2001

The algebraic application of Lewis structures in conjunction with combinatorial analysis in global molecular identification from graphs

Jonathan Geach

The algebraic application of Lewis structures in conjunction with combinatorial analysis in global molecular identification from graphs

Jonathan Geach and Ray Hefferlin

Southern Adventist University, Collegedale, TN 37315

(Received April 17, 2001)

Keywords:

Graphs
Lewis diagrams
Lewis structures
Bonding
Closed-shell molecules
Octet rule

Abstract

Closed shell bonding can be described by a theory that relates the manner that atoms fill their outer shells to algebraic formulas. Equations can be derived that are capable of deciding whether a given electron configuration gives a closed shell electron structure or not. A combinatorial analysis can be used to try every possible electron configuration for a given structure. The equations are then used to determine whether each structure gives a closed shell electron configuration. Some of these molecules may be considered impossible because they are bonded to noble gases. However, a list of molecules that contain a noble gas species is included to show that bonding to noble gases is not impossible. An initial list of triatomic linear molecules is presented.

Introduction

G. N. Lewis was one of the first to note that the electron configurations of the noble gases are very stable and unreactive. He went on to suggest that atoms could attain similar electronic structures as the noble gases by sharing their electrons. In 1916, he coined the term covalent bond to describe this phenomenon. This model quickly became known as the Lewis model and is one of the most well known models for describing molecular bonding.[1]

The Lewis model for covalent bonding is only concerned with electrons in the outermost principal energy level. These electrons are commonly referred to as valence electrons. Valence electrons can be easily represented as dots around a chemical symbol. For example, the following is a representation of a Beryllium atom.

·Be·

The atom has only two valence electrons and these are represented by the two dots. These symbols can be joined together to create covalent bonds by combining the structures in such a way so that the same electron configuration as a noble gas is attained around each atom. When all the atoms in a molecule have closed outer electron shells, the energy of the system is minimized. Because all noble gases except helium have eight electrons in the outermost shell, the this configuration is now called the rule of eight.

The basic ideas used in determining Lewis structures can easily be translated into an algebraic form.[2] For any molecule that follows the rule of eight and has atoms with no formal charge, the number of electrons held by one atom, plus the electrons it receives from another atom must equal eight.[3] In other words, an equation is written that requires each atom to have the number of electrons requisite for a closed shell.

For example, in a two-atom molecule the following equation can be written for the first atom, $c_1 + v_{12} = 8$, where c_1 refers to the number of electrons in atom 1 and v_{12} refers to the number of atoms that c_1 receives from c_2.[2] A similar equation can be written for the other atom in the molecule. The equations can then be combined algebraically to make one large equation that when satisfied will give a molecule that contains only closed shell atoms as long as a few simply constraints are met. Two atom molecules will again be used as an example. Assuming that v_{12} is equal to v_{21}, the following equations can be derived. The first one is derived by using addition of equations and the second by subtractions. Equations can be combined in any manner of addition or subtraction and this will not affect the validity of the equation.

$$\begin{array}{r} c_1+v_{12}=8 \\ +\quad c_2+v_{21}=8 \\ \hline c_1+c_2+2v_{12} = 16 \end{array}$$

$$\begin{array}{r} c_1+v_{12}=8 \\ -\quad c_2+v_{21}=8 \\ \hline c_1-c_2 = 0 \end{array}$$

Note how the two solutions both have benefits and drawbacks. The first equation, arrived at through the addition method includes a term for the bonding. This is good if one wants to know the order of the bonds; however, it is an extra variable that can add considerable time to calculations. On the other hand, the subtraction method is valuable because it can predict the specific atoms in the molecule with the fewest variables possible. Simply satisfy that equation and a few simple constraints and a closed shell molecule will be produced. When used together the different techniques can be very powerful. The subtraction method can first be used to determine the atoms in a molecule, and then the addition method can be used to find the bond order. Although the addition method alone would work as well, the use of both of them together can be much faster for large molecules.

The use of these two methods is very good at finding a small subset of molecules. However, it fails to take into effect coordinate covalent bonding. In coordinate covalent bonding, one molecule donates both electrons from a lone pair to a shared bond. The most common example of this is carbon monoxide. Carbon is triple bonded to oxygen, however carbon donates both electrons to form the third bond, thus there are two electrons from oxygen and four from carbon making a triple bond. Although the constraints are still being discovered for coordinate covalent bonding, it can easily be incorporated into our equations. Coordinate covalent bonding can be taken into effect by adding a 2 in front of each of the variables that represents the atom that initiates the coordinate covalent bond.

Methodology

Triatomic linear molecules were chosen to be studied because diatomic molecules had already been studied in detail.[4] Another benefit of triatomic linear molecules is that they are easily represented and can be graphed according to each atom in the molecule.

Once the number of molecules and geometric structure were determined, the equations for triatomic linear molecules had to be determined. The first step is to find the specific equation for each atom in the molecule. Because the first molecule has c_1 valence electrons and receives v_{12} electrons from the second atom, its equation is $c_1 + v_{12} = 8$. The second atom is slightly more complex. It not only receives electrons from the third atom, it receives electrons from the first. So the equation for the second atom has an extra term, $c_2 + v_{12} + v_{23} = 8$. The third atom is in the same situation as the first, so its equation is $c_3 + v_{23} = 8$. These equations can be combined using the addition method to make one large equation.

$$
\begin{array}{rl}
 & c_1 + v_{12} = 8 \\
 & c_2 + v_{12} + v_{23} = 8 \\
+ & c_3 + v_{23} = 8 \\
\hline
 & c_1 + c_2 + c_3 + 2(v_{12}) + 2\,(v_{23}) = 24
\end{array}
$$

Once the general equation has been found, combinatorial analysis can be used to find out which atoms are in the molecule, and what bond order is between them. So a computer program can simply plug in all possible values one at a time and check to see if the equation is true. As an example, the analysis finds that $7 + 6 + 7 + 2 * 1 + 2 * 1 = 24$ is a true statement. So we can say that a triatomic linear molecule that has an outer

electron configuration of 7, 6, 7, that is joined by two single bonds is a closed shell molecule. By substituting into the second of the periodic table, one can find the molecule F – O – F.

There has been a constant evolution in the methods of combinatorial analysis used to determine these molecules. The first method used was a pen and paper. One can guess which possible atoms should be in a molecule, then enter the values in to the equations to determine if it is a valid molecule. The pen and paper method is slower, but it is the most accurate and is still used to check the newer methods.

The second method employed was a spreadsheet. In the spreadsheet, each line had one possible configuration for a molecule. The spreadsheet then checked to see if the variables used in that line, when put into the equation gave a true statement. Using the addition method, this took around 1600 lines. This also did not take coordinate covalent bonding into effect.

A third method was developed using a Java™ computer program to try all the possible values. This first program did not take coordinate covalent bonds into effect and was very rudimentary. It was also only able to find molecules for triatomic linear molecules. It had to be rewritten and recompiled to find any other configuration. However, it did find the majority of covalent molecules after some bugs were worked out. The computer code for this program has been included in appendix A.

The program was then rewritten to be able to address some of the major problems associated with the first program. The biggest feature of this program is that no rewrite is necessary to change the geometric and atomic structures being used. The computer code used is included in appendix B.

Results

The molecules found from the combinatorial analysis are listed in table 1. These molecules have both covalent and coordinate covalent bonding. The molecules were then graphed according to the number of valence electrons in each atom. Figure 1 shows the entire coordinate system used to graph the molecules. The first item of note from figure 1 is that there are three main planes moving from the upper left to the lower right. Another item of note is that only atoms in the s and p levels have been included. If d atoms had been included the entire coordinate system would have been filled. Figure 2 is a closer view that is centered on the lowest plane in the figure. Figure 3 is very similar, however it is centered on the middle plane. Figure 4 gives a completely different view. This is a picture taken from above the origin looking down at the planes. Figure 5 is a cut away from the top plane in the figures 1-4. Figures 6 and 7 are both cut away from the middle and bottom planes respectively. Figure 8 shows molecules that have entropy data for them. The picture on the left is the standard picture. The boxes in the picture on the right are proportional to the entropy of each molecule. Since entropy is a measure of disorder, the smaller boxes should be molecules that are more stable, note CO_2.

Discussion

One supposed problem with our study is that many of the molecules predicted by our theories, especially using coordinate covalent bonding, are bonded to the noble gases. Anyone who has taken a high school chemistry class knows that noble gases rarely bond to other molecules. That is why they used to be called the inert gases. However, current research has shown many molecules that contain noble gases. According to the National

Institute of Standards and Technologies webBook database the elements below are all molecules with either spectral or thermodynamic data.[4]

ArBrXe	NeXeBr	NeKrCl	Ne_2F	NeXeF	BeOXe	BrNeXe
C_5CrO_5Xe	C_5MoO_5Xe	ClFXe	ClKrXe	F_2Xe	HBrXe	ArFXe

Simply because many molecules appear extremely unstable in no way negates the value of this method. A closed shell molecule may not always be a stable molecule due to steric strain or other factors.

The theory presented here is an attempt to reformulate Lewis-structure chemistry in a general and rigorous way.[6] Nearly 20 million molecules have been characterized and the majority of them have a least partial Lewis structures. It seems that the methodology employed here should be useful for identifying new species as well as systematically finding any molecule needed, known or not.[6]

The most recent research is showing clear trends in stability, with the most stable molecules near the center of three isoelectronic planes. One way these stability Formal charge is a way of finding the apparent charge on any one atom in a molecule. One way of determining whether a Lewis structure's is valid is by minimizing formal charge. Figures 5, 6 and 7 show the three isoelectronic triangles from figures 1-4. These triangular graphs contain the sum of formal charge for the entire molecule.

Another good indicator of stability is entropy. Because entropy is a measure of the disorder of a molecule, the smaller the entropy value, the more stable a molecule should be[1]. Figure 8 shows a graph of entropy with the molecules that contain stability data. The left hand side of the figure contains the molecules as they would normally appear. The

right hand side contains the same molecules, but the size of the boxes has been scaled to be proportional to entropy.

This theory could some day allow pharmaceutical companies to find new drugs faster, by systematically finding all related molecules. Many different drugs have very similar atomic structures. One could use a program similar to the one in appendix B to input a known structure, while giving the program certain variables of what the parts of the molecule the researchers would like to vary. Hopefully some of the molecules generated would produce useful molecules not thought of initially.

The ability to predict the majority of electronically stable molecules from a set of basic principles should be a huge asset to the scientific community. Although many molecules do appear to be short lived, the fact that they follow the rule of eight and fall into such regular patterns is astounding. Still, much more research is needed before more practical applications can be developed.

Acknowledgements

We are especially indebted to Chris Walters who helped write a large portion of the computer code appearing in the appendix. We are also indebted to Bryan James, Ken Caviness, Rhonda Scott-Ennis, Brent Hamstra and Bruce Schilling for their help, insights and suggestions.

References

[1]W. Masterton, Chemistry: Principles and Reactions, Saunders College Publishing, New York, 1997, p. 175, 471-473.

[2]R. Hefferlin, The textures of chemical spaces, Journal of Molecular Structure (Theochem) 506 (2000) 71-86.

[3]R. Cavanough, *et al*, Adjacent DIM isoelectronic molecules and chemical similarity among triatomics, Journal of Molecular Structure 382 (1996) 137-145.

R. Hefferlin, *et al,* Global Molecular Identification from Graphs, in preperation for publication.

[5]M.E. Jacox, "Vibrational and Electronic Energy Levels of Polyatomic Transient Molecules" in NIST Chemistry WebBook, NIST Standard Reference Database Number 69, Eds. W.G. Mallard and P.J. Linstrom, February 2000, National Institute of Standards and Technology, Gaithersburg MD, 20899 (http://webbook.nist.gov).

[6]R. Hefferlin, *et al,* The Algebraic Representation of Lewis Structures, submitted to Croatica Cehmica Acta.

Table 1: This is a list of all linear triatomic molecules. The notation needs explantion. Normal bond notation is used for covalent bonds, but for coordinate covalent bonds direction must be noted. The arrows show which atoms donated two atoms for a bond. The symbol -= represents a covalent triple bond.

Ne>O>>>Be	F-F>>>Be	O<Ne>>>Be	Ne>N->>B	F-O->>B
O<F->>B	Ne>>C>>C	Ne>C=>C	F->N>>C	F-N=>C
O=O>>C	O<O=>C	Ne>O>>C	N-<F>>C	F-F>>C
C<<Ne>>C	O<Ne>>C	Ne>>B->N	Ne>B-=N	F->C->N
F-C-=N	O=N->N	O<N-=N	Ne>N->N	N-<O->N
F-O->N	C<<F->N	O<F->N	Ne>>>Be>O	Ne>>Be=O
Ne>Be=<O	F->>B>O	F->B=O	F-B=<O	O=>C>O
O<C=<O	O=C=O	Ne>>C>O	Ne>C=O	N-=N>O
N-<N=O	F->N>O	F-N=O	C=<O>O	C<<O=O
O=O>O	O<O=O	Ne>O>O	B-<<F>O	N-<F>O
F-F>O	Be<<<Ne>O	C<<Ne>O	O<Ne>O	Ne>>>Li-F
Ne>>Li-<F	Ne>Li-<<F	F->>Be-F	F->Be-<F	F-Be-<<F
O<B-<<F	O=>B-F	O=B-<F	Ne>>B-F	Ne>B-<F
N-<C-<F	N-=C-F	F->C-F	F-C-<F	C<<N-<F
C=<N-F	O<N-<F	O=N-F	Ne>N-F	B-<<O-F
N-<O-F	F-O-F	Be<<<F-F	C<<F-F	O<F-F
F->>Li<Ne	F->Li<<Ne	F-Li<<<Ne	O<Be<<<Ne	O=>Be<Ne
O=Be<<Ne	Ne>>Be<Ne	Ne>Be<<Ne	N-<B<<Ne	N-=B<Ne
F->B<Ne	F-B<<Ne	C<<C<<Ne	C=<C<Ne	O<C<<Ne
O=C<Ne	Ne>C<Ne	B-<<N<Ne	N-<N<Ne	F-N<Ne
Be<<<O<Ne	C<<O<Ne	O<O<Ne		

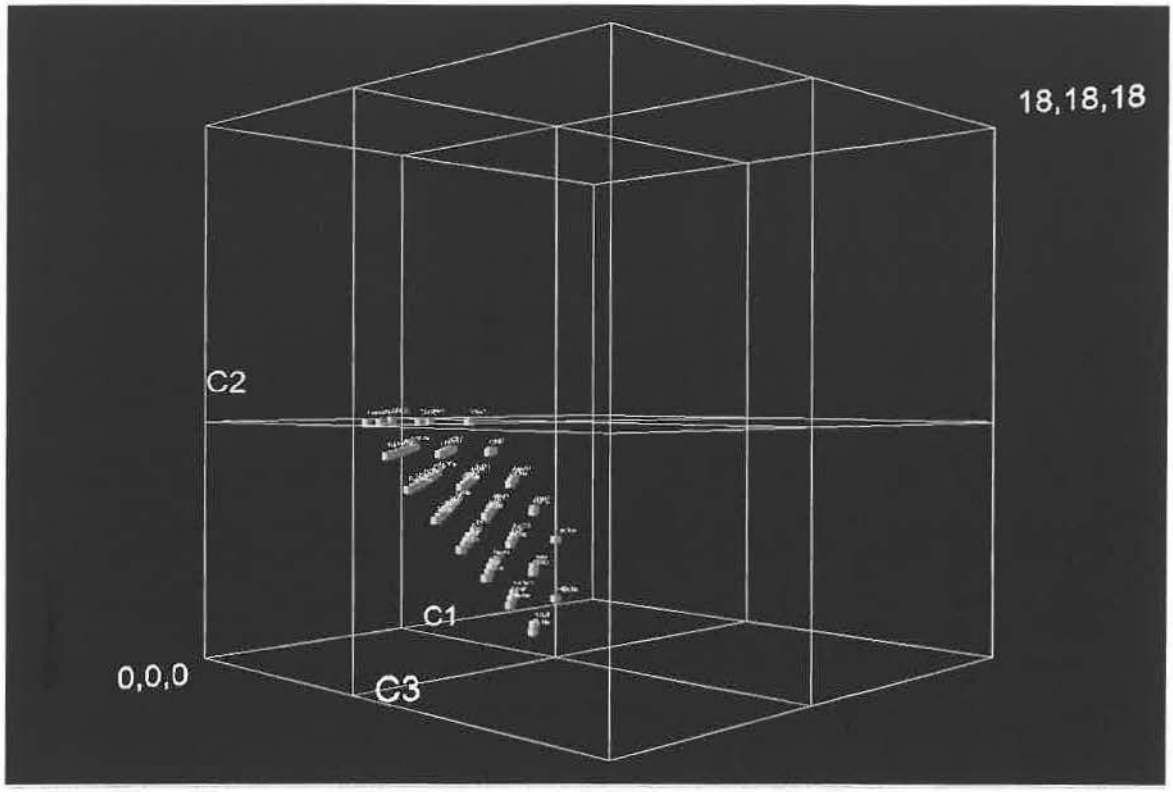

Figure 1: Distant view showing coordinate system from a distance

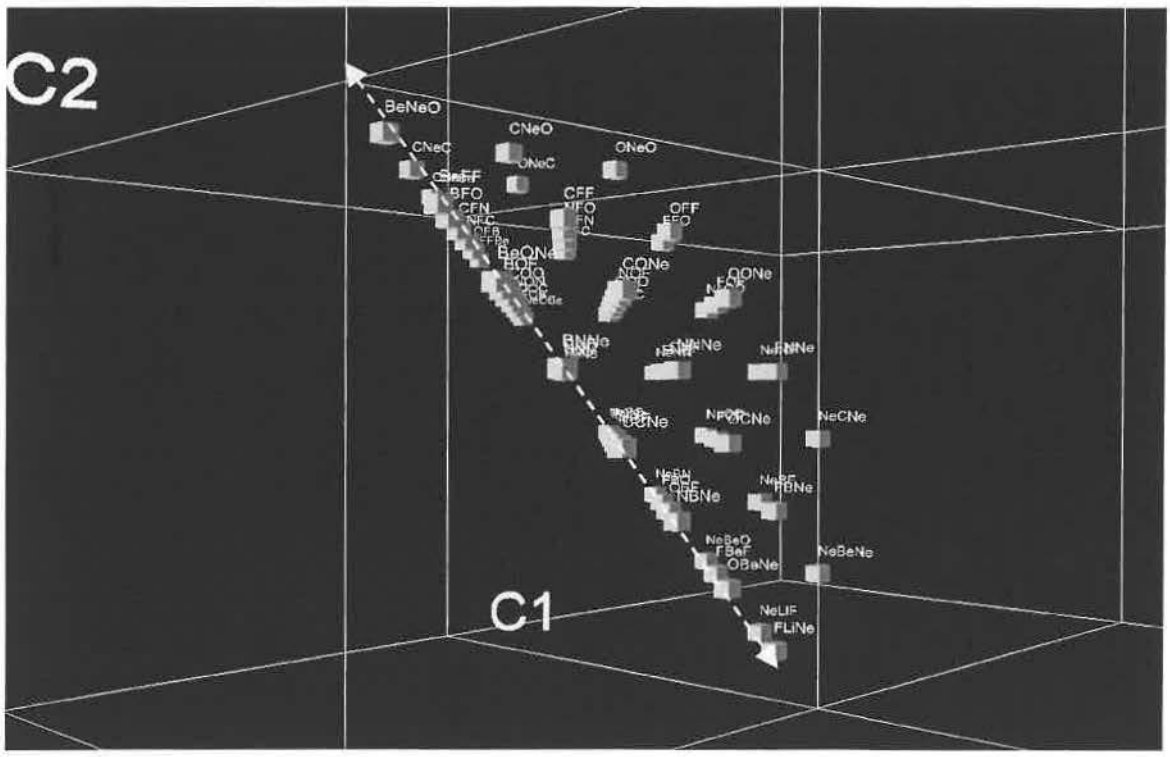

Figure 2: Closer view centered on lowest plane

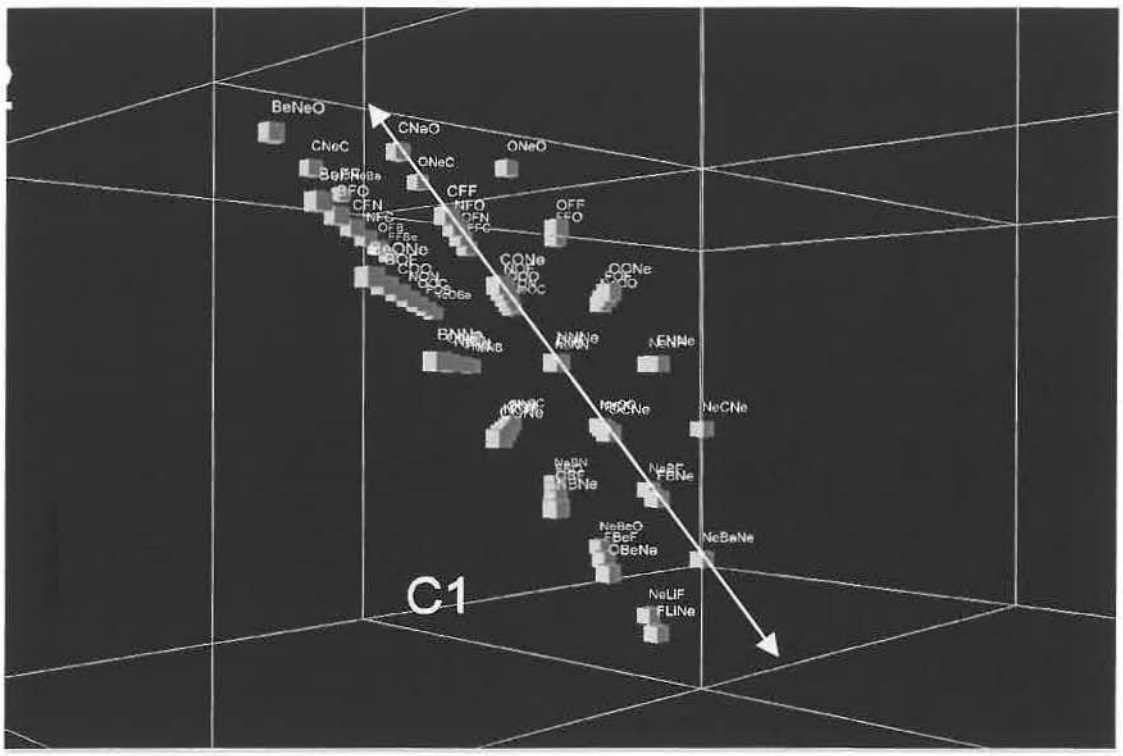

Figure 3: A closer look at molecules centered on second plane

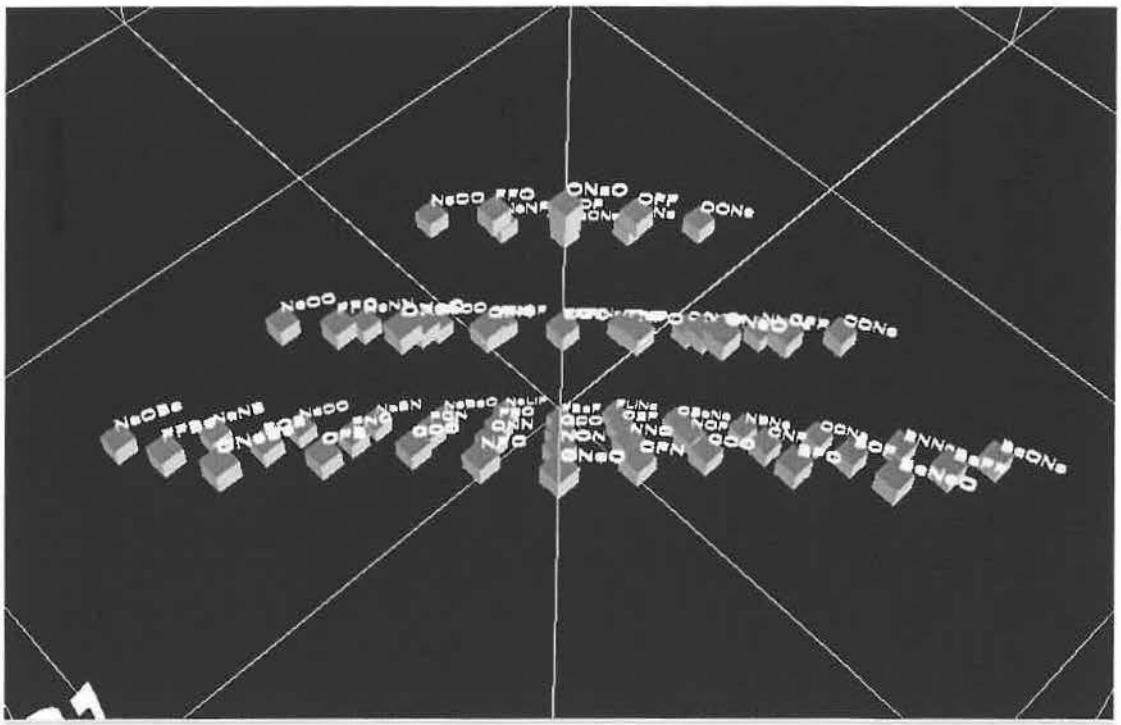

Figure 4: View from above origen looking down along planes

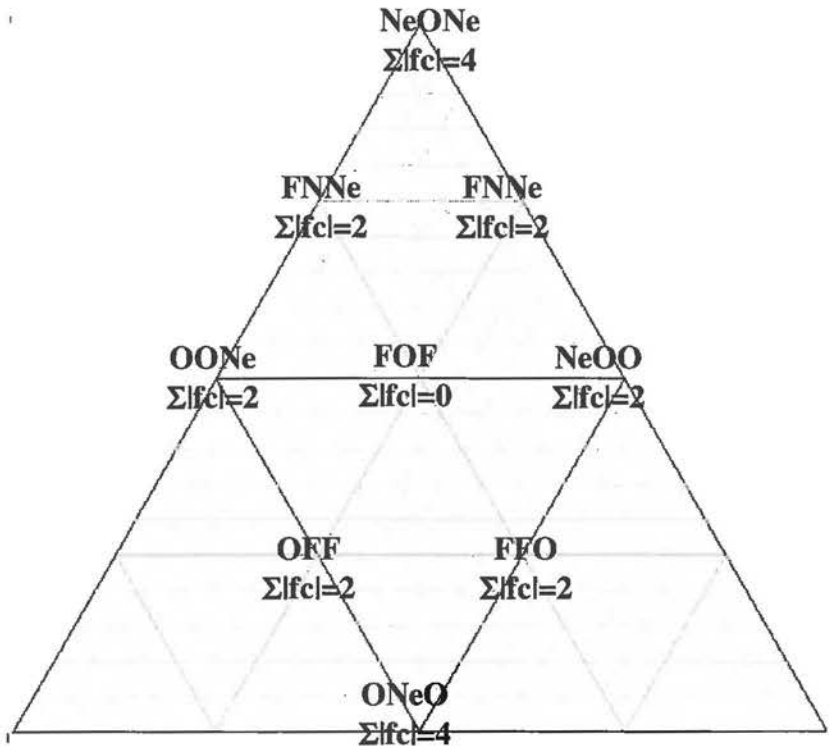

Figure 5: Diagram of highest plane showing the location and formal charge of all molecules

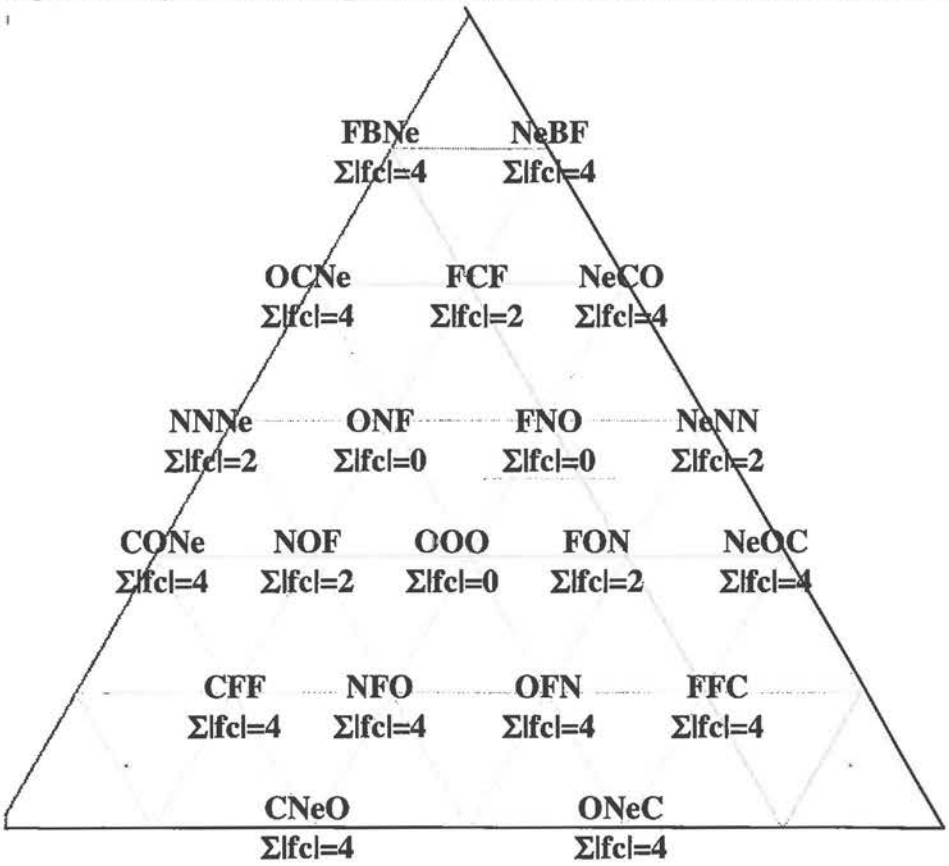

Figure 6: Diagram showing molecules in middle plane

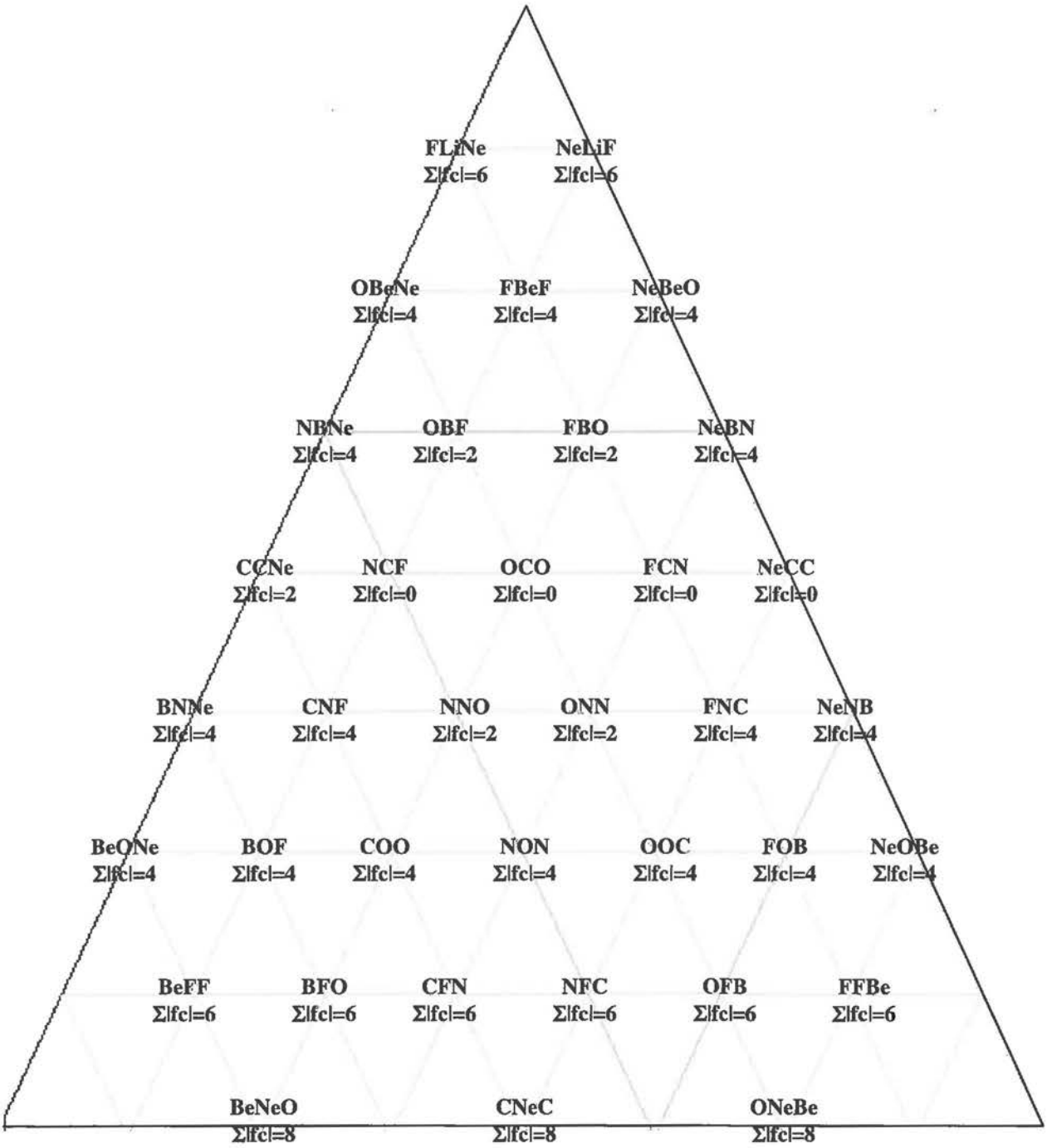

Figure 7: Diagram showing molecules from the lowest plane in graph and showing the summation of the absolute value of the formal charge for each molecule

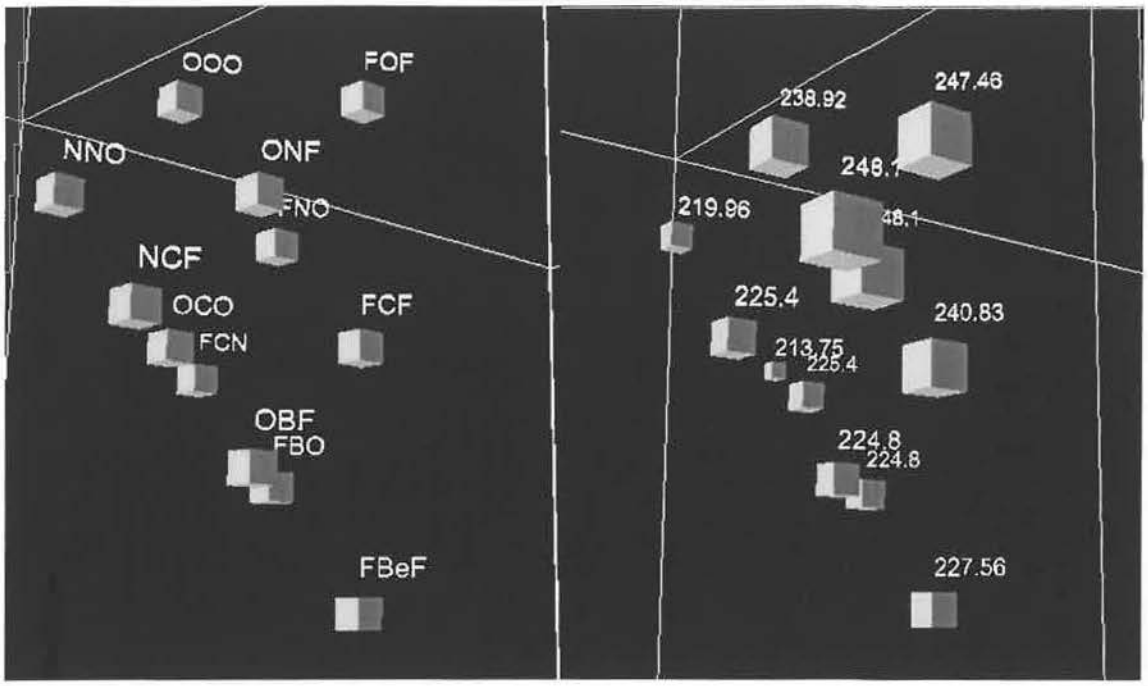

Figure 8: Standard Entropies at 25°C (J/mol · K) of compounds at 1 atm. Two views of molecules with entropy data. The picture on the left shows the molecules as they appeared in the earlier graphs. In the picture on the right, the molecules have been edited for size according to their entropy data. The numbers above are the actual values

Appendix

A. First Java Computer Program

```
import com.sun.j3d.utils.applet.MainFrame;
import mvs.*;
import java.util.Vector;
import java.awt.*;
import ViewMol;
import javax.media.j3d.*;
public class Model3Ln extends Canvas
{
        protected int state = 0;
        protected String lookup[] = { "", "Li", "Be", "B", "C", "N", "O", "F",
                                                                "Ne", "Sc", "Ti", "V", "Cr", "Mn",
"Fe",
                                                                "Co", "Ni", "Cu", "Zn"};
        protected String lookup2[] = { "", "-", "=", "-=" };
        protected Data temp;
        public Vector answer = new Vector(20,20);
        protected int c1 = 0, c2 = 0, c3 = 0, c4 = 0, c5 = 0;
        protected int a = 8, b = 8, c = 8, d = 8, e = 8;    public void setState( int a ){
                state = a;       }
  public Model3Ln ()
  {
        this.setSize(400, 800);   this.setBackground(Color.cyan);   this.setVisible(true);
    compute ();
    sort ();     repaint();
  }
  public Model3Ln (int a1, int b1, int c1)
  {
        this.setSize(400, 800);   this.setBackground(Color.cyan);   this.setVisible(true);
    compute ();
    sort ();     repaint();
  }
  public void setData (int a1, int b1, int c1)
```

```
{
  a = a1;
  b = b1;
  c = c1;
  compute ();
  sort ();
}
public void compute ()
{
  for (int i = a / 2; i < a; i++)
    {
      for (int j = b / 2; j < b; j++)
        {
          for (int k = c / 2; k < c; k++)
            {
              for (int v12 = 1; v12 < 4; v12++)
                {
                  for (int v23 = 1; v23 < 4; v23++)
                    {

                                if (i + j + k == a + b + c - 4 - 2 * v12 - 2 * v23
                                && v12 + v23 <= 4)
                                {
                                        c1 = a - v12;
                                        c2 = b - 4 - v12 - v23;
                                        c3 = c - v23;
                                        if (c1 > 0 && c2 > 0 && c3 > 0)
                                                {
                                                temp =
                                                new Data (c1, c2, c3, 0, 0, v12 + 1, v23 + 1, 0, 0,
0);
                                                answer.addElement (temp);
                                        }
                                }
                        }
                }
            }
        }
    }
}
public void sort ()
{  for( int m =0; m < answer.size(); m++){
  for (int i = 0; i < answer.size () - 1; i++)
    {
      for (int j = i + 1; j < answer.size (); j++)
        {
          if (((Data) answer.elementAt (i)).value () ==
              ((Data) answer.elementAt (j)).value ())
            {
              answer.removeElementAt (j);
            }
        }
    }      }

}

public String[] getResults ()
{
  String[] results = new String[answer.size ()];
  for (int i = 0; i < answer.size (); i++)
    {
      results[i] =
        lookup[((Data) answer.elementAt (i)).c1] +
        lookup2[((Data) answer.elementAt (i)).v12] +
        lookup[((Data) answer.elementAt (i)).c2] +
        lookup2[((Data) answer.elementAt (i)).v23] +
        lookup[((Data) answer.elementAt (i)).c3];
    }
  return results;
}  public void paint( Graphics g){         int xpos = 30, ypos = 10, linc = 20, sinc = 10;
        //paint generic shape
        g.drawString("c1", xpos, ypos);
        g.drawString("c2", xpos +linc, ypos +sinc);
        g.drawString("c3", xpos +2*linc, ypos);
        g.drawLine( xpos + 10, ypos, xpos+linc-2, ypos+5);
        g.drawLine( xpos +linc+12, ypos+5, xpos+2*linc, ypos);       ypos += 20;
        for (int i = 0; i < answer.size (); i++)
    {
              g.drawString(lookup[((Data) answer.elementAt (i)).c1], xpos, ypos);
      g.drawString(lookup[((Data) answer.elementAt (i)).c2], xpos +linc, ypos +sinc);
      g.drawString(lookup[((Data) answer.elementAt (i)).c3], xpos +2*linc, ypos);
```

```
                g.drawLine( xpos + 10, ypos, xpos+linc-2, ypos+5);          if( ((Data)
answer.elementAt(i)).v12 == 2){
                        g.drawLine( xpos + 10, ypos-2, xpos+linc-2, ypos+3);                }
        if( ((Data) answer.elementAt(i)).v12 == 3){
                        g.drawLine( xpos + 10, ypos-2, xpos+linc-2, ypos+3);
                        g.drawLine( xpos + 10, ypos+2, xpos+linc-2, ypos+7);                }
                g.drawLine( xpos +linc+12, ypos+5, xpos+2*linc, ypos);              if( ((Data)
answer.elementAt(i)).v23 == 2){
                        g.drawLine( xpos +linc+12, ypos+3, xpos+2*linc, ypos-2);            }
        if( ((Data) answer.elementAt(i)).v23 == 3){
                        g.drawLine( xpos +linc+12, ypos+3, xpos+2*linc, ypos-2);
                        g.drawLine( xpos +linc+12, ypos+7, xpos+2*linc, ypos+2);            }
                        ypos+=20;
          }
         }

}
```

B. Second Computer Program

```
public class Generate
{
        protected String lookup[] = { "", "Li", "Be", "B", "C", "N", "O", "F",
                                                                "Ne", "Sc", "Ti", "V", "Cr", "Mn",
"Fe",
                                                                "Co", "Ni", "Cu", "Zn"};
        protected String lkbond[] = { "", "-", "=", "-=" };
        private boolean atom = true;
        public static final int NUMATOMS = 3;
        public static final int MAXATOM = 8;
        public static final int MINATOM = 1;
        public static void main(String[] args)
        {
                //list of covalent bonds, 2,1 and 1,2 are both required
                int[] bondlist = {
                0,1,
                1,0,
                1,2,
                2,1};
                /*2,3,
                3,2,
                3,4,
                4,3,
                4,5,
                5,4}*/
                //list of cordinate covalent bonds 1,2 represents atom[1] giveing 2 two electrons
                // note 1,2 isn't the same as 2,1
                int[] ccbondlist ={1,2};
                //the electron count for atom ec[i], corrosponds to atom[i]
                int[] ec = {8,8,8,8,8,8,8,8};
                Generate test = new Generate(NUMATOMS,bondlist,ccbondlist,ec);
        }
        int numbonds;
        int numccbonds;
        int[] bondlist;
        int[] ccbondlist;
        int[] elecount;
        int[] bond;// the covalent bond numbers that will change eveytime checkmol is called
        int[] ccbond;// the ccbond numbers that will change eveytime checkmol is called
        int numatoms;
        int atomn[];// the atom numbers that will change eveytime checkmol is called

        public Generate(int na,int[] bd,int[] cbd,int[] ec)
        {
                numatoms = na;
                numbonds = (int)bd.length/4;
                numccbonds = (int)cbd.length/2;
                bondlist = bd;
                ccbondlist = cbd;
                elecount = ec;
                atomn = new int[na];
                bond = new int[numbonds];
                ccbond = new int[numccbonds];
                fillarray(atomn,1);
                fillarray(bond,1);
                fillarray(ccbond,1);
                compute(na);

        }
```

```
public static void fillarray(int list[], int value)
{
        for(int i = 0; i < list.length;i++)
        {
                list[i] = value;
        }
}

public void compute(int pos)
{

        for(int i = MINATOM; i <= MAXATOM;i++)
        {
                atomn[pos-1] = i;

                bondcalc(numbonds);//recursive algorithm for bond numbers;
                if(pos-2 != -1)
                {
                        compute(pos -1);
                }
        }
        atomn[pos-1] = 1;
}

public void bondcalc (int pos)
{

        for(int i = 1; i <= 3;i++)
        {
                bond[pos-1] = i;

                ccbdcalc(numccbonds);//recursive algorithm for bond numbers;
                if(pos-2 != -1)
                {
                        bondcalc(pos -1);
                }
        }
        bond[pos-1] = 1;
}
public void ccbdcalc (int pos)
{
        if (ccbondlist.length == 0)
        {
                checkmol();
                return;
        }
        for(int i = 1; i <= 3;i++)
        {
                ccbond[pos-1] = i;

                // print();
                checkmol();//recursive algorithm for bond numbers;
                if(pos-2 != -1)
                {
                        ccbdcalc(pos -1);
                }
        }
        ccbond[pos-1] = 1;
}
public void print(int[] list)
{
        System.out.print("[ ");
        if(atom){
                for(int i = 0; i < list.length;i++)
                {
                        System.out.print(","+lookup[list[i]]);
                }
                System.out.print(" ]");
        }
        else {
                for(int i = 0; i < list.length;i++)
                {
                        System.out.print(","+list[i]);
                }
                System.out.print(" ]");
        }
}
public void printAll()
{
        for( int i = 0; i < atomn.length; i++){
                System.out.print(lookup[atomn[i]]);
```

```
                if(i < bond.length )
                {
                        System.out.print(lkbond[bond[i]]);
                }
        }
        System.out.println();
}
public int count(int list[],int value)
{
        int c = 0;
        for(int i = 0;i<list.length;i++)
        {
                if (list[i] == value)
                {
                        c++;
                }
        }
        return c;
}//count close
public int cccount(int list[],int value)
{
        int c = 0;
        for(int i = 0;i<list.length;i++)
        {
                if (i%2==0)//if even continue
                {
                        continue;
                }
                if (list[i] == value)
                {
                        c++;
                }
        }
        return c;
}//count close
public void checkmol()
{

for(int i = 0;i < NUMATOMS;i++)
{
        //10's place contains # of covalent bonds of atom i,
        // 1's place contains # of cordinate covalent bonds in atom i
        int bndtype = 0;

        //gets the number of covalent bonds atom i has and stores in 10's bndtype
        switch ((int)count(bondlist,i)/2)
        {
                case 0:
                        break;
                case 1:
                        bndtype += 10;
                        break;
                case 2:
                        bndtype += 20;
                        break;
                case 3:
                        bndtype += 30;
                        break;
                case 4:
                        bndtype += 40;
                        break;
                default:
                        System.out.println("error atom "+i+" can't be bonded to >4 atoms");
        }
      // System.out.println(cccount(ccbondlist,i)+" <<cc "+count(bondlist,i)/2);
        //get the number of cc bonds atom i has and stores in bndtype
        switch ((int)cccount(ccbondlist,i))
        {
                case 0:
                        break;
                case 1:
                        bndtype += 1;
                        break;
                case 2:
                        bndtype += 2;
                        break;
                case 3:
                        bndtype += 3;
                        break;
                case 4:
                        bndtype += 4;
```

```
                        break;
                default:
                        System.out.println("covalent bond error in atom "+i);
        }

        //the previous two switch's are used the determine the bonding of atom 'i' and this
switch statement
        //tests the correct formula
        switch (bndtype)//since the atom donating cc bonds dosn't need a formula there is no
case for them
        {
                case 0:
                        System.out.println("error bondtype 0");
                        //System.exit(1);
                        break;
                case 1://receiving one ccbond
                        if(!((atomn[i]) + (0 + 2) == elecount[i]))//if this statement is
false start next recursion
                                return;
                        break;
                case 2: //receiving two ccbonds
                        if(!((atomn[i]) + (0 + 4) == elecount[i]))
                                return;
                        break;
                case 3: //receiving three ccbonds
                        if(!((atomn[i]) + (0 + 6) == elecount[i]))
                                return;
                        break;
                case 4: //receiving two ccbonds
                        if(!((atomn[i]) + (0 + 8) == elecount[i]))
                                return;
                        break;
                case 10://one covalent bond
                        //System.out.println("i= "+i+" atom #: "+atomn[i]+" bondtype:
"+bond[pos(i,1)]);
                        if(!(atomn[i] + bond[pos(i,1)]  == elecount[i]))
                                return;
                        break;
                case 11: //one ccbond and one covalent bond
                        //   c1   +     ( v12 +  2 )  == 2,8,18
                        if(!(atomn[i] + bond[pos(i,1)] + 2 == elecount[i]))
                                return;
                        break;
                case 12: //two ccbond and one covalent bond
                        //   c1   +     ( v12 +  4 )  == 2,8,18
                        if(!(atomn[i] + bond[pos(i,1)] + 4 == elecount[i]))
                                return;
                        break;
                case 13: //three ccbond and one covalent bond
                        //   c1   +     ( v12 +  6 )  == 2,8,18
                        if(!(atomn[i] + bond[pos(i,1)] + 6 == elecount[i]))
                                return;
                        break;
                case 20://two covalent bonds
                        //    c1    +    v12     + v21  == 2,8,18
                        if(!(atomn[i] + bond[pos(i,1)] + bond[pos(i,2)] == elecount[i]))
                                return;
                        break;
                case 21://two covalent bonds, and one cc bond
                        //    c1    +    ( v12 + 2)    + v21  == 2,8,18
                        if(!(atomn[i] + bond[pos(i,1)] + 2 + bond[pos(i,2)] == elecount[i]))
                                return;
                        break;
                case 22://two covalent bonds, and two cc bond
                        //    c1    +    ( v12 + 4)    + v21  == 2,8,18
                        if(!(atomn[i] + bond[pos(i,1)] + 4 + bond[pos(i,2)] == elecount[i]))
                                return;
                        break;
                case 30: // three covalent bonds
                        //    c1    +    v12     +    v13    +     v14     ==     2,8,18
                        if(!(atomn[i] + bond[pos(i,1)] + bond[pos(i,2)] + bond [pos(i,3)] ==
elecount[i]))
                                return;
                        break;
                case 31: // three covalent bonds and one ccbond
                        //    c1    +    (v12 + 2)     +      v13   +   v14  ==    2,8,18
                        if(!(atomn[i] + bond[pos(i,1)] + bond[pos(i,2)] + bond [pos(i,3)] ==
elecount[i]))
                                return;
                        break;
                case 40: // four covalent bonds
```

```
                                //    c1    +    v12     + v13 + v14  + v15  == 2,8,18
                                if(!(atomn[i] + bond[pos(i,1)] + bond[pos(i,2)] + bond[pos(i,3)] +
bond[pos(i,4)]== elecount[i]))
                                        return;
                                break;



        }//big switch close
        }//atom for loop close
                atom = true;
                System.out.println();
                /*System.out.print("atom ");
                print(atomn);
                atom = false;
                System.out.print("  bond  ");
                print(bond);
                System.out.print("  ccbond  ");
                print(ccbond);
                System.out.println();*/
                printAll();
                //Keyboard.readInt();


/*
        System.out.println("Molecule: "+lookup[atomn[0]]+lkbond[bond[0]]+lookup[atomn[1]]);
        System.out.println();
        System.out.println();
*/

        }//checkmol close


        /*  This method is use to find the 'int bondnum' 'th  occurance of atom 'int atom'
                int the covalent bonding array. This is used to calculate the number of electrons
                that the atom 'int atom' is getting from it's 'int bondnum' covalent bond.
                Eg. bondnum = 1  v12
                                = 2  v13
                                = 3  v14 etc.
                */

        public int pos(int atom,int bondnum)
        {
                int count = 0;
                for(int i = 0;i< bondlist.length;i+=2)
                {
                if (bondlist[i] == atom){
                        count++;
                        }
                if (count == bondnum){
                        int cc = 0;
                        while (i > 4)
                        {
                                i -= 4;
                                cc++;
                        }
                        return cc;
                        }
                }
                return 0;
        }//pos close
}//class close
```

SOUTHERN SCHOLARS SENIOR PROJECT

Name: Jonathon Geach Date: 10/17/00 Major: Biochemistry

SENIOR PROJECT

A significant scholarly project, involving research, writing, or special performance, appropriate to the major in question, is ordinarily completed the senior year. The project is expected to be of sufficiently high quality to warrant a grade of A and to justify public presentation.

Under the guidance of a faculty advisor, the Senior Project should be an original work, should use primary sources when applicable, should have a table of contents and works cited page, should give convincing evidence to support a strong thesis, and should use the methods and writing style appropriate to the discipline.

The completed project, to be turned in in duplicate, must be approved by the Honors Committee in consultation with the student's supervising professor three weeks prior to graduation. Please include the advisor's name on the title page. The 2-3 hours of credit for this project is done as directed study or in a research class.

Keeping in mind the above senior project description, please describe in as much detail as you can the project you will undertake. You may attach a separate sheet if you wish:

Signature of faculty advisor Ray Hefflin D Expected date of completion 3/2001

Approval to be signed by faculty advisor when completed:

This project has been completed as planned: ✓

This in an "A" project: ✓

This project is worth 2-3 hours of credit: ✓

Advisor's Final Signature Ray Hefflin 4/16/1

Chair, Honors Committee _____ Date Approved: _____

Dear Advisor, please write your final evaluation on the project on the reverse side of this page. Comment on the characteristics that make this "A" quality work.

Title: Algebraic Use of Closed Shell Electron Configurations in Predicting Stable Molecular Species

Name: Jonathan Geach

Faculty Adviser: Dr. Ray Hefferlin

My project will be to determine the validity of a model proposed by Dr. Hefferlin for determining molecular structures based up on the added stability of closed-shell electron configurations. I will do this by using the model proposed by Dr. Hefferlin to find all predicted molecules for a given number of atoms. I will then determine whether the molecules predicted are in fact real molecules.

Finding all the predicted models involves determining all the possible three-dimensional structures and all of the possible bonding conditions and solving a set of equations for each case and then finding what molecules the solved equations predict.

After finding the molecules and determining if they exist I will use statistical methods to attempt to show statistical viability.