Isomers Generator

Lorenzo Tinacci

March 2020

The program takes as input the number of atoms in a chemical stoichiometric formula, for now it is only available for this list of atoms: H, C, N and O. The code's goals is to give all the possible way to combine, with a chemical meaning, the input atoms. So it is a code that will produce all the structural isomers and resonance structures. The easy way to work with the molecules, for this kind of purpose, it is to use a graph based approach (from now on we will refer to it using the term: *molecular-graph*).
The code is it organized in a flow chart of operation to achieve the molecules. In the next figure it is show the tree-graph related to all the process of the code, in the case of **HCN**.

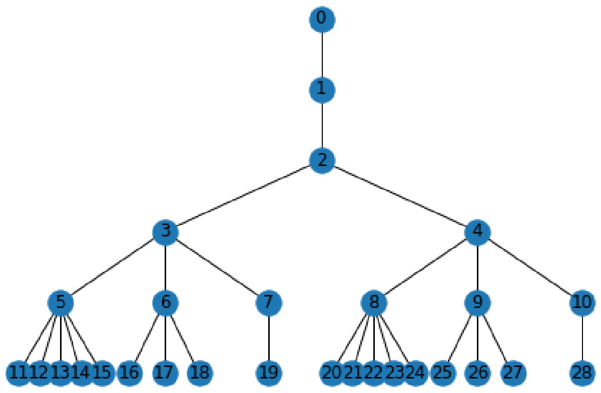


Figure 0.1: Genealogical molecular graph tree of the operations to achieve the molecules, in the **HCN** example.

The first step, node 0 or root of the *molecular-graph*, of the code is to extract only the possible atoms that will form the "skeleton" of the molecules, so we are not going to consider hydrogen atoms because are always terminals. Moreover we are not consider the atom type. In the next figure it is showed this step.

Figure 0.2: Node 0 of the molecular graph tree 0.1.

The second step is to create all the possible way to connect the "skeleton" molecules. In the case that we are studying now, there is only possibility. In the next figure it is showed this step.

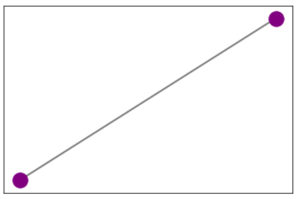


Figure 0.3: Node 1 of the molecular graph tree 0.1.

The third step is to create all the possible combination to colored the "skeleton" molecules, in other words: assigning the atoms in the node 1, figure 0.3. In the case that we are studying now, there is only possibility. In the next figure it is showed this step. In our figures we will refer to H, C, N and O respectively with the grey, black, blue and red colors.

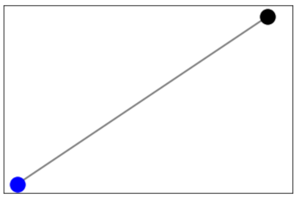
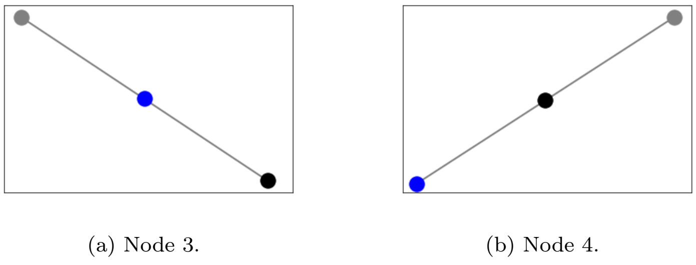


Figure 0.4: Node 2 of the molecular graph tree 0.1.

The next step is to create all the possible combination to add the hydrogen atoms in the colored molecules in the node 2, figure 0.4. In the case that we are studying now, there are two possibilities. In the next figure it is showed this step.

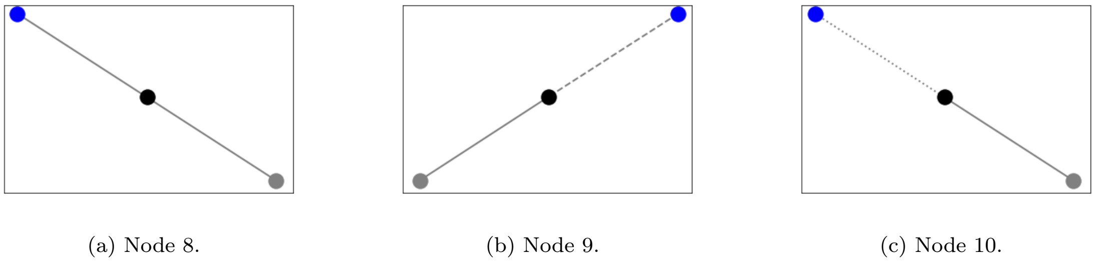


(a) Node 3.

(b) Node 4.

Figure 0.5: Node 3 and 4 of the molecular graph tree 0.1.

The next step is to create all the possible combination to add the bonds, double and triple, to the molecules in the 0.5. To show how the code works we will show only the case of the node 4, figure 0.5b, in this case there are three possibilities. In the next figure it is showed this step. In our figures we will refer to single, double and triple bond respectively with the solid, dashed and dot line style.



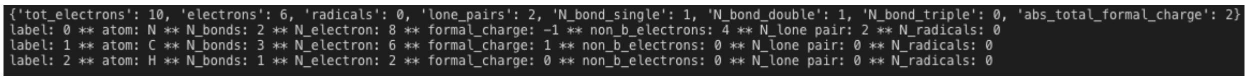
(a) Node 8.

(b) Node 9.

(c) Node 10.

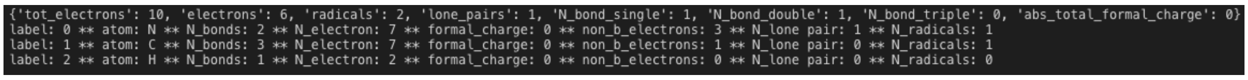
Figure 0.6: Node 8,9 and 10 of the molecular graph tree 0.1.

The next step is to create all the possible electronic structures of the molecules. In other words, in this step, we will put the electrons not used in bonds in all the allowed possibilities to assign non-bond electron to the atoms. Then we will used the max spin maximum spin multiplicity principle, or Hund's Rule, to assign the number of radicals and lone pairs of each atoms. To show how the code works we will show the case of the node 9, figure 0.6b, and 10, 0.6c. In next figure, related to the node 9, figure 0.6b, we will show the properties of the three possibilities.



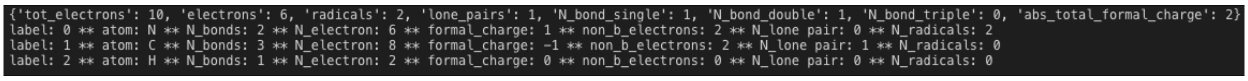
```
{'tot_electrons': 10, 'electrons': 6, 'radicals': 0, 'lone_pairs': 2, 'N_bond_single': 1, 'N_bond_double': 1, 'N_bond_triple': 0, 'abs_total_formal_charge': 2}
label: 0 ** atom: N ** N_bonds: 2 ** N_electron: 8 ** formal_charge: -1 ** non_b_electrons: 4 ** N_lone pair: 2 ** N_radicals: 0
label: 1 ** atom: C ** N_bonds: 3 ** N_electron: 6 ** formal_charge: 1 ** non_b_electrons: 0 ** N_lone pair: 0 ** N_radicals: 0
label: 2 ** atom: H ** N_bonds: 1 ** N_electron: 2 ** formal_charge: 0 ** non_b_electrons: 0 ** N_lone pair: 0 ** N_radicals: 0
```

(a) Node 25.



```
{'tot_electrons': 10, 'electrons': 6, 'radicals': 2, 'lone_pairs': 1, 'N_bond_single': 1, 'N_bond_double': 1, 'N_bond_triple': 0, 'abs_total_formal_charge': 0}
label: 0 ** atom: N ** N_bonds: 2 ** N_electron: 7 ** formal_charge: 0 ** non_b_electrons: 3 ** N_lone pair: 1 ** N_radicals: 1
label: 1 ** atom: C ** N_bonds: 3 ** N_electron: 7 ** formal_charge: 0 ** non_b_electrons: 1 ** N_lone pair: 0 ** N_radicals: 1
label: 2 ** atom: H ** N_bonds: 1 ** N_electron: 2 ** formal_charge: 0 ** non_b_electrons: 0 ** N_lone pair: 0 ** N_radicals: 0
```

(b) Node 26.

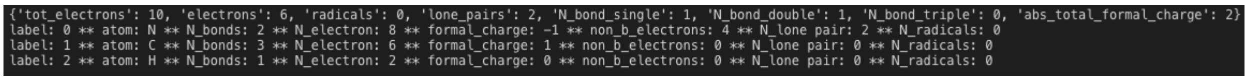


```
{'tot_electrons': 10, 'electrons': 6, 'radicals': 2, 'lone_pairs': 1, 'N_bond_single': 1, 'N_bond_double': 1, 'N_bond_triple': 0, 'abs_total_formal_charge': 2}
label: 0 ** atom: N ** N_bonds: 2 ** N_electron: 6 ** formal_charge: 1 ** non_b_electrons: 2 ** N_lone pair: 0 ** N_radicals: 2
label: 1 ** atom: C ** N_bonds: 3 ** N_electron: 8 ** formal_charge: -1 ** non_b_electrons: 2 ** N_lone pair: 1 ** N_radicals: 0
label: 2 ** atom: H ** N_bonds: 1 ** N_electron: 2 ** formal_charge: 0 ** non_b_electrons: 0 ** N_lone pair: 0 ** N_radicals: 0
```

(c) Node 27.

Figure 0.7: Node 25,26 and 27 of the molecular graph tree 0.1.

In the next figure, related to the node 10, figure 0.6c, we will show the properties of the one possibility.



```
{'tot_electrons': 10, 'electrons': 6, 'radicals': 0, 'lone_pairs': 2, 'N_bond_single': 1, 'N_bond_double': 1, 'N_bond_triple': 0, 'abs_total_formal_charge': 2}
label: 0 ** atom: N ** N_bonds: 2 ** N_electron: 8 ** formal_charge: -1 ** non_b_electrons: 4 ** N_lone pair: 2 ** N_radicals: 0
label: 1 ** atom: C ** N_bonds: 3 ** N_electron: 6 ** formal_charge: 1 ** non_b_electrons: 0 ** N_lone pair: 0 ** N_radicals: 0
label: 2 ** atom: H ** N_bonds: 1 ** N_electron: 2 ** formal_charge: 0 ** non_b_electrons: 0 ** N_lone pair: 0 ** N_radicals: 0
```

Figure 0.8: Node 28 of the molecular graph tree 0.1.

To try the code, and visualize the compute molecules, go to this link: GoogleColabTinacci