

Combimol B.

A Bainalabs Product

(c) William Bains. Version 3.8, Jun 2017.

Contents

Description of the program	2
Algorithm in brief	2
Running Combimol	2
More detail on the algorithm.....	4
Rules for bond changes.....	4
Rules for atom substitution.....	5
Special atoms and molecular size.....	5
Some minor aspects of atom swapping.....	6
Stability	6
Aromatic bonds.	7
More details on the input files.....	7
The SDF file.....	7
A common problem with SDF files	8
The combimol-atoms.txt file	8
Combimol-bonds.txt file	9
Stupid error messages	9
Batch and parallelizing Combimol.....	10
Batch	10
Multiple instances	10
Other notes.	11
Combimol configuration statements.....	14
Example file sets.....	17
Butanes	17
Amino acids.....	17

Description of the program

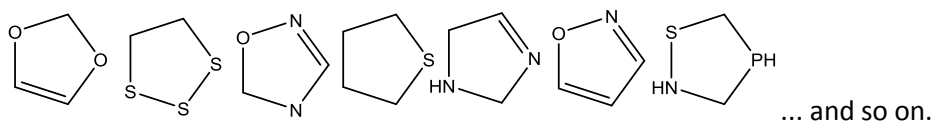
Combimol generates all the variants on a chemical structure that are consistent with bonding rules, filtered for stability. You provide the basic atomic topology (i.e. what shape you want the molecule to be), Combimol substitutes the atoms.

Algorithm in brief

Combimol makes a lot of molecules with different atoms in a molecular topology that you define. Say you want a lot of five-membered rings. You input this structure



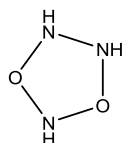
i.e. a five-membered ring made of carbon atoms linked by single bonds, into Combimol. It outputs



The reason for not getting Combimol to generate its own topology is so you can select what topology you think is relevant for your project.

The program does this in two stages. First it generates 'cores' by adding double, triple and aromatic bonds, according to a set of rules. For each of these cores it then runs through and changes atoms wherever it can, according to the rules laid out in the *atoms* and *bonds* files.

This would result in a) huge duplication and b) some spectacularly stupid structures like



You can elect to keep the stupid structures, or to filter them out. The filter is not 100% effective, but it is pretty good. There are a variety of rules about which atoms can be substituted where, which stops you make trivalent chlorine etc..

The duplication is dealt with at three levels; making sure that cores are unique (very fast, as there are not many of them), making sure that output SMILES are unique (fast) and making sure that output molecules are unique (slower).

Output is as a text file containing SMILES (which can also have InChI keys and InChI codes in it), and/or an SDF file (+/- InChI keys and codes)

Running Combimol

Nothing could be easier, old chap! Providing you have a Windows PC. This does not run under Unix (sorry) or on a Mac¹.

Combimol and its data files need to be in the same directory. If they are not, it will say something is missing, and stop. These files are:

- Your structure input file. This provides a list of structures you want permutations on.
- `combimol-b-config.txt`. This provides the way to configure what the program does with your structures. It is a text file with a series of simple statements in. See the section 'Combimol

¹ If you have a Mac, all I can say is – go out and buy a computer.

configuration statements' for a list of them and what they do. This file can be empty (there are defaults for everything), but the file must be there.

- Combimol-atoms.txt . This defines what atoms Combimol will use, and some of their characteristics
- Combimol-bonds.txt . This defines the bonds that Combimol is allowed to make between the atoms
- Inchi-1.exe . This is an independent executable that calculates InChI codes. If you use any of the InChI-using options, you need this. (this is sourced from <http://www.inchi-trust.org/downloads/>).

Everything except the structure input file can be used 'off the shelf' as provided. I also provide two sample sets of inputs in the folders *butane* and *amino acids*. See the section at the end of this document on what they do.

You can run Combimol in three ways. Double clicking the Combimol-b.exe file opens the program, which will then ask for your filename. Note that you have to include the three-letter extension (i.e. *butane.sdf*, not just *butane*). If you enter an incorrect filename, it will ask you for another.

Alternatively, you can drag your structure file and drop it onto the program. Lastly, you can run it from the DOS command line with the command

```
>combimol filename
```

where *filename* is the name of your structure file *including* the 3-letter extension. If you enter an incorrect filename in the DOS command line version, it will just quit. Thus the DOS version requires no mouse input at all. This may be useful for setting up batch file runs if you are going to set some machine to generate structures for a month or two (see 'running in batch' below).

Output is any of three files, depending on the configuration.

- A file of SMILES strings. If your input file name was *molfile.sdf*, then the SMILES will be in the file *molfile_smiles_output.txt*. You can elect (again in the configuration file) to have this as a list of SMILES alone, or a list of SMILES, and InChI codes and InChI keys for those SMILES.
- An SDF structure file of your output. This will have fields for InChI keys and InChI codes as well, but if you have not used any InChI-generating option in the output then the InChI code field will be blank and the InChI key field will have *N/A* in it. The file will be called *molfile_output.sdf*
- A trace file. If you have opted for this, then the screen output (plus one or two other things to help me track how long the program takes to run) is output to a file called *molfile_trace.txt*.

Screens while it is running look like this. This is the manual input version, the drag-and-drop and batch versions look the same, but do not request a filename at the start.

Enter your input file name when asked here. No other input needed

As it runs through the cores it keeps you updated

This number counts the SMILES it has on the current core, mainly just to let you know it is doing *something*

```

Combimol-B
V 3.5
William Bains' random chemical generation program
Please refer to documentation to explain what this does, and how

(c) William Bains 2016, 2017. All rights reserved.

Press F10 to abort at any time
What filename? pentane.sdf
CCCC
Generated intermediate molecules. Starting atom substitution. Stand by ...
Core number 1 [ CCCCC ]:- 4134 SMILES: Total 4134
Core number 2 [ CCCC=C ]:- 1066 SMILES: Total 5200
Core number 3 [ CCCC#C ]:- 378 SMILES: Total 5578
Core number 4 [ CCC=CC ]:- 838 SMILES: Total 6416
Core number 5 [ CCC=C=C ]:- 260 SMILES: Total 6676
352
  
```

I had inchi=check turned on, so the program checks for duplicates in InChIs, and tells you that

Final number less than the last "total" because of duplicates removed

Press return to close this screen.

```

Combimol-B
Core number 4 [ CCC=CC ]:- 838 SMILES: Total 6416
Core number 5 [ CCC=C=C ]:- 260 SMILES: Total 6676
Core number 6 [ CCC#CC ]:- 423 SMILES: Total 7099
Core number 7 [ CC=CC=C ]:- 162 SMILES: Total 7261
Core number 8 [ CC=CC#C ]:- 48 SMILES: Total 7309
Core number 9 [ CC=C=CC ]:- 196 SMILES: Total 7505
Core number 10 [ CC=C=C=C ]:- 56 SMILES: Total 7561
Core number 11 [ CC#CC=C ]:- 72 SMILES: Total 7633
Core number 12 [ CC#CC#C ]:- 18 SMILES: Total 7651
Core number 13 [ C=CCC=C ]:- 320 SMILES: Total 7971
Core number 14 [ C=CCC#C ]:- 122 SMILES: Total 8093
Core number 15 [ C=CC=C=C ]:- 52 SMILES: Total 8145
Core number 16 [ C=C=CC#C ]:- 16 SMILES: Total 8161
Core number 17 [ C=C=C=C=C ]:- 16 SMILES: Total 8177
Core number 18 [ C#CCC#C ]:- 48 SMILES: Total 8225

Finished raw list output. Calculating InChI codes
Sorting and outputting final list. Stand by .....
5679 final records

Started at 21:32:28 on 01-03-2017
Finished at 21:34:41 on 01-03-2017
? _
  
```

For the manual entry version, the program pauses at the end until you press return. For the batch or drag-and-drop version, the window just closes.

More detail on the algorithm

Rules for bond changes

Any bond between two carbons in the original molecule can be changed, along certain rules. These are hard coded into the program. They are:

- Only single bonds between uncharged carbon atoms can be changed
- You cannot make a double bond to a carbon that already has a double, triple or aromatic bond to it if it is in a ring
- If 'cumulenes=no' in the Config file. then you cannot make a double bond to a carbon that already has a double or an aromatic bond to it anywhere. NOTE if you say that the 'special

atoms' S=O, P=O etc. can support double bonds, then you can still generate C=S=O structures even if 'cumulenes=no'.

- You cannot make a triple bond in a ring
- You cannot make an aromatic bond to a carbon which already has a double or triple bond to it (note that you *can* make an aromatic bond to a carbon which has another aromatic bond to it).
- The total number of bonds to a carbon cannot exceed 4.

Rules for atom substitution

Any uncharged carbon atom in the 'cores' can be changed for any of the atoms listed in *combimol-atoms.txt*, with the following rules. These *are* all coded in the configuration files, and so can be changed.

- Only uncharged carbon atoms can be swapped. An atom can only be substituted into a site with at least a *minimum* number of bonds to it and no more than a *maximum*. The minimum and maximum are set in the file *combimol-atoms.txt*. Thus if you want an atom that has to have exactly three bonds to it, you set *minimum* = *maximum* =3.
- If there is a double bond to that atom, check that atom can support double bonds. If it cannot, that swap is not allowed
- Check each of the bonds from the swapped atom to all other atoms in the molecule. Look up those bonds in the matrix in *combimol-bonds.txt*. If that bond is allowed, the matrix will have a '1', if not, it will have a '0'. If any of the bonds to that atom are '0', then the swap is not allowed.

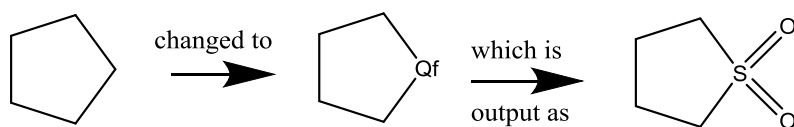
Special atoms and molecular size

Combimol cannot handle atoms with valencies of >4. To get round this for sulphur, phosphorus, nitrogen and arsenic, Combimol has some special 'atoms' defined internally:

- Qd – S=O
- Qe – S=S
- Qf – S(=O)=O
- Qg – S(=S)=O
- Qh - P=O
- Qi - P=S
- Qk – N([O-])=O (nitro)
- Qm – As(=O) (arsenate)

You can set the valencies for these 'atoms' in *combimol-atoms.txt*, but the file that comes with Combimol has suggested valencies (Qd, Qe, Qf, Qg = 2, Qh, Qi = 2 or 3, Qk=1, Qm=3). The actual letter codes are purely used internally and are replaced by the appropriate real atoms. However note that:

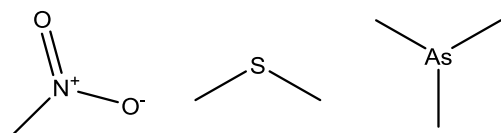
- 1) If you want to include these 'atoms', then they need to be defined in the *combimol_atoms* and *combimol_bonds* files.
- 2) Because they contain more than one actual atom each, they result in molecules that are larger than the original cores. Thus if you include Qf as an 'atom' that can be used to replace carbon, you get this



If molecular size is important to you, you will need to post-process the file to cope with this.

Some minor aspects of atom swapping

As noted, when Combimol swaps an atom for another atom, it checks three things. This means that all the atoms in the input molecule must be specified in the two files above. If they are not, then Combimol says 'Oh, I cannot decide whether I can change an atom in here, so I had better not'. This causes problems if, for example, you want to run variants of these molecules



with a *combimol-atoms.txt* file that only allows C, N, O and P. Combimol will see the carbons in those molecules as being bonded to atoms it does not recognise, and so cannot look up in the *combimol-bonds* table, and therefore not will replace them.

The way round this is to generate a *combimol-atoms.txt* file, and corresponding *combimol-bonds.txt* file, that defines the atoms, but in a way that means they are never used as replacement atoms. For the example above, this *combimol-atoms.txt* file would work:

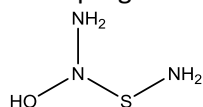
```
6
N, 1, 3, 0, 1, 2
O, 1, 2, 0, 1, 1
P, 2, 3, 0, 0, 1
S, 9, 9, 0, 0, 0
N+, 9, 9, 1, 0, 0
As, 9, 9, 1, 0, 1
```

Thus As, S and N+ are defined, but the file says that you can only put them in a molecule to replace a C atom with 9 bonds to it. As that never occurs, the replacement will never happen.

Stability

The stability of a molecule is really hard to evaluate. It depends on temperature and chemical environment. There are two ways of making sure that a molecule output by Combimol is stable.

- 1) The bond matrix. The file *combimol-bonds.txt* lists which atoms you can link with a bond. If some bond types are known to be unstable under your conditions of choice, then you can eliminate them. Thus for example Si-Cl bonds are hydrolysed rapidly under any conditions where water is liquid, and so for stability in aqueous solution you can just put the Si-Cl bond to '0' in the file *combimol-bonds.txt*
- 2) Filters in the program. There are three settings of the program (defined in *combimol-config.txt*) that switch these on
 - a. You can ignore stability. If you want to get everything, do this, and then use some other program to filter out output that looks like this



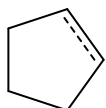
- b. You can select for molecules in which every atom is joined to a carbon atom (the 'carbon' setting)
- c. You can screen out molecules in which atoms are either bonded to a carbon, or are bonded to an atom which is bonded to a carbon (the 'chains' setting). Which the program chooses depends on the *combimol-atoms.txt* file (i.e. you can change it). See below for more on that file. For atoms where the 'chain' parameter is >0, then it will look to see if the atoms it is bonded to are themselves bonded to a carbon.

Note that both these stability criteria are aimed at organic chemicals. Thus they will exclude phosphoric acid, sulphuric acid, hydrogen peroxide as being 'unstable'.

None of these are 100% reliable – any will rule out some perfectly sensible molecules, and include a few that are unstable.

Aromatic bonds.

For reasons lost in the mists of time, the program can insert 'isolated' aromatic rings into structures where they make no chemical sense, like this:-



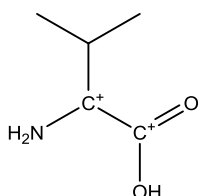
Combimol can insert these into rings only, into any structure (inserting them into chains makes even less chemical sense than into rings), or leave them out. For any real molecule, leave them out. This is controlled by a switch in the *combimol-b-config.txt* file. Note that anything that uses the InChI code generator will produce odd results if there are isolated aromatic rings in the output, as this makes no sense to InChI.

More details on the input files

The SDF file.

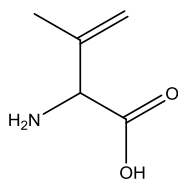
You define the topology in a chemical structure file. You provide a list of carbon atoms and their bonds, which Combimol takes as its starting point. The input structure file can be a MOL file (single molecule) or an SDF file (multiple molecules). Many structure drawing programs output these formats.

The program will swap only carbon atoms in the structure you input, so other atoms are left unchanged. You can also flag a carbon as 'do not change' by giving it a positive charge. (This may be a chemically unrealistic structure – sorry, it's the easiest way to flag an atom). Thus this structure

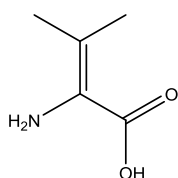


will give a lot of variants of valine with only the side-chain changed. The charge is only used as a flag in the input file – it is removed from the output, so there will be no charged carbons in the output (unless you define them as such in the *combimol-atoms.txt* file - but if you do that do not call them "C+" as I am not sure what that will do).

The program will only change bonds that have a changeable carbon on both ends (i.e. the non-charged carbons). Thus the structure above will generate



But not



In principle you can also have pentavalent or hexavalent carbon atoms defined in your file too, but the resulting molecules would be a bit odd.

A common problem with SDF files

If you manipulate the input SDF file by hand, do not put an extra <cr> after the final \$\$\$\$\$. Doing so will lead Combimol to think there is another molecule coming, and produce a 'Line959 input past end of file' error.

The combimol-atoms.txt file

The file *combimol-atoms.txt* has quite a lot of text in it describing what the various components are for. You can add atoms as you want (You want to define an element Yg, which has a valency of 7 and can only bond to phosphorus atoms? Be my guest!) Note that the data elements for each atom have to be separated by commas, not tabs.

The data columns are:

- First – letter code for atom
- Second and third – minimum and maximum number of bonds for that atom
- Fourth – '0' if this is a default element in the SMILES specification, '1' otherwise
- Fifth – can it support double bonds? 1= yes, 0=no
- Sixth – how many atoms can there be between this atom and a carbon atoms for a stable molecule? 0 = this must be bonded to carbon. [NB the non-zero number currently does not matter here – one day I will implement a more sophisticated chain-counting algorithm to use the >0 values in this column.]

Note that in the definition of the atoms, if you define an atom that is not part of the standard SMILES set, then SMILES will have problems with it *unless* you define it in its various hydrogen-linked forms. Thus selenium has to be defined as two atoms: [SeH] (which has a valency of 1) and [Se] (which has a valency of two). See the example in the *butanes* example folder. Non-standard atoms that are described in this way will be 'split' into their core atoms and explicit hydrogens in the output SDF file. Whether SeH is coded as [SeH] or [Se][H] in the output SMILES depends on the combinations of configuration statements used.

If you are going to have 'atoms' like SeH or SiH in your set, then you should also define Hydrogen as an atom. For the standard SMILES set of atoms you do not need to define Hydrogen – the SMILES

rules (and SDF rules) are that for the standard atom set unfilled valencies are assumed to be filled by Hydrogen.

Carbon is not on the list. The 'default' atom is carbon, and Combimol uses this list as a list of atoms to exchange for carbon.

Combimol-bonds.txt file

This is an array of NxN 0s or 1s, where N is the number of atoms in the *combimol-atoms.txt* file. 1 means you can have a bond between those two atoms, 0 means you cannot. Note that if the file is not symmetric, Combimol will say you can have bonds between (e.g.) P and N, but not between N and P, which makes no sense – checking this is up to you.

Again, this is a comma-separated file. The rows and columns refer to the atoms in the same order as in the *combimol-atoms.txt* file. To make this comprehensible, I also include an Excel spreadsheet version. If you want to change the data file, copy the array of 1s and 0s, open a new text document, paste-special as text. Global replace every tab with a comma. Save as a new *combimol-bonds.txt* file. I suggest you keep the old one.

Note that the *bonds* file assumes the same number of atoms in the same order as in the *atoms* file. There is no checking on this, so if you change the order or number of atoms in the *atoms* file, you must also change the *bonds* file or get silly results.

Note that it is assumed that anything can be bonded to carbon – this does not have to be explicitly stated in the file. Thus in the example in the *butanes* folder, selenium is listed as not bonding to anything! But in fact this means it will not bond to anything *other than* carbon..

Stupid error messages

No-one ever puts helpful error messages into their programs. It is all "Exception at 10aa132." Or "The program has terminated unexpectedly" to which one is tempted to say "No shit!". So I have a few error traps that give unhelpful errors messages, which mean as follows

- "Warning! Megastructure problem". When filtering the output by InChI key to ensure unique molecules only, the program has reached the maximum number of InChI keys it can handle.
- "Warning! Bathtub overflow". More than 100 atoms (the maximum Combimol can handle) in a molecule in the input SDF file
- "Fishing net occurrence". More than 100 bonds in an input molecule.
- "Unpopular penguin error!" A charged atom that has a symbol of >1 letter has been found in the input molecule. Combimol only expects C, N, O, S, P to be charged in input structures. Other elements (As, Cl etc) cause it to boggle.
- "BFG error". Too many SMILES generated in the intermediate atom substitution stage. The program carries on, but produces unreliable results.

These do not trap all the errors that might occur. A couple of others are

- 'Line 1107 input past end of file' or 'bad file number' pop up window error. This is because the file INCHI-1.EXE is present (which the program checks for), but has not run. This may be because you have installed the 64 bit version on a 32 bit machine, or some similar incompatibility.

Batch and parallelizing Combimol.

Batch

As above, Combimol can be run in batch mode. Create a file called

Filename.bat

And windows will run it as a batch file². The file just contains the following

```
Combimol filename_1.mol
Combimol filename_2.sdf
etc
```

where *filename_1*, *filename_2* are the names of the structure files you want Combimol to run on. This will run through running Combimol on all those files automatically. Neat, eh?

Why do this and not just put all the molecules in an SDF file? Two reasons

1. You might want the output separated into different files. This is a convenient way to do that – otherwise Combimol just outputs one vast SDF or SMILES file
2. There may be so many combinations that the InChI de-duplication checking fails, so you want to split your list up, or it does not fail but never actually finishes.

The folder ‘amino acids’ provides an example of running a batch file to generate variants on all the 19 chiral proteinaceous amino acids, using only C, N, O and S(II).

Multiple instances

Combimol is written in good old 1960s-style linear code, with no multiple threading etc.. You cannot run two instances of Combimol from the same folder, as there are some common files that the program needs to write to, and having two instances of the program write to them at the same time will be *bad*.

If you want to run more than one instance of Combimol at once (e.g. you have a 4-processor PC and want to run 4 instances on different input files), then copy the program, all the files listed above in the *running Combimol* section, and your data file into a new folder. You can then start the program from that folder and it will run. You can do this as many times as you want, but I find if you run more instances than your PC has processors then the overall speed is reduced.

If you are running your PC flat out on Combimol, it might be a good idea to set their priority to ‘low’, so that the keyboard and mouse take priority over Combimol, and so you are not effectively frozen out of your own PC. Set them *all* to ‘low’, or one will hog all the processor time. On a PC, do this by:

- Press <ctl><alt>
- Select ‘start task manager’
- Select ‘processes’
- Look for ‘Combimol’. Right click on that, go down the menu to ‘Set priority’, click, go down the sub-menu to ‘low’.

If you do not plan to use the PC for anything else until Combimol is finished, this is not useful. If you have compiled this for a Mac, well, there really is not much I can do to help you.

² I do this by creating a text file called *filename.txt*, and then changing the *txt* to *bat*. Windows comes up with a warning asking if I really want to do this, to which I say “Sure, man, why not?” This only works if you have the folder option showing the three-letter extensions switched on. There may be better ways ...

Other notes.

- SMILES. I wrote my own SMILES generator for Combimol, which as a result has an extremely eccentric output style. In essence, it does not use brackets, it uses branching numbers instead, so isobutane is output as CC1C.C1, not as CC(C)C. This should not be a problem, it just looks weird if you are used to reading SMILES. As noted, the program will translate 'non-SMILES standard' elements that have hydrogens in them into one of two formats, one with explicit hydrogens, one (as coded in the combimol-atoms.txt file) without explicit hydrogens. Both are acceptable to the SMILES definition, but they can look even odder. Thus disilane (Si_2H_6) can be output as

[SiH3][SiH3]

Or

[H][Si]12[Si]34[H].[H]1.[H]2.[H]3.[H]4

- While it has the same name as my old Combimol program, this is completely different in usage, method, output, even language. I doubt you could reproduce the old Combimol output using any combinations of inputs to the new program. This is probably a good thing.
- Any atoms added by Combimol, either through parsing the internal Qx atoms or adding hydrogens to non-standard atoms as in the last item, will not have correct 3D coordinates. Combimol has no understanding of the 3D arrangement of atoms in space.
- It is assumed that SDF files do **not** explicitly code hydrogens. Some programs code methane like this

```
methane
ChemDraw01031714432D

1  0  0  0  0  0  0  0  0999 V2000
  0.0000  0.0000  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0
M  END
```

Some code it like this

```
methane
Chem3D Core 15.101031714413D

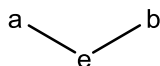
5  4  0  0  0  0  0  0  0999 V2000
-0.7539 -0.3008 -0.0188 H  0  0  0  0  0  0  0  0  0  0  0  0
 0.3586 -0.2707 -0.0062 C  0  0  0  0  0  0  0  0  0  0  0  0
 0.7010  0.7884 -0.0062 H  0  0  0  0  0  0  0  0  0  0  0  0
 0.7539 -0.7884 -0.9087 H  0  0  0  0  0  0  0  0  0  0  0  0
 0.7334 -0.7817  0.9087 H  0  0  0  0  0  0  0  0  0  0  0  0
1  2  1
2  3  1
2  4  1
2  5  1
M  END
```

The second is *bad*. (More specifically, Combimol will not try to replace the hydrogens with anything, as they are not carbon, it will only replace carbon with things that can take 4 bonds, because it thinks there are 4 bonds to the carbon from other elements, and so this will output only methane using the basic element set.)

Note however that Combimol *does* output explicit hydrogens for some atoms, as noted above. As it is pointless using Combimol's own output as its input, this is not a problem for this program.

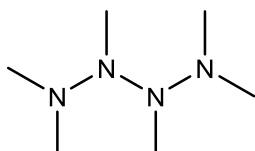
If in doubt, open your SDF file in a text editor and see if it has “H” atoms listed like the second example above. The first example is from Perkin Elmer’s ChemDraw, the second from Chem3D.

- There is no checking for isomerism. The most common problem this gives rise to is it outputting keto and enol isomers of carbonyl compounds as separate compounds.
- There is a specific kludge in the ‘stability=chains’ stability check for detecting and removing structures of the sort

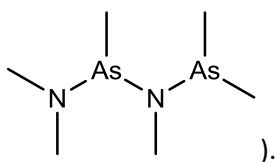


Where a and b are N, O or S, and e is C or N. This removes cis diols and related structures. It may also remove some valid structures as well – I have not checked exhaustively.

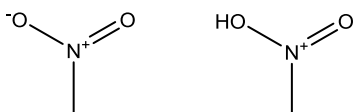
- If you have an atom with a formal charge on it (as I do in the example in the *butanes* folder) then Combimol will include that in the structure according to the rules you specify, but will *not* add counter-ions.
- Formally, both the ‘chains’ and ‘carbon’ stability criteria can allow for structures like this



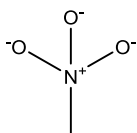
There is a specific fix for this built into the program which is implemented by the ‘stability=chains’ option, which should weed out chains of N, P and B. However other trivalent elements will still be liable to such chains if you allow them to bond to themselves or to other trivalent atoms (for example, if you include arsenic in the atoms and bonds definitions, and allow N-As or P-As bonds, then you will get this



- Combimol gets a bit muddled over charged atoms. Specifically, it cannot distinguish



and the cis-diol filter does not recognise O⁻ atoms as equivalent to OH, and so allows structures like this



I could fix both these, but then it would not generate nitro groups or pyridine oxides etc.. I have therefore included a specific atom type for nitro, and you will just have to do without pyridine oxides.

- InChI deduplication does some tautomer removal, e.g. it detects CS(=O)S as being the same as CS(=S)O, and removes the latter. However it does not understand keto-enol isomerism.

- The InChI Key generated only varies in the topological first 14 characters (Layer 1 of the InChI), as there is no steric information in the structures. The program pastes the last 16 characters into every key just to make it the 'right' length, but they are all the same.

Combimol configuration statements

V 3.8

These are statements that go into the file 'combimol-b-config.txt'. The file must exist, but can be empty. Note that Combimol does not parse these in a clever way – it just looks precisely for the strings 'arom=no' etc..

arom=

This can be arom=yes, arom=rings or arom=no .

- arom=yes allows the insertion of isolated 'aromatic' bonds, as in CC:CC .
- arom=rings allows the insertion of aromatic bonds, but only in rings.
- arom=no does not allow isolated aromatic bonds.

default is 'arom=no' . NB all real molecules are 'arom=no'

stability=

This can be stability=carbon, stability=chains, stability=no. This is a filter for molecular stability

- Stability=carbon requires every atom to be bonded to at least one carbon atom. This will disallow quite a lot of valid structures, like phosphate, sulphate etc., but it is simple and reliable.
- Stability=chains requires an atom be no more than CHAIN atoms away from at least one carbon. See the notes on the atom definition file for description of CHAIN. This will disallow a few genuine structures, like phosphate (which has no carbon in at all) and persulfate and diphosphate. NOTE at the moment, the program only distinguishes between CHAIN=0 and CHAIN<>0. I hope one day to have it count the number of atoms properly, but ... that is not today.
- Stability=no means there is no stability check. Note that this will produce some spectacularly stupid structures, like N1NONN1 or PPPP .

The default is 'stability=no'.

smiles=

- smiles=yes means that the structures are output as a text file of SMILES strings.
- smiles=no means that they are not.

Default is 'smiles=yes'

sdf=

- sdf=yes means that the structures are output as an SDF file.
- sdf=no means that they are not.

Default is 'sdf=no'

core=

This says whether you want to remove duplicate intermediate cores before substituting atoms in.

This is done using the InChI key generator, and is therefore incompatible with any *arom* option other than arom=no. Options are

- core=yes – remove them
- core=no – don't

Default is 'core=yes'.

inchi=

This uses an external code (<http://www.inchi-trust.org/downloads/>) to generate an InChI key, and uses this to de-duplicate the structures. (this is the same InChI-generating code as used in the *core* parameter above). This is much better than the SMILES-based system, but has two drawbacks

1. It is slower
2. It is incompatible with isolated aromatic bonds generated by *arom=yes* or *arom=rings*

So *arom=rings* or *arom=yes* are incompatible with anything that uses the InChI generator (*inchi=yes*, *inchioutput=yes*, *core=yes*). The program will warn you of this, and not check or label structures that the InChI generator has a problem with. Probably. Options are:

- *Inchi=check* . Generate InChI keys and only output structures with unique InChI keys. Will also output InChI keys to the SDF file if one is generated. NB There is nothing to stop you doing *inchi=check* and *core=no*, but this would be a waste of processor time.
- *inchi=no* . Just remove identical SMILES strings from the output, no InChI generation

Default is 'inchi=no'

inchioutput=

This generates InChI codes and InChI keys, and can output them to the SMILES file as a CSV file with the InChI code in column 2 and the key in column 3. If you generate InChIs and are generating an SDF file, it will automatically output InChIs to the SDF.

- *inchioutput=yes* – output inchi codes and keys in csv with SMILES. NB if *inchioutput=yes*, then *smiles=yes*
- *inchioutput=no* – output SMILES only.

The default in *inchioutput=no*

trace=

copies the screen output to a file called *name_trace.txt* (where *name* is the filename you entered, minus the 3-letter extension), and also the start and finish times. Options:

- *trace=yes*
- *trace=no*

this does not add much more information, but it can be useful for seeing just how long it is taking to get a result. The default is *trace=no*

cleanup=

This controls whether the program deletes its rather incontinent output of working files on exit. (The reason for *not* doing so is to track weird or unexpected results) . NB this deletes them at a DOS level, so they are not in your Trash – they are gone for good.

- *cleanup=yes* – deletes working files
- *cleanup=no* – leaves working files

The default is *cleanup=yes*

cumulenes=

This controls whether you can have adjacent double bonds in the 'cores', i.e. structures such as C=C=C.

- cumulenes=yes – allow cumulene structures
- cumulenes=no – do not allow cumulene structures

The default is cumulenes=yes.

Example file sets

I have provided two example file sets for you to see how Combimol work.

Butanes

This is to illustrate the basic operation on an SDF file. The file *butanes.sdf* contains a number of topologies of C_4H_n compounds. Drag and drop this onto the Combimol-B executable, and it will generate all the variants of these that are predicted to be stable under the 'Chains' stability rules, containing C, N, O, S, P and Se. The run outputs just a SMILES file, with InChI codes next to the SMILES. It checks for stability using the rather simplistic 'carbon' method.

The folder contains *combimol-b-config.txt*, *combimol-atoms.txt* and *combimol-bonds.txt* files, and the INCHI-1.exe needed to run the InChI code part of the program. It also contains a spreadsheet version of the bonds file, which is a bit easier to visualize.

Note that the Combimol-bonds.txt file specifies that any atom may join to any other atom. The Combimol-bonds.xls file has a version in which only a subset of bonds are allowed that are more likely to be stable to aqueous hydrolysis. Have a go at converting the Combimol-bonds.xls data into a *combimol-bonds.txt* file, re-running and seeing the difference.

I have blocked cumulene (adjacent double bond) formation in the *combimol-b-config.txt* file. However, note a (deliberate) quirk – I had said that the 'S=O' special 'atom' can support double bonds (this is set in the *combimol-atoms.txt* file), and so this can generate structures containing $C=S=O$, which you may consider unrealistic. However the $O=S=O$ 'atom' is defined as not being able to support double bonds, and so the structure $C=S(=O)=O$ will not be generated.

Amino acids

This folder is to illustrate running Combimol in batch mode. The folder has 19 of the 20 proteinaceous amino acids in it as MOL files (Glycine is not included because you cannot change anything in it and still have it be an α -amino acid), and a file *_amino_acids.bat* that runs each amino acid as a separate file generating variants on the side chain that just contain C, N, O and S atoms. Note the specific *combimol-atoms.txt* and *combimol-bonds.txt* files needed to make an NOS-only substitution. Also note that some of these will produce no variants at all, because the substitution rules forbid it. Output is as a series of SDF files, one for each amino acid, and also a series of text files with SMILES in, again one for each amino acid.