Transformación de compuestos químicos en estructuras de Lewis

Juan Diego Araya Garbanzo B30457
Ricardo Muñoz Molina B34793

9 de diciembre de 2015

Definición del proyecto

Tema del proyecto

El objetivo de proyecto radica en la transformación de compuestos químicos en su equivalente en estructura de Lewis, y modelado gráfico, por lo que debe ser capaz de descartar fórmulas cuya estructura no sea válida, bajo la regla del octeto. Las reglas van a ser delimitadas, en general, por su tipo de elemento, excluyendo elementos como los lantánidos y actínidos, además de muchas excepciones de elementos con cualidades particulares que se toman como excepciones a las reglas de Lewis.

Descripción del lenguaje de entrada

El lenguaje de entrada para el análisis léxico y pasos posteriores consiste en el abecedario inglés, mayúsucas y minúscalas, cada elemento incluido en el sistema debe comenzar con mayúscula, seguido o no de una minúscula dependiendo del elemento en cuestión. Además, los subíndices serán representados por números seguiendo al elemento con el que están relacionados. Se formarán moléculas constituídas por un elemento centra que no tiene número, y un elemento secundario que debe tener número i.e: CH2. No se permite que el elemento central tenga un número, porque las fórmulas estructurales no funcionan de esa manera. Se da la opción de agregar enlaces entre moléculas con el carácter de guión, todo esto representa un enlace sencillo, no se admite la entrada de enlaces dobles.

Descripción del producto que se desea obtener

El producto del sistema consiste en la estructura de Lewis, tratando de respetar las geometrías moleculares del elemento lo más posible, ya habiendo sido procesado por una serie de restricciones que delimitan el rango de elección de estructuras posibles; se esboza con la biblioteca gráfica VPython.Se grafican tres modelos moleculares en tres dimensiones: modelos de barras y esferas (stick ball), varillas (line) y espacio lleno (spacefill). Por último, cada elemento tiene un tono de color diferente para distinguirlos.

Lista provisional de tokens

Para el presente proyecto, se toma como lista provisional de tokens, los siguientes:

- Model
- Element
- Number
- Bond
- Lcurl
- Rcurl

Expresiones regulares para los tokens

Para el presente proyecto, se toma como lista provisional de tokens, los siguientes:

- Model: (spaceFill|line|stickBall)
- Element: (Br|Mg|Li|Pb|Si|As|Sn|Rb|Sb|Po|Se|Be|Fr|Sr|Tl|Ba|Cl|Ca|Te|Al|At|Ga|Na|Cs|Bi|Ra|B|C|F|S|H|K|O|P|I|N)
- Number: [1-9]
- Bond: (-)
- Lcurl: ({)
- Rcurl: (})

Gramática final que representa el lenguaje

El producto del sistema consiste en la estructura de Lewis, del elemento en cuestión, ya habiendo sido procesado por una serie de restricciones que delimitan el rango de elección de estructuras posibles; esbozado en pantalla utilizando una biblioteca gráfica de Python. La lista de producciones final, es la siguiente:

- S: A LCURL L RCURL
- A: MODEL
- L: B
- B: ELEMENT
- B: ELEMENT ELEMENT NUMBER
- B: ELEMENT ELEMENT NUMBER BOND B

Versión final

Casos de prueba válidos

Para el siguiente proyecto, se incluyen los siguientes casos de pruebas válidos con su gramática respectiva:

Figure 1: data = ”line { SeF3-CH2-IF5-NH1-SF4-TeCl4-CH3 }”

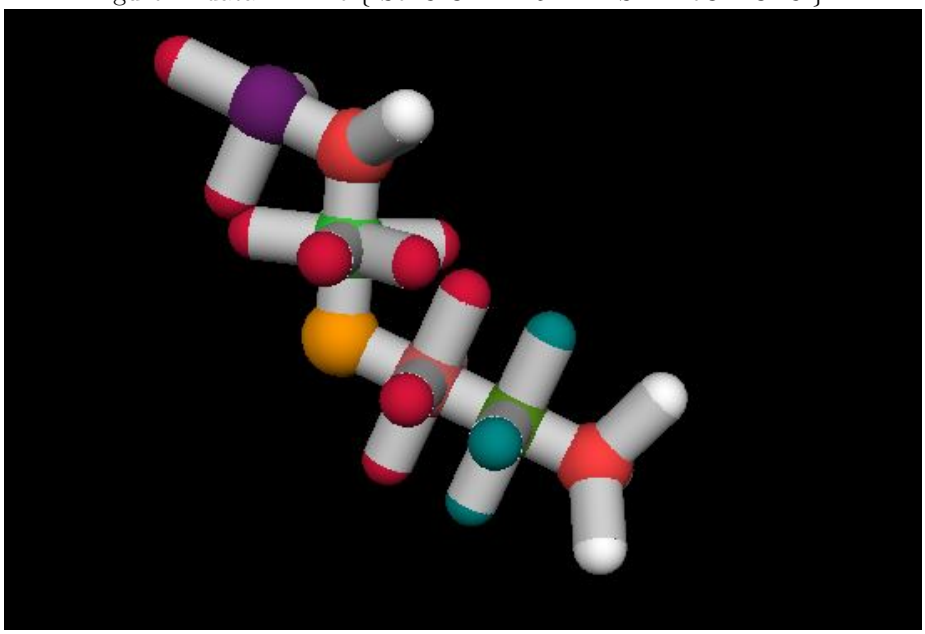


Figure 2: data = ”spaceFill{ BiH3 }”

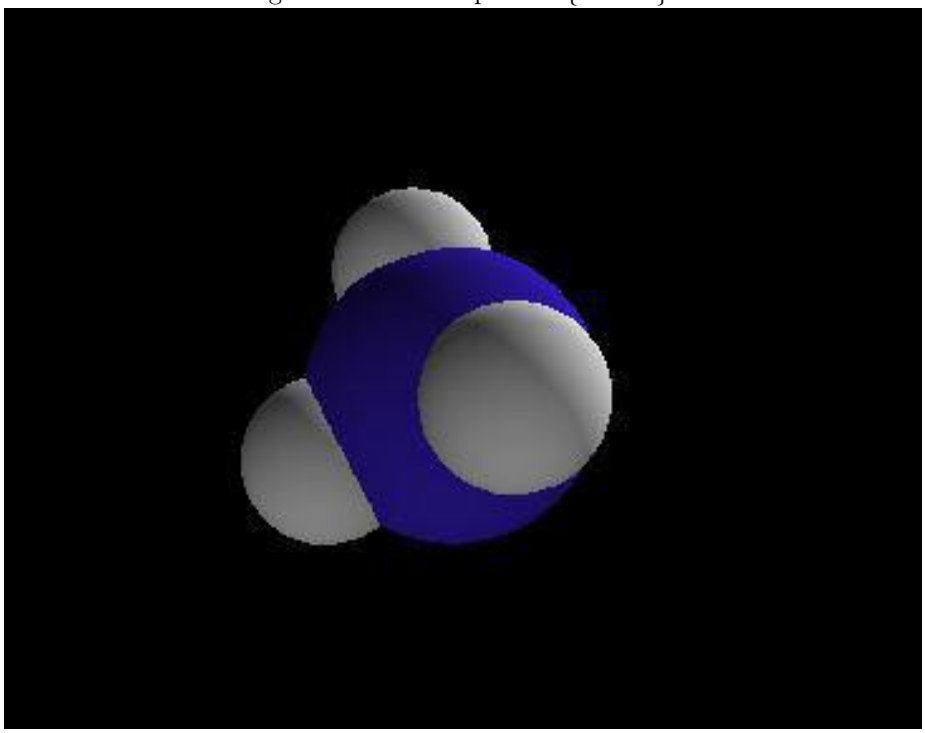


Figure 3: data = "stickBall { CH3-PCl1-SF5 }"

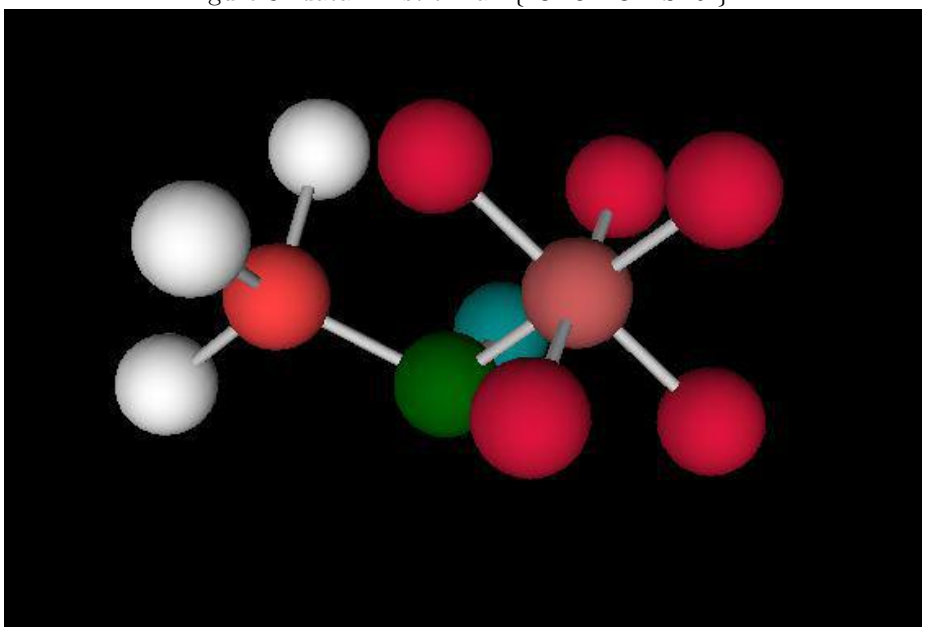


Figure 4: data = "stickBall { I }"

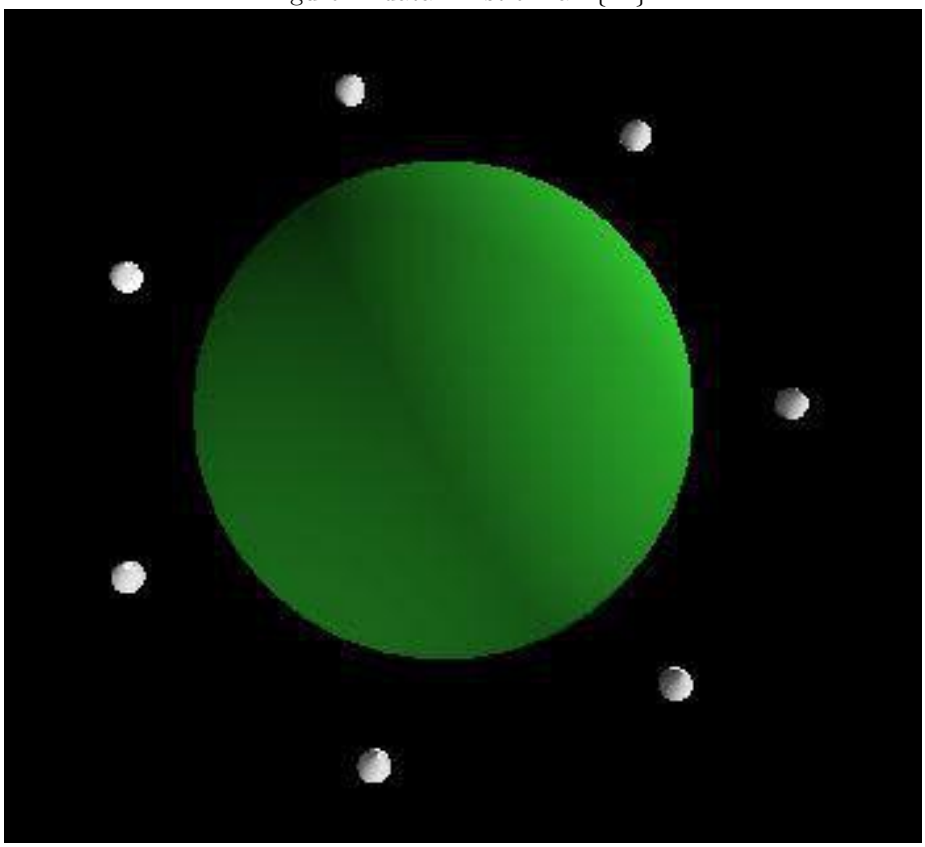
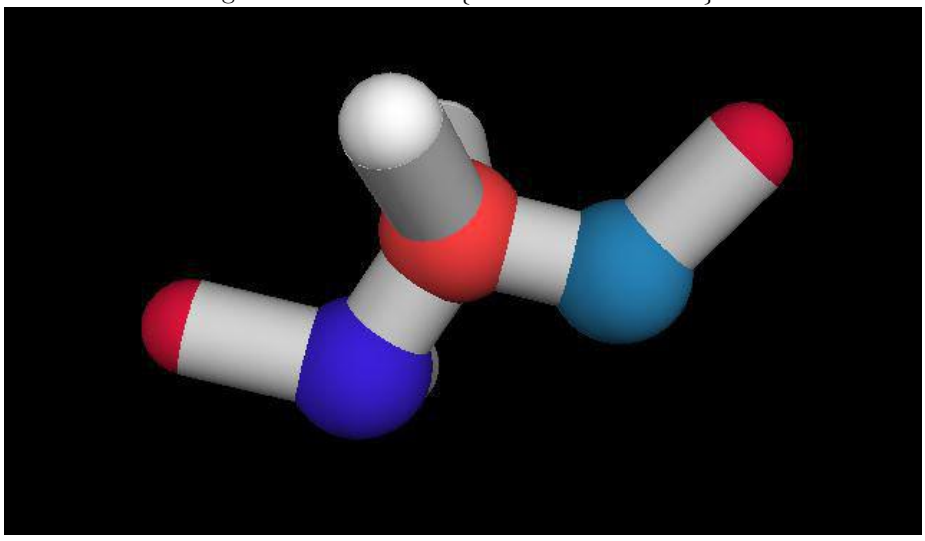


Figure 5: data = "line{ BiF2 - CH2 - OF1 }"



Casos de prueba inválidos

- Error léxico: "data = line { FeH2 }"
- Error sintáctico: "data = spaceFill{ -ClF-O- }"
- Error semántico: "data = spaceFill{ FH1 - CH2 - AtF1 }"

Versiones y bibliotecas

Para el siguiente proyecto, se tomaron solo dos bibliotecas externas, las cuales son:

- VPython: "VPython-Win-64-Py2.7-6.11"
- PLY: "ply-3.0"

Todo lo anterior, para Python v2.7.9, 64bit.

Manual de usuario

Para poder ejecutar el programa, se tiene que insertar el string a ser parseado, a mano, puesto que el módulo VPython tenía problemas a la hora de recibir comandos desde consola, y con un display de interfaz gráfica simple. Para ello, hay que abrir el archivo MoleculeParser.py con cualquier editor de texto, y cambiar el valor del string data, a cualquier valor que quiera ser parseado, los casos de prueba incluidos están comentados en el código. Para hacer una prueba, debe poner el tipo de modelo que quiera elegir, seguido por dos corchetes, y dentro de ello insertar la molécula conformada por elemento principal y acompañante, o cadenas de moléculas enlazadas por el guión. Los elementos que podrá usar están establecidos en el archivo Elements.py, además, debe procuparar que sea una molécula que respete las reglas del octeto, reglas que se pueden consultar en este link: `es.wikipedia.org/wiki/Regla_del_octeto`. Por ejemplo: *line { SeF3-CH2-IF5-NH1-SF4-TeCl4-CH3 }*.
Para instalar ambos módulos, es necesario descargar ambos archivos de su página oficial, para VPython: `http://vpython.org/contents/download_windows.html` y para PLY `http://www.dabeaz.com/ply/`. VPython es un archivo ejecutable, que si no reconoce la ubicación del python instalado, pide una, y lo instala. Ply por otro lado, viene con el setup.py y basta con llamar a Python y ese setup para instalarlo.
El proyecto consta de seis módulos python, los cuales son:

- MoleculeParser: Donde se hace el análisis léxico, sintáctico y semántico, y es donde se invoca el método yacc.parse(@param), con lo que va a ser parseado.
- Elements.py: Contiene un diccionario con todos los elementos disponibles.
- Geometries.py: Se encarga de graficar todas las geometrías.
- Molecule.py: Guarda atributos clave para graficar la geometría.
- MolValidator.py: Tiene métodos para la validación semántica.
- Graph.py: Módulo que llama a Geometries.py, recibe una lista de objetos tipo Molecule del parser, y las envía a Geometries.py para que las grafique.