de los grados de libertad redundantes de un sistema que no modifican ninguna propiedad física observable. Estas transformaciones pueden ser locales o globales. Si la simetría es local, le diremos **simetría de norma** y las transformaciones (de norma) forma un grupo de Lie.
- **Simetría quiral**: está vinculada con las transformaciones de las proyecciones quirales de los campos fermiónicos bajo un grupo de simetría que puede ser local o global. Esta simetría es global.
- **Supersimetría**: esta simetría relaciona a cada partícula con otra llamada *supercompañera*, cuyos espines difieren por $1/2\hbar$.

La construcción de un teoría implica crear un lagrangiano que describa la dinámica de los grados de libertad del sistema. Estos grados de libertad están descritos por campos, que son los bloques sobre los cuales se edifica la teoría.

El concepto de **campo** es la generalización de coordenada generalizada de la mecánica lagrangiana, y es el elemento crucial de la denominada *Teoría Cuántica de Campos*, que es la unión de la mecánica cuántica con la teoría de la relatividad especial [22]. Esta es la herramienta adecuada para describir fenómenos a distancias pequeñas y grandes velocidades. En este nuevo paradigma, el concepto de partícula es sustituido por el de campo, que es una función del espacio-tiempo, cuyas perturbaciones dan origen a lo que se conoce como partícula.

Debido que los campos son funciones del espacio-tiempo, se ven afectados por transformaciones de Lorentz. Dependiendo del efecto de dicha transformación existen diferentes tipos de campos:

- Escalares, $s = 0$,
- Vectoriales, $s = 1$,
- Espinoriales, $s = 1/2$,
- Tensoriales, $s \geq 2$,

donde s se refiere al valor del espín del campo.

En general, un campo es **bosónico** si tiene espín entero, y **fermiónico** si tiene espín semientero. El espín de las partículas es igual al espín del campo que las crea. Los campos fermiónicos están relacionados con los fermiones, que son las partículas que forman la materia. Mientras que las interacciones son modeladas por el intercambio de partículas bosónicas.

Una **teoría de norma** es una teoría cuántica de campos que tiene asociado un lagrangiano que es invariante bajo una transformación de norma [24]. Las transformaciones de norma forman un grupo de Lie. Entonces una transformación infinitesimal estará completamente definida por los generadores en el álgebra de Lie correspondiente. Para construir un lagrangiano invariante de norma, debemos partir de un lagrangiano libre e invariante ante una simetría global (interna). Los parámetros de la simetría global pasan de ser constantes del espacio-tiempo a ser funciones del mismo. Esta acción debe ir acompañada de la inclusión de campos adicionales, llamados **campos de norma** [1]. Por cada generador en el álgebra de Lie, se debe incluir un campo de norma. Debido a que convertimos los parámetros en funciones del espacio-tiempo, aparecerán términos de interacción entre los campos de norma y campos originales del lagrangiano libre. Con estos términos se define una derivada covariante, que sustituirá a la derivada del espacio-tiempo en el lagrangiano original. Además, debemos incluir los términos cinéticos de los

[1] En general, los campos de norma son campos vectoriales

campos de norma que sean invariantes ante la simetría de norma. Con esto obtendremos un lagrangiano invariante de norma. Al procedimiento anterior se le conoce como *principio de invarianza de norma*.

Si un campo interacciona a través de un campo de norma, se dirá que está "cargado" ante el grupo de norma. Las cargas estarán determinadas por las constantes que acompañan a los términos de interacción en el lagrangiano. El principio de invarianza de norma, además de dar los términos de interacción invariantes, determina como los campos deben transforma en alguna de las irreps del grupo de norma. Sin embargo, la elección de la *irrep* sólo estará determinada por los hechos experimentales.

Para ejemplificar el principio de invarianza de norma, consideremos el caso electromagnético, donde el grupo de norma es $U(1)$. El lagrangiano original es:

$$\mathcal{L}_0 = \bar{\psi}(x)(i\not{\partial} - m)\psi(x) \tag{2.1}$$

donde $\psi(x)$ y su adjunto son espinores de Dirac, $\not{\partial} = \gamma^\mu \partial_\mu$ y γ^μ son las matrices de Dirac. Usando el principio de invarianza llegamos a las siguientes transformaciones de norma,

$$\psi \to e^{-i\beta}\psi \Rightarrow \psi \to e^{-i\beta(x)}\psi \quad \text{y} \quad A_\mu \to A_\mu - \frac{1}{e}\partial_\mu \beta, \tag{2.2}$$

donde A_μ es el campo de norma y e es la *carga eléctrica* del campo $\psi(x)$.

Ahora se define la derivada covariante, que es invariante antes la transformación de norma

$$D_\mu \psi = (\partial_\mu + ieA_\mu)\psi \to e^{-i\beta(x)} D_\mu \psi. \tag{2.3}$$

Se debe agregar el término cinético del campo de norma invariante,

$$\mathcal{L} = \bar{\psi} \not{D} \psi - \frac{1}{4} F_{\mu\nu} F^{\mu\nu} \tag{2.4}$$

Las partículas que forman la materia son fermiones, entonces deben ser representadas por campos fermiónicos. Estos campos son descritos por espinores de Dirac, los cuales se pueden descomponer en sus proyecciones izquierdas y derechas. A esta cualidad se le conoce como **quiralidad**. Estas proyecciones quirales son grados de libertad independientes, $\psi = \psi_R + \psi_L$, por lo tanto, pueden tener diferentes formas de transformación ante un grupo de simetría global o local. Si las proyecciones izquierdas y derechas transforman de diferente manera, la transformación es quiral; de otra manera, es no quiral también llamada vectorial.

Si la transformación de norma es quiral, se tiene una *teoría de norma quiral*. Un ejemplo es la teoría electrodébil, en ésta los bosones de norma solamente están acoplando a los espinores izquierdos. Las teorías quirales están vinculadas a la violación de paridad (simetría discreta, que cambia el signo de las coordenadas espaciales, físicamente implica un cambio de sistema de coordenadas derecho a izquierdo). Además, la simetría quiral prohíbe un término de masa en el lagrangiano para los fermiones ya que este término no es invariante ante las transformaciones de norma, por lo tanto, las simetrías quirales involucran a fermiones no masivos.

2.2. Ruptura de simetría

Las simetrías de las ecuaciones de movimiento pueden ser rotas *explícitamente* por la inclusión de términos en el lagrangiano que no son invariantes o *espontáneamente* en las soluciones.

Para ilustrar este fenómeno, considérese el caso del campo escalar complejo

$$\mathcal{L}_0 = \partial^\mu \varphi^\dagger \partial_\mu \varphi - V(\varphi), \quad V(\varphi) = \mu^2 \varphi^\dagger \varphi + \lambda(\varphi^\dagger \varphi)^2, \tag{2.5}$$

donde $\lambda > 0$ para que $V(\varphi)$ este acotada inferiormente. Este lagrangiano es invariante ante una transformación $\varphi \to e^{i\beta}\varphi$, con una carga conservada que corresponde al número de partículas. Es conveniente hacer la descomposición en campos hermíticos de la forma $\varphi = (\varphi_1 + i\varphi_2)/\sqrt{2}$, el lagrangiano adquiere la forma

$$\mathcal{L}_0 = \frac{1}{2}\left[(\partial_\mu\varphi_1)^2 + (\partial_\mu\varphi_2)^2\right] - V(\varphi_1,\varphi_2), \quad V = \frac{\mu^2}{2}\left(\varphi_1^2+\varphi_2^2\right) + \frac{\lambda}{4}(\varphi_1^2+\varphi_2^2)^2. \tag{2.6}$$

Por lo anterior, la transformación $U(1)$ toma la forma de una rotación en un espacio bidimensional.

$$\begin{pmatrix}\varphi_1\\ \varphi_2\end{pmatrix} \to \begin{pmatrix}\cos\beta & -\operatorname{sen}\beta\\ \operatorname{sen}\beta & \cos\beta\end{pmatrix}\begin{pmatrix}\varphi_1\\ \varphi_2\end{pmatrix} \tag{2.7}$$

El vacío corresponde a la solución clásica de las ecuaciones de movimiento

$$\left(\partial_t^2 - \nabla^2\right)\varphi_a = \left[\mu^2 + \lambda(\varphi_1^2+\varphi_2^2\right]\varphi_a \tag{2.8}$$

con la energía mínima. Esto implica que una constante $\nu_a \equiv \langle\varphi_a\rangle$ con

$$\left.\frac{\partial V}{\partial\varphi_a}\right|_{\nu_1,\nu_2} = 0. \tag{2.9}$$

La condición de mínimo requiere que los eigenvalores $m_{1,2}^2$ de la matriz hessiana de $V(\varphi_1,\varphi_2)$ deben ser no negativos. Estos son interpretados como las masas cuadradas de los estados físicos cuando se expanden alrededor del mínimo.

Cuando $\mu^2 > 0$, el mínimo está en $\nu_1 = \nu_2 = 0$. Se puede cuantizar alrededor de este punto, obteniendo los estado $\varphi_{1,2}$ degenerados con la misma masa cuadrada μ^2, pero se sigue conservando el número de partículas. Por lo tanto, se tiene que la simetría no está rota, tanto en las ecuaciones de movimiento como en el estado fundamental. Esto es conocido como la realización de *Wigner-Weyl* de la simetría.

Existen dos maneras en las cuales se puede romper la simetría del lagrangiano. Una es añadiendo un término pequeño que rompa explícitamente. Por ejemplo, si $\mu^2 > 0$ y

$$\mathcal{L} = \mathcal{L}_0 - \frac{\varepsilon}{2}\varphi_2^2,$$

la simetría $U(1)$ se rompe, con estados no degenerados con masas

$$m_1^2 = \mu^2,\ m_2^2 = \mu^2 + \varepsilon.$$

que corresponden a las masas de los estados φ_1 y φ_2 respectivamente.

La otra posibilidad es un *rompimiento espontáneo de simetría*, también conocido como realización de *Nambu-Goldstone* de la simetría. [26, 27] Si $\mu^2 < 0$ el potencial toma la forma de un sombrero mexicano, que tiene un mínimo fuera del origen. La simetría rotacional de potencial conduce a que el mínimo esté degenerado en un círculo

$$\varphi_1^2 + \varphi_2^2 = \nu^2 \equiv \frac{-\mu^2}{\lambda} > 0. \tag{2.10}$$

Por lo tanto, la simetría y la carga original del campo clásico se han perdido.

Otra consecuencia de la ruptura es la existencia de bosones no masivos. El teorema de Nambu-Goldstone afirma que por cada generador roto espontáneamente de una simetría global continua aparecen partículas sin masa y espín

0, llamadas bosones de Goldstone. En nuestro ejemplo, eligiendo sin perdida de generalidad la dirección de los ejes, de tal manera que $\nu_1 = \nu$ y $\nu_2 = 0$. Entonces,

$$\varphi_1 = \nu + \varphi_1', \quad \varphi_2 = \varphi_2' \tag{2.11}$$

donde $\langle \varphi_i' \rangle = 0$. Entonces se puede cuantizar φ_i' de la manera usual. En términos de φ_i',

$$\begin{aligned} \mathcal{L} &= \frac{1}{2} \sum_{i=1}^{2} \left(\partial_\mu \varphi_i' \right)^2 - V(\varphi_1', \varphi_2') \\ V &= \frac{-\mu^4}{4\lambda} - \mu^2 \varphi_1'^2 + \lambda \nu \varphi_1' \left(\varphi_1^2 + \varphi_2'^2 \right) + \frac{\lambda}{4} \left(\varphi_1'^2 + \varphi_2'^2 \right)^2 . \end{aligned} \tag{2.12}$$

Se nota que el primer término es constante y en principio es irrelevante [2], el segundo término implica que φ_1' tiene una masa al cuadrado igual $m_1^2 = -2\mu^2 > 0$ y los otros términos inducen interacciones. Además, no aparece un término de masa para φ_2', es decir, éste es un bosón de Goldstone. La interpretación es que el potencial es plano cuando uno se mueve del mínimo en la dirección de φ_2', de tal manera que las excitaciones del estado de vacío no masivas se convierte en oscilaciones masivas.

Es posible combinar rompimientos espontáneos con explícitos, el resultado será que los bosones de Goldstone adquieren masa del rompimiento explícito.

En la Tabla 2.1 se da un resumen de lo anterior.

Simetría exacta del lagrangiano ($[U_G, \mathcal{L}] = 0$)	
$U_G \left\|0\right\rangle = \left\|0\right\rangle$	$U_G \left\|0\right\rangle \neq \left\|0\right\rangle$
Simetría exacta	Rompimiento espontáneo de simetría
(Wigner-Weyl)	(Nambu-Goldstone)
Multiplete degenerado	caso quiral: fermiones adquieren masa
cargas conservadas	caso global: bosones de Goldstone
caso quiral: fermiones no masivos	caso de norma: bosones adquieren masas
caso de norma: bosones de norma no masivos	por Higgs o mecanismo dinámico
Rompimiento explicito ($[U_G, \mathcal{L}] \neq 0$) (solo global)	
separación de multipletes	separación de multipletes
caso quiral: fermiones adquieren masa	bosones de Goldstone con masa

Tabla 2.1: Tipos de ruptura de simetrías

Mecanismo de Higgs

Se sabe que en las teorías de norma no están permitidos términos de masa para los bosones de norma, porque estos romperían la invarianza de norma y destruyen la renormalización. Eso parece problemático para las interacciones débiles, las cuales son de corto alcance y requieren mediadores masivos. Otro problema para teorías con rompimiento espontáneo de simetría es la aparición de bosones de Goldstone, que no son observados en las interacciones de la naturaleza.

[2] Este término toma relevancia cuando consideremos a la gravedad, ya que viene a ser la contribución a la constante cosmológica

Afortunadamente, si la simetría rota es de norma, los dos problemas de los bosones de Goldstone no deseados y de los bosones de norma no masivos, se resuelven entre si, cuando la simetría es rota espontáneamente. Una manera simple de hacerlo es a través del mecanismo de Higgs, que involucra campos de espín 0. En lugar de existir partículas de espín 0, los grados de libertad asociados a los bosones de Goldstone aparecen como las componentes longitudinales un bosón de norma, que por este proceso adquiere masa. Es decir, los bosones de Goldstone son absorbidos por los campos de norma. Además, el mecanismo de Higgs preserva la renormalización de la teoría.

Supersimetría

Existen simetrías del espacio-tiempo y simetrías de norma. Surge la pregunta: ¿es posible unirlas en una sola álgebra de Lie? La respuesta la dieron Coleman y Mandula [28], diciendo que no es posible unirlas de manera no trivial. Sin embargo, existe una alternativa: cambiar el álgebra de Lie por un álgebra gradada que, además de incluir a los generadores del grupo de Poincaré y el grupo de norma, incluye a nuevos elementos llamados supercargas (son espinores de Weyl, cuya acción sobre los campos modifica el espín de éstos por $\hbar/3$). Este tipo de álgebra, además de tener relaciones de conmutación entre los generadores, tiene relaciones de anticonmutación dependiendo del carácter fermiónico o bosónico de los generadores involucrados.

Como resultado de la implementación de estas ideas, surge una simetría entre campos de espín determinado con campos que difieren en su espín por $\hbar/2$. Si la supersimetría no está rota, los supercompañeros deberían tener los mismos números cuánticos incluyendo la masa; sin embargo, el selectrón que es el supercompañero bosónico del electrón no ha sido observado a la escala de energía que el electrón. Lo anterior indica que si la supersimetría existe, debe estar rota para que los supercompañeros adquieran masas aún no exploradas experimentalmente.

2.3. El Modelo Estándar

Los experimentos en aceleradores de partículas han sondeando diversas escalas de energía, buscando evidencias de las interacciones fundamentales de la materia. Actualmente, no existen indicios que contradigan que los constituyentes fundamentales de la materia son los llamados **quarks** y **leptones**. Existen diferentes variedades de quarks: u, d, c, s, t y b; y los leptones existen en varios tipos: e, μ, τ, ν_e, ν_μ y ν_τ. Por otro lado, se sabe que las interacciones fundamentales presentes en la naturaleza son: electromagnéticas, nucleares fuertes, nucleares débiles y gravitacionales. Los leptones no participan en la interacción nuclear fuerte, y los neutrinos son eléctricamente neutros. Además, la interacción nuclear débil tiene la propiedad de violar paridad; es decir, las proyecciones quirales de las partículas no participan de la misma forma en la interacción débil.

Todas las interacciones son el resultado del intercambio de bosones de norma. Estas partículas son introducidas por el *principio de invarianza de norma*. El fotón es el responsable de la interacción electromagnética que se basa en una simetría $U(1)$, esta partícula es no masiva y por lo tanto, la interacción es de largo alcance. Por otro lado, la interacción débil es de corto alcance y esta mediada por el intercambio de partículas masivas, los bosones $W^{\pm}$ y el bosón Z^0. Finalmente, la interacción fuerte es transmitida por partículas no masivas llamadas *gluones*. Estos gluones son los bosones de norma de grupo de "color" $\mathrm{SU}(()3)$ y existen 8 campos de este tipo. Además, debido a que $\mathrm{SU}(3)$ es no abeliano, los campos de norma autointeractuán entre si, es decir, están cargados bajo el grupo de color, a diferencia del caso electromagnético donde el fotón no tiene carga eléctrica. Además, los quarks nunca han sido vistos de manera aislada, están *confinados* en hadrones o mesones. Sin embargo, a alta energías y por periodos cortos de tiempo existen evidencias de su existencia.

El SM está basado en el grupo de norma $\mathrm{SU}(3)_C \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y$. El grupo $\mathrm{SU}(3)_C$ está asociado la interacción fuerte, en lo que se conoce como *Cromodinámica Cuántica* (QCD, por sus siglas en ingles). Esta teoría es no quiral, tiene un acoplamiento g_s y 8 campos de norma que actúan en los indices de color de las proyecciones

izquierdas y derechas de los quarks. Está teoría no está rota espontáneamente, por lo tanto, los gluones permanecen no masivos.

En contraste a la QCD, la *Teoría Electrodébil* cuyo grupo de norma es $\mathrm{SU}(2)_L \times \mathrm{U}(1)_Y$ es quiral. El grupo $\mathrm{SU}(2)_L$ tiene un acoplamiento de norma g, y bosones de norma W^i, $i = 1, 2, 3$ y actúan sobre las proyecciones izquierdas de los fermiones en los índices de sabor. El factor abeliano $\mathrm{U}(1)_Y$ tiene un acoplamiento g' y un bosón de norma B. Este también es quiral, ya que actúa sobre los fermiones L y R, pero con diferentes cargas. Después del rompimiento espontáneo de simetría, el grupo $\mathrm{SU}(2)_L i \times \mathrm{U}(1)_Y$ es roto a un solo grupo $\mathrm{U}(1)_Q$, que resulta ser el grupo de la *Electrodinámica Cuántica* con el fotón como una combinación lineal de W^3 y B. La combinación ortogonal Z al fotón, junto con los bosones $W^{\pm}$ adquieren masa.

$$\mathrm{SU}(3)_C \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y \to \mathrm{SU}(3)_c \times \mathrm{U}(1)_Q.$$

Todos los campos excepto ν_R poseen un hipercarga débil Y, la cual se define por

$$Y = Q - T_L^3 \tag{2.13}$$

donde T_L^3 es el tercer generador de $\mathrm{SU}(2)_L$ y Q es la carga eléctrica.

Naturaleza dicta las irreps de los grupos de norma en las cuales deben de transformar los campos fermiónicos bajo los grupos de norma. De esta manera, los quarks son tripletes de color, y los leptones son singletes de color. En particular, las proyecciones izquierdas de los quarks u y d, transforman como un doblete de $\mathrm{SU}(2)_L$ $q_L = [u_L, d_L]^T$ con hipercarga $1/6$. Mientras que las proyecciones derechas u_R y d_R transforman como singletes bajo $\mathrm{SU}(2)_L$ y tienen hipercargas $-2/3$ y $1/3$, respectivamente.

En el caso de los leptones, las proyecciones izquierdas del e y ν^e transforman como un doblete de $SU(2)_L$ $l^T = [\nu_L^e, e_L]^T$ con hipercarga igual a $-1/2$. Las proyecciones derechas son singletes bajo $\mathrm{SU}(2)_L$ con hipercargas 0 y 1, respectivamente.

En la Tabla 2.2 se da un resumen de estos resultados.

En el SM, todos los fermiones son espinores de Dirac izquierdos (simetría quiral), e inicialmente no poseen masa (ya que un término de masa en el lagrangiano rompería la simetría quiral). Para dar masa a los quarks, se introducen interacciones con el campo de Higgs, que están determinadas por los acoplamientos de Yukawa.

2.4. Problemas del Modelos Estándar

El SM ha sido verificado por muchos experimentos y con una altísima precisión; sin embargo, el modelo tiene varios problemas que nos indican que este no puede ser la teoría final. Estos pueden ser de carácter experimental o teórico. Nosotros nos centraremos en los teóricos.

Gravedad. El SM solo describe tres de las cuatro interacciones fundamentales del Universo. La teoría de la *Relatividad General* es la que describe a la gravedad, pero desde un punto de vista clásico. Por lo cual, no se tiene aún una descripción de los fenómenos gravitacionales, a una escala de energía del orden de la energía de Planck, en la que suponemos que fenómenos cuánticos-gravitacionales serían inevitables.

La unificación de los acoplamientos de norma. Usando las ecuaciones del grupo de renormalización en el SM, para la evolución de los tres acoplamientos, las curvas tienden a coincidir en una escala energética M_{GUT} entre $\sim 10^{15}$ y $\sim 10^{16}$ GeV. [3]. Esto indica la posible existencia de una simetría de norma más grande a altas

[3]Esta situación mejora al considerar SUSY, bajo la cual las tres constantes de acoplamiento coinciden (en los márgenes de error existentes) en la escala 1.2×10^{16} GeVa

		Familias					
		Primera	Segunda	Tercera	$\mathrm{SU}(3)_C$	$\mathrm{SU}(2)_L$	$\mathrm{U}(1)_Y$
Fermiones s = 1/2	Leptones	(ν_L^e, e_L)	(ν_L^μ, μ_L)	(ν_L^τ, τ_L)	**1**	**2**	$-1/2$
		e_R	μ_R	τ_R	**1**	**1**	1
		ν_R^e	ν_R^μ	ν_R^τ	**1**	**1**	0
	Quarks	(u_L, d_L)	(c_L, s_L)	(t_L, b_L)	**3**	**2**	$1/6$
		u_R	c_R	t_R	$\bar{\mathbf{3}}$	**1**	$-2/3$
		d_R	s_R	b_R	$\bar{\mathbf{3}}$	**1**	$1/3$
Bosones	s=1	B_μ Hipercarga			**1**	**1**	0
		$W_\mu^{1,2,3}$ Débiles			**1**	**3**	0
		$g_\mu^{1,\ldots,8}$ Gluones			**8**	**1**	0
	s = 0	H Higgs			**1**	**2**	$-1/2$

Tabla 2.2: Partículas elementales y bosones de norma del Modelo Estándar, con sus respectivas cargas ante los grupos de norma

energías, donde los acoplamientos se unifican en uno solo. Usando esta idea se buscan crear **Teoría de Gran Unificación**, y obtener el grupo de norma del SM a través de rupturas de simetrías de norma de un grupo de Lie de mayor rango.

El problema de jerarquía. La masa de los bosones W's es obtenida experimentalmente con un valor ~ 80 GeV. Esta masa es obtenida a través del mecanismo de Higgs, entonces m_W es proporcional a la masa del bosón de Higgs. Pero esta recibe correcciones radiativas que son cuadráticamente divergentes. Estas divergencias se deben cancelar "a mano" en todos los órdenes en la teoría de perturbaciones, por un proceso de *ajuste fino*. Tal cancelación es un "milagro" y da lugar a cuestionarse si existen otras razones para la estabilidad electrodébil. Soluciones a este problema son dadas por SUSY y **Teorías de Dimensiones Extras**.

Número de parámetros. El SM contiene 19 parámetros libres, 9 masas para los fermiones (en el SM, los neutrinos no tienen masa), 3 acoplamientos de norma, 4 ángulos de mezcla de la matriz CKM, el vev y el acoplamiento cuadrático del potencial de Higgs, y la θ_{QCD}. Todos estos parámetros son fijados por los experimentos. No existe una relación u origen dado por la teoría. Sin embargo, existen correlaciones entre estos que se podría explicar por la introducción de simetrías adicionales, lo que reduciría el número de

parámetros [4].

Para resolver estos problemas existen dos métodos:

- "Bottom-Up". Se usa al SM como una teoría efectiva y se buscan extensiones para completar la teoría a niveles ultravioletas.
- "Top-Down". En este método se supone una simetría de norma más grande, que por rompimientos espontáneos de simetría se recupera el grupo de norma del SM. Este método tiene la ventaja de ser más sistemático y genera GUT con o sin SUSY. Sin embargo, este método no solo incluye a las GUT's sino a todas las teorías que parten de principios más fundamentales y obtienen al SM en un aproximación.

2.5. Teorías de Gran Unificación

El problema de la unificación de los acoplamientos de norma, sugiere la posibilidad de una *gran unificación* [4, 11] Las *Teorías de Gran Unificación* (GUT's por sus siglas en ingles) buscan unificar interacciones en un solo grupo de norma, el cual puede ser simple o semisimple. Las GUT's son la generalización de **Teoría de Campo Unificado**.

Históricamente, el primer intento de campo unificado lo hizo Einsten buscando unir la gravedad con el electromagnetismo, pero fracasó. Previamente Maxwell había logrado unir la electricidad con el magnetismo creado la Teoría Electromagnética. Tiempo después Glashow, Salam y Weinberg lograron unificar al electromagnetismo y la interacción débil en lo que se denomina **Teoría Electrodébil**. El siguiente paso es unificar la cromodinámica con la teoría electrodébil, en una GUT. El último paso sería unir GUT a la Teoría de la Relatividad General para obtener un **Teoría de Todo**.

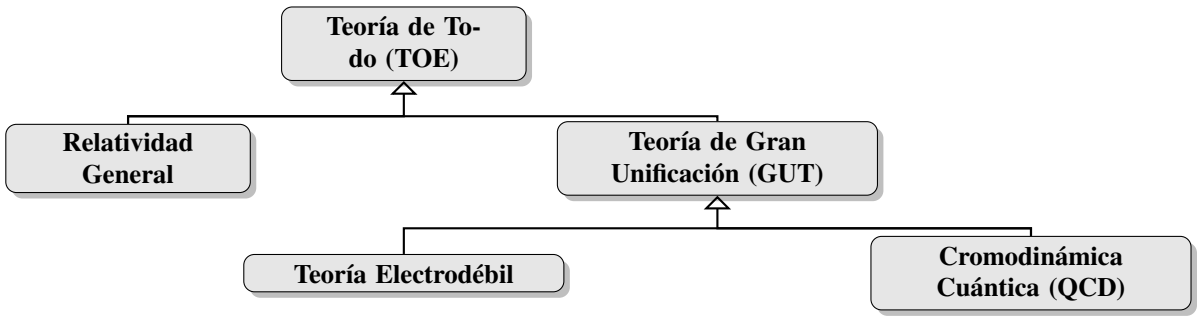


Figura 2.1: Proceso de unificación de interacciones

El primer intento de una GUT fue hecho por Pati y Salam (PS) [4], que proponían que el número leptónico fuera un cuarto color. Con esto se agranda el grupo de color de $\mathrm{SU}(3) \to \mathrm{SU}(4)$. Ellos mostraron que una familia de quarks y leptones podrían ser introducidos en dos irreps del grupo simétrico izquierdo-derecho $\mathrm{SU}(4) \times \mathrm{SU}(2)_L \times \mathrm{SU}(2)_R$ con una hipercarga débil dada por $Y/2 = 1/2(B-L) + T_{3R}$ [5]. Por lo tanto, en el modelo de PS la carga eléctrica está cuantizada.

[4] Lo ideal, sería tener solo un parámetro libre, como en la Teoría de Cuerdas.

[5] Por supuesto, la simetría derecha-izquierda requiere la introducción de los neutrinos derechos

Poco después de ser propuesto el modelo de PS, se encontró que el grupo $\mathrm{SO}(10)$ contenía a al grupo del modelo de PS como un subgrupo y unificaba una familia de quarks y leptones en la irrep **16** de $\mathrm{SO}(10)$.

Las GUT's tienen varias predicciones, unas de ellas son la unificación de los acoplamientos y el decaimiento rápido del protón. Los primeros experimentos de aceleradores de partículas excluyeron GUT's sin SUSY. Por otro lado, las GUT's supersimétricas (requieren supercompañeros para todas las partículas del SM con masas del orden de la escala electrodébil) fueron más favorecidas por los datos experimentales y a la vez suprimían el decaimiento del protón.

El espectro de la mínima teoría supersimétrica incluye a todas a las partículas del SM más sus supercompañeros. También incluye un (o más) par de dobletes de Higgs. Uno para dar masa a los quarks u, c, t y neutrinos; y otro para los quarks tipo d, b, s y leptones cargados. Los dos dobletes tienen hipercarga Y opuesta. También son necesarios para cancelar las anomalías que surgen por los nuevos acoplamientos. Finalmente, es importante reconocer que a energías por debajo de la escala de ruptura de SUSY (escala en la cual los supercompañeros de las partículas del SM obtienen masa) es necesario resolver el problema de jerarquía.

Las extensiones supersimétrica del SM tienen la propiedad de que sus efectos se desacoplan cuando la escala efectiva de ruptura de SUSY es incrementada. Cualquier teoría más allá del SM debe tener esta propiedad simplemente por que el SM trabaja perfectamente. Sin embargo, la escala de ruptura de SUSY no puede ser aumentada indiscriminadamente ya que podría volver a tener el problema de jerarquía.

Las GUT's supersimétricas dan varias explicaciones y predicciones, tales como:

- La condición $M_Z \ll M_{GUT}$ es natural.
- Existe una explicación para la cuantización de la carga.
- La predicción de la unificación de los acoplamientos.
- La predicción de la detección de supercompañeros en el LHC.
- El decaimiento no tan rápido que en ausencia de SUSY del protón.
- Predicción de la unificación de los acoplamientos de Yukawa.
- Candidatos a materia oscura supersimétricos y ligeros.
- Bariogénesis por leptogénesis.

Una GUT está formada por:

1. **Grupo de norma:** éste resulta ser un grupo de Lie compacto y la mayoría de veces simple.
2. **Una lagrangiano invariante de norma:** lo que implica introducir campo de norma, en una derivada covariante, con lo que se construye una teoría de Yang-Mills. Estos campos de norma deben estar en la representación adjunta del grupo de norma.
3. **Sector de Higgs:** que está formado por campos escalares agrupado en una irrep real o compleja del grupo de norma.
4. **Campos fermiónicos:** estos deben ser espinores de Weyl, representa a la materia y deben estar en un irrep compleja.

Muchos de los posibles rompimientos nos conduce a grupos de Lie no semisimples. Que genera una extensión natural del SM con un nuevo bosón Z' asociado a otra simetría $U(1)$

$$G \to SU(3)_C \times SU(2)_L \times U(1)_Y \times U'(1) \times \text{más grupos}$$

Modelos $\mathrm{U}(1) \times \mathrm{U}'(1)$

Muchas de las extensiones del SM involucran factores $\mathrm{U}(1)$'y bosones de norma Z' [29–35]. Estos incluyen teorías de supercuerdas, teorías de gran unificación, y muchos modelos que involucran una nueva escala física, tales como rompimiento dinámico de simetría y modelos *little Higgs*. Los bosones de norma extras podría ser extremadamente pesados, no masivos o muy ligeros o cualquier punto intermedio.

Hay un número enorme de posibles modelos, que se distinguen por la masa de Z', los acoplamientos quirales para los quarks y leptones y si son una familia universal, el sector extendido de Higgs, posibles campos exóticos que pueden ser necesarios para evitar anomalías del sector extendido de norma, posibles acoplamientos a un sector oculto, posibles mezclas cinéticas, etc.

Capítulo 3

DYNKIN

En la construcción de modelos "top-down" se busca partir de principios básicos y recobrar el SM. Una perspectiva son las GUT's cuya idea principal es que el grupo de norma de SM, está embebido en el grupo de norma de la GUT, y a través de rompimientos de simetría, se recupera el grupo de SM junto con factores $\mathrm{U}(1)$ adicionales. Para hacer este proceso de una manera automática se creó un programa informativo escrito en `Python3` llamado `dynkin`. El cual se reconstruyeron varios de los modelos encontrados en la literatura, en particular los relacionados con $\mathrm{SU}(5)$ y E_6.

3.1. Resumen del programa

Nombre del programa: `dynkin`
Obtener el programa de: `http://www.github.com/Dynkin/`
Formato de distribución: zip
Sistemas operativos: Probado Linux (Ubuntu 18, Debian 9, ElementaryOS 0.3)
Dependencias: `Sympy`
Tiempo de procesamiento: Algunos minutos dependiendo del rompimiento.

Para hacer los cálculos se implementaron todos los algoritmos, ecuaciones y definiciones del capítulo 1, en un software llamado `dynkin`. Este software está escrito en `Python3`. Se eligió este lenguaje como base debido a su portabilidad, legibilidad y el gran número de bibliotecas y herramientas con las que ya cuenta.

Se creó una clase llamada *álgebra* en la cual se incluyen todas las características que definen un álgebra, tales como:

- Tipo: "A", "D" o "E".
- Rango: para las álgebras del tipo "A" este puede ser cualquier número natural. En el caso de álgebra tipo "D", el rango puede tomar valores naturales a partir de 4. Y, como sabemos, el rango de las álgebras excepcionales "E" sólo pueden ser 6, 7 y 8.
- El conjunto de raíces simples en la base de Dynkin. El software toma como punto de partida a este conjunto para construir todas las demás propiedades del álgebra y sus irreps.
- La raíz máxima, es el peso máximo de la representación adjunta. A partir de esta, `dynkin` encuentra el sistema completo de raíces simples.,

- Raíces positivas.
- Dimensión del álgebra.
- Matriz de Cartan.
- Tensor métrico.
- Diagrama de Dynkin.

Además, el software puede construir el sistema de pesos asociado a una irrep del álgebra. A partir del sistema de peso, se extrae información acerca del irrep. Esta información están dada por invariantes aritméticos definidos en el capítulo 1:

- Dimensión de la irrep.
- Índice.
- Clase de congruencia.
- Conjugación.

Adicionalmente, `dynkin` permite calcular productos tensoriales de irreps de una misma álgebra.

Finalmente, es posible encontrar los rompimientos de un álgebra dada. Hasta ahora, el software sólo puede hacer rompimientos maximales no semisimples. Para realizar el rompimiento, es necesario introducir la regla de rompimiento (*branching rule*, en inglés) y el peso máximo de la irrep del álgebra que se quiere romper.

3.2. Descarga e instalación

Para ejecutar `dynkin` es necesario disponer de `Python3` y la biblioteca llamada `Sympy`. La mayoría de las distribuciones GNU/Linux contienen `Python3` por defecto, o se encuentran en sus repositorios de software. Sin embargo, si no ocurre lo anterior, `Python3` y su manual de instalación también están disponibles para su descarga en el sitio oficial de este intérprete de comandos [36]. Si se desea ejecutar el software en sistemas operativos tales como MS-Windows o Mac-OS, la mejor opción es `Anaconda` que está disponible en su sitio oficial [37]. `Anaconda` cuenta con una gran documentación sobre su instalación y uso.

Nos centraremos en la implementación sobre distribuciones GNU/Linux basada en Debian.

Verifiquemos que la versión de `Python` es la correcta, para ello tecleamos en una terminal

```
$ python -V
Python 3.5.1
```

Lo cual es correcto.

Para instalar la biblioteca `Sympy`, usaremos el manejador de las bibliotecas de `Python`, `pip`, es preciso instalarlo, para lo cual empleamos el comando

```
$ sudo apt-get install python3-pip
```

Finalmente, con ayuda de `pip` es posible instalar `Sympy` mediante el comando

```
$ sudo pip3 install sympy
```

Para ejecutar `dynkin`, primero es preciso descargarlo del sitio web

www.github.com/elhacs/Dynkin

Una vez descargado el archivo en el directorio en el que deseamos realizar la ejecución, descomprimimos con los siguientes comandos:

```
$ unzip Dynkin-master.zip
$ cd Dynkin-master
```

Debido a que `Python3` es un intérprete de comandos no hay necesidad de compilar nada. Podemos ejecutar directamente el archivo **dynkin** que se encuentra en la carpeta en la que estamos actualmente.

```
$ ./dynkin
```

3.3. Como usar el programa

Inmediatamente después de ejecutar el archivo `dynkin`, nos deberá aparecer un pantalla como la siguiente

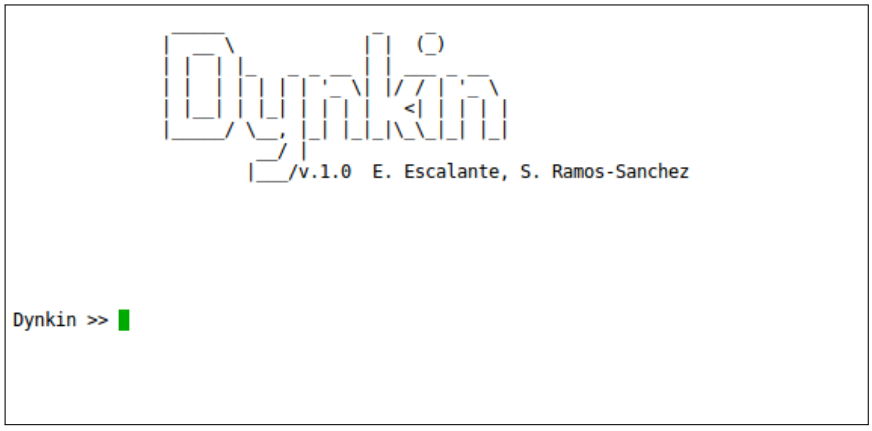


Figura 3.1: Pantalla inicial de Dynkin

3.3.1. Creando un álgebra de Lie

Debemos crear el objeto principal con el que trabaja `dynkin` que es un álgebra de Lie. `dynkin` está diseñado para trabajar inicialmente con las álgebras del tipo "A", "D" y "E". Por ejemplo, para crear un álgebra del tipo "A" y rango 4, sólo debemos teclear

```
Dynkin >> A4 = alg('A', 4)
```

Es importante notar la presencia de comillas para no generar errores en la creación del álgebra. El nombre "A4" solo es una etiqueta para el álgebra recién creada, por lo que se puede elegir cualquier otro. Sin embargo, es recomendable elegir nombres que tengan una relación mnemotécnica y evitar incluir símbolos no alfanuméricos. Es importante aclarar que `Python` distingue entre mayúsculas y minúsculas, por lo que "A4" y "a4" son diferentes en el entorno de `Python`.

Tres de las funciones más usadas de `dynkin` son:

- `p(arg)`, que se encarga de imprimir en la pantalla los resultados `arg`. Por ejemplo

```
Dynkin >> p(simples(A4))
[[ 2, -1, 0, 0]
[-1, 2, -1, 0]
[0, -1, 2, -1]
[0, 0, -1, 2]]
```

- `h` nos muestra la documentación relacionada con un comando específico.

```
Dynkin >> h(alg)
alg(tipo,rango)
---------------------

Debemos crear el objeto principal con él que trabaja \code{dynkin} que
    es
un álgebra de Lie, este software esta diseñado para trabajar
inicialmente con las álgebras tipo ``A'', ``D'' y ``E''. Por lo tanto,
para crear un álgebra tipo ``A'' y rango 4, solo debemos
teclear

    Dynkin >> A4 = alg('A', 4)
```

- `Comandos`. Su impresión aporta una lista con todos los comandos importantes de `dynkin`. Debemos recordar que `dynkin` también acepta comandos generales del entorno `Python` donde fue desarrollado.

```
Dynkin >> p(comandos)
'alg'
'complejidad'
'congruencia'
'ddynkin'
'dim'
'espectro'
'h'
'ind'
'maxima'
'mcartan'
'nombre'
'orden'
'p'
'pesos'
'positivas'
'ptensorial'
'ruptura'
'simples'
'tmetrico'
'ug'
'yug'
```

Además de las funciones anteriores, `dynkin` posee la característica de autocompletado, que se ejecuta con la tecla “TAB”.

Una vez creado el álgebra, podemos obtener diversas características. Usaremos como ejemplo el álgebra "A4" que anteriormente construimos. Podemos imprimir en pantalla el resultado de los comandos, o asignarles una etiqueta como se hizo con el comando `alg`. Optaremos por la primera opción para ejemplificar el uso de `dynkin`.

Los comandos disponibles son los siguientes:

- `simples(algebra)` nos devuelve el conjunto de raíces simples del álgebra en la base de Dynkin. Podemos asignar una etiqueta especial a este conjunto, o simplemente imprimirlo en pantalla el resultado.

```
Dynkin >> p(simples(A4))
[[ 2, -1, 0, 0]
[-1, 2, -1, 0]
[0, -1, 2, -1]
[0, 0, -1, 2]]
```

- `positivas(algebra)` nos da el conjunto de raíces positivas del álgebra.

```
Dynkin >> p(positivas(A4))
[[ 1, 0, 0, 1]
[-1, 1, 0, 1]
[1, 0, 1, -1]
[0, -1, 1, 1]
[-1, 1, 1, -1]
[1, 1, -1, 0]
[0, 0, -1, 2]
[0, -1, 2, -1]
[-1, 2, -1, 0]
[2, -1, 0, 0]]
```

- `maxima(algebra)`. Con este comando obtenemos el peso máximo de la representación adjunta del álgebra.

```
Dynkin >> p(mxima(A4))
[1, 0, 0, 1]
```

- `orden(algebra)` calcula la dimensión del álgebra.

```
Dynkin >> p(orden(A4))
24
```

- `mmcartan(algebra)` nos devuelve la matriz de Cartan definida en 1.49. Aunque el resultado se parece al que obtenemos con el comando `simples`, internamente `dynkin` trata a estos dos objetos de maneras distintas, por lo que es importante no mezclarlos.

```
Dynkin >> p(mcartan(A4))
[ 2, -1, 0, 0]
[-1, 2, -1, 0]
[0, -1, 2, -1]
[0, 0, -1, 2]
```

- `tmetrico(algebra)` determina el tensor métrico del álgebra, definido en 1.54.

```
Dynkin >> p(tmetrico(A4))
[4/5, 3/5, 2/5, 1/5]
[3/5, 6/5, 4/5, 2/5]
[2/5, 4/5, 6/5, 3/5]
[1/5, 2/5, 3/5, 4/5]
```

- `ddynkin(algebra)` imprime el diagrama de Dynkin asociado al álgebra, con las raíces simples enumeradas.

```
Dynkin >> p(ddynkin(A4))
0---0---0---0
1   2   3   4
```

3.3.2. Creando sistemas de pesos

Una vez construida el álgebra, podemos generar sus irreps conociendo el peso máximo de la representación. `dynkin` puede obtener los sistemas de pesos y calcular todos los invariantes aritméticos que definen completamente a la irrep.

El listado completo de comandos es el siguiente:

- `pesos(pesomaximo,algebra)`. Con este comando calculamos el sistema de pesos en la base de Dynkin. Es importante notar el formato del peso máximo (junto con la obvia restricción de que ninguna de sus componentes puede ser negativa). En nuestro ejemplo, podemos definir

```
Dynkin >> hwv1 = [1,0,0,0]
Dynkin >> hwv2 = [0,1,0,0]
```

que son los pesos máximos de dos irreps del álgebra "A4". Ahora calculamos el sistema de pesos para uno de los pesos máximos dados.

```
Dynkin >> p(pesos(hwv1,A4)
[[ 1, 0, 0, 0]
[-1, 1, 0, 0]
[0, -1, 1, 0]
[0, 0, -1, 1]
[0, 0, 0, -1]]
```

- `dim(pesomaximo,algebra)`. De esta manera obtenemos uno de los principales invariantes de la irrep.

```
Dynkin >> p(dim(hwv1, A4))
5
```

- `ind(pesomaximo,algebra)`. Así calculamos el índice de la irrep que tiene un papel importante en el cálculo de la función beta en física de altas energías.

```
Dynkin >> p(ind(hwv1,A4))
1
```

- `congruencia(pesomaximo,algebra)`. Este invariante nos permite distinguir irreps cuyos índices y dimensión son iguales.

```
Dynkin >> p(congruencia(hwv1,A4))
1
```

- `conjugacion(pesomaximo,algebra)`. Con este comando obtenemos el peso máximo de la irrep conjugada con respecto a las reglas dadas en 1.5

```
Dynkin >> Imp(conjugacion(hwv1,A4))
[0, 0, 0, 1]
```

- `complejidad(pesomaximo,algebra)`. Este comando nos devuelve "*" si la irrep es compleja con respecto al sistema de nombramiento definido antes, y no devuelve ningún valor cuando la irrep no es compleja, con el sistema antes mencionado

```
Dynkin >> p(complejidad(hw1,A4))

Dynkin >> p(complejidad(conjugacion(hwv1,A4),A4))
*
```

- `nombre(pesomaximo,algebra)` nos devuelve la dimensión y la complejidad de una irrep.

```
Dynkin >> p(nombre(hwv1,A4))
5
```

- `ptensorial(pesomaximo1,pesomaximo2,algebra)`, mediante este comando, `dynkin` calcula productos tensoriales de irreps de un álgebra.

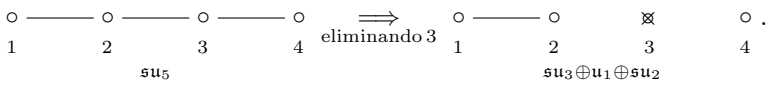
```
Dynkin >> p(ptensorial(hwv1,hwv2,A4))
5 x 10 = 40* + 10*
```

Esta salida de `dynkin` en notación usual de álgebras de Lie corresponde a $\mathbf{5} \otimes \mathbf{10} = \overline{\mathbf{40}} \oplus \overline{\mathbf{10}}$.

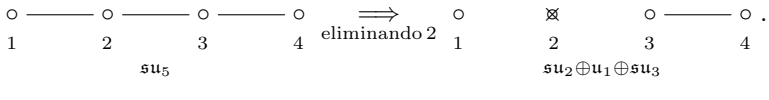
3.3.3. Ruptura de representaciones

Una vez creada una irrep de un álgebra, `dynkin` es capaz de romper una irrep a irreps de una subálgebra no semisimple. Primero debemos saber el formato de la regla de rompimiento para que `dynkin` pueda trabajar con ella. Consideremos el álgebra que hemos usado en los ejemplos, es decir, $A_4 = \mathfrak{su}_5$. Estamos interesados en el rompimiento a la subálgebra $\mathfrak{su}_3 \times \mathfrak{su}_2 \times \mathfrak{u}_1$. Recordando el algoritmo detallado en el capítulo 1, existen dos maneras de llegar a esta subálgebra.

Veamos los diagramas de Dynkin correspondientes:



o



Las reglas de rompimiento asociadas a cada ruptura en el entorno de `dynkin` son:

```
Dynkin >> regla1 = {'A2 A1 U1':[[1,2],[4],[3]]}
Dynkin >> regla2 = {'A2 A1 U1':[[3,4],[1],[2]]}
```

para el primer y segundo rompimiento, respectivamente. Es importante notar el formato de la regla de rompimiento, para que `dynkin` pueda trabajar en el rompimiento. Los índices corresponden a las etiquetas de las raíces simples en el diagrama de Dynkin. En esta versión `dynkin` solo puede trabajar con rompimientos maximales no semisimples; en una versión futura se implementarán los rompimientos maximales semisimples.

Para analizar los rompimientos tenemos a nuestra disposición los siguientes comando:

- `ug(regla, algebra)`, nos devuelve una lista con los generadores de las simetrías $U(1)$, compatibles con la regla de rompimiento.

  ```
  Dynkin >> p(ug(regla2, A4))
  [[3/5, 6/5, 4/5, 2/5]]
  ```

- `ruptura(pesomaximo, regla, algebra)`, obtiene las rupturas de cada peso del sistema asociado al *pesomaximo*, en las irreps de la subálgebra dada por la *regla*.

  ```
  Dynkin >> p(ruptura(hwv1, regla2, A4))
  5 = (1 ,2 )_3/5 + (3 ,1 )_-2/5
  Dynkin >> p(ruptura(hwv2, regla2, A4))
  10 = (1 ,1 )_6/5 + (3 ,2 )_1/5 + (3*,1 )_-4/5
  ```

  que se escribe en notación convencional como $\mathbf{5} = (\mathbf{3},\mathbf{1})_{-2/5} \oplus (\mathbf{1},\mathbf{2})_{3/5}$ y $\mathbf{10} = (\overline{\mathbf{3}},\mathbf{1})_{-4/5} \oplus (\mathbf{3},\mathbf{2})_{1/5} \oplus (\mathbf{1},\mathbf{1})_{6/5}$, respectivamente,

3.3.4. Asignación de hipercarga

Hasta el momento, se pueden obtener las rupturas de cualquier irrep. Sin embargo, falta el proceso de asignación de la hipercarga correspondiente a las partículas del SM. Sabemos de la tabla 1 cual debe ser la hipercarga, este proceso lo hace `dynkin` por medio de los siguientes comandos:

- `yug(pesosSM, algebra, regla)`, calcula el conjunto de generadores Y y U(1) adicionales compatibles con la regla de ruptura. Sabemos que el conjunto no es único debido a que la asignación no es única. Por lo tanto, obtenemos una lista con los correspondientes generadores de la hipercarga y de los generadores adicionales que son ortogonales a la dirección de la hipercarga. El primer elemento de cada conjunto es el generador de hipercarga.

  ```
  Dynkin >> pesosSM = [[0,0,0,1],[0,1,0,0],[0,0,0,0]]
  Dynkin >> p(yug(pesosSM, A4, regla2))
  [[[1/2, 1, 2/3, 1/3]]]
  ```

- `espectro(pesos, algebra, regla, pesosSM)`, nos presenta un resultado similar al comando, `ruptura` con la diferencia que ya se cuenta con la asignación de la hipercarga para cada irrep. La lista *pesos* contiene a todos los pesos para los cuales se quiere conocer su rompimiento, y *pesosSM* contiene las irreps asociadas a las partículas del SM.

  ```
  Dynkin >> p(espectro(pesosSM, A4, regla2, pesosSM))
  ['5*',
  ['3*', '1 ', 1/3]
  ```

```
['1 ', '2 ', -1/2]]]
['10 ',
[['1 ', '1 ', 1]
['3 ', '2 ', 1/6]
['3*'. '1 '. -2/3]]]
['1 ',
[['1 ', '1 ', 0]]]]
Dynkin >> p(espectro([[1,0,0,1]], A4, regla2, pesosSM))
[['24 ',
[['3*', '2 '. 5/6]
['8 '. '1 ', 0]
['1 ', '3 ', 0]
['1 ', '1 ', 0]
['3 ', '2 ', -5/6]]]]
```

En notación usual corresponde a:

- $\overline{\mathbf{5}} = (\bar{\mathbf{3}}, \mathbf{1})_{1/3} \oplus (\mathbf{1}, \mathbf{2})_{-1/2}$,
- $\mathbf{10} = (\mathbf{1}, \mathbf{1})_1 \oplus (\mathbf{3}, \mathbf{2})_{1/6} \oplus (\bar{\mathbf{3}}, \mathbf{1}))_{-2/3}$,
- $\mathbf{1} = (\mathbf{1}, \mathbf{1})_0$,
- $\mathbf{24} = (\bar{\mathbf{3}}, \mathbf{2})_{5/6} \oplus (\mathbf{8}, \mathbf{1})_0 \oplus (\mathbf{1}, \mathbf{3})_0 \oplus (\mathbf{1}, \mathbf{1})_0 \oplus (\mathbf{3}, \mathbf{2})_{-5/6}$,

respectivamente.

En el siguiente capítulo demostraremos la utilidad del software `dynkin` en el estudio de la física de partículas. Utilizando este programa, obtendremos rompimientos del grupo excepcional E_6 que, entre otras cualidades, contengan las simetrías de norma del SM y simetrías Z' adicionales. Asimismo, exigiremos que a partir de la representación **27** de E_6 surjan las representaciones en las que se transforman los quarks y leptones del SM. El propósito de este proceso será el análisis de la viabilidad fenomenológica de las simetrías Z' identificadas con este método.

Capítulo 4

E_6 y sus simetrías $\mathrm{U}(1)'$

Muchos de los modelos que incluyen dos factores extras $\mathrm{U}(1)$ ocurren de la descomposición de la GUT E_6 [29, 38], es decir, $\mathrm{E}_6 \to \mathrm{SO}(10) \times \mathrm{U}(1)_\psi$ y $\mathrm{SO}(10) \to \mathrm{SU}(5) \times \mathrm{U}(1)_\chi$. Consideraremos solo ejemplos libres de anomalías y campos exóticos. Para el caso de E_6 los fermiones que forman la materia están en la irrep fundamental **27**, el cual se descompone como $\mathrm{E}_6 \to \mathrm{SO}(10) \to \mathrm{SU}(5)$ como

$$\mathbf{27} \to \mathbf{16} \oplus \mathbf{10} \oplus \mathbf{1} \to \left(\mathbf{10} + \bar{\mathbf{5}} + \mathbf{1}\right) + \left(\mathbf{5} + \bar{\mathbf{5}}\right) \oplus \mathbf{1}$$

Adicionalmente a los 15 fermiones cargados bajo el grupo de norma de SM y el ν^c, cada irrep **27** contienen un segundo singlete, S. Tanto ν^c y S pueden estar cargados bajo $\mathrm{U}(1)'$. Hay también un triplete de color exótico de quarks $\mathcal{D}$ con una carga $-1/3$ y el conjugado $\mathcal{D}^c$, ambos son singletes bajo $\mathrm{SU}(2)$, y un par de singletes de color y dobletes de $\mathrm{SU}(2)$, h_u y h_d. Los campos exóticos, los cuales son necesarios para cancelar anomalías, y todos son singletes o no quirales bajo el grupo de norma de SM, por lo tanto no producen grandes correcciones. Sin embargo, ellos son usualmente quirales bajo $\mathrm{U}(1)$', es decir, son *cuasi-quirales*.

Los modelos E_6 puede considerarse en ambas versiones no-supersimétricas y supersimétricas. En el caso supersimétrico, los compañeros escalares de S (o ν^c) pueden desarrollar *vev* para romper la simetría $\mathrm{U}(1)$'. Similarmente, los compañeros escalares de uno de los pares $h_{u,d}$ pueden ser interpretados como los dos dobletes de Higgs del *Minimal Supersymmetric Standard Model*. Las dos familias adicionales $h_{u,d}$ puede ser interpretadas como pares adicionales de Higgs o como leptones exóticos (h_d tiene los mismo números cuánticos de SM como un doblete ordinario de leptones, mientras h_u seria conjugado al un doblete derecho exótico)

$\mathrm{U}(1)_\psi \times \mathrm{U}(1)_\chi$ podría sobrevivir a bajas energías, aunque muchos estudios asumen únicamente a $\mathrm{U}(1)_\psi$ o $\mathrm{U}(1)_\chi$ o una combinación lineal de éstas. Ejemplos de estas combinaciones son $Q_\eta = \sqrt{\frac{3}{5}} Q_\chi - \sqrt{\frac{5}{8}} Q_\psi$, el cual ocurre en algunas compactificaciones de cuerdas heteróticas, o $Q_N = \frac{1}{4} Q_\chi + \frac{\sqrt{15}}{4} Q_\psi$, el cual permite el mecanismo de *sewsaw* para los neutrinos con la masa grande de ν^c o puede evitar las constricciones cosmológicas o astrofísicas en los neutrinos de Dirac.

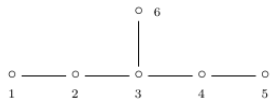
Todos los modelos supersimétricos LR y E_6 incluyen un par vectorial de supercampos quirales, lo que lleva a un versión del problema de la masa μ del Higgs, o no son consistentes con la forma simple de encontrar la unificación en el MSSM. Los modelos de unificación minimal [39] remedia esto empezando con el contenido de partículas del MSSM y entonces eligiendo los exóticos que preservan la unificación del MSSM a nivel de árbol, con cargas $\mathrm{U}(1)'$ elegidas para cancelar anomalías. Al menos dos singletes S_i con diferentes cargas $\mathrm{U}(1)'$ son necesarios para dar masa a todos los exóticos. Es posible pero no trivial asegurar que ellos adquieren *vev*, y evitar simetrías globales

accidentales y los bosones de Goldstone asociados [40]. Este trabajo se centra en los rompimientos maximales no semisimples de E_6 para obtener modelo de Z'. Para hacer esto se usará el algoritmo visto en el sección 1.5.

4.1. Álgebra E_6

El álgebra asociada tiene las siguientes características:

- Su dimensión es 78.
- El rango es 6.
- La irrep **27** es fundamental y real, y la adjunta es la **78** que también es real.
- El sistema de raíces es 6-dimensional y tiene 78 elementos todos con la misma longitud.
- El diagrama de Dynkin asociado es:



Generalizando los rompimientos no semisimples a través de eliminar más de una raíz simple a la vez, se encuentran que todas las posibles topología del diagrama de Dynkin que conducen al álgebra del SM son:

4.2. Simetrías y cargas emergentes del rompimiento de E_6

Se procede al cálculo del contenido de partículas usado `dynkin`. Para lo anterior, introducimos las cinco reglas de rompimiento en `dynkin`de la siguiente manera:

```
Dynkin >> E6 = alg('E', 6)
Dynkin >> regla1 = {'A2 A1 U1 U1 U1': [[4,5],[1],[2],[3],[6]]}
Dynkin >> regla2 = {'A2 A1 U1 U1 U1': [[4,5],[2],[1],[3],[6]]}
Dynkin >> regla3 = {'A2 A1 U1 U1 U1': [[4,5],[6],[1],[2],[3]]}
Dynkin >> regla4 = {'A2 A1 U1 U1 U1': [[3,6],[5],[1],[2],[4]]}
Dynkin >> regla5 = {'A2 A1 U1 U1 U1': [[2,3],[5],[1],[4],[6]]}
```

Como sabemos, el contenido de partículas del SM se encuentra en la irrep **27** de E_6, mientras que los bosones de norma se encuentran en la irrep **78**. Por lo tanto, se hacen los rompimientos de estas dos irrep y se buscan todas las posibles asignaciones de hipercargas para cada regla de rompimiento. Se debe recordar que las cargas adicionales $\mathrm{U}(1)$ son el resultado de dos generadores ortogonales entre si y a el generador de hipercarga. Es por esto que las cargas $\mathrm{U}(1)$ adicionales no tienen una normalización convencional. En la práctica, la cargas $\mathrm{U}(1)'$ aportadas en las siguientes tables puede renormalizarse arbitrariamente.

Para todas las reglas de rompimiento se encontraron tres asignaciones de hipercarga para las partículas del SM. A continuación se enlistan los resultados para cada una de las reglas de rompimiento antes descritas.

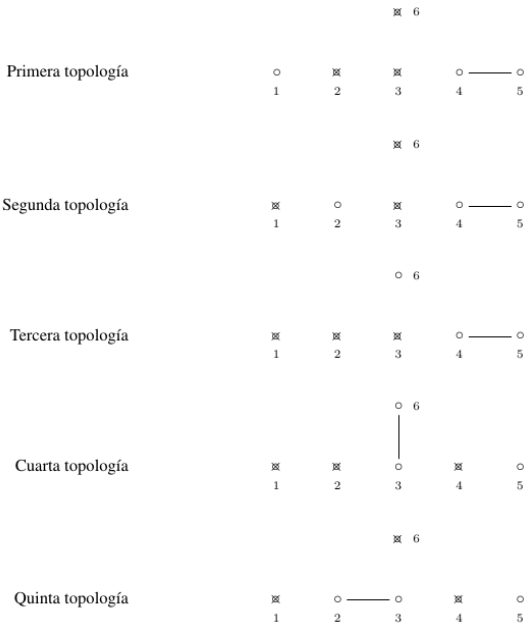


Figura 4.1: Rompimientos no semisimples de E_6

Primera regla de rompimiento

Irrep de E_6	Irrep de SM	1 $U(1)' \times U(1)'$	2 $U(1)' \times U(1)'$	3 $U(1)' \times U(1)'$
27	$(\mathbf{3},\mathbf{2})_{1/6}$	-32/61, -3/65	104/137, 99/829	86/101, -129/1189
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	6/61, 3/65	-279/137, 125/829	-243/101, -230/1189
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	-3/61, 31/65	345/137, -110/829	273/101, 185/1189
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	-3/61, -34/65	-66/137, -15/829	-30/101, 45/1189
	$(\mathbf{1},\mathbf{2})_{-1/2}$	35/61, 37/65	-38/137, -84/829	-56/101, 84/1189
	$(\mathbf{1},\mathbf{2})_{-1/2}$	35/61, -28/65	-449/137, 11/829	-359/101, -56/1189
	$(\mathbf{1},\mathbf{1})_{1}$	-70/61, -9/65	448/137, 73/829	415/101, -28/1189
	$(\mathbf{1},\mathbf{1})_{0}$	-1, 28/65	2, 213/829	2, -303/1189
	$(\mathbf{1},\mathbf{1})_{0}$	-1, -37/65	-1, 208/829	-1, -28/1189
	$(\mathbf{3},\mathbf{1})_{-1/3}$	64/61, 6/65	-208/137, -198/829	-172/101, 258/1189
	$(\mathbf{1},\mathbf{2})_{1/2}$	26/61, 0	175/137, -224/829	157/101, 359/1189
78	$(\mathbf{8},\mathbf{1})_{0}$	0, 0	0, 0	0, 0
	$(\mathbf{1},\mathbf{3})_{0}$	0, 0	0, 0	0, 0
	$(\mathbf{1},\mathbf{1})_{0}$	0, 0	0, 0	0, 0
	$(\mathbf{3},\mathbf{2})_{1/6}$	29/61, 34/65	241/137, -209/829	-116/101, 6/41
	$(\mathbf{3},\mathbf{2})_{1/6}$	29/61, -31/65	-170/137, -114/829	187/101, 314/1189
	$(\mathbf{3},\mathbf{2})_{-5/6}$	38/61, 6/65	-383/137, 26/829	-329/101, -101/1189

Irrep de E_6	Irrep de SM	$1\ U(1)' \times U(1)'$	$2\ U(1)' \times U(1)'$	$3\ U(1)' \times U(1)'$
	$(\mathbf{3},\mathbf{1})_{2/3}$	-67/61, 5/13	142/137, 183/829	445/101, -73/1189
	$(\mathbf{3},\mathbf{1})_{2/3}$	-67/61, -8/13	553/137, 88/829	142/101, -213/1189
	$(\mathbf{3},\mathbf{1})_{-1/3}$	-58/61, -3/65	-71/137, 323/829	-71/101, -488/1189
	$(\bar{\mathbf{3}},\mathbf{2})_{-1/6}$	-29/61, -34/65	170/137, 114/829	116/101, -6/41
	$(\bar{\mathbf{3}},\mathbf{2})_{-1/6}$	-29/61, 31/65	-241/137, 209/829	-187/101, -314/1189
	$(\bar{\mathbf{3}},\mathbf{2})_{5/6}$	-38/61, -6/65	383/137, -26/829	329/101, 101/1189
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	67/61, -5/13	-142/137, -183/829	-445/101, 73/1189
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	67/61, 8/13	-553/137, -88/829	-142/101, 213/1189
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	58/61, 3/65	71/137, -323/829	71/101, 488/1189
	$(\mathbf{1},\mathbf{2})_{1/2}$	96/61, 9/65	312/137, 297/829	258/101, -387/1189
	$(\mathbf{1},\mathbf{2})_{-1/2}$	-96/61, -9/65	-312/137, -297/829	-258/101, 387/1189
	$(\mathbf{1},\mathbf{1})_{1}$	-9/61, 28/65	624/137, -235/829	213/101, 275/1189
	$(\mathbf{1},\mathbf{1})_{1}$	-9/61, -37/65	213/137, -140/829	516/101, 415/1189
	$(\mathbf{1},\mathbf{1})_{0}$	0, 1	-3, -95/829	3, 140/1189
	$(\mathbf{1},\mathbf{1})_{0}$	0, -1	3, 95/829	-3, -140/1189
	$(\mathbf{1},\mathbf{1})_{-1}$	9/61, 37/65	-213/137, 140/829	-213/101, -275/1189
	$(\mathbf{1},\mathbf{1})_{-1}$	9/61, -28/65	-624/137, 235/829	-516/101, -415/1189

Segunda regla de rompimiento

Irrep de E_6	Irrep de SM	$1\ U(1)' \times U(1)'$	$2\ U(1)' \times U(1)'$	$3\ U(1)' \times U(1)'$
27	$(\mathbf{3},\mathbf{2})_{1/6}$	86/61, 126/1189	68/65, -6/53	-178/337, 12/361
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	-243/101, 197/1189	-207/65, -140/901	-42/377, 15/361
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	273/101, -236/1189	201/65, 149/901	21/377, 173/361
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	-30/101, 39/1189	6/65, -9/901	21/377, -188/361
	$(\mathbf{1},\mathbf{2})_{-1/2}$	-56/101, -165/1189	-74/65, 111/901	157/377, 176/361
	$(\mathbf{1},\mathbf{2})_{-1/2}$	-359/101, 110/1189	-269/65, -47/901	157/377, -185/361
	$(\mathbf{1},\mathbf{1})_{1}$	415/101, 55/1189	343/65, -64/901	-314/377, 9/361
	$(\mathbf{1},\mathbf{1})_{0}$	2, 213/1189	2, -195/901	-1, 212/361
	$(\mathbf{1},\mathbf{1})_{0}$	-1, 488/1189	-1, -353/901	-1, -149/361
	$(\mathbf{3},\mathbf{1})_{-1/3}$	-172/101, -252/1189	-136/65, 12/53	356/377, -24/361
	$(\mathbf{1},\mathbf{2})_{1/2}$	157/101, -323/1189	139/65, 242/901	220/377, -27/361
78	$(\mathbf{8},\mathbf{1})_{0}$	0, 0	0, 0	0, 0
	$(\mathbf{1},\mathbf{3})_{0}$	0, 0	0, 0	0, 0
	$(\mathbf{1},\mathbf{1})_{0}$	0, 0	0, 0	0, 0
	$(\mathbf{3},\mathbf{2})_{1/6}$	187/101, -362/1189	133/65, 251/901	199/377, 161/361
	$(\mathbf{3},\mathbf{2})_{1/6}$	-116/101, -3/41	-62/65, 93/901	199/377, -200/361

Irrep de E_6	Irrep de SM	$1\ U(1)' \times U(1)'$	$2\ U(1)' \times U(1)'$	$3\ U(1)' \times U(1)'$
	$(\mathbf{3},\mathbf{2})_{-5/6}$	-329/101, 71/1189	-55/13, -38/901	136/377, 3/361
	$(\mathbf{3},\mathbf{1})_{2/3}$	445/101, 16/1189	337/65, -55/901	-335/377, 197/361
	$(\mathbf{3},\mathbf{1})_{2/3}$	142/101, 291/1189	142/65, -213/901	-335/377, -164/361
	$(\mathbf{3},\mathbf{1})_{-1/3}$	-71/101, 449/1189	-71/65, -344/901	-398/377, 39/361
	$(\bar{\mathbf{3}},\mathbf{2})_{-1/6}$	116/101, 3/41	62/65, -93/901	-199/377, 200/361
	$(\bar{\mathbf{3}},\mathbf{2})_{-1/6}$	-187/101, 362/1189	-133/65, -251/901	-199/377, -161/361
	$(\bar{\mathbf{3}},\mathbf{2})_{5/6}$	329/101, -71/1189	55/13, 38/901	-136/377, -3/361
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	-142/101, -291/1189	-142/65, 213/901	335/377, 164/361
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	-445/101, -16/1189	-337/65, 55/901	335/377, -197/361
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	71/101, -449/1189	71/65, 344/901	398/377, -39/361
	$(\mathbf{1},\mathbf{2})_{1/2}$	258/101, 378/1189	204/65, -18/53	-534/377, 36/361
	$(\mathbf{1},\mathbf{2})_{-1/2}$	-258/101, -378/1189	-204/65, 18/53	534/377, -36/361
	$(\mathbf{1},\mathbf{1})_{1}$	516/101, -433/1189	408/65, 17/53	63/377, 158/361
	$(\mathbf{1},\mathbf{1})_{1}$	213/101, -158/1189	213/65, 131/901	63/377, -203/361
	$(\mathbf{1},\mathbf{1})_{0}$	3, -275/1189	3, 158/901	0, 1
	$(\mathbf{1},\mathbf{1})_{0}$	-3, 275/1189	-3, -158/901	0, -1
	$(\mathbf{1},\mathbf{1})_{-1}$	-213/101, 158/1189	-213/65, -131/901	-63/377, 203/361
	$(\mathbf{1},\mathbf{1})_{-1}$	-516/101, 433/1189	-408/65, -17/53	-63/377, -158/361

Tercera regla de rompimiento

Irrep de E_6	Irrep de SM	$1\ U(1)' \times U(1)'$	$2\ U(1)' \times U(1)'$	$3\ U(1)' \times U(1)'$
27	$(\mathbf{3},\mathbf{2})_{1/6}$	4/101, 36/557	18/137, 36/869	0, 0
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	659/303, 306/557	743/411, -84/869	5/3, 24/41
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	-178/303, -534/557	-166/411, -690/869	-1/3, -54/41
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	-481/303, 228/557	-577/411, 774/869	-4/3, 30/41
	$(\mathbf{1},\mathbf{2})_{-1/2}$	469/303, -264/557	523/411, -810/869	-4/3, -30/41
	$(\mathbf{1},\mathbf{2})_{-1/2}$	166/303, 498/557	112/411, 654/869	1/3, 54/41
	$(\mathbf{1},\mathbf{1})_{1}$	-635/303, -234/557	-635/411, 156/869	-5/3, -24/41
	$(\mathbf{1},\mathbf{1})_{0}$	5/3, -156/557	5/3, -702/869	4/3, -30/41
	$(\mathbf{1},\mathbf{1})_{0}$	2/3, 606/557	2/3, 762/869	1/3, 54/41
	$(\mathbf{3},\mathbf{1})_{-1/3}$	-8/101, -72/557	-36/137, -72/869	0, 0
	$(\mathbf{1},\mathbf{2})_{1/2}$	-671/303, -342/557	-797/411, 48/869	-5/3, -24/41
78	$(\mathbf{8},\mathbf{1})_{0}$	0, 0	0, 0	0, 0
	$(\mathbf{1},\mathbf{3})_{0}$	0, 0	0, 0	0, 0
	$(\mathbf{1},\mathbf{1})_{0}$	0, 0	0, 0	0, 0
	$(\mathbf{3},\mathbf{2})_{1/6}$	-190/303, -570/557	-220/411, -66/79	-1/3, -54/41

Irrep de E_6	Irrep de SM	$1\ U(1)' \times U(1)'$	$2\ U(1)' \times U(1)'$	$3\ U(1)' \times U(1)'$
	$(\mathbf{3},\mathbf{2})_{1/6}$	-493/303, 192/557	-631/411, 738/869	-4/3, 30/41
	$(\mathbf{3},\mathbf{2})_{-5/6}$	647/303, 270/557	689/411, -120/869	5/3, 24/41
	$(\mathbf{3},\mathbf{1})_{2/3}$	-154/303, -462/557	-58/411, -618/869	-1/3, -54/41
	$(\mathbf{3},\mathbf{1})_{2/3}$	-457/303, 300/557	-469/411, 846/869	-4/3, 30/41
	$(\mathbf{3},\mathbf{1})_{-1/3}$	683/303, 378/557	851/411, -12/869	5/3, 24/41
	$(\bar{\mathbf{3}},\mathbf{2})_{-1/6}$	493/303, -192/557	631/411, -738/869	4/3, -30/41
	$(\bar{\mathbf{3}},\mathbf{2})_{-1/6}$	190/303, 570/557	220/411, 66/79	1/3, 54/41
	$(\bar{\mathbf{3}},\mathbf{2})_{5/6}$	-647/303, -270/557	-689/411, 120/869	-5/3, -24/41
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	457/303, -300/557	469/411, -846/869	4/3, -30/41
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	154/303, 462/557	58/411, 618/869	1/3, 54/41
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	-683/303, -378/557	-851/411, 12/869	-5/3, -24/41
	$(\mathbf{1},\mathbf{2})_{1/2}$	12/101, 108/557	54/137, 108/869	0, 0
	$(\mathbf{1},\mathbf{2})_{-1/2}$	-12/101, -108/557	–54/137, -108/869	0, 0
	$(\mathbf{1},\mathbf{1})_{1}$	-279/101, -840/557	-303/137, -606/869	-2, -78/41
	$(\mathbf{1},\mathbf{1})_{1}$	-380/101, -78/557	-440/137, 78/79	-3, 6/41
	$(\mathbf{1},\mathbf{1})_{0}$	1, -762/557	1, -1464/869	1, -84/41
	$(\mathbf{1},\mathbf{1})_{0}$	-1, 762/557	-1, 1464/869	-1, 84/41
	$(\mathbf{1},\mathbf{1})_{-1}$	380/101, 78/557	440/137, -78/79	3, -6/41
	$(\mathbf{1},\mathbf{1})_{-1}$	279/101, 840/557	303/137, 606/869	2, 78/41

Cuarta regla de rompimiento

Irrep de E_6	Irrep de SM	$1\ U(1)' \times U(1)'$	$2\ U(1)' \times U(1)'$	$3\ U(1)' \times U(1)'$
27	$(\mathbf{3},\mathbf{2})_{1/6}$	82/483, -246/877	124/591, -6/19	25/39, 40/329
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	799/483, 234/877	883/591, -156/589	-35/39, -8/47
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	-158/483, 474/877	-146/591, 390/589	37/39, -204/329
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	-641/483, -708/877	-737/591, -234/589	-2/39, 260/329
	$(\mathbf{1},\mathbf{2})_{-1/2}$	559/483, 954/877	613/591, 420/589	-23/39, -300/329
	$(\mathbf{1},\mathbf{2})_{-1/2}$	76/483, -228/877	22/591, -204/589	-62/39, 164/329
	$(\mathbf{1},\mathbf{1})_{1}$	-635/483, -726/877	-635/591, -216/589	85/39, 136/329
	$(\mathbf{1},\mathbf{1})_{0}$	5/3, 216/877	5/3, -138/589	4/3, -180/329
	$(\mathbf{1},\mathbf{1})_{0}$	2/3, -966/877	2/3, -762/589	1/3, 284/329
	$(\mathbf{3},\mathbf{1})_{-1/3}$	-164/483, 492/877	-248/591, 12/19	-50/39, -80/329
	$(\mathbf{1},\mathbf{2})_{1/2}$	-881/483, 12/877	-1007/591, 18/31	10/39, 16/329
78	$(\mathbf{8},\mathbf{1})_{0}$	0, 0	0, 0	0, 0
	$(\mathbf{1},\mathbf{3})_{0}$	0, 0	0, 0	0, 0
	$(\mathbf{1},\mathbf{1})_{0}$	0, 0	0, 0	0, 0

Irrep de E_6	Irrep de SM	1 $U(1)' \times U(1)'$	2 $U(1)' \times U(1)'$	3 $U(1)' \times U(1)'$
	$(\mathbf{3},\mathbf{2})_{1/6}$	-80/161, 720/877	-90/197, 576/589	4/13, -244/329
	$(\mathbf{3},\mathbf{2})_{1/6}$	-241/161, -462/877	-287/197, -48/589	-9/13, 220/329
	$(\mathbf{3},\mathbf{2})_{-5/6}$	239/161, 480/877	253/197, 30/589	-20/13, -96/329
	$(\mathbf{3},\mathbf{1})_{2/3}$	2/161, -18/877	34/197, 18/589	29/13, -124/329
	$(\mathbf{3},\mathbf{1})_{2/3}$	-159/161, -1200/877	-163/197, -606/589	16/13, 340/329
	$(\mathbf{3},\mathbf{1})_{-1/3}$	321/161, -258/877	377/197, -528/589	5/13, 24/329
	$(\bar{\mathbf{3}},\mathbf{2})_{-1/6}$	241/161, 462/877	287/197, 48/589	9/13, -220/329
	$(\bar{\mathbf{3}},\mathbf{2})_{-1/6}$	80/161, -720/877	90/197, -576/589	-4/13, 244/329
	$(\bar{\mathbf{3}},\mathbf{2})_{5/6}$	-239/161, -480/877	-253/197, -30/589	20/13, 96/329
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	159/161, 1200/877	-163/197, 606/589	-16/13, -340/329
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	-2/161, 18/877	-34/197, -18/589	-29/13, 124/329
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	-321/161, 258/877	-377/197, 528/589	-5/13, -24/329
	$(\mathbf{1},\mathbf{2})_{1/2}$	82/161, -738/877	124/197, -18/19	25/13, 120/329
	$(\mathbf{1},\mathbf{2})_{-1/2}$	-82/161, 738/877	-124/197, 18/19	-25/13, -120/329
	$(\mathbf{1},\mathbf{1})_{1}$	-319/161, 240/877	-343/197, 546/589	24/13, -148/329
	$(\mathbf{1},\mathbf{1})_{1}$	-480/161, -942/877	-540/197, -78/589	11/13, 316/329
	$(\mathbf{1},\mathbf{1})_{0}$	1, 1182/877	1, 624/589	1, -464/329
	$(\mathbf{1},\mathbf{1})_{0}$	-1, -1182/877	-1, -624/589	-1, 464/329
	$(\mathbf{1},\mathbf{1})_{-1}$	480/161, 942/877	540/197, 78/589	-11/13, -316/329
	$(\mathbf{1},\mathbf{1})_{-1}$	319/161, -240/877	343/197, -546/589	-24/13, 148/329

Quinta regla de rompimiento

Irrep de E_6	Irrep de SM	1 $U(1)' \times U(1)'$	2 $U(1)' \times U(1)'$	3 $U(1)' \times U(1)'$
27	$(\mathbf{3},\mathbf{2})_{1/6}$	-38/843, -210/803	-74/195, -3/23	400/483, -3/38
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	238/281, -196/803	142/65, -2/23	-104/161, -9/76
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	-104/161, -9/76	-6/65, 3/23	52/161, 85/152
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	-400/281, 46/803	-136/65, -1/23	52/161, -67/152
	$(\mathbf{1},\mathbf{2})_{-1/2}$	1238/843, 164/803	482/195, 4/23	-556/483, 79/152
	$(\mathbf{1},\mathbf{2})_{-1/2}$	-448/843, 60/803	92/195, 0	-556/483, -73/152
	$(\mathbf{1},\mathbf{1})_{1}$	-790/843, -224/803	-574/195, -4/23	1112/483, -3/76
	$(\mathbf{1},\mathbf{1})_{0}$	4/3, -466/803	4/3, -5/23	4/3, 43/152
	$(\mathbf{1},\mathbf{1})_{0}$	-2/3, -570/803	-2/3, -9/23	4/3, -109/152
	$(\mathbf{3},\mathbf{1})_{-1/3}$	76/843, 420/803	148/195, 6/23	-800/483, 3/19
	$(\mathbf{1},\mathbf{2})_{1/2}$	-676/843, 406/803	-352/195, 5/23	1-88/483, 15/76
78	$(\mathbf{8},\mathbf{1})_{0}$	0, 0	0, 0	0, 0
	$(\mathbf{1},\mathbf{3})_{0}$	0, 0	0, 0	0, 0

Irrep de E_6	Irrep de SM	$1\ U(1)' \times U(1)'$	$2\ U(1)' \times U(1)'$	$3\ U(1)' \times U(1)'$
	$(\mathbf{1},\mathbf{1})_0$	0, 0	0, 0	0, 0
	$(\mathbf{3},\mathbf{2})_{1/6}$	524/843, 360/803	56/195, 6/23	-244/483, 97/152
	$(\mathbf{3},\mathbf{2})_{1/6}$	-1162/843, 256/803	–334/195, 2/23	-244/483, -55/152
	$(\mathbf{3},\mathbf{2})_{-5/6}$	752/843, 14/803	100/39, 1/23	-712/483, -3/76
	$(\mathbf{3},\mathbf{1})_{2/3}$	410/843, -270/803	-166/195, -3/23	956/483, 61/152
	$(\mathbf{3},\mathbf{1})_{2/3}$	-1276/843, -34/73	-556/195, -7/23	956/483, -91/152
	$(\mathbf{3},\mathbf{1})_{-1/3}$	638/843, -56/73	278/195, -8/23	488/483, -21/76
	$(\bar{\mathbf{3}},\mathbf{2})_{-1/6}$	1162/843, -256/803	334/195, -2/23	244/483, 55/152
	$(\bar{\mathbf{3}},\mathbf{2})_{-1/6}$	-524/843, -360/803	-56/195, -6/23	244/483, -97/152
	$(\bar{\mathbf{3}},\mathbf{2})_{5/6}$	-752/843, -14/803	-100/39, -1/23	712/483, 3/76
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	1276/843, 34/73	556/195, 7/23	-956/483, 91/152
	$(\bar{\mathbf{3}},\mathbf{1})_{-2/3}$	-410/843, 270/803	166/195, 3/23	-956/483, -61/152
	$(\bar{\mathbf{3}},\mathbf{1})_{1/3}$	-638/843, 56/73	-278/195, 8/23	-488/483, 21/76
	$(\mathbf{1},\mathbf{2})_{1/2}$	-38/281, -630/803	-74/65, -9/23	400/161, -9/38
	$(\mathbf{1},\mathbf{2})_{-1/2}$	-38/281, 630/803	74/65, 9/23	-400/161, 9/38
	$(\mathbf{1},\mathbf{1})_1$	-76/281, 346/803	-148/65, 5/23	156/161, 103/152
	$(\mathbf{1},\mathbf{1})_1$	-638/281, 22/73	-278/65, 1/23	156/161, -49/152
	$(\mathbf{1},\mathbf{1})_0$	2, 104/803	2, 4/23	0, 1
	$(\mathbf{1},\mathbf{1})_0$	-2, -104/803	-2, -4/23	0, -1
	$(\mathbf{1},\mathbf{1})_{-1}$	638/281, -22/73	278/65, -1/23	-156/161, 49/152
	$(\mathbf{1},\mathbf{1})_{-1}$	76/281, -346/803	148/65, -5/23	-156/161, -103/152

Los resultados anteriores nos indican potenciales modelos para extensiones del SM con simetrías $U(1)'$, motivadas por la teoría de gran unificación E_6. Debemos mencionar que la elección de las cargas mostrada en cada caso de las tablas anteriores no es única. En la naturaleza, quizá sólo una combinación lineal (del infinito de posibilidades) de las $U(1)'$ podría manifestarse en, por ejemplo, colisionadores de partículas, pero no es posible sin una guía fenomenológica determinar esa combinación lineal y sus consecuencias observables.

Por esta razón, el presente trabajo se restringe a estos resultados. Sin embargo, es claro que el siguiente paso consiste en encontrar un generador ortogonal al generador de hipercarga (una combinación de las $U(1)'$), que conduzca a diversos escenarios ya conjeturados desde la perspectiva *bottom-up*. Por ejemplo, se puede solicitar que bajo el nuevo generador sólo estén cargadas ciertas partículas, tales como, leptones o quarks o el bosón de Higgs del SM. De la misma manera, se puede buscar que sólo se excluyan de las interacciones cierto tipo de partículas, dando lugar a un sector oscuro que pudiera explicar la aparente existencia de materia oscura. Relegamos esta investigación a un futuro proyecto.

Capítulo 5

Resultados y conclusiones

La búsqueda de un teoría que complete la física de partículas codificada en el SM dando respuesta a las preguntas que el SM no es capaz de contestar es un problema abierto en la física actual. Este reto se estudia desde diferentes frentes. Desde la perspectiva *bottom-up*, se extiende el SM añadiendo partículas o interacciones nuevas, las cuales pueden conducir a física nueva que es actualmente puesta a prueba en diversos laboratorios de partículas y astropartículas. La mayoría de los trabajos en fenomenología de partículas emplean este tratamiento.

Por otro lado, desde la perspectiva *top-down*, que es la que adoptamos en este trabajo, se considera que el SM es una teoría efectiva, remanente de una teoría más fundamental que unifica a todas las fuerzas del SM y cuyo grupo de simetrías es roto a una escala energética mucho mayor a la explorada actualmente en los aceleradores de partículas. Es este rompimiento (espontáneo o no) el que daría lugar a las simetrías de norma de la física de partículas observada, $\mathrm{SU}(3)_C \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y$. Además, para que a partir de este rompimiento surjan los fermiones y bosones del SM se requiere de campos que se transformen en representaciones del álgebra válida a altas energías. Las reglas de ramificación bajo el rompimiento de la simetría original deben conducir a campos con las cualidades de los del SM y tal vez algunos más que sean considerados propiedades de la nueva física proveniente de la teoría conjeturada. Esta propuesta es la que da lugar a las llamadas teorías de gran unificación, las cuales se rigen típicamente por un álgebra del tipo "A", "D" y "E" (tales como $\mathrm{SU}(5)$, $\mathrm{SU}(6)$, $\mathrm{SO}(10)$, $\mathrm{SO}(14)$, E_6, E_7, E_8, entre otras).

Debido a que el rango del álgebra del grupo del SM es cuatro y que todas las propuestas de teorías de gran unificación, salvo $\mathrm{SU}(5)$, tienen rango superior, es natural esperar que surjan del rompimiento fuerzas adicionales a bajas energías que aún no han sido observadas. Frecuentemente, las fuerzas obtenidas son Abelianas y continuas, es decir, forman grupos de norma $\mathrm{U}(1)$ adicionales, comúnmente llamados $\mathrm{U}(1)'$. Uno de los grandes objetivos de la presente tesis ha sido proveer herramientas para el estudio general y sistemático de cualquier grupo de gran unificación, de las simetrías $\mathrm{U}(1)'$ (y los bosones de norma exóticos Z') que pueden surgir de ellos, y de las cargas de los campos resultantes a las energías en las que las simetrías del SM dominan.

Con este objetivo en mente, hemos desarrollado una herramienta informática denominada `dynkin`. Este es el principal resultado de este trabajo. El software `dynkin` ha sido diseñado en Python, por lo que cuenta con una gran portabilidad y fiabilidad en los cálculos efectuados. Además, ha sido liberado bajo la *GNU General Public Licence*, por lo que es de acceso gratuito, libre, y se puede compartir, distribuir y editar sin ningún inconveniente. El software está disponible en

`www.github.com/elhacs/Dynkin`

El programa cuenta con diversas funciones que se detallan en el capítulo 3. Además de funciones específicas para el estudio de las principales propiedades de las álgebras del tipo "A", "D" y "E" y sus representaciones, la

herramienta más importante incluida en `dynkin` es su capacidad de aportar las consecuencias de los rompimientos maximales semisimples de estas álgebras. Los datos de entrada son el álgebra original, la *topología* del rompimiento basada en la estructura del diagrama de Dynkin del álgebra que se investiga, y los pesos máximos de las representaciones de los campos (bosónicos y fermiónicos) de la teoría original. El programa determina entonces las reglas de ramificación para las representaciones dadas en la nueva teoría, lo que conduce a las representaciones bajo las simetrías no rotas y, en particular, las cargas $\mathrm{U}(1)'$ de los nuevos campos.

Durante el diseño del programa se buscó que el número de suposiciones fuera limitado y tuvieran un carácter geométrico, por lo que `dynkin` tiene una gran fortaleza a la hora de realizar rompimientos ya que solo es necesario saber la topología del rompimiento del diagrama de Dynkin en el cual estemos interesados. En este trabajo nos enfocamos en buscar simetrías $\mathrm{U}(1)$ adicionales, sin embargo, `dynkin` es capaz de romper un grupo de Lie arbitrario a otro que no exhiba simetrías Abelianas. Esta ventaja en `dynkin`, permite hacer búsquedas de física nueva, que no están directamente relacionadas con simetrías $\mathrm{U}(1)'$.

Adicionalmente, `dynkin` está escrito con el paradigma de *programación orientada objetos* (o POO por sus siglas en inglés), por lo que es fácilmente ampliable y reutilizable, ya que cada parte del código se separó en diferentes módulos que cumple una función similar. Además se optó por un enfoque simbólico a la hora de realizar cálculos en lugar de un enfoque numérico, debido a la exactitud manejada por Python. Lo cual evita un problema presente en el cálculo numérico producto del manejo de números con punto flotante. En las pruebas y cálculos realizados, se nota un mayor consumo de tiempo de máquina debido a este enfoque, sin embargo, no es demasiado considerando los resultados en forma de números racionales que presenta `dynkin`, en las asignaciones de las cargas $\mathrm{U}(1)$ de los rompimientos. Consideramos que esta herramienta tendrá muchas más aplicaciones que la empleada en esta tesis, incluso en campos ajenos a la física de partículas.

Las rutinas desarrolladas son utilizadas en el capítulo 4 para un ejemplo de grupo de gran unificación poco estudiado en la literatura, el grupo E_6. En este contexto, con la intención de clasificar *todas* las posibles rupturas maximales no semisimples que conducen al SM y a dos simetrías $\mathrm{U}(1)'$ y las propiedades de los campos resultantes bajo estas simetrías, estudiamos las reglas de rompimiento, especificando las combinaciones de las $\mathrm{U}(1)$s que conducen a la hipercarga y a las dos simetrías $\mathrm{U}(1)'$. Notamos que cada rompimiento no semisimple conduce a tres posibles elecciones independientes de hipercarga, por lo que también se obtienen tres pares de simetrías $\mathrm{U}(1)'$.

Una vez conseguido este resultado, considerando que una generación de fermiones se encuentra contenida en la representación **27** y que esta representación también puede emplearse en un multiplete bosónico que dé lugar al campo de Higgs, se determinan las cargas $\mathrm{U}(1)'$ para todas las partículas exóticas de materia resultantes. Lo mismo se hace para la representación **78**, la cual contiene los 14 bosones de norma de la simetría de norma remanente $\mathrm{SU}(3)_C \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y \times \mathrm{U}(1)'_1 \times \mathrm{U}(1)'_2$ y otros bosones de materia exótica. Las cargas de estas partículas son determinadas sistemáticamente en cada caso, lo cual conduce a un conjunto de cargas para las partículas observadas (16 de la representación **27** y 14 de la **78**) y las exóticas. Este, nuestro segundo resultado importante, se enlista en las tablas de la sección 4.2.

Para concluir, es preciso discutir las posibles extensiones del presente trabajo. Como el código de `dynkin` es modular, se puede fácilmente ampliar. Una futura implementación es la búsqueda de modelos con Z' que exhiban ciertas características fenomenológicamente preferidas en la literatura. Por ejemplo, se puede solicitar que la nueva interacción sólo actúe sobre los leptones del SM, conduciendo a modelos *leptofilicos* [14, 41], o bien, que se excluya de la interacción del bosón de Higgs (en modelos denominados *higgsfóbicos*) [42]. La física de estos escenarios estaría relacionada con la estabilidad del potencial de Higgs y de los nucleones, así como también con la explicación de señales anómalas, incluyendo el problema del momento dipolar eléctrico anómalo del muón. Este estudio, así como otras posibles aplicaciones será parte de nuestra futura investigación.

Apéndice

Apéndice A

Módulos de `dynkin`

Debido a que el resultado principal de este trabajo es esta herramienta informática, concebida para el estudio de diversas álgebras y sus rompimientos maximales semisimples, como se presenta en el capítulo 3, en este apéndice aportamos una descripción y los detalles de los módulos y funciones incluidas en `dynkin`. Como mencionamos antes, el código fuente, con la documentación correspondiente, puede ser descargado del sitio

`www.github.com/elhacs/Dynkin`

A.1. Módulo Álgebra

Este módulo es fundamental, ya que se construye la clase llamada ALGEBRA, encargada de crear objetos con las características que definen a un álgebra de Lie, explicada en la sección 1.1, que son:

- Raíces simples
- Raíz máxima
- Orden
- Tipo
- Rango
- Matriz de Cartan
- Tensor métrico
- Diagrama de Dynkin

```
1 # -*- coding: utf-8 -
2
3 # ------------------------------------------------------------
4 # Enrique Escalante Notario
5 # Instituto de Fisica, UNAM
```

```
# email enriquescalante at gmail.com
# Dynkin1.0
# Módulo algebra
# ------------------------------------------------------------------

# Anotaciones
# 1) el programa solo necesitara el HWV, Algebra & Rango
# 2) solo vamos esta interesados en algebras A, D, E
# 3) Matrix([[]]) -> vector fila; Matrix([]) -> vector columna
# 4) la salida peso es una lista con enteros

# importacion de paquete

from math import *
import sympy as sp


def __verificaralgebra__(algebra):
    try:
        if algebra not in ["A", "D", "E"]:
            raise ValueError
        else:
            return algebra
    except:
        print("Algebra invalida: Solo A, D y E")


def __verificarrango__(algebra, rango):
    try:
        if algebra == "A":
            if rango < 1:
                raise ValueError
            else:
                return rango
        elif algebra == "D":
            if rango < 4:
                raise ValueError
            else:
                return rango
        elif algebra == "E":
            if rango not in [6, 7, 8]:
                raise ValueError
            else:
                return rango
        else:
            return rango
    except:
        print("Rango invalido: rango(A) > 1, rango(D) > 3, rango(E) = 6,7,8")


```

```
class alg():
    """
    alg(tipo,rango)
    -------------------

    Debemos crear el objeto principal con el que trabajar \code{dynkin} que es
    una álgebra de Lie, este software está diseñado para trabajar inicialmente
    con las álgebras del tipo A, D y E. Por lo tanto, para crear una álgebra del
    tipo A y rango 4, solo debemos teclar

        Dynkin>> A4 = alg('A',4)


    """
    def __init__(self, tipo, rango):
        self.tipo = __verificaralgebra__(tipo)
        self.rango = __verificarrango__(self.tipo, rango)
        self.positivas = []
        self._simpleroots = []
        self._ortogonalroots = []
        self._highroot = []
        self._orden = 0
        self._cartanmatrix = []
        self._metrictensor = []

    def raicessimples(self):
        # Creamos las raices simples del algebra en la base de Dynkin
        # la salida es un lista cuyos elementos son simpy.matrix
        if self._simpleroots == []:
            if self.tipo == "A":
                for i in range(self.rango):
                    _simple = sp.zeros(1, self.rango)
                    _simple[i] = 2
                    if i > 0:
                        _simple[i - 1] = -1
                    if i < self.rango - 1:
                        _simple[i + 1] = -1
                    self._simpleroots.append(_simple)
            elif self.tipo == "D":
                if self.rango > 3:
                    for i in range(self.rango):
                        _simple = sp.zeros(1, self.rango)
                        _simple[i] = 2
                        if i > 0:
                            if i == self.rango - 2:
                                _simple[i - 1] = -1
                            elif i == self.rango - 1:
                                _simple[i - 2] = -1
                                _simple[i - 1] = 0
                            else:
```

```
                                _simple[i - 1] = -1
                        if i < self.rango - 1:
                            if i == self.rango - 3:
                                _simple[i + 1] = -1
                                _simple[i + 2] = -1
                            if i == self.rango - 2:
                                _simple[i + 1] = 0
                            else:
                                _simple[i + 1] = -1
                        self._simpleroots.append(_simple)
            elif self.tipo == "E":
                if self.rango == 6:
                    self._simpleroots = [
                        sp.Matrix([[2, -1, 0, 0, 0, 0]]),
                        sp.Matrix([[-1, 2, -1, 0, 0, 0]]),
                        sp.Matrix([[0, -1, 2, -1, 0, -1]]),
                        sp.Matrix([[0, 0, -1, 2, -1, 0]]),
                        sp.Matrix([[0, 0, 0, -1, 2, 0]]),
                        sp.Matrix([[0, 0, -1, 0, 0, 2]])
                    ]
                elif self.rango == 7:
                    self._simpleroots = [
                        sp.Matrix([2, -1, 0, 0, 0, 0, 0]),
                        sp.Matrix([-1, 2, -1, 0, 0, 0, 0]),
                        sp.Matrix([0, -1, 2, -1, 0, 0, -1]),
                        sp.Matrix([0, 0, -1, 2, -1, 0, 0]),
                        sp.Matrix([0, 0, 0, -1, 2, -1, 0]),
                        sp.Matrix([0, 0, 0, 0, -1, 2, 0]),
                        sp.Matrix([0, 0, -1, 0, 0, 0, 2])
                    ]
                elif self.rango == 8:
                    self._simpleroots = [
                        sp.Matrix([2, -1, 0, 0, 0, 0, 0, 0]),
                        sp.Matrix([-1, 2, -1, 0, 0, 0, 0, 0]),
                        sp.Matrix([0, -1, 2, -1, 0, 0, 0, 0]),
                        sp.Matrix([0, 0, -1, 2, -1, 0, 0, 0]),
                        sp.Matrix([0, 0, 0, -1, 2, -1, 0, -1]),
                        sp.Matrix([0, 0, 0, 0, -1, 2, -1, 0]),
                        sp.Matrix([0, 0, 0, 0, 0, -1, 2, 0]),
                        sp.Matrix([0, 0, 0, 0, -1, 0, 0, 2])
                    ]
        elif self._simpleroots != []:
            pass
        return self._simpleroots

    def __raicesortogonales(self):
        # Definimos las raices en un base orthogonal
        # la salida es un lista cuyo elementos son simpy.matrix
        if self._ortogonalroots == []:
            if self.tipo == "A":
```

```
                for i in range(self.rango):
                    _simple = sp.zeros(1, self.rango + 1)
                    _simple[i] = 1
                    if i + 1 < self.rango + 1:
                        _simple[i + 1] = -1
                    self._ortogonalroots.append(_simple)
            elif self.tipo == "D":
                if self.rango > 3:
                    for i in range(self.rango):
                        _simple = sp.zeros(1, self.rango)
                        _simple[i] = 1
                        if i + 1 < self.rango:
                            _simple[i + 1] = -1
                        elif i + 1 == self.rango:
                            _simple[i - 1] = 1
                        self._ortogonalroots.append(_simple)
            elif self.tipo == "E":
                if self.rango == 6:
                    self._ortogonalroots = [
                        sp.Matrix(
                            [0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, 0.5]),
                        sp.Matrix([-1, 1, 0, 0, 0, 0, 0, 0]),
                        sp.Matrix([0, -1, 1, 0, 0, 0, 0, 0]),
                        sp.Matrix([0, 0, -1, 1, 0, 0, 0, 0]),
                        sp.Matrix([0, 0, 0, -1, 1, 0, 0, 0]),
                        sp.Matrix([1, 1, 0, 0, 0, 0, 0, 0])
                    ]
                elif self.rango == 7:
                    self._ortogonalroots = [
                        sp.Matrix(
                            [0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, 0.5]),
                        sp.Matrix([-1, 1, 0, 0, 0, 0, 0, 0]),
                        sp.Matrix([0, -1, 1, 0, 0, 0, 0, 0]),
                        sp.Matrix([0, 0, -1, 1, 0, 0, 0, 0]),
                        sp.Matrix([0, 0, 0, -1, 1, 0, 0, 0]),
                        sp.Matrix([0, 0, 0, 0, -1, 1, 0, 0]),
                        sp.Matrix([1, 1, 0, 0, 0, 0, 0, 0])
                    ]
                elif self.rango == 8:
                    self._ortogonalroots = [
                        sp.Matrix(
                            [0.5, -0.5, -0.5, -0.5, -0.5, -0.5, -0.5, 0.5]),
                        sp.Matrix([-1, 1, 0, 0, 0, 0, 0, 0]),
                        sp.Matrix([0, -1, 1, 0, 0, 0, 0, 0]),
                        sp.Matrix([0, 0, -1, 1, 0, 0, 0, 0]),
                        sp.Matrix([0, 0, 0, -1, 1, 0, 0, 0]),
                        sp.Matrix([0, 0, 0, 0, -1, 1, 0, 0]),
                        sp.Matrix([0, 0, 0, 0, 0, -1, 1, 0, 0]),
                        sp.Matrix([1, 1, 0, 0, 0, 0, 0, 0])
                    ]
```

```
206             elif self._ortogonalroots != []:
207                 pass
208             return self._ortogonalroots
209 
210         def raizmaxima(self):
211             # Obtenemos las raiz maxima del algebra en cuention
212             # la salida es un lista
213             if self._highroot == []:
214                 if self.tipo == "A":
215                     if self.rango == 1:
216                         self._highroot = sp.Matrix([2])
217                     elif self.rango != 1:
218                         self._highroot = sp.zeros(1, self.rango)
219                         self._highroot[0] = 1
220                         self._highroot[self.rango - 1] = 1
221                 if self.tipo == "D":
222                     self._highroot = sp.zeros(1, self.rango)
223                     self._highroot[1] = 1
224                 if self.tipo == "E":
225                     if self.rango == 6:
226                         self._highroot = sp.Matrix([[0, 0, 0, 0, 0, 1]])
227                     if self.rango == 7:
228                         self._highroot = sp.Matrix([1, 0, 0, 0, 0, 0, 0])
229                     if self.rango == 8:
230                         self._highroot = sp.Matrix([1, 0, 0, 0, 0, 0, 0, 0])
231             elif self._highroot != []:
232                 pass
233             return self._highroot.tolist()[0]
234 
235         def orden(self):
236             # El orden del algebra es igual a la dimension de la representacion
237             # adjunta del algebra
238             # la salida es un entero
239             if self._orden == 0:
240                 if self.tipo == "A":
241                     self._orden = int((self.rango + 1)**2 - 1)
242                 if self.tipo == "D":
243                     self._orden = int(
244                         (2 * self.rango) * ((2 * self.rango) - 1) / 2)
245                 if self.tipo == "E":
246                     if self.rango == 6:
247                         self._orden = 78
248                     if self.rango == 7:
249                         self._orden = 133
250                     if self.rango == 8:
251                         self._orden = 248
252             elif self._orden != 0:
253                 pass
254             return self._orden
255 
```

```
256     def matrizcartan(self):
257         # Define la matriz de Cartan del algebra
258         # la salida es un sympy.matrix
259         if self._cartanmatrix == []:
260             self._cartanmatrix = sp.Matrix(self.raicessimples())
261         elif self._cartanmatrix != []:
262             pass
263         return self._cartanmatrix
264 
265     def tensormetrico(self):
266         # Define un tensor usado en el producto escalar del espacio de pesos,
267         # que es la inversa de la matriz de Cartan"""
268         # la salida es un sp.Matrix rankxrank
269         if self._metrictensor == []:
270             self._metrictensor = self.matrizcartan().inv()
271         elif self._metrictensor != []:
272             pass
273         return self._metrictensor
274 
275     def diagramadynkin(self):
276         """Nos visualizar el diagrama de Dynkin asociado al algebra"""
277         n = self.rango
278         if self.tipo == "A":
279             diag = "---".join("0" for i in range(1, n + 1)) + "\n"
280             diag += "   ".join(str(i) for i in range(1, n + 1))
281         if self.tipo == "D":
282             diag = " " * 4 * (n - 3) + str(n - 1) + "\n"
283             diag += " " * 4 * (n - 3) + "0\n"
284             diag += " " * 4 * (n - 3) + "|\n"
285             diag += " " * 4 * (n - 3) + "|\n"
286             diag += "---".join("0" for i in range(1, n)) + "\n"
287             diag += "   ".join(
288                 str(i) for i in range(1, n - 1)) + "   " + str(n)
289         if self.tipo == "E":
290             diag = " " * 8 + str(Rank) + "\n"
291             diag += " " * 8 + "0\n"
292             diag += " " * 8 + "|\n"
293             diag += "---".join("0" for i in range(1, n)) + "\n"
294             diag += "" + "   ".join(str(i) for i in range(1, n))
295         return diag
```

A.2. Módulo Información

Es el módulo que extrae información sobre el objeto ALGEBRA, tales como:

- Raíces simples
- Raíz máxima

- Orden del álgebra, que coincide con la dimensión de la representación adjunta
- Matriz de Carta
- Tensor métrico del espacio de pesos del álgebra
- Diagrama de Dynkin
- Rango del álgebra
- Tipo del álgebra

```
# -*- coding: utf-8 -*-

# ------------------------------------------------------------
# Enrique_Escalante-Notario
# Instituto de Fisica, UNAM
# email: <enriquescalante@gmail.com>
# Distributed under terms of the GPLv3 license.
# Módulo informacion
# ------------------------------------------------------------


def simples(algebra):
    """
    simples(algebra)
    -----------------------

    Nos devuelve el conjunto de raíces simples del
    álgebra en la base de Dynkin. Podemos asignar una etiqueta especial a esta
    conjunto, o simplemente podemos imprimir en pantalla el resultado.


        Dynkin >> p(simples(A4))
        [[ 2, -1, 0, 0]
        [-1, 2, -1, 0]
        [0, -1, 2, -1]
        [0, 0, -1, 2]]


    """
    return algebra.raicessimples()


def maxima(algebra):
    """
    maxima(algebra)
    -----------------------

    Con este comando obtenemos el peso máximo de la representación adjunta del
```

```
    álgebra.

        Dynkin >> p(maxima(A4))
        [1, 0, 0, 1]


    """
    return algebra.raizmaxima()


def orden(algebra):
    """
    orden(algebra)
    ------------------------

    Calcula la dimensión del álgebra.

        Dynkin >> p(orden(A4))
        24

    """
    return algebra.orden()


def mcartan(algebra):
    """
    mcartan(algebra)
    --------------------------
    Nos devuelve la matriz de Cartan. Aunque el resultado se parece al que
    obtenemos con el comando RaicesSimples, internamente dynkin trata a estos
    dos objetos de maneras distintas, por lo que es importante no mezclarlos.

        Dynkin >> p(mcartan(A4))
        [ 2, -1, 0, 0]
        [-1, 2, -1, 0]
        [0, -1, 2, -1]
        [0, 0, -1, 2]

    """
    return algebra.matrizcartan()


def tmetrico(algebra):
    """
    tmetrico(algebra)
    -------------------------

    Obtenemos el tensor métrico del álgebra.

        Dynkin >> p(tmetrico(A4))
```

```
          [4/5, 3/5, 2/5, 1/5]
          [3/5, 6/5, 4/5, 2/5]
          [2/5, 4/5, 6/5, 3/5]
          [1/5, 2/5, 3/5, 4/5]


    """
    return algebra.tensormetrico()


def ddynkin(algebra):
    """
    ddynkin(algebra)
    ------------------------

    Imprime el diagrama de Dynkin asociado al álgebra, con las raíces simples
    enumeradas.

        Dynkin >> p(ddynkin(A4))
        0---0---0---0
        1   2   3   4
    """
    return algebra.diagramadynkin()


def tipo(algebra):
    return algebra.tipo


def rango(algebra):
    return algebra.rango
```

A.3. Módulo Pesos

Con el objeto álgebra creado a partir de la instanciación de la clase ALGEBRA, podemos empezar a construir el sistema de pesos para un peso máximo dado. A partir de la definición de este peso máximo podemos obtener los siguientes resultados.

- Raíces positivas.
- Pesos.

```
# -*- coding: utf-8 -

# ------------------------------------------------------------------
# Enrique Escalante Notario
# Instituto de Fisica, UNAM
```

```
6  # email <enriquescalantegmail.com>
7  # Distributed under terms of the GPLv3 lincense.
8  # ------------------------------------------------------------
9
10 # Anotaciones
11 # 1) el programa solo necesitara el HWV, Algebra & Rango
12 # 2) solo vamos esta interesados en algebras A, D, E
13 # 3) Matrix([[]]) -> vector fila; Matrix([]) -> vector columna
14 # 4) la salida "peso" es un lista de enteros.
15
16 # importacion de paquetes
17 import sympy as sp
18
19
20 def __espesomaximo__(vector):
21     for elemento in vector:
22         valor = True
23         if elemento < 0:
24             valor = False
25             break
26     return valor
27
28
29 def __bajada__(peso, raicessimples):
30     # Es el primer paso para construir el sistema de pesos, out [peso's]
31     _lista_resultante = []
32     _peso = sp.Matrix([peso])
33     for i in range(len(raicessimples)):
34         _componente = int(peso[i])
35         if _componente > 0:
36             for j in range(_componente):
37                 _lista_resultante.append(
38                     (_peso - (j + 1) * raicessimples[i]).tolist()[0])
39     return _lista_resultante
40
41
42 def __bajada1__(_bajada1, raicessimples):
43     # Es la generalizacion de __bajada__ para un lista, out[peso's]
44     _bajada2 = [[]]
45     _bajada3 = [[]]
46     for _peso in _bajada1:
47         _bajada = __bajada__(_peso, raicessimples)
48         _bajada2.extend([_elemento for _elemento in _bajada
49                          if _elemento not in _bajada2])
50     del(_bajada2[0])
51     _bajada3.extend([elemento for elemento in _bajada2
52                      if elemento not in _bajada1])
53     del(_bajada3[0])
54     return _bajada3
55
```

```
56
57  def __pesos__(peso, raicessimples):
58      # Construye el sistema de peso sin considerar las multiplicidades
59      # out[peso's]
60      _bajada1 = __bajada__(peso, raicessimples)
61      _bajada2 = __bajada1__(_bajada1, raicessimples)
62      _bajada3 = _bajada1 + _bajada2
63      while _bajada2 != []:
64          _bajada4 = __bajada1__(_bajada2, raicessimples)
65          _bajada3.extend([_elemento for _elemento in _bajada4
66                           if _elemento not in _bajada3])
67          _bajada2 = _bajada4
68      _bajada3.insert(0, peso)
69      return _bajada3
70
71
72  def __dynkin2dual__(peso, tensormetrico):
73      # cambia de la base de dynkin a la de raices simples, out peso
74      return (sp.Matrix([peso]) * tensormetrico).tolist()[0]
75
76
77  def __dual2dynkin__(peso, matrizcartan):
78      # cambia de la base de raices simples a la de dynkin, out peso
79      return (sp.Matrix([peso]) * matrizcartan).tolist()[0]
80
81
82  def escalar(peso1, peso2, tensormetrico):
83      # define el producto eucliano(base de cartan-weyl) en la base de dynkin,
84      # out int
85      return (sp.Matrix([peso1]) * tensormetrico * sp.Matrix([peso2]).T)[0, 0]
86
87
88  def __conjunto__(lista):  # este no lo he usado
89      _set = set()
90      for _element in lista:
91          _set.add(str(_element))
92      return _set
93
94
95  def positivas(algebra):  # out [peso's]
96      """
97      positivas(algebra)
98      ----------------------------
99
100     Nos da el conjunto de raíces positivas del álgebra.
101
102         Dynkin >> p(positivas(algebra))
103         [[ 1, 0, 0, 1]
104         [-1, 1, 0, 1]
105         [1, 0, 1, -1]
```

```
106             [0, -1, 1, 1]
107             [-1, 1, 1, -1]
108             [1, 1, -1, 0]
109             [0, 0, -1, 2]
110             [0, -1, 2, -1]
111             [-1, 2, -1, 0]
112             [2, -1, 0, 0]]
113
114         """
115         if algebra.positivas == []:
116             raizmaximA = algebra.raizmaxima()
117             raicessimpleS = algebra.raicessimples()
118             tensormetricO = algebra.tensormetrico()
119             _raices = __pesos__(raizmaximA, raicessimpleS)
120             _raicespositivas = []
121             for _raiz in _raices:
122                 _raizdual = __dynkin2dual__(_raiz, tensormetricO)
123                 for _componente in _raizdual:
124                     if _componente > 0:
125                         _raicespositivas.append(_raiz)
126                         break
127                     if _componente < 0:
128                         break
129             algebra.positivas = _raicespositivas
130             return _raicespositivas
131         elif algebra.positivas != []:
132             pass
133             return algebra.positivas
134
135
136     # out [peso's]
137     def __raicespositivas__(raizmaxima, raicessimples, tensormetrico):
138         # Construye el sistema de raices positivas asociada a una
139         # algebra en particular
140         _raices = __pesos__(raizmaxima, raicessimples)
141         _raicespositivas = []
142         for _raiz in _raices:
143             _raizdual = __dynkin2dual__(_raiz, tensormetrico)
144             for _componente in _raizdual:
145                 if _componente > 0:
146                     _raicespositivas.append(_raiz)
147                     break
148                 if _componente < 0:
149                     break
150         return _raicespositivas
151
152
153     def __ks__(peso, pesos, raicespositivas, tensormetrico):
154         # Verifica la condicion en la formula de freudental, out {pesodestino:raiz}
155         _ks = {}
```

```
    _peso = sp.Matrix([peso])
    for _raiz in raicespositivas:
        for _peso1 in pesos:
            k = escalar(
                sp.Matrix([_peso1]) - _peso, _raiz, tensormetrico) / 2.0
            _peso2 = (_peso + k * sp.Matrix([_raiz])).tolist()[0]
            if k > 0 and _peso1 == _peso2:
                _ks[str(_peso1)] = _raiz
    return _ks


def __freudental__(pesos, raicessimples, tensormetrico, raizmaxima):
    # Nos permite calcular las multiplicades de toda una representacion,
    # out {peso,,multi}
    _pesos = pesos[:]
    _pesomaximo = _pesos[0]
    _delta = sp.ones(1, len(_pesomaximo))
    _pesomasdelta = sp.Matrix([_pesomaximo]) + _delta
    _producto = escalar(_pesomasdelta, _pesomasdelta, tensormetrico)
    _raicespositivas = __raicespositivas__(
        raizmaxima, raicessimples, tensormetrico)
    _multiplicidades = {}
    while _pesos != []:
        _pesosnocomputados = []
        for _peso in _pesos:
            if (_peso == _pesomaximo):
                _multiplicidades[str(_peso)] = 1
            if (_peso != _pesomaximo):
                _pesoaux = sp.Matrix([_peso])
                _ks = __ks__(_peso, _pesos, _raicespositivas, tensormetrico)
                if set(_ks.keys()) <= set(_multiplicidades.keys()):
                    _coeficiente = 2.0 / (_producto -
                                          escalar(
                                              _pesoaux + _delta, _pesoaux +
                                              _delta, tensormetrico))
                    _multipeso = 0
                    for _etiqueta in _ks.keys():
                        _multipeso1 = _multiplicidades.get(_etiqueta)
                        _peso1 = eval(_etiqueta)
                        _raiz1 = _ks.get(_etiqueta)
                        _multipeso = _multipeso + _multipeso1 * \
                            escalar(_peso1, _raiz1, tensormetrico)
                    _multiplicidades[str(_peso)] = int(
                        round(_multipeso * _coeficiente))
                elif _ks.keys() not in _multiplicidades.keys():
                    _pesosnocomputados.append(_peso)
        _pesos = _pesosnocomputados
    return _multiplicidades


```

```
def pesos(peso, algebra):
    """
    pesos(pesomaximo,algebra)
    -------------------------

    Con este comando calculamos el sistema de pesos en la base de Dynkin.
    Es importante notar el formato del peso máximo al igual de la restricción
    de que ninguna de sus componentes puede ser negativa. Para abreviar la
    escritura podemos definir

        hwv1 = [1,0,0,0]
        hwv2 = [0,1,0,0]


    que son los pesos máximos de dos irreps del álgebra ``A4''.
    Ahora podemos calcular el sistema de pesos para uno de los pesos máximos
    anteriores.

        Dynkin >> p(pesos(hwv1,A4)
        [[ 1, 0, 0, 0]
        [-1, 1, 0, 0]
        [0, -1, 1, 0]
        [0, 0, -1, 1]
        [0, 0, 0, -1]]

    """
    # out [peso's]
    if __espesomaximo__(peso):
        raicessimpleS = algebra.raicessimples()
        tensormetricO = algebra.tensormetrico()
        raizmaximA = algebra.raizmaxima()
        _pesos = __pesos__(peso, raicessimpleS)
        _multi = __freudental__(
            _pesos,
            raicessimpleS,
            tensormetricO,
            raizmaximA)
        _sistema = []
        for _peso in _pesos:
            for i in range(int(round(_multi.get(str(_peso))))):
                _sistema.append(_peso)
        return _sistema
    if __espesomaximo__(peso) == False:
        raise TypeError
```

A.4. Módulo Invariantes

Una vez creado el sistema de peso representado por un peso máximo, obtenemos el sistema de pesos. A partir de éste, podemos obtener las siguientes cantidades.

- Dimensión de la representación irreducible asociada al peso máximo.
- Índice de la representación irreducible asociada al peso máximo.
- Clase de congruencia de la representación irreducible asociada al peso máximo.
- Conjugación de la representación irreducible asociada al peso máximo.
- Complejidad de la representación irreducible asociada al peso máximo.
- Nombre del la representación irreducible, que es la suma de la dimensión de la representación más su complejidad.

```
# ------------------------------------------------------------
# Enrique_Escalante-Notario
# Instituto de Fisica, UNAM
# email: <enriquescalante@gmail.com>
# Distributed under terms of the GPLv3 license.
# Módulo invariantes
# ------------------------------------------------------------


import sympy as sp
from pesos import *


def dim(pesomaximo, algebra):
    """
    dim(pesomaximo,algebra)
    -----------------------------------

    De esta manera obtenemos uno de los primeros invariantes de la irrep,
    el numero de elemento de la irrep que es igual a la dimension del espacio
    de Hilbert donde actua el algebr
    a

        Dynkin >> p(dim(hwv1, A4))
        5


    """
    tensormetricO = algebra.tensormetrico()
    raicespositivaS = positivas(algebra)
    _dim = 1
    _delta = sp.ones(1, len(pesomaximo))
```

```
    _pesomasdelta = sp.Matrix([pesomaximo]) + _delta
    for _raiz in raicespositivaS:
        _dim = _dim * escalar(_pesomasdelta, _raiz, tensormetricO)
        _dim = _dim / escalar(_delta, _raiz, tensormetricO)
    return int(round(_dim))


def ind(pesomaximo, algebra):
    """
    ind(pesomaximo, algebra):
    ---------------------------------

    De esta manera obtenemos el indice de la irrep

            Dynkin >> p(ind([0,0,0,1], a4))
            1

    Para el caso de SU(5), tenemos

    """
    tensormetricO = algebra.tensormetrico()
    raicespositivaS = positivas(algebra)
    _delta = sp.ones(1, len(pesomaximo))
    _dim_adj = 2 * len(raicespositivaS) + len(pesomaximo)
    _pesomas2delta = sp.Matrix([pesomaximo]) + 2*_delta
    _indice = dim(pesomaximo, algebra) * escalar(
        pesomaximo, _pesomas2delta, tensormetricO)/_dim_adj
    return int(round(_indice))


def congruencia(pesomaximo, algebra):
    """
    congruencia(pesomaximo, algebra):
    ---------------------------------

    Calcula la clase de congruencia de la irrep, usamos el siguiente comando

        Dynkin >> p(congruencia([0,0,0,1],A4))
        4

    La clase de congruencia es la generación del concepto de trialidad en el
    grupo SU(3).

    """
    tipO = algebra.tipo
    if tipO == "A":
        i = 1
        _clase = 0
        for _componente in pesomaximo:
            _clase = int(_componente)*i + _clase
```

```
            i = i + 1
        _clase = _clase % (len(pesomaximo) + 1)
    elif tipO == "D":
        _clase = sp.zeros(1, len(pesomaximo))
        rangomenosdos = int(len(pesomaximo) - 2)
        for i in range(len(pesomaximo) / 2):
            _clase = _clase + sp.Matrix([0, 2 * int(pesomaximo[2*i+1])])
            _clase = _clase + sp.Matrix([0, rangomenosdos * int(
                pesomaximo[rangomenosdos]) + (rangomenosdos + 2) * int(
                    pesomaximo[rangomenosdos + 1])])
            _clase = _clase + sp.Matrix([int(pesomaximo[rangomenosdos]) + int(
                pesomaximo[rangomenosdos + 1] % 2), 0])
            _clase[1] = _clase[1] % 4
    elif tipO == "E":
        if len(pesomaximo) == 6:
            _clase = (pesomaximo[0] - pesomaximo[1] + pesomaximo[3] -
                      pesomaximo[4])
            _clase = _clase % 3
        elif len(pesomaximo) == 7:
            _clase = pesomaximo[3] + pesomaximo[5] + pesomaximo[6] % 2
        elif len(pesomaximo) == 8:
            _clase = 0
    return _clase


def conjugacion(pesomaximo, algebra):
    tipO = algebra.tipo
    """
    conjugacion(pesomaximo, algebra):
    ---------------------------------


    Calcula  peso máximo de la representación conjugada,


        Dynkin >> p(conjugacion([0,0,0,1],a4))
        [1, 0, 0, 0]

    La conjugación respeta las reglas teóricas.
    """
    if tipO == "A":
        _conjugado = pesomaximo.copy()
        _conjugado.reverse()
    elif tipO == "D":
        if len(pesomaximo) % 2 == 1:
            _conjugado = pesomaximo.copy()
            _aux = _conjugado.pop(-2)
            _conjugado.append(_aux)
        if len(pesomaximo) % 2 == 0:
            _conjugado = pesomaximo.copy()
```

```
    elif tipO == "E":
        if len(pesomaximo) == 6:
            _conjugado = pesomaximo.copy()
            _conjugado.reverse()
            _aux = _conjugado.pop(0)
            _conjugado.insert(5, _aux)
        elif len(pesomaximo) == 7:
            _conjugado = pesomaximo.copy()
        elif len(pesomaximo) == 8:
            _conjugado = pesomaximo.copy()
    return _conjugado


def complejidad(pesomaximo, algebra):
    """
    complejidad(pesomaximo, algebra):
    -------------------------------


    Calcula si una irrep es compleja o no

        Dynkin >> p(complejidad([1,0,0,1], a4))
        *

    Los posibles resultados son: "*" si la irrep
    es compleja y "" si la irrep es


    """
    _conjugado = conjugacion(pesomaximo, algebra)
    _clase = congruencia(pesomaximo, algebra)
    if pesomaximo == _conjugado:
        _complejo = " "
    if pesomaximo != _conjugado:
        _clase_conjugado = congruencia(_conjugado, algebra)
        if _clase > _clase_conjugado:
            _complejo = "*"
        if _clase < _clase_conjugado:
            _complejo = " "
        if _clase == _clase_conjugado:
            pe = int(''.join(str(e) for e in pesomaximo))
            co = int(''.join(str(e) for e in _conjugado))
            if pe < co:
                _complejo = "*"
            if pe > co:
                _complejo = " "
    return _complejo


def nombre(pesomaximo, algebra):
```

```
    """
    nombre(pesomaximo, algebra)
    ---------------------------

    Nos devuelve el nombre de la irrep, dando el pesomaximo y el algebra
    donde con nombre nos referimos a dim+complejidad

        Dynkin >> p(nombre([1,0,0,0],A4))
        '5 '
        Dynkin >> p(nombre([0,0,1,0],A4))
        '10*'

    """
    _complejidad = complejidad(pesomaximo, algebra)
    _dimension = dim(pesomaximo, algebra)
    _nombre = str(_dimension)+_complejidad
    return _nombre
```

A.5. Módulo Rupturas

A partir del peso máximo de un representación irreducible de una álgebra, podemos calcular el rompimiento del sistema de pesos asociado a éste peso máximo en las representaciones irreducibles de las subálgebras dadas por la regla de rompimiento. El módulo ruptura se encarga de este proceso a través de las funciones:

- Rompimiento
- Ruptura.

Dependiendo si buscamos una lista de las dimensiones y complejidades de las irreps de las subálgebras asociadas por la regla de rompimiento o una cadena con la misma información.

```
# -*- coding: utf-8 -*-

# ------------------------------------------------------------
# Enrique_Escalante-Notario
# Instituto de Fisica, UNAM
# email: <enriquescalante@gmail.com>
# Distributed under terms of the GPLv3 license.
# Módulo ruptura
# ------------------------------------------------------------


from algebra import *
from pesos import *
from invariantes import *
import sympy as sp

# Anotaciones
```

```
# 1) raices_eliminadas = lista de posiciones de las raices desde 1.
# 2) Sympy inicia sus contadores desde 1.
# 3) la salida "peso" es una lista de enteros.


# tensormetico del algebra a romper
def __ruptura__(peso, regla, tensormetrico, contador):
    # Hacemos la ruptura para un peso respetando la regla
    # out [peso1,peso2,..]
    _ruptura = []
    _particion = regla.get(list(regla.keys())[0])
    _algebras = list(regla.keys())[0].split()
    _ruptura.append(contador)
    for i in range(len(_particion)):
        _alg = _algebras[i]
        if _alg != "U1":
            _parte = []
            for _indice in _particion[i]:
                _parte.append(peso[_indice - 1])
            _ruptura.append(_parte)
        if _alg == "U1":
            _ind = int(_particion[i][0])
            # aqui es donde se asigna la carga para las U1
            _carga = tensormetrico.row(_ind-1).dot(peso)
            _ruptura.append(_carga)
    return _ruptura


# algebra se refiere al algebra a romper, regla es branching rule, y pesomaximo
# el peso maximo del irrep que queremos romper
def __rompimiento__(pesomaximo, algebra, regla):
    # Hacemos el rompimiento para un sistema de pesos completo
    # out [peso1, peso2,...]
    tensormetricO = algebra.tensormetrico()
    _pesos = pesos(pesomaximo, algebra)
    _sistema = []
    contador = 0
    for _peso in _pesos:
        _sistema.append(__ruptura__(_peso, regla, tensormetricO, contador))
        contador = contador + 1
    return _sistema


# raicessimples de la subalgebra
def __esmaximo__(sistema, raicessimples):
    # Determina si un peso es un peso maximo en un sistema de pesos
    maximo = []
    for _peso in sistema:
        _subidos = []
        for _raiz in raicessimples:
```

```
            _peso1 = (sp.Matrix([_peso]) + sp.Matrix([_raiz])).tolist()[0]
            _subidos.append(_peso1)
            _aux = [elem for elem in _subidos if elem not in sistema]
            if len(_aux) == len(_peso) and all(com >= 0 for com in _peso):
                maximo = _peso
    return maximo


# el tipo y rango son de subalgebra
def __busqueda__(subrompimiento, tipo, rango):
    # Nos devuelve los indices de los pesos con respecto la lista original
    # del sistema de pesos resultado de un rompimiento y el peso maximo de
    # dicha irrep de la subalgebra
    # out {[indices], pesomaximo}
    _subgrupo = []
    _complemento_sub = []
    _indice = []
    for _ruptura in subrompimiento:
        if len(_ruptura) > 2:
            _subgrupo.append(_ruptura[1])  # _ruptura[0] es el indice con respecto
                al sistema original
            _complemento_sub.append([_ruptura[0]] + _ruptura[2:])
        if len(_ruptura) == 2:
            _subgrupo.append(_ruptura[1])
            _complemento_sub.append(_ruptura[:])
    _indices_dim = {}
    if tipo != "U":
        Alg = alg(tipo, rango)  # subalgebra
        while _subgrupo != []:
            _maximo = __esmaximo__(_subgrupo, Alg.raicessimples())
            _indice_maximo = _subgrupo.index(_maximo)
            _pesos_maximo = pesos(_maximo, Alg)
            _complemento_maximo = _complemento_sub[_indice_maximo][1:]
            _indices_maximo = []
            for _peso_maximo in _pesos_maximo:
                for i in range(len(_subgrupo)):
                    if (_subgrupo[i] == _peso_maximo and
                            _complemento_sub[i][1:] == _complemento_maximo):
                        _indices_maximo.append(_complemento_sub[i][0])
                        _complemento_sub.pop(i)
                        _subgrupo.pop(i)
                        break
            _indices_dim[repr(_indices_maximo)] = _maximo
        # se unen todos los indices para un mismo peso maximo
        _indices_dim_finales = {}
        for _indices1 in _indices_dim.keys():
            _indices_aux = []
            for _indices2 in _indices_dim.keys():
                if _indices_dim.get(_indices1) == _indices_dim.get(_indices2):
                    _indices_aux.extend(eval(_indices2))
```

```
117                 _indices_dim_finales[repr(_indices_aux)] = _indices_dim.get(_indices1)
                        # devuelve {[indices]: pesomaximo}
118             # construimos el conjunto de complementos asociados a cada peso maximo
119             _complementos_maximo = []
120             for _indices in _indices_dim_finales:
121                 _Indices = eval(_indices)
122                 _complemento_peso = []
123                 for _indice_peso in _Indices:
124                     for _ruptura in subrompimiento:
125                         if _indice_peso == _ruptura[0]:
126                             _complemento_peso.append([_ruptura[0]] + _ruptura[2:])
127                 _complementos_maximo.append(_complemento_peso)
128             return _indices_dim_finales, _complementos_maximo
129         if tipo == "U":
130             while _subgrupo != []:
131                 _maximo = max(_subgrupo)
132                 _indice_maximo = _subgrupo.index(_maximo)
133                 _complemento_maximo = _complemento_sub[_indice_maximo][1:]
134                 _indices_maximo = []
135                 for _peso_maximo in _subgrupo:
136                     _indice_peso = _subgrupo.index(_peso_maximo)
137                     if _peso_maximo == _maximo and _complemento_maximo ==
                            _complemento_sub[_indice_peso][1:]:
138                         _indices_maximo.append(_complemento_sub[_indice_peso][0])
139                         _complemento_sub.pop(_indice_peso)
140                         _subgrupo.pop(_indice_peso)
141                         break
142                 _indices_dim[repr(_indices_maximo)] = _maximo
143             # se unen todos los indices para un mismo peso maximo
144             _indices_dim_finales = {}
145             for _indices1 in _indices_dim.keys():
146                 _indices_aux = []
147                 for _indices2 in _indices_dim.keys():
148                     if _indices_dim.get(_indices1) == _indices_dim.get(_indices2):
149                         _indices_aux.extend(eval(_indices2))
150                 _indices_dim_finales[repr(_indices_aux)] = _indices_dim.get(_indices1)
151             # construimos el conjunto de complementos asociados a cada peso maximo
152             _complementos_maximo = []
153             for _indices in _indices_dim_finales:
154                 _Indices = eval(_indices)
155                 _complemento_peso = []
156                 for _indice_peso in _Indices:
157                     for _ruptura in subrompimiento:
158                         if _indice_peso == _ruptura[0]:
159                             _complemento_peso.append([_ruptura[0]] + _ruptura[2:])
160                 _complementos_maximo.append(_complemento_peso)
161             return _indices_dim_finales, _complementos_maximo
162 
163 
164     # algebra es el algebra a romper
```

```
165 def __descomposicion__(pesomaximo, regla, algebra):
166     # Nos devuelve una listas con arrays que contiene los pesosmaximos de cada
        subalgebra.
167     # out [[peso1,..],...]
168     _subalgebras = list(regla.keys())[0].split()
169     _rompimi = __rompimiento__(pesomaximo, algebra, regla)  # es una lista con los
        rompimientos.
170     _conrompi = [_rompimi]
171     _dicc_irreps = []  # es un array con dicc's de todas las irreps de todas las
        subalgebras
172     # se obtiene la lista con los diccionarios para cada subalgebra
173     for _posicion in range(len(_subalgebras)):
174         _conaux = []
175         _subalg = _subalgebras[_posicion][0]
176         _rango = int(_subalgebras[_posicion][1:])  # IMPORTANTE: depende de la
            correcta escritura de la branching rule
177         for _subrompimiento in _conrompi:
178             _indices, _conaux1 = __busqueda__(_subrompimiento, _subalg, _rango)
179             _dicc_irreps.append(_indices)
180             _conaux.extend(_conaux1)
181         _conrompi = _conaux
182     _descom = []
183     # corremos sobre los indices de los rompimientos
184     for _indice in range(len(_rompimi)):
185         _irrep_subalg_dim = []
186         # tomamos un diccionario
187         for _subalgeb in _dicc_irreps:  # _subalgeb == {'[indices]':pesomaximp}
188             for _indices in _subalgeb.keys():
189                 # obtenemos los indices de cada irreps de una subalgebra
190                 _Indices = eval(_indices)
191                 for _Indice in _Indices:
192                     # corremos por los indices de la irreps de un subalagebra
193                     if _Indice == _indice:
194                         _irrep_subalg_dim.append(_subalgeb.get(_indices))  # agrego
                            el peso de la irrep
195         _descom.append(_irrep_subalg_dim)
196     _descomposicion = []
197     for _key in _descom:  # elimina los registros repetidos
198         if _key not in _descomposicion:
199             _descomposicion.append(_key)
200     return _descomposicion
201
202
203 # algebra es la algebra a romper.
204 def rompimiento(pesomaximo, regla, algebra):
205     # Nos devuelve las dimensiones y complexificaciones del rompimiento
206     # out [dim1,..]
207     tipO = algebra.tipo
208     rangO = algebra.rango
209     _descomposicion = __descomposicion__(pesomaximo, regla, algebra)
```

```
210         _subalgebras = list(regla.keys())[0].split()  # IMPORTANTE: depende de la
                correcta escritura de la branching rule.
211         _descomposiciones = []
212         for _descom in _descomposicion:
213             _dim_hwv = []
214             for i in range(len(_descom)):
215                 _tipo = _subalgebras[i][0]
216                 _rango = int(_subalgebras[i][1:])
217                 _pesomaximo = _descom[i]
218                 if _tipo != "U":
219                     z = alg(_tipo, _rango)  # se crea el objeto subalgebra
220                     _dim = dim(_pesomaximo, z)
221                     _compl = complejidad(_pesomaximo, z)
222                     _dim_hwv.append(str(_dim)+_compl)  # se almacenan cadenas
223                 if _tipo == "U":
224                     _dim_hwv.append(_pesomaximo)  # se almacena un numero
225             _descomposiciones.append(_dim_hwv)
226         return _descomposiciones
227
228
229     def ruptura(pesomaximo, regla, algebra):
230         """
231         ruptura(pesomaximo, regla, algebra)
232         ----------------------------------------
233
234         Nos devuelve la suma de irreps del subalgebra, que estan de acuerdo con
235         la regla dada, suponiendo una regla del la forma
236
237             Dynkin >> regla = {'A2 A1 U1': [[3,4],[1],[2]]}
238
239         Donde podemos intuir que el Algebra A_2 esta formado por las raices
240         simples etiquetas con los indices 3 y 4 en el diagrama de Dynkin.
241         Si hacemos el rompimiento de la irreps 5 y 10 de A_4
242
243             Dynkin >> p(ruptura([1,0,0,0], regla, A4))
244             5 = (1 ,2 )_3/5 + (3 ,1 )_-2/5
245             Dynkin >> p(ruptura()[0,1,0,0], regla, A4)
246             10 = (1 ,1 )_6/5 + (3 ,2 )_1/5 + (3*,1 )_-4/5
247
248         La salida corresponde en notacion común a 5 = 1x2_{3/5} + 3x1_{-2/5}
249         y 10 = 1x1_{6/5} + 3x2_{1/5} + 3*x1_{-4/5}.
250         En el producto de irreps, la primera correspode a A_2, la segunda
251         a A_1 y el ultimo factor a la simentria U_1.
252         """
253         _romp = rompimiento(pesomaximo, regla, algebra)
254         nom = nombre(pesomaximo, algebra)
255         descom = []
256         for elemento in _romp:
257             irreps = []
258             cargas = []
```

```
        for componente in elemento:
            if type(componente) == str:
                irreps.append(componente)
            if type(componente) != str:
                cargas.append(componente)
        irrepsstr = '('+','.join(irreps)+')'
        cargasstr = ','.join(str(e) for e in cargas)
        descom.append(irrepsstr+'_'+cargasstr)
    return nom+' = '+ '.join(descom)
```

A.6. Módulo Tensoriales

El módulo Tensoriales nos entrega el producto tensorial de dos representaciones irreducibles de una álgebra definida para dos pesos máximo asociados, a través de la función:

- Ptensorial.

<++>

```
# -*- coding: utf-8 -*-

# ------------------------------------------------------------
# Enrique_Escalante-Notario
# Instituto de Fisica, UNAM
# email: <enriquescalante@gmail.com>
# Distributed under terms of the GPLv3 license.
# Módulo tensoriales
# ------------------------------------------------------------


import sympy as sp
from pesos import *
from invariantes import *


def __esmaximo__(sistema, raicessimples):
    # Determina si un peso es un peso maximo en un sistema de pesos
    maximo = []
    for _peso in sistema:
        _subidos = []
        for _raiz in raicessimples:
            _peso1 = (sp.Matrix([_peso]) + sp.Matrix([_raiz])).tolist()[0]
            _subidos.append(_peso1)
            _aux = [elem for elem in _subidos if elem not in sistema]
            if len(_aux) == len(_peso) and all(com >= 0 for com in _peso):
                maximo = _peso
    return maximo

```

```
30
31  def __producto__(peso1, peso2, algebra):
32      # Calcula el producto tensorial de dos irrep de una algebra
33      tip0 = algebra.tipo
34      raicessimpleS = algebra.raicessimples()
35      dimensiones = []
36      sistema1 = pesos(peso1, algebra)
37      sistema2 = pesos(peso2, algebra)
38      peso3 = (sp.Matrix([peso1]) + sp.Matrix([peso2])).tolist()[0]
39      sistema3 = pesos(peso3, algebra)
40      dimensiones.append(nombre(peso1, algebra))
41      dimensiones.append(nombre(peso2, algebra))
42      dimensiones.append(nombre(peso3, algebra))
43      sistema4 = []
44      for _peso1 in sistema1:
45          for _peso2 in sistema2:
46              sistema4.append(
47                  (sp.Matrix(
48                      [_peso1]) +
49                      sp.Matrix(
50                      [_peso2])).tolist()[0])
51      for _peso3 in sistema3:
52          sistema4.remove(_peso3)
53      while sistema4 != []:
54          _maximo = __esmaximo__(sistema4, raicessimpleS)
55          sistema5 = pesos(_maximo, algebra)
56          dimensiones.append(nombre(_maximo, algebra))
57          for _peso5 in sistema5:
58              sistema4.remove(_peso5)
59      return dimensiones
60
61
62  def ptensorial(peso1, peso2, algebra):
63      """
64      ptensorial(peso1, peso2, algebra)
65      -----------------------------------
66
67      El software dynkin puede calcular productos tensoriales de irreps
68      de una algebra determinada
69
70          Dynkin >> hwv1 = [1,0,0,0]
71          Dynkin >> hwv2 = [0,1,0,0]
72          Dynkin >> A4 = alg('A',4)
73          Dynkin >> p(ptensorial(hwv1,hwv2,A4)
74          5 x 10 = 40* + 10*
75
76
77      """
78      _produc = __producto__(peso1, peso2, algebra)
79      _factores = ' x '.join(_produc[0:2])
```

```
80      _result = ' + '.join(_produc[2:])
81      return _factores+' = '+_result
```

A.7. Módulo Espectro

Una vez conseguido el rompimiento de una representación irreducible de una álgebra dada, podemos obtener el asignación de la hipercarga para éste rompimiento junto con las U(1) extras (si existen en la regla de rompimiento) ortogonales a la hipercarga. La función encargada de este proceso es:

- Espectro.

Además podemos obtener los generadores de U(1) y los generadores de la hipercarga junto con los generadores ortogonales a dicho generador, a través de las funciones:

- Yg.
- Yug,

Respectivamente.

```
1   # -*- coding: utf-8 -*-
2
3   # ------------------------------------------------------------
4   # Enrique_Escalante-Notario
5   # Instituto de Fisica, UNAM
6   # email: <enriquescalante@gmail.com>
7   # Distributed under terms of the GPLv3 license.
8   # Módulo espectro
9   # ------------------------------------------------------------
10
11  from algebra import *
12  from rupturas import *
13  from invariantes import *
14  import sympy as sp
15
16
17  # Anotaciones
18  # 1) raices_eliminadas = lista de posiciones de las raices desde 1
19  # 2) Sympy inicia sus contadores desde 1
20  # 3) la salida "peso" es una lista de enteros
21  # 4) en la mayoria rompimiento debe contener a las
22  #    particulas del modelo estandar
23
24  def _variables(regla):
25      # Nos devuelve una lista con varibles de sympy
26      # Este sera usado para resolver sistema de ecuaciones
27      subalgebras = list(regla.keys())[0].split()
28      numdeU = 0
```

```
    var = []
    for i in range(len(subalgebras)):
        tipo = subalgebras[i][0]
        if tipo == "U":
            numdeU += 1
    for j in range(numdeU):
        var.append(sp.symbols("x"+str(j+1)))
    return (var)


def _variablesop(numero):
    # Nos devuelve una lista con varibles de sympy
    var = []
    for i in range(numero):
        var.append(sp.symbols("x"+str(i+1)))
    return var


def _quiralidad(rompimiento):
    # rompimiento debe de contener a las particulas del SM
    # Nos devuelve la quiralidad del rompimiento que tiene a las particulas
    # del modelos estandar
    # Out {indicequark:quiralidad}
    ind_quiral = {}
    for i in range(len(rompimiento)):
        if rompimiento[i][0:2] == ["3 ", "2 "]:
            ind_quiral[str(i)] = 1
        if rompimiento[i][0:2] == ["3*", "2 "]:
            ind_quiral[str(i)] = -1
    return ind_quiral


def _combinaciones(rompimiento, regla, indice_quark, quiral):
    # Nos devuelve los coeficientes de todas las posibles combinaciones lineales
    # de los generadores de U(1), tal manera que podemos asignar el operador de
    # hipercarga,
    # IMPORTANTE: rompimiento debe contener a las particulas de SM
    # out [[combi],...]
    quark = {}
    rquark = {}
    lepton = {}
    rlepton = {}
    soluciones = []
    if quiral == 1:
        RQuark = ["3*", "1 "]
        Lepton = ["1 ", "2 "]
        RLepton = ["1 ", "1 "]
    if quiral == -1:
        RQuark = ["3 ", "1 "]
        Lepton = ["1 ", "2 "]
```

```
79              RLepton = ["1 ", "1 "]
80          indice = 0  # este es el indice de la descomposion
81          cargasQuark = rompimiento[int(indice_quark)][2:]
82          for ruptura in rompimiento:
83              if ruptura[0:2] == RQuark:
84                  rquark[str(indice)] = ruptura[2:]
85              if ruptura[0:2] == Lepton:
86                  lepton[str(indice)] = ruptura[2:]
87              if ruptura[0:2] == RLepton:
88                  rlepton[str(indice)] = ruptura[2:]
89              indice = indice + 1
90          for indLepton, cargasLepton in lepton.items():
91              for indRlepton1, cargasRlepton1 in rlepton.items():
92                  for indRlepton2, cargasRlepton2 in rlepton.items():
93                      for indRquark1, cargasRquark1 in rquark.items():
94                          for indRquark2, cargasRquark2 in rquark.items():
95                              if indRquark1 != indRquark2:
96                                  if indRlepton1 != indRlepton2:
97                                      # [q, l, e, v, u, d] Nota: confirmar hipercargas
98                                      A = sp.Matrix(
99                                          [cargasQuark, cargasLepton,
100                                             cargasRlepton1, cargasRlepton2,
101                                             cargasRquark1, cargasRquark2])
102                                     b = sp.Matrix(
103                                         [sp.Rational(1, 6),
104                                             sp.Rational(-1, 2),
105                                             sp.Integer(1),
106                                             sp.Integer(0),
107                                             sp.Rational(-2, 3),
108                                             sp.Rational(1, 3)])
109                                     solucion = sp.linsolve(
110                                         (A, b), _variables(regla))  # A var = b
111                                     if solucion != sp.EmptySet():
112                                         solucion1 = []
113                                         for elemento in list(
114                                                 list(solucion)[0]):
115                                             solucion1.append(elemento)
116                                         soluciones.append(repr(solucion1))
117         aux = list(set(soluciones))
118         for i in range(len(aux)):
119             aux[i] = sp.sympify(aux[i])
120         return aux
121
122
123 def ug(algebra, regla):
124     """
125     ug(algebra, regla)
126     --------------------------------
127
128     Nos devuelve el conjunto de generadores de U(1), compatibles con
```

```
129         con la regla de rompimiento
130 
131             Dynkin >> p(ug(E6, regla))
132             [[5/3, 10/3, 4, 8/3, 4/3, 2],
133             [4/3, 8/3, 4, 10/3, 5/3, 2],
134             [2/3, 4/3, 2, 5/3, 4/3, 1]]
135 
136 
137         """
138         # out [[ugenerador1],[ugenerador2],...]
139         tensorMetrico = algebra.tensormetrico()
140         particion = regla.get(list(regla.keys())[0])
141         ugenerators = []
142         subalgebras = list(regla.keys())[0].split()
143         for i in range(len(particion)):
144             subalgebra = subalgebras[i]
145             if subalgebra == "U1":
146                 ind = int(particion[i][0]) - 1
147                 ugenerators.append(tensorMetrico.row(ind))
148         return ugenerators
149 
150 
151     def yug(hwvsSM, algebra, regla):
152         """
153         yug(pesosSM, algebra, regla)
154         --------------------------------
155 
156         Calcula el conjunto de generadores Y y U(1) adicionales, compatibles con
157         la regla de ruptura. Sabemos que el conjunto no es unico debido a que la
158         asignacion no es unica. Por lo tanto, obtenemos con lista con los
159         correspondientes generadores de la hipercarga y de los genedorea adicionales
160         que son ortogonales a la direccion de la hipercarga.
161         El primer elemento de cada conjunto es el generador de hipercarga.
162         La lista pesosSM debe de contener a las partículas de SM.
163 
164             Dynkin >> p(yug([[1,0,0,0,0,0]],E6,regla))
165             [[[-1/2, -1, -4/3, -1, 0, -2/3]
166             [-76/483, -152/483, 4/161, 170/483, 5/3, 2/161]
167             [38/877, 76/877, -6/877, -85/877, 36/877, -3/877]]
168             [[1/2, 1, 2/3, 0, 0, 1/3]
169             [-20/39, -40/39, 20/13, 10/3, 5/3, 10/13]
170             [2/85, 4/85, -6/85, -13/85, 36/85, -3/85]]
171             [[-1/2, -1, -4/3, -1, -1, -2/3]
172             [-22/591, -44/591, 68/197, 350/591, -635/591, 34/197]
173             [-34/589, -68/589, -6/589, 59/589, 36/589, -3/589]]]
174 
175         """
176         # out [[ygenerador], [ugenerador],...]
177         rango = algebra.rango
178         ugeneradores = ug(algebra, regla)
```

```
    rompi = []
    for pesomaximo in hwvsSM:
        rompi.extend(rompimiento(pesomaximo, regla, algebra))
    quiralidades = _quiralidad(rompi)
    combtotales = []
    for indice, quiral in quiralidades.items():
        combtotales.extend(_combinaciones(rompi, regla, indice, quiral))
    generadores = []
    for comb in combtotales:
        ygenerador = sp.zeros(1, rango)
        ugeneradores1 = ugeneradores.copy()
        for i in range(len(comb)):
            ygenerador = ygenerador + ugeneradores[i]*comb[i]  # genhipercarga
        for i in range(len(comb)):
            if comb[i] != 0:
                ugeneradores1.pop(i)
                break
        ugeneradores1.insert(0, ygenerador)
        generadores.append(sp.GramSchmidt(ugeneradores1))
    return generadores


def _yyucoeficientes(ugeneradores, yyugeneradores):
    # Devuelve la matriz de cambio de base entre generadores originales
    # y los generadores con y y u_ortogonales
    # out [[coeficiones1],[coeficientes2],...]
    combs = []
    numdeU = len(ugeneradores)
    A = sp.Matrix(ugeneradores).T
    for orthogonal in yyugeneradores:
        b = sp.Matrix(orthogonal).T
        combs.append(
            list(
                list(
                    sp.linsolve(
                        (A, b), _variablesop(numdeU)))[0]))
    return combs


def _hipercarga(rompimiento, yyucoeficientes):
    # Devuelve la reasignación de cargas U1 con Y adecuadamente fija para todo
    # el rompimiento.
    # Out [peso1, ...]
    rup = []
    coeficientes_matrix = sp.Matrix(yyucoeficientes)
    for descom in rompimiento:
        nuevaDescom = descom[:2]  # hay que generalizar
        cargaDescom = sp.Matrix([descom[2:]])
        rup.append(nuevaDescom + list(coeficientes_matrix * cargaDescom.T))
    return rup
```

```
229
230
231 def espectro(hwvs, algebra, regla, particulas):
232     """
233     espectro(pesos, algebra, regla, particulas)
234     -------------------------------------------
235
236     Nos devuelve el rompimiento en las irreps de las subalgebra (respetando
237     la regla) junto con la asignación de la hipercarga y las cargas U1 (si
238     existen) para cada peso de la lista pesos, la asignación se la hipercarga
239     se hizo tomando en cuenta las irreps del algebra que contienen a las
240     partículas de SM, estas se enlistan en el conjunto particulas.
241
242         Dynkin >> E6 = alg('E', 6)
243         Dynkin >> regla = {"A2 A1 U1 U1 U1": [[3,6],[1],[2],[4],[5]]}
244         Dynkin >> p(espectro([[1,0,0,0,0,0],[0,0,0,0,0,1]],E6,regla,[[1,0,0,0,0,0]])
                )
245         [['27 ',
246         [['1 ', '2 ', -1/2, -76/483, 38/877]
247         ['3 ', '1 ', -1/3, 164/483, -82/877]
248         ['3*', '1 ', 1/3, 641/483, 118/877]
249         ['3*', '1 ', -2/3, -799/483, -39/877]
250         ['1 ', '2 ', 1/2, 881/483, -2/877]
251         ['1 ', '2 ', -1/2, -559/483, -159/877]
252         ['1 ', '1 ', 0, -2/3, 161/877]
253         ['3 ', '2 ', 1/6, -82/483, 41/877]
254         ['3*', '1 ', 1/3, 158/483, -79/877]
255         ['1 ', '1 ', 1, 635/483, 121/877]
256         ['1 ', '1 ', 0, -5/3, -36/877]]
257
258         [['1 ', '2 ', -1/2, -22/591, -34/589]
259         ['3 ', '1 ', -1/3, 248/591, 2/19]
260         ['3*', '1 ', -2/3, -883/591, -26/589]
261         ['3*', '1 ', 1/3, 737/591, -39/589]
262         ['1 ', '2 ', -1/2, -613/591, 70/589]
263         ['1', '2 ', 1/2, 1007/591, 3/31]
264         ['1 ', '1 ', 0, -2/3, -127/589]
265         ['3 ', '2 ', 1/6, -124/591, -1/19]
266         ['3*', '1 ', 1/3, 146/591, 65/589]
267         ['1 ', '1 ', 0, -5/3, -23/589]
268         ['1 ', '1 ', 1, 635/591, -36/589]]
269
270         [['1 ', '2 ', 1/2, -20/39, 2/85]
271         ['3 ', '1 ', -1/3, 100/39, -2/17]
272         ['3*', '1 ', 1/3, -35/39, 46/85]
273         ['3*', '1 ', 1/3, -35/39, -39/85]
274         ['1 ', '2 ', -1/2, 85/39, 2/5]
275         ['1 ', '2 ', -1/2, 85/39, -3/5]
276         ['1 ', '1 ', 1, -170/39, 1/5]
277         ['3 ', '2 ', 1/6, -50/39, 1/17]
```

```
278           ['3*', '1 ', -2/3, 70/39, -7/85]
279           ['1 ', '1 ', 0, -5/3, 49/85]
280           ['1 ', '1 ', 0, -5/3, -36/85]]]
281
282           ['78 ',
283           [['3*', '1 ', -2/3, 2/161, -3/877]
284           ['1 ', '2 ', -1/2, 82/161, -123/877]
285           ['1 ', '1 ', 0, 1, 197/877]
286           ['3 ', '2 ', 1/6, 241/161, 77/877]
287           ['1 ', '1 ', -1, -319/161, 40/877]
288           ['3 ', '2 ', -5/6, -239/161, -80/877]
289           ['3*', '1 ', 1/3, 321/161, -43/877]
290           ['3*', '2 ', -1/6, -80/161, 120/877]
291           ['3*', '1 ', -2/3, -159/161, -200/877]
292           ['8 ', '1 ', 0, 0, 0]
293           ['1 ', '1 ', 1, 480/161, 157/877]
294           ['1 ', '3 ', 0, 0, 0]
295           ['1 ', '1 ', 0, 0, 0]
296           ['3 ', '1 ', 2/3, 159/161, 200/877]
297           ['1 ', '1 ', -1, -480/161, -157/877]
298           ['3 ', '2 ', 1/6, 80/161, -120/877]
299           ['3 ', '1 ', -1/3, -321/161, 43/877]
300           ['3*', '2 ', 5/6, 239/161, 80/877]
301           ['3*', '2 ', -1/6, -241/161, -77/877]
302           ['1 ', '1 ', 1, 319/161, -40/877]
303           ['1 ', '2 ', 1/2, -82/161, 123/877]
304           ['1 ', '1 ', 0, -1, -197/877]
305           ['3 ', '1 ', 2/3, -2/161, 3/877]]
306
307           [['3*', '1 ', -2/3, 34/197, -3/589]
308           ['1 ', '2 ', -1/2, 124/197, 3/19]
309           ['1 ', '1 ', -1, -343/197, -91/589]
310           ['3 ', '2 ', -5/6, -253/197, 5/589]
311           ['1 ', '1 ', 0, 1, -104/589]
312           ['3 ', '2 ', 1/6, 287/197, -8/589]
313           ['3*', '1 ', -2/3, -163/197, 101/589]
314           ['3*', '2 ', -1/6, -90/197, -96/589]
315           ['3*', '1 ', 1/3, 377/197, 88/589]
316           ['8 ', '1 ', 0, 0, 0]
317           ['1 ', '1 ', -1, -540/197, 13/589]
318           ['1 ', '3 ', 0, 0, 0]
319           ['1 ', '1 ', 0, 0, 0]
320           ['3 ', '1 ', -1/3, -377/197, -88/589]
321           ['1 ', '1 ', 1, 540/197, -13/589]
322           ['3 ', '2 ', 1/6, 90/197, 96/589]
323           ['3 ', '1 ', 2/3, 163/197, -101/589]
324           ['3*', '2 ', -1/6, -287/197, 8/589]
325           ['3*', '2 ', 5/6, 253/197, -5/589]
326           ['1 ', '1 ', 0, -1, 104/589]
327           ['1 ', '2 ', 1/2, -124/197, -3/19]
```

```
328             ['1 ', '1 ', 1, 343/197, 91/589]
329             ['3 ', '1 ', 2/3, -34/197, 3/589]]
330 
331             [['3*', '1 ', 1/3, 10/13, -3/85]
332             ['1 ', '2 ', -1/2, 50/13, -3/17]
333             ['1 ', '1 ', 1, -35/13, 53/85]
334             ['3 ', '2 ', 1/6, 5/13, 41/85]
335             ['1 ', '1 ', 1, -35/13, -32/85]
336             ['3 ', '2 ', 1/6, 5/13, -44/85]
337             ['3*', '1 ', -2/3, 45/13, 29/85]
338             ['3*', '2 ', 5/6, -40/13, 12/85]
339             ['3*', '1 ', -2/3, 45/13, -56/85]
340             ['8 ', '1 ', 0, 0, 0]
341             ['1 ', '1 ', 0, 0, 1]
342             ['1 ', '3 ', 0, 0, 0]
343             ['1 ', '1 ', 0, 0, 0]
344             ['3 ', '1 ', 2/3, -45/13, 56/85]
345             ['1 ', '1 ', 0, 0, -1]
346             ['3 ', '2 ', -5/6, 40/13, -12/85]
347             ['3 ', '1 ', 2/3, -45/13, -29/85]
348             ['3*', '2 ', -1/6, -5/13, 44/85]
349             ['3*', '2 ', -1/6, -5/13, -41/85]
350             ['1 ', '1 ', -1, 35/13, 32/85]
351             ['1 ', '2 ', 1/2, -50/13, 3/17]
352             ['1 ', '1 ', -1, 35/13, -53/85]
353             ['3 ', '1 ', -1/3, -10/13, 3/85]]]]
354 
355         """
356         # out [[simpy,matrix]],
357         # el primer de las lista mas interna es un nombre de la la irrep
358         # que se rompio del algebra original
359         _ugen = ug(algebra, regla)
360         _contenido = []
361         _yygen = yug(particulas, algebra, regla)
362         for hwv in hwvs:
363             romp = rompimiento(hwv, regla, algebra)
364             nomb = nombre(hwv, algebra)
365             rompi = []
366             for yyug in _yygen:
367                 rompi.append(_hipercarga(romp, _yyucoeficientes(_ugen, yyug)))
368             rompi.insert(0, nomb)
369             _contenido.append(rompi)
370         return _contenido
```

A.8. Módulo Main

El módulo Main se centran en presentan la información de una manera clara al usuario, Esto se consigue con las funciones:

- P.
- H.

```
# -*- coding: utf-8 -*-

# ------------------------------------------------------
# Enrique_Escalante-Notario
# Instituto de Fisica, UNAM
# email: <enriquescalante@gmail.com>
# Distributed under terms of the GPLv3 license.
# Módulo main
# ------------------------------------------------------

from algebra import *
from pesos import *
from invariantes import *
from tensoriales import *
from rupturas import *
from informacion import *
from espectro import *
import fcntl
import termios
import struct
import sys
import time
import os
import rlcompleter
import readline
import re
readline.parse_and_bind('tab:complete')  # para completar los comandos con tab
version = 1.0  # Aquí va la versión del programa.


def prompt():
    """Este es el prompt de Dynkin"""
    __limpiarpantalla__()
    print(__logo__())
    cmd = None
    try:
        while True:
            cmd = input("Dynkin >> ")
            if cmd not in ["salir"]:
                try:
                    exec(cmd)
                except:
                    print("Comando erroneo: Obtener el listado de comandos con "
                          "p(comandos) y documentación de los comandos "
                          "h(comando)")
            if cmd in ["salir", "exit"]:
```

```
                print("\nAdiós...")
                time.sleep(0.1)
                __limpiarpantalla__()
                sys.exit(0)
    except KeyboardInterrupt:
        print("\nAdiós...")
        time.sleep(0.1)
        __limpiarpantalla__()
        sys.exit(0)


def __logo__():
    """Return text logo"""
    logo_txt = r"""
 ______              _
|  __ \             | |   (_)
| |  | |_   _ _ __  | | ___ _ __
| |  | | | | | '_ \| |/ / | '_ \
| |__| | |_| | | | |   <| | | | |
|_____/ \__, |_| |_|_|\_\_|_| |_|
         __/ |
        |___/"""
    autores = " E. Escalante-Notario, S. Ramos-Sánchez"
    logo_txt = logo_txt + "v."+str(version)+" "+autores
    lines = logo_txt.split("\n")
    length = max(len(x) for x in lines)
    x, y = __terminal_size__()
    indent = (x - length - 1) // 2
    newlines = (y - 12) // 2
    indent, newlines = (0 if x < 0 else x for x in (indent, newlines))
    lines = [" " * indent + l for l in lines]
    logo_txt = "\n".join(lines) + "\n" * newlines
    return logo_txt


def __terminal_size__():
    """Esta función obtiene el tamaño de la pantalla"""
    h, w, hp, wp = struct.unpack(
        'HHHH', fcntl.ioctl(
            0, termios.TIOCGWINSZ, struct.pack(
                'HHHH', 0, 0, 0, 0)))
    return w, h


def __limpiarpantalla__():
    LIMPIAR = "clear" if sys.platform.startswith("linux") else "cls"
    os.system(LIMPIAR)


def p(argumento):
```

```
97      """
98      p(comando)
99      -------------------------------
100
101     Este comando imprime los resultados en la pantalla.
102
103         Dynkin >> p(simples(A4))
104         [[ 2, -1, 0, 0]
105         [-1, 2, -1, 0]
106         [0, -1, 2, -1]
107         [0, 0, -1, 2]]
108
109     """
110     if argumento != comandos:
111         arg = str(argumento)
112         arg = arg.replace("Matrix", "")
113         arg = arg.replace("([", "")
114         arg = arg.replace("])", "")
115         arg = arg.replace("], [", "]\n[")
116         arg = arg.replace("[[", "\n[[")
117         print(arg)
118     else:
119         arg = str(argumento)
120         arg = arg.replace("[", "")
121         arg = arg.replace("]", "")
122         arg = arg.replace(",", "\n")
123         print(arg)
124
125
126 def h(argumento):
127     """
128     h(comando)
129     -------------------------------
130
131     Este comando se encarga de presentar la documentacion sobre los comandos
132     soportados por Dynkin
133
134         Dynkin >> Ayuda(Ayuda)
135
136
137     """
138     print(argumento.__doc__)
139
140
141 comandos = sorted(["h", "p", "alg", "simples",
142                    "positivas",
143                    "maxima", "orden", "mcartan", "tmetrico",
144                    "ddynkin", "pesos", "dim", "ind",
145                    "congruencia", "conjugacion", "complejidad",
146                    "ptensorial", "ruptura", "nombre",
```

```
                      "ug", "yug", "espectro"
                      ])
prompt()
```

Bibliografía

[1] Georges Aad, T Abajyan, B Abbott, J Abdallah, S Abdel Khalek, AA Abdelalim, O Abdinov, R Aben, B Abi, M Abolins, et al. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29, 2012.

[2] Howard Georgi. H. georgi and sl glashow, phys. rev. lett. 32, 438 (1974). *Phys. Rev. Lett.*, 32:438, 1974.

[3] H Fritzsch. H. fritzsch and p. minkowski, ann. phys.(ny) 93, 193 (1975). *Ann. Phys.(NY)*, 93:193, 1975.

[4] Jogesh C Pati and Abdus Salam. Lepton number as the fourth "color". In *Selected Papers Of Abdus Salam: (With Commentary)*, pages 343–357. World Scientific, 1994.

[5] A De Rújula, H Georgi, and SL Glashow. Trinification of all elementary particle forces. In *Fifth Workshop on Grand Unification*, page 88. World Scientific, Singapore, 1984.

[6] A Hartanto and LT Handoko. Grand unified theory based on the su (6) symmetry. *Physical Review D*, 71(9):095013, 2005.

[7] Feza Gürsey, Pierre Ramond, and Pierre Sikivie. A universal gauge theory model based on e6. *Physics Letters B*, 60(2):177–180, 1976.

[8] Richard Slansky. *Group theory for unified model building*, volume 79. Elsevier, 1981.

[9] Gregory W Anderson and Tomás Blazek. E 6 unification model building. i. clebsch–gordan coefficients of 27x27. *Journal of Mathematical Physics*, 41(7):4808–4816, 2000.

[10] In-Guy Koh, J Patera, and C Rousseau. Clebsch–gordan coefficients for e 6 and so (10) unification models. *Journal of mathematical physics*, 25(10):2863–2872, 1984.

[11] Howard Georgi and Sheldon L Glashow. Unity of all elementary-particle forces. *Physical Review Letters*, 32(8):438, 1974.

[12] Murray Gell-Mann, Pierre Ramond, and R Slansky. Color embeddings, charge assignments, and proton stability in unified gauge theories. *Reviews of Modern Physics*, 50(4):721, 1978.

[13] Oleg Lebedev and Yann Mambrini. Axial dark matter: the case for an invisible z'. *Physics Letters B*, 734:350–353, 2014.

[14] Nicole F Bell, Yi Cai, Rebecca K Leane, and Anibal D Medina. Leptophilic dark matter with z' interactions. *Physical Review D*, 90(3):035027, 2014.

[15] Stefano Di Chiara, Venus Keus, and Oleg Lebedev. Stabilizing the higgs potential with a z′. *Physics Letters B*, 744:59–66, 2015.

[16] Junji Hisano, Yu Muramatsu, Yuji Omura, and Masato Yamanaka. Flavor violating z′ from so (10) susy gut in high-scale susy. *Physics Letters B*, 744:395–400, 2015.

[17] Eduardo Rojas and Jens Erler. Alternative z'bosons in e_6. *JHEP*, 10:063, 2015.

[18] Garrett Birkhoff and Saunders Mac Lane. *A survey of modern algebra*. Universities Press, 1965.

[19] Israel Nathan Herstein. *Abstract algebra*. Macmillan, 1990.

[20] Robert Gilmore. *Lie groups, Lie algebras, and some of their applications*. Courier Corporation, 2012.

[21] EB Dynkin. Semisimple subalgebras of semisimple lie algebras. *Selected Papers*, pages 175–308, 1972.

[22] Ashok Das. *Lectures on quantum field theory*, volume 4. World Scientific, 2008.

[23] Paul Langacker. *The standard model and beyond*. CRC press, 2009.

[24] Stefan Pokorski. *Gauge field theories*, volume 136. Cambridge University Press, 2000.

[25] Kurt Sundermeyer. Constrained dynamics with applications to yang-mills theory, general relativity, classical spin, dual string model. 1982.

[26] Yoichiro Nambu. Axial vector current conservation in weak interactions. *Physical Review Letters*, 4(7):380, 1960.

[27] Jeffrey Goldstone. Field theories with «superconductor» solutions. *Il Nuovo Cimento (1955-1965)*, 19(1):154–164, 1961.

[28] Sidney Coleman and Jeffrey Mandula. All possible symmetries of the s matrix. *Physical Review*, 159(5):1251, 1967.

[29] JoAnne L Hewett and Thomas G Rizzo. Low-energy phenomenology of superstring-inspired e6 models. *Physics Reports*, 183(5-6):193–381, 1989.

[30] F Del Aguila. The physics of z'bosons. *arXiv preprint hep-ph/9404323*, 1994.

[31] MIRJAM CVETIC and Stephen Godfrey. Discovery and identification of extra gauge bosons. In *Electroweak symmetry breaking and new physics at the TeV scale*, pages 383–415. World Scientific, 1996.

[32] M Cvetic and P Langacker. Z'physics and supersymmetry. In *Perspectives on supersymmetry*, pages 312–331. World Scientific, 1998.

[33] Arnd Leike. The phenomenology of extra neutral gauge bosons. *Physics Reports*, 317(3-4):143–250, 1999.

[34] Thomas G Rizzo. Z'phenomenology and the lhc. In *Colliders And Neutrinos: The Window into Physics beyond the Standard Model (TASI 2006)*, pages 537–575. World Scientific, 2008.

[35] Paul Langacker. The physics of heavy z'gauge bosons. *Reviews of Modern Physics*, 81(3):1199, 2009.

[36] Python3. `www.python.org/downloads`, 2016.

[37] Anaconda. `https://www.continuum.io/downloads`, 2016.

[38] Richard Wallace Robinett and Jonathan L Rosner. Mass scales in grand unified theories. *Physical Review D*, 26(9):2396, 1982.

[39] Jens Erler. Chiral models of weak scale supersymmetry. *arXiv preprint hep-ph/0006051*, 2000.

[40] Paul Langacker, Gil Paz, and Itay Yavin. Scalar potentials and accidental symmetries in supersymmetric u(1)′ models. *Physics Letters B*, 671(2):245–249, 2009.

[41] Gardner Marshall and Marc Sher. Supersymmetric leptophilic higgs model. *Physical Review D*, 83(1):015005, 2011.

[42] Nan Chen, Ying Zhang, Qing Wang, Giacomo Cacciapaglia, Aldo Deandrea, and Luca Panizzi. Higgsphobic and fermiophobic z'as a single dark matter candidate. *Journal of High Energy Physics*, 2014(5):88, 2014.