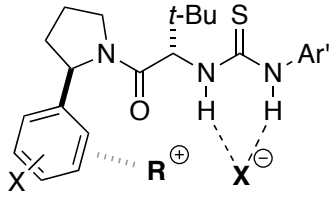
Chem 106: Computational Handout 5

Conformational Analysis of Thiourea Catalysts

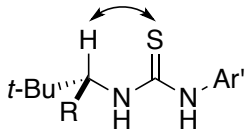
Objectives: rapid optimization, conformational analysis, overlaying structures

In This Exercise: we will consider the conformational behavior of chiral amide catalysts (*Org. Lett*. **2016**, *18*, 3214). Arylpyrrolidine amido-thioureas such as these are known to catalyze many interesting transformations:

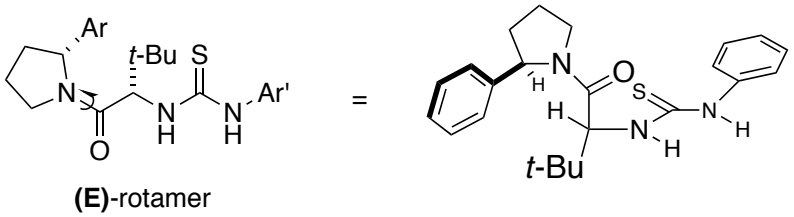


These catalysts are thought to work by hydrogen bonding to anions such as chloride through the thiourea. This binding generates an ion pair in a chiral environment, which can lead to highly enantioselective reactions. Computational studies and linear free energy relationships have demonstrated that precise positioning of the aryl group on the pyrrolidine fragment is crucial for enantioselectivity. In several cases, the enantioselectivity has been shown to arise from an attractive cation-π interaction between the aromatic surface and the cationic partner of the ion pair.

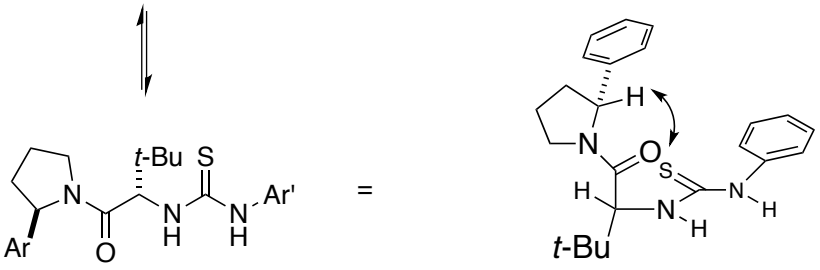
As a result, the conformational behavior of these catalysts is of considerable interest. The conformation about the amino acid is controlled by 1,3-allylic strain:



Placing the indicated hydrogen and thiourea in the same plane minimizes this strain.



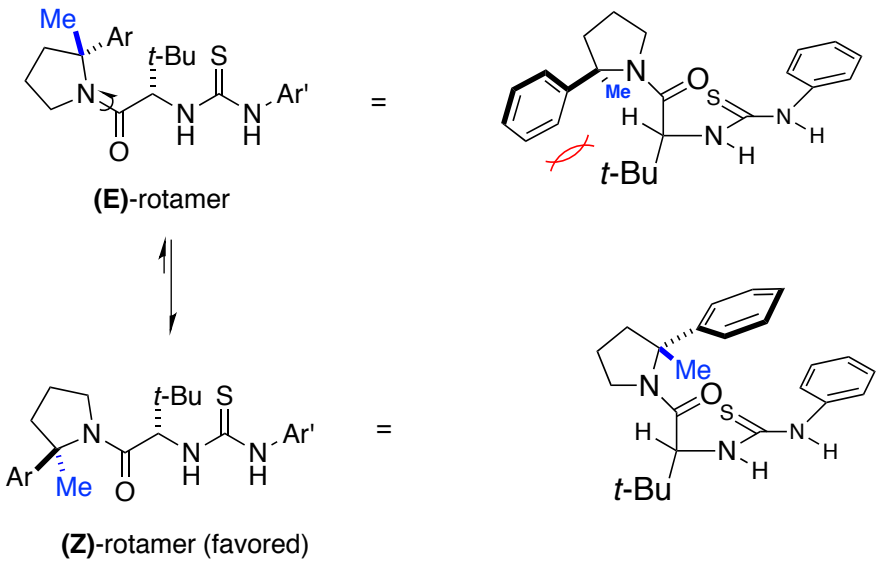
(E)-rotamer



(Z)-rotamer

Similarly, 1,3-allylic strain between the amide and the pyrrolidine hydrogen requires co-planarity. However, which of the E or Z rotamers is favored is not clear from simple drawings and indeed, NMR spectra show a mixture of rotamers.

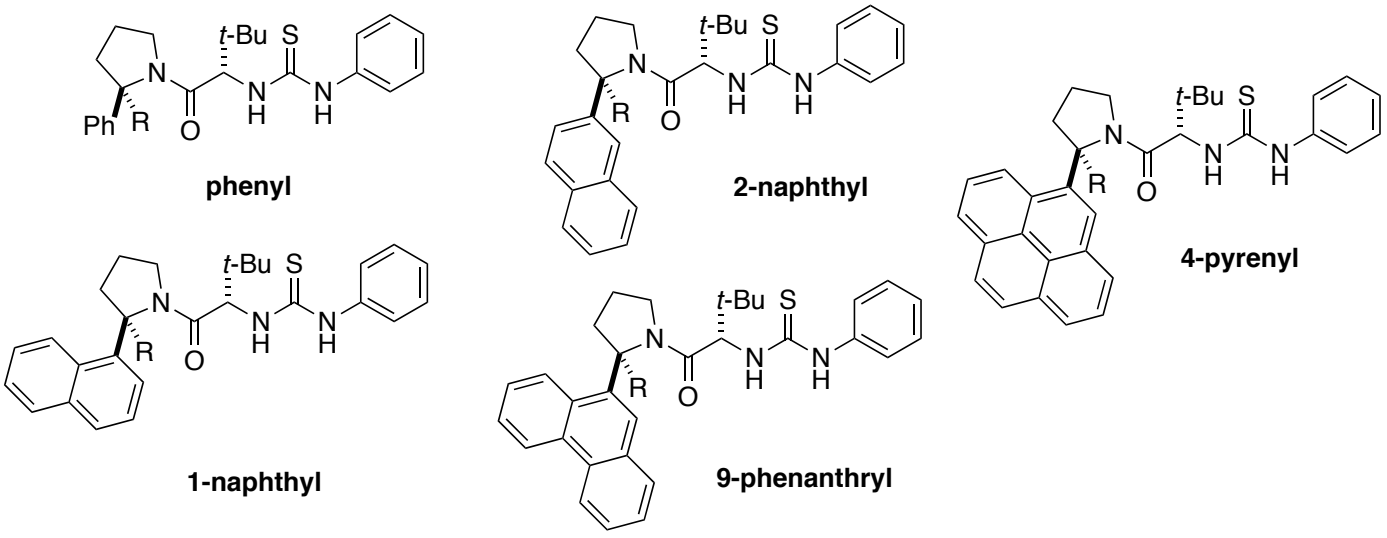
The addition of a methyl group adjacent to the amide is known to bias this equilibrium in favor of the E rotamer:



Accordingly, the NMR spectrum of the this "fully substituted" catalyst shows only the (Z)-rotamer. In this exercise, you will use calculations to predict the size of this effect as a function of the identity of the aryl group. You will also compare structures to determine the nature of this subtle conformational gearing effect.

Structures

Please consider the following ten catalysts, where R = H or R = Me:



Draw each in the E and Z rotamers for a total of twenty structures. Setup a PDDG optimization:

```
%nprocshared=4
%mem=3GB
#p pddg opt freq=noraman scrf=(pcm,solvent=diethylether)
```

Because the structures are relatively big, these jobs will take at least several hours. PDDG will be particularly helpful here because it is difficult to draw the structures perfectly.

Use these filenames:

```
cat_H-amide_E-1nap-pddg-pcm.gjf   cat_Me-amide_E-1nap-pddg-pcm.gjf
cat_H-amide_E-2nap-pddg-pcm.gjf   cat_Me-amide_E-2nap-pddg-pcm.gjf
cat_H-amide_E-4pyr-pddg-pcm.gjf   cat_Me-amide_E-4pyr-pddg-pcm.gjf
cat_H-amide_E-9phen-pddg-pcm.gjf  cat_Me-amide_E-9phen-pddg-pcm.gjf
cat_H-amide_E-Ph-pddg-pcm.gjf     cat_Me-amide_E-Ph-pddg-pcm.gjf
cat_H-amide_Z-1nap-pddg-pcm.gjf   cat_Me-amide_Z-1nap-pddg-pcm.gjf
cat_H-amide_Z-2nap-pddg-pcm.gjf   cat_Me-amide_Z-2nap-pddg-pcm.gjf
cat_H-amide_Z-4pyr-pddg-pcm.gjf   cat_Me-amide_Z-4pyr-pddg-pcm.gjf
cat_H-amide_Z-9phen-pddg-pcm.gjf  cat_Me-amide_Z-9phen-pddg-pcm.gjf
cat_H-amide_Z-Ph-pddg-pcm.gjf     cat_Me-amide_Z-Ph-pddg-pcm.gjf
```

It will be tedious to fix the headers for each of these files, so I have provided a shell (BASH) script called `fix_header.sh` to do it:

```
for i in *.gjf; do
    awk -f fix_header.awk head.txt $i tail.txt > temp.gjf
    mv temp.gjf $i
    echo $i
done
```

This simply goes through each gjf file and runs `fix_header.awk` on it. The text in `head.txt` is pre-pended to the geometry and the `text` in tail.txt is appended to make the final file:

```
~/chem106_calculations/thiourea $ more head.txt
%nprocshared=4
%mem=3GB
#p pddg opt scrf=(pcm,solvent=diethylether)

Title Card Required
```

In this case, `tail.txt` is just a couple of blank lines. Here is part of the awk script:

```
fileCount == 1 || fileCount == 3 {print}

fileCount == 2 {
    if ( length($0) == 0 )
        spaces++
    if ( spaces == 2 )
        {
            getline
            printf "\n"
            spaces += 2
        }
    if ( spaces == 4)
        print $0
}
```

Files 1 and 3 are printed as is, but only the geometry is printed from the gjf file. The geometry is extracted by finding the third “paragraph” of the file.

To run the script, place all the gjf files in the same directory as fix_header.sh, fix_header.awk, head.txt, and tail.txt and type `./fixheader.sh`. This is a very handy tool that you should use to reduce the amount of time it takes to setup calculations.

Optimization

1. Confirm that all structures are converged and are correctly labeled.

2. Re-optimize the structures using B3LYP-D3(BJ)/MIDI!:

```
#p b3lyp midix empiricaldispersion=gd3bj opt freq=noraman scrf=(pcm,solvent=diethylether)
```

3. It will be tedious to open and save every structure in GaussView, so we will use an awk script (this is `extract_geom.awk` in the repository under `thiourea/`). Here's how to run it:

```
awk -f extract_geom.awk *pddg*.out
```

Remember, the `-f` flag tells awk to run code from a text file. This is useful when you want to run multiple awk commands at a time. The wildcard specification feeds all files matching the pddg description to awk, one at a time, one line at a time. Here is a step-by-step explanation:

```
FNR == 1 {
    file_count++
    filenames[file_count]=FILENAME
}
```

You've already seen this part, which keeps track of the filenames (`FNR==1` is a pattern that matches the first line of every file).

```
/Standard orientation/,/Rotational constants/ {
    if ( NF == 6 && match($0,"[a-zA-Z]") == 0 )
        {
            symbol[file_count,$1]=$2
            x[file_count,$1]=$4
            y[file_count,$1]=$5
            z[file_count,$1]=$6
            atoms[file_count]=$1
        }
}
```

This is a "range pattern," which asks awk to find all the text between `Standard orientation` and `Rotational constants`. We automatically get the last geometry in each file because the geometry on each step is successively overwritten in the arrays. `Symbol`, `x`, `y`, `z`, and `atoms` store the atomic symbol (actually the atomic number), the Cartesian positions in Å, and the number of atoms in each file, respectively.

Arrays in awk are "associative," meaning that each array is a collection of mappings from one string to another. In this case, `file_count,$1` is evaluated to its value (e.g., `5,1`) and mapped to some other value (e.g., the x position). (The comma is replaced with an invisible character, so the key is, in fact, a string. Similar constructs are available in other languages such as `HashMap<K,V>` in Java and `dict` in Python.)

```
END {
    for (i=1; i <= file_count; i++)
        {
            filename = filenames[i]
            gsub("pddg","b3lyp_d3bj-midix",filename)
            gsub(".out",".gjf",filename)
            print filename
            print "%nprocshared=4\n%mem=3GB\n#p b3lyp midix empiricaldispersion=gd3bj opt
freq=noraman scrf=(pcm,solvent=diethylether)\n\ntitle\n\n0 1" > filename
```

```
            for (j=1; j <= atoms[i]; j++)
                printf "%2d %15.8f %15.8f %15.8f\n", symbol[i,j], x[i,j], y[i,j], z[i,j]
>> filename
            printf "\n\n" >> filename
        }
}
```

At the end of the file, we loop through each file. We construct a new filename using the `gsub` command. This is a command that performs "search and replace" on a string (similar to `sed` on the command line). The syntax is `gsub(regex,replacement,target)`, where `regex` is the text to search for, `replacement` is the text to replace it with, and `target` is the name of the variable to perform the replacement on. You can think of regular expressions as generalized wildcards (like `*`). They are very useful, but outside the scope of this exercise. For now, avoid using any punctuation (because punctuation marks have special meaning in the regex language). If you must use punctuation, you must escape each mark with a backslash (`\`).

We print out the filename to the screen (known as `stdout`), but then we print out the header/route card to `filename`. The `>` sign means that the file, if it exists, will be replaced with the indicated string. We then loop through the geometry and use `printf` to print out the coordinates (`%2d` = integer with a width of two characters, `%15.8f` = floating point number with a width of 15 characters and 8 numbers after the decimal place, `\n` = newline character). The `>>` sign means that the given string will be *appended* to the file.

This is a very useful script and you should considering modifying it as needed for future steps in this exercise and for your project/research.

4. Use one-liners to verify that each job converged correctly. Use the above script to re-optimize the final geometry at B3LYP-D3(BJ)/6-31G*/PCM.

5. Perform single points at B3LYP-D3(BJ)/cc-pVTZ/PCM.

These jobs will take on the order of six hours. The 6-31G* optimization will take about as long as the triple-zeta single points because the former involves a number of steps while the latter involves only one energy calculation.

Analysis

Note: If you ran out of time, the output files are provided in the git repository.

You will only need the cc-pVTZ electronic energies. Let's write an awk script to pull out the energies and save them into a comma-separated values (csv) file, which can be opened in Excel (this is `catalyst.awk` in `thiourea/out/`):

```
FNR == 1 {
    file_count++
    filenames[file_count]=FILENAME
}

/SCF Done/ { energy[file_count] = $5 }

END {
print "substituent,rotamer,R_group,energy" > "analysis.csv"
for (i=1; i <= file_count; i++)
        {
            filename = filenames[i]
            n_fields = split(filename,fields,"[_-]")
            printf  "%s,%s,%s,%.8f\n",  fields[2],  fields[4],  fields[5],  energy[i]    >>
"analysis.csv"
        }
}
```

The only new part here is the `split` command, which takes a string and cuts it into fields based on a separator you designate. The syntax is `split(var,target,fieldsep)`, which cuts `var` based on `fieldsep` and places the result in `target`. The actual return value of split is the number of fields (`n_fields` here). Here, I used a very simple regex, `[_-]`, to tell `split` to consider both _ and - as field separators.

1. Run this script on all the single point files.

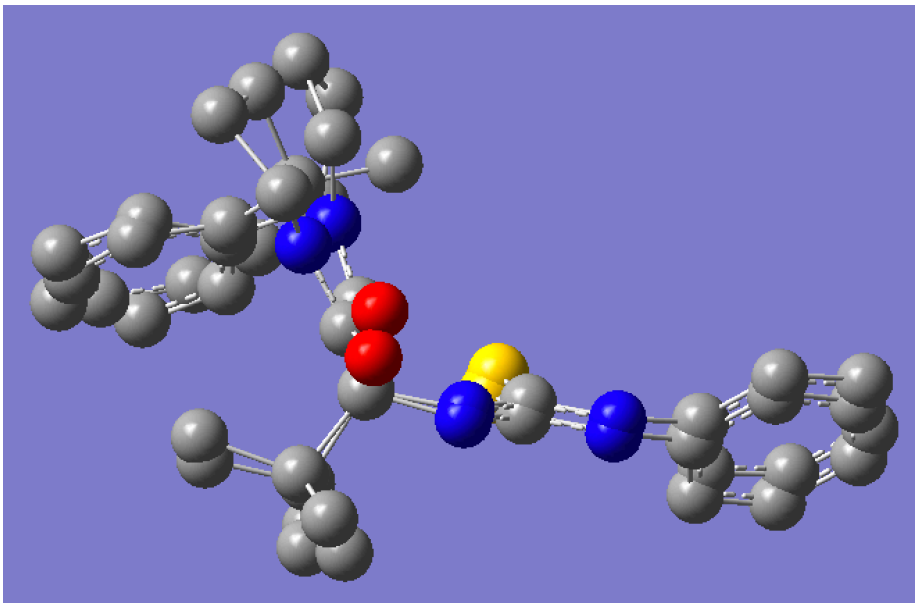
2. Open the csv file in Excel. Sort the results and calculate the E/Z energy differences for each catalyst pair.

Discussion

Aryl Group	Preference for Z Rotamer (kcal/mol)	
	R=H	R=Me
Ph	0.41	3.39
1-naphthyl	0.40	5.36
2-naphthyl	0.44	2.99
9-phenanthryl	1.27	3.50
4-pyrenyl	1.51	4.51

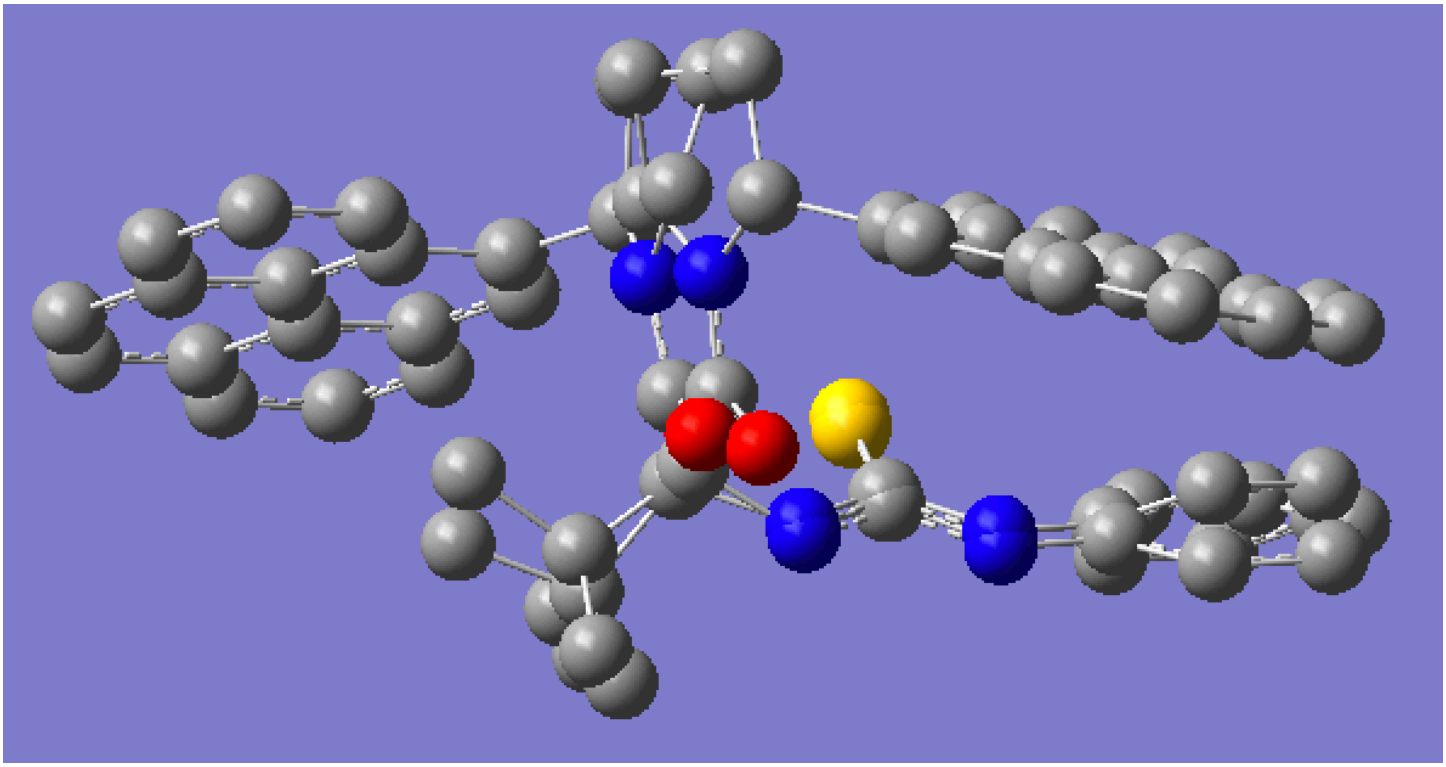
1. Adding a methyl group to the phenyl catalysts dramatically increases the preference for the Z rotamer. Open the E rotamer files for the R=H and R=Me catalysts. In one window, click on Edit…Copy to copy the structure to the clipboard. Switch to the other structure and click on Edit…Paste…Append Molecule.

2. Turn off the hydrogens with View…Hydrogens. Rotate the structures until the thioureas are aligned. What is the effect of adding a methyl group on the geometry of the arylpyrrolidine?



Evidently, adding the methyl group rotates the aryl group towards the tert-butyl group. Confirm that this is the case by comparing some of the distances.

3. As the size of the aryl group is increased, the preference for the Z rotamer increases in the R=H series. Overlay the E and Z rotamers for the R=H 4-pyrenyl catalyst. What do you think is driving the preference?



The E conformation seems to have an unavoidable steric interaction between the pyrene and the *tert*-butyl group, but there may also be an π-π interaction between the pyrene and the phenyl group. This attractive interaction is presumably stronger with the larger catalyst.

4. Many of the structures seem to have a non-planar relationship between the thiourea and its adjacent aryl group, which may be suspicious. How would you evaluate whether this is a realistic geometrical preference using computations? How could you corroborate this experimentally?

5. More generally, what are the limitations of this analysis in terms of predicting the E/Z preference quantitatively?

Eugene Kwan
March 2017