



OpenEye
Scientific Software

FILTER

Release 2.1.1

OpenEye Scientific Software, Inc.

June 08, 2010

CONTENTS

1	Front Matter	1
2	Installation and Platform Notes	3
2.1	Licenses	3
2.2	General Installation	3
3	Filtering Theory	7
3.1	Introduction	7
4	FILTER Application Usage	13
4.1	Command Line Interface	13
5	Filter Preprocessing	17
5.1	Metal Removal	17
5.2	Salt Removal	17
5.3	pKa Normalization	17
5.4	Normalization	18
5.5	Reagent Selection	18
5.6	Type Checking	18
6	Molecular Properties and Predictors	19
6.1	Structural and Chemical Features	19
6.2	Functional Groups	20
6.3	LogP	20
6.4	LogS	21
6.5	Polar Surface Area	21
6.6	Lipinski and Hydrogen-bonds	21
6.7	Aggregators	22
7	Filter Files	23
7.1	Physical Property Limits	23
7.2	Functional Group Rules	29
7.3	New Rules	78
7.4	Selection Statements	78
8	Release Notes	81
8.1	FILTER 2.1.1	81
8.2	FILTER 2.1.0 (<i>May 2010</i>)	81
8.3	FILTER 2.0.3 (<i>November 2009</i>) (Toolkit only)	81
8.4	FILTER 2.0.2 (<i>March 2009</i>)	82

8.5	FILTER 2.0.1 (<i>January 2007</i>)	82
8.6	FILTER 2.0.0 (<i>October 2005</i>)	83
	Bibliography	85
	Index	87

FRONT MATTER

Copyright 1997-2010 OpenEye Scientific Software, Santa Fe, New Mexico. All rights reserved.

All rights reserved. This material contains proprietary information of OpenEye Scientific Software. Use of copyright notice is precautionary only and does not imply publication or disclosure.

The information supplied in this document is believed to be true but no liability is assumed for its use or the infringement of the rights of others resulting from its use. Information in this document is subject to change without notice and does not represent a commitment on the part of OpenEye Scientific Software.

This package is sold/licensed/distributed subject to the condition that it shall not, by way of trade or otherwise, be lent, re-sold, hired out or otherwise circulated without OpenEye Scientific Software's prior consent, in any form of packaging or cover other than that in which it was produced. No part of this manual or accompanying documentation, may be reproduced, stored in a retrieval system on optical or magnetic disk, tape, CD, DVD or other medium, or transmitted in any form or by any means, electronic, mechanical, photocopying recording or otherwise for any purpose other than for the purchaser's personal use without a legal agreement or other written permission granted by OpenEye.

This product should not be used in the planning, construction, maintenance, operation or use of any nuclear facility nor the flight, navigation or communication of aircraft or ground support equipment. OpenEye Scientific software, shall not be liable, in whole or in part, for any claims arising from such use, including death, bankruptcy or outbreak of war.

Windows is a registered trademark of Microsoft Corporation. Apple, OS X, and Macintosh are registered trademarks of Apple Computer, Inc. AIX and IBM are registered trademarks of International Business Machines Corporation. UNIX is a registered trademark of the Open Group. RedHat is a registered trademark of RedHat, Inc. Linux is a registered trademark of Linus Torvalds. SPARC is a registered trademark of SPARC International Inc.

SYBYL is a registered trademark of TRIPOS, Inc. MDL is a registered trademark and ISIS is a trademark of Symyx Technologies, Inc. SMILES, SMARTS, and SMIRKS may be trademarks of Daylight Chemical Information Systems. Macromodel is a trademark of Schrodinger, Inc.

Python is a trademark of the Python Software Foundation. Java is a trademark or registered trademark of Sun Microsystems, Inc. in the U.S. and other countries.

Other products and software packages referenced in this document are trademarks and registered trademarks of their respective vendors or manufacturers.

INSTALLATION AND PLATFORM NOTES

2.1 Licenses

To run FILTER you will need to obtain a license file for FILTER from OpenEye Scientific Software (business@eyesopen.com). The license file should be pointed to by the environment variable **OE_LICENSE**.

On Windows, the environment variables can be set under the system Control Panel.

2.2 General Installation

By default, all OpenEye applications are installed into a single distribution directory tree on the specified machine. The default location for this tree is platform specific and will be detailed below.

The root of the tree (i.e. the `openeye` directory) contains the following subdirectories:

- admin** This directory is intended to contain any administrative scripts and tools associated with the installed applications. Currently, this directory is simply a placeholder on all platforms except for Microsoft Windows, where it contains the uninstaller executables.
- arch** This directory contains the collection of platform specific subdirectories. Each subdirectory contains the actual installed executables and support libraries for the associated platform. In the platform specific subdirectory, there will be a subdirectory for each application and within that will be another subdirectory for each version of that application.
- bin** This directory contains a startup script for each application that has been installed. This script determines at run-time what the current platform is and then calls the appropriate executable in the `arch`. This script enables the easy co-existence of multiple platforms and versions of any OpenEye application in the same distribution tree.
- data** This directory contains all of the associated data for the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.
- docs** This directory contains all of the documentation associated with the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.
- examples** This directory contains all of the examples associated with the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

The startup script discussed in the section on the `bin` directory above will have the same name as the installed executable with which it is associated. When the script is called, it will attempt to determine the current platform and run the appropriate executable if installed. If an appropriate executable cannot be found, the script will report that information as well as a list of the currently installed platforms. The auto-detection can be overridden by setting one of two environment variables:

- **OE_ARCH** can be used to specify a colon separated list of compatible distributions for the current platform such as:

```
redhat-RHEL5-x64:redhat-RHEL4-x64
```

Specification of this environment variable overrides the auto-detection process if it is present. If none of the compatible distributions listed are found, the script will fall back to the auto-detection process.

- **APPNAME_OE_ARCH** can be used to specify a colon separated list of compatible distributions for a specific application (as specified by changing the **APPNAME** text in the environment variable name) just like **OE_ARCH** as detailed above.

Specification of this environment variable overrides the **OE_ARCH** environment variable as well as the auto-detection process. If none of the compatible distributions listed are found, the script will fall back to the **OE_ARCH** list first and then to the auto-detection process.

Specifying this variable provides a simple way to customize the behavior for individual applications on non-standard platforms.

The startup script also supports a few commandline arguments including:

- | | |
|---------------------|--|
| -path | Specifying this argument will output the full path of the executable to be run. The executable will not be started if this argument is present. |
| -print_arch | Specifying this argument will output the details of the current platform as detected by the script as well as which platform-version of the executable is being run. The executable will be started if this argument is present. |
| -use_version | Specifying this argument followed by a specific version number allows the user to control which released version of the executable to run. |

2.2.1 Linux/Unix

Linux/Unix distributions are provided as a gzipped tarball of the distribution tree described above. Installation is performed by untarring the file in the desired location. Multiple distributions can be installed in the same location without any challenge.

To ensure that the installed applications can be called from the command line, be sure to add the full path of the `openeye/bin` subdirectory to the **PATH** environment variable. For instance, if the distribution was installed into `/usr/local/openeye`, the **PATH** environment variable should contain: `/usr/local/openeye/bin`.

2.2.2 Windows

Windows distributions are provided as a standard EXE installer. By default, all OpenEye applications will install into the `C:\OpenEye` directory.

An OpenEye group with an application specific subgroup will be added to the *Start* menu. The application specific subgroup will contain links to the documentation, the uninstaller, as well as to a Windows command shell which has the appropriate **PATH** settings already defined to allow the user to simply type the executable name at the prompt without concern for where the executable is actually installed.

For GUI applications, a link to the application will be created on the desktop as well as in the application specific subgroup of the *Start* menu.

2.2.3 Mac OS X

Mac OS X distributions are provided as a standard *pkg* installer delivered as a *dmg* disk image. By default, all OpenEye applications will install into the `/Applications/OpenEye` directory.

To ensure that the installed applications can be called from the command line in the *Terminal*, be sure to add `/Applications/OpenEye/bin` to the **PATH** environment variable.

For GUI applications, an application bundle which can be clicked on to start, will be present in the `/Applications/OpenEye` directory. This bundle cannot be moved independent of the `OpenEye` directory. For instance, the entire `OpenEye` directory can be moved as one piece, but moving the application bundle or the contents of any of the subdirectories in the `OpenEye` directory may cause the application to not start. However, the bundle can still be dragged into the Dock and run from there without any problem.

FILTERING THEORY

3.1 Introduction

Filtering attempts to eliminate inappropriate or undesirable compounds from a large set before beginning to use them in modelling studies. The goal is to remove all of the compounds that should not be suggested to a medicinal chemist as a potential hit. This exercise is obviously case dependent, depending on ease of the assay, intended target, personal bias of the modeller & medicinal chemist, strengths of the company, etc.

To match this need, FILTER's default filter encapsulates many of the standard filtering principles, such as removal of unstable, reactive, and toxic moieties. In addition, FILTER allows the customization of the filtering criteria to fit specific needs.

The criteria for passing or failing a given molecule fall into three categories.

- Physical properties
 - Molecular weight
 - Topological polar surface area (TPSA)
 - logP
 - Bioavailability
- Atomic and functional group content
 - Absolute and relative content of heteroatoms
 - Limits on a very wide variety of functional groups
- Molecular graph topology
 - Number and size of ring systems
 - Flexibility of the molecule
 - Size and shape of non-ring chains

All of the data FILTER generates in filtering molecules can be written to a tab-separated file for easy import into a spreadsheet. This function allows for combining the values dynamically for a variety of purposes, including, but not limited to, determining which filter values best fit each project's needs.

3.1.1 History

When OpenEye's work on filtering technology began in 2000, it was designed simply to remove compounds with reactive or otherwise undesirable functional groups. Over the years, the understanding of lead-like and drug-like

compound selection has advanced. In addition, with the publication of Lipinski’s “Rule of 5” [Lipinski-1997], more and more pharmacokinetic properties have been pushed earlier into the virtual screening process.

In addition to providing basic functional group selection, the technology is a one-stop database preparation tool aimed at generating databases suitable for high-throughput virtual screening.

- Cheminformatics quality-control
 - Valence-state validation
 - Aromaticity perception
 - Implicit hydrogen perception
 - Bond-order perception
- Database preparation
 - Setting pKa states
 - Applying normalizations (tautomers & dative or hypervalent states)
- Compound selection
 - Physical properties (see *above*)
 - Assay counter-indicators (aggregators and dyes)
 - PK ([Martin-2005], [Veber-2002], [Egan-2000], [Lipinski-1997])

Finally, it should be pointed out that in the virtual screening world, time is of the essence. Algorithms for preliminary database preparation should not take large amounts of time. Because of this, all the calculations included in FILTER are 2D or graph-based algorithms. While this does occasionally limit the technology, it allows for the delivery of a product that is appropriate for the task of virtual-screening database preparation.

3.1.2 The Rant!

Nearly every computational tool used in early drug discovery yields statistically predictive, rather than absolutely definitive results. In nearly every case, prudence demands that one consider the causes of false-positives and false-negatives and make an attempt to optimize the area under the receiver-operator curve (ROC) for the computational tool. However, there are well known methods for improving statistical predictions of this nature that are independent of the absolute false-positive and false-negative rates. These methods include filtering the population to which a test will be applied. By applying a test to smaller populations that only contain molecules appropriate for the specific application at hand, the negative impact of the false-positive rate on the predictive results can be dramatically improved.

A familiar example from the medical world will serve to illustrate this principle. Assume we have a test for the presence of the new *foo virus* which has an exceptional ROC curve with false-positive and false-negative values (1/1,000 and 1/1,000 respectively). Let us assume that the *foo-syndrome*, caused by the *foo virus*, effects 1 person in 20,000. If we gave this test to 100,000 people from the general population, we would expect 5 to actually have the *foo syndrome*. With this test, there is only a 0.05% percent chance that any of them would not be detected (i.e. be a false-negative). However, we would expect there to be 100 false positive test results. Thus of the 105 total positive test results, only 4.8% would actually have the *foo syndrome* (positive predictive value = 4.8%).

Table 3.1: Confusion table for the unfiltered *foo virus* test (prevalence 1 in 20,000)

	Actual Positive	Actual Negative	
Predicted Positive	True Positives = 5	False Positives = 100	Positive Predictive Value = 4.8%
Predicted Negative	False Negatives = 0	True Negatives = 99,895	

Alternatively, we could start by using very simple screening before applying the test. We first eliminate people who do not have any risk factors for contracting the *foo virus*. Next we may eliminate people whose blood is incompatible with

the test for the *foo virus*. Further, we may want to eliminate people who acknowledge that they will refuse treatment for the *foo virus* even if we determine that they do have it. By these admittedly simple screens, we apply the test for the *foo virus* to a much smaller group with a decidedly higher prevalence of the virus. For instance, after the filtering, we may be left with a group of only 1,000 people who have a 1 in 200 chance of having the syndrome. Now, we still have the same 5 people who actually have the disease, but we only expect 1 false positive test. Suddenly, there are 6 total positive tests, and 83% of them actually have the syndrome! This is reflected in a much more reasonable (83%) positive predictive value.

Table 3.2: Confusion table for the filtered foo virus test (prevalence 1 in 200)

	Actual Positive	Actual Negative	
Predicted Positive	True Positives = 5	False Positives = 1	Positive Predictive Value = 83%
Predicted Negative	False Negatives = 0	True Negatives = 994	

Bringing the discussion back to drug design. If we have a ligand-based design tool such as ROCS, we can imagine that the receiver-operator curve may have a false positive rate as low as 1 in 10,000. For this exercise, let's assume no false negatives. When using that to identify 50 inhibitors from a database of 2.5 million available compounds, we'd identify 300 potential inhibitors, and 5 out of every 6 of these would be a false positive (positive predictive value of 17%)! If we first run filter and eliminate 65% of the 2.5 million compounds, this leaves us with 875,000 compounds to push through ROCS. There will be about 88 false positives to go with the 50 true positives and the positive predictive value will increase over two-fold with relatively little work.

Table 3.3: Confusion table for the unfiltered ROCS virtual screen

	Actual Positive	Actual Negative	
Predicted Positive	True Positives = 50	False Positives = 250	Positive Predictive Value = 17%
Predicted Negative	False Negatives = 0	True Negatives = 2,499,700	

Table 3.4: Confusion table for the filtered ROCS virtual screen

	Actual Positive	Actual Negative	
Predicted Positive	True Positives = 50	False Positives = 88	Positive Predictive Value = 36%
Predicted Negative	False Negatives = 0	True Negatives = 874,862	

3.1.3 Filtering Principles

The same principle of increasing positive predictive value by removing obvious true negatives applies to screening for lead candidates, regardless of whether it is virtual screening or high-throughput screening. While both are reasonable screens, each can be plagued by very low positive-predictive values (despite low false-positive rates), particularly when applied to all available compounds, or large virtual libraries. Simple filtering techniques focus the set of compounds passed on to more computationally intensive screening methods.

The first approach to consider is filtering based on *functional groups*. Generally speaking, there are toxic and reactive functional groups that you simply do not want to consider (alkyl-bromides, metals etc). There are also functional groups that are not strictly forbidden, but are not desired in large quantities. For instance, parafluoro-benzene, or trifluoromethyl have specific purposes, but heavily fluorinated molecules can be eliminated.

Beyond simple functional group filtering, you can consider both simple and complex physical properties which can be used to characterize the kinds of compounds you would like to keep and those you would like to eliminate. These properties attempt to consider "drug-likeness", such as bio-availability, solubility, toxicity, and synthetic accessibility even before the primary high-throughput or virtual screening, which primarily are geared toward detecting potency alone. The best known of the physical property filters is *Lipinski's "rule-of-five"*, which focuses on bioavailability [Lipinski-1997]. However, many other physical properties, such as *solubility*, *atomic content*, *ring structures*, and *surface area ratios* can also be considered. FILTER provides algorithms for calculating many of these properties, and applying them with filters based on literature studies.

Finally, you should eliminate the types of compounds that can be troublesome at later stages. For instance, Shoichet's *aggregating compounds* often produce false positives that can waste enormous resources if they were identified by virtual or high-throughput screening [McGovern-2003] [Seidler-2003]. Similarly, *dyes* can appear to be inhibitors by interfering with colorimetric or fluorometric assays or binding non-specifically to the target protein.

3.1.4 Variations of Filters

Different types of filters are appropriate under different circumstances. Very early in a project, when little or no SAR is available, very strict drug-like filters can be applied. This prevents a project team from spending chemistry resources pursuing difficult compounds that may not be modifiable to introduce appropriate properties. However, when considering compounds for purchase for HTS, different filters can be applied. Oprea, et al, pointed out that the best molecules for initial HTS are smaller and less functionalised than drugs, but with some activity [Oprea-2000]. Therefore, strict lead-like filters can be applied to ensure that hits identified from HTS have sufficient "room" for elaboration into (usually larger and more highly functionalised) leads. However, when SAR suggests that particular compounds or series may yield valuable information, filtering criteria can be loosened, because the secondary screens (QSAR models, similarity to known actives) that are being applied are effective in detecting useful compounds. Reflecting back on the medical analogy, this is the case where an improved primary screen with a dramatically improved false-positive rate (say 1 in 100,000) can be safely applied to a larger population without terrible effects on the positive-predictive value.

FILTER provides the following "light" (lenient) and "heavy" (restrictive) filters:

- BlockBuster
- Lead

The BlockBuster filter is based on 141 best-selling, non-antibiotic, prescription drugs. We designed the physical property portion of the filter so that it passes all of the compounds. The physical property values in this filter are quite good. However, the functional group filters in this filter are probably too restrictive because 141 compounds is not sufficient to span all acceptable functionality.

Note: The original [Oprea-2000] Drug filter is provided as well. However, experience has shown it to be too restrictive. The BlockBuster filter was developed in response to complaints about the Drug filter being too restrictive.

Hint: We recommend the BlockBuster filter, the default, for most purposes. If your project is unusual, or you are unsatisfied with the results we recommend you review the *filter file* for your specific filtering needs.

If you decide to modify the *filter file* the depictions found in the *Functional Group Rules* section can be particularly helpful in determining what functional groups are indicated by each name in the file.

3.1.5 Accumulation of Rules

FILTER contains numerous rules that judge the quality of molecules on many different facets. When examined individually, each of these rules seems quite reasonable and even profitable. However, when each molecule is tested against hundreds of individual filters, the fraction of molecules that pass all the filters can be surprisingly small. Sometimes less than 50% of vendor databases pass the filters. If this is unacceptable we recommend you examine the predicted aggregator, solubility, and Veber filters. In our experience, these are the most common failures. The best method of investigating failures is looking at the filter log.

The BlockBuster filter was adjusted to demonstrate this in a tangible way. For each value, rather than spanning the entire range, it's properties were set to cover from the 2.5th percentile to the 97.5th percentile. The differences between the original BlockBuster filter and the adjusted filter are both in reasonable ranges. For instance, the full range of molecular weight for the BlockBuster filter spans 130 to 781, while the 2.5th percentile is 145 and the 97.5th percentile is 570. The remarkable result is that when the reduced filter is used, only 75 of the 141 original molecules pass the filter! This demonstrates how slight changes to many filters can lead to a significant reduction in the number of compounds that pass all of the filters.

Taking the opposite approach of allowing everything to pass can be equally futile. To demonstrate this a filter was designed around the “small molecule drug” file available from the DrugBank [website](#). In order to pass every one of these molecules, including some that would not be acceptable for modern project work, many of the individual filters must be set to unreasonable values. For instance, the molecular weight range is 30 to 1500 and the hetero-atom count range is 1 to 60!

Hint: OpenEye can not magically devine the needs of every project. You should personally inspect the *filter file*. The depictions found in the *Functional Group Rules* section can be particularly helpful in discerning what functional groups are desirable and undesirable.

FILTER APPLICATION USAGE

4.1 Command Line Interface

A description of the command line interface can be obtained by executing `FILTER` with the `--help` option.

```
prompt> filter --help
```

will generate the following output:

```
Help functions:
  filter --help simple      : Get a list of simple parameters
  filter --help all        : Get a complete list of parameters
  filter --help <parameter> : Get detailed help on a parameter
  filter --help html       : Create an html help file for this program
```

4.1.1 Required Parameters

-in

File containing one or more molecular connection tables from which you would like to remove the non-medicinal compounds.

-out

File to fill with the molecules that pass all of the specified filters.

Execute Options

-param

The argument for this flag is the name of a file containing control parameters. The control parameter file acts to either replace or augment the command line interface. All parameters necessary for program execution may be provided in the control parameter file, although any command given explicitly on the command line will supercede options found in the parameter file. `FILTER` generates a new parameter file containing the full set of execution parameters upon every execution. The name of the parameter file written by `FILTER` is created by combining the prefix base name with the `'.parm'` extension.

Optional Parameters

-filter

This optional parameter specifies a filter file to be used in place of the default filter. If only simple additions to

the default filters are desired, please see the *-newrule* parameter. The file format for this file is described in the *Filter Files* chapter. There are two special reserved strings for this parameter. If the *-filter* parameter is “lead”, then the default lead-based filter will be used. If the *-filter* parameter is “drug”, then the default drug-based filter will be used. [default = lead]

-fail

This specifies an optional molecular output file where the molecules that fail to pass the filter will be written. If this parameter is specified, then every molecule from *-in* will either be written to *-out* or to *-fail*. [default = null]

-prefix

For an execution of the FILTER program, three general purpose files are written in addition to the output file specified on the command-line. These files are the “info”, “log” and “param” files. Normally, they all begin with the prefix “filter”. However, this can be overridden with the *-prefix* parameter. This is particularly useful if you want to run multiple FILTER jobs in the same directory without overwriting files.

-info

Normally, FILTER writes an “info” file during the progress of any execution. At regular intervals during the execution, the info file is updated to reflect the most recent progress. If you are interested in seeing the progress of a run, it is best to either use the *-dots* parameter or look at the info file. Normally, the info file is saved as *filename.info* where “filename” is the prefix specified by the *-prefix* flag. However, one may use the *-info* flag to specify an info file with a separate name. [default = null]

-newrule

This optional parameter can specify a file that contains filter rules to supplement the default filter or the filter specified with the *-filter* parameter. This parameter can be used to extend the functional group list used to filter. New filter rules can also be added directly to the filter file specified with *-filter*. [default = null]

-typecheck

This boolean flag controls whether the valence states of atoms will be checked. This check identifies molecules that are poorly specified, or represent nonsensical chemical states. For example, an oxygen with eight hydrogens attached or a carbon with a +9 formal charge would be rejected. [default = true]

-select

This parameter is a SMARTS string that allows a user to require a specific functional group or substructure be present in all molecules that pass the filter. This feature is particularly useful for identification of reagents for library design. Selection items can also be added directly to the filter file specified with *-filter*. The command-line argument only allows specification of the SMARTS pattern, and exactly one copy of that functional group is required. If a user wants to specify a selection SMARTS with minimum and maximum number of occurrences other than 1, then they can use a SELECT statement inside the filter file. [default = null]

-log

This flag determines where the logging information is written. The logging information includes a listing of the filter used, followed by a one line comment about why each molecule failed, or if it passed, an assessment of the probability that the compound lies in drug-like space.

-table

This flag specifies a file for a tab-separated format table that includes all of the filter data. These data files are ready for import into a spreadsheet program for easy examination. Each column of the table includes one of the filter categories (such as “Molecular Weight”) and each row of the table corresponds to a single molecule. The table contains complete entries for all of the molecules in the input file regardless of whether they pass or fail. NOTE: Setting this flag will cause the program to slow down. [default = null]

-tableFlag

This flag specifies that, if a table is being written, any values in the table that would cause a molecule to fail a filter will be flagged with an asterix. This provides a means of seeing all the filters a molecule might fail, as the log file typically only provides the first failure. [default = false]

-interval

This is the interval at which data is written to the *filter.info* file. The *filter.info* file contains running totals that

are relevant to a FILTER run. Examining the *filter.info* file is the best means of checking on the progress of a FILTER execution. If this flag is 50, then the *filter.info* file is re-written every 50 molecules. [default = 5000]

-pkanorm

This boolean flag determines whether compounds will be modified to reflect a pH=7.4 model. Notice, this will modify the molecules permanently. [default = true]

-normalize

This flag indicates an optional SMIRKS file. This file should contain the set of reactions you wish to use to normalize the connection table of your molecules. Please note: These reactions are applied before the filtering process and can significantly slow the filtering process. [default = null]

-salt

This flag specifies a molecule file that you consider to be salts. If any molecular entries contain multiple disconnected fragments, then any fragment contained in the “salt” file will be removed. If no file is specified, or if there are multiple disconnected fragments in a molecule record that are not in the salt file, then the first largest remaining fragment will be retained and all others discarded. [default = null]

-sdtag

This boolean flag indicates whether you want the molecular properties used for the filtering run (see *-filter*) to be attached to output molecules as SD tag data. This parameter will only work for *.sdf* or *.oeb* formats. [default = false]

-dots

Boolean flag that determines whether FILTER writes a single dot (.) to the terminal (stdout) for every 500 compounds that are processed.

4.1.2 Example Executions

This section has a series of example FILTER command-line executions. Each example is followed by a brief description of its behavior.

```
prompt> filter drugs.smi drugs.oeb.gz
prompt> filter -in drugs.smi -out drugs.oeb.gz
```

These two commands will yield identical results. These execute FILTER with the default parameters. The file *drugs.smi* is opened in SMILES format for input, and the output is written to the file *drugs.oeb.gz* in gzipped OE-Binary version 2 format.

```
prompt> filter -in drugs.smi -out drugs.sdf -filter myfilter
```

This command is the same as the one above except for the *-filter* flag. It executes FILTER with the parameters found in the *myfilter* file. The file *drugs.smi* is opened in SMILES format for input, and the output is written to the file *drugs.oeb.gz* in gzipped OE-Binary version 2 format.

```
prompt> filter -param myparameters drugs.smi drugs.oeb.gz
prompt> filter drugs.smi drugs.oeb.gz -param myparameters
```

The first of these two commands will yield exactly the same results as the example above. The file *drugs.smi* will be mapped to the *-in* flag and *drugs.oeb.gz* will be mapped to the *-out* flag being the second to last and last command-line arguments respectively. Unfortunately, the second of these two commands, will fail to parse because the implicit input and output arguments are not the final two arguments in the list.

```
prompt> filter -in drugs.smi -out drugs.oeb.gz -table drugs.table
```

This executes FILTER on the file *drugs.smi* and writes molecules that pass the filter to the file *drugs.oeb.gz*. It also writes the all of the filter data to the tab-separated value file *drugs.table*.

```
prompt> cat maybridge.05-1.sdf |filter -in .sdf -out .ism|omega .ism m.oeb.gz
```

This command presumes that you have an SD format file called *maybridge.05-1.sdf*. That file is catenated and piped to the FILTER program. The *-in .sdf* flag indicates that FILTER should read .sdf format from std:in. Since no *-filter* flag is specified, the default filter will be used. The *-out .ism* flag indicates FILTER will write isomeric smiles format to std:out. The output would then be piped into OMEGA.

```
prompt> filter -in drugs.smi -out drugs.oeb.gz -select "[N;$(*-a)]"
```

This command will filter the compounds in *drugs.smi* with the default filter and write the output to *drugs.oeb.gz*. It also requires that molecules contain exactly one instance of the aniline substructure defined by the SMARTS pattern “[N;\$(*-a)]”.

FILTER PREPROCESSING

Before the applying any of the *molecular property filters* a preprocessing step occurs that can alter the molecule significantly to fit the criteria needed for most modeling applications. This filtering preprocessing step is a precisely defined series of stages that occur on the molecule in the following order:

1. *Metal Removal*
2. *Salt Removal*
3. *pKa Normalization*
4. *Normalization*
5. *Reagent Selection*
6. *Type Checking*

5.1 Metal Removal

Metal removal is the first stage of elemental based filtering. This stage will remove specified metal complexes from the molecule. It will not reject a molecule for having a metal complex. This allows the filter to treat atoms in the counter-ion portion of a molecule separately from the atoms in the primary portion of the molecular record.

For instance, this allows organic molecules that are complexed with silver to be eliminated based on their metal chelate even though they themselves are acceptable while at the same time eliminating a sulphate counter-ion from another molecule before it leads to elimination of the acceptable cationic molecule.

See Also:

Elemental Filters

5.2 Salt Removal

This step deletes all atoms that are not part of the largest connected component of a compound. This effectively eliminates all non-covalently bound portions of the compound.

5.3 pKa Normalization

FILTER has a rule-based system to set the ionization state of input molecules. If pKa normalization is turned on, the molecule is set to its most energetically favorable ionization state for $pH=7.4$. The rule-based nature of this calcu-

lation allows it to be quite fast. Further, despite being rule-based, this approach takes into account many secondary charge interactions.

While more advanced levels of theory can be found for predicting ionization states, this method is very well suited to virtual-screening database preparation. However, for hit-to-lead or lead optimization a higher level of theory is recommended for pKa estimation.

5.4 Normalization

In addition to *pKa normalization*, FILTER allows any number of additional molecular normalizations. Since normalizations are usually specific to a particular company or site, FILTER provides the ability for users to input normalizations, such as the nitro tautomer state, but does not provide default implementations.

5.5 Reagent Selection

Reagent selection for small linear library synthesis or large combinatorial library synthesis is still a necessary task at many pharmaceutical companies. For a user hoping to identify a set of acyl-halide reagents, they can specify a selection parameter to require that each compound have exactly one acyl-halide. In addition they might want to modify the filter to exclude functional groups (such as primary amines) that may be acceptable for typical lead-like molecules, but are not acceptable for the specific reagent the user has in mind.

Therefore, the selection parameter is the reverse of a filtering parameter. The molecule must *include* the given substructure in order to pass the filter.

See Also:

The *select parameter* in filter files.

5.6 Type Checking

This checks the valence state and formal charge of the entire molecule. The check identifies molecules that are poorly specified, or represent nonsensical chemical states, often from corrupt input data. For example, an oxygen with eight hydrogens attached or a carbon with a +9 formal charge would be rejected.

MOLECULAR PROPERTIES AND PREDICTORS

FILTER provides a range of properties to predictors to be used as molecular filters. Molecular properties are distinct physical properties that can be measured such as the following:

- *Structural and Chemical Features*
- *Functional Groups*

There have been several attempts at developing fast-approximate QSAR models for bioavailability that have been published. The first of these was *Lipinski's work* ([Lipinski-1997]) and it has been followed by work at Pharmacopia [Egan-2000], Abbott [Martin-2005], and GSK [Veber-2002]. The simplest and probably most trusted is the work of *LogP* and *PSA* used by Egan. The most recent work by Martin, the Abbott Bioavailability Score (ABS) appears to be a refinement of the first generation models and is designed specifically to categorize a molecule's probability of having a bioavailability > 10% in rats.

See Also:

The *Pharmacokinetic Predictors* section in the Filter Files chapter.

6.1 Structural and Chemical Features

There are a number of important structural and chemical features of molecules that it is desirable to limit for virtual screening. The following simple measures are provided as filterable properties:

- Molecular weight
- Ring count
- Ring-system size
- Size of non-ring structures
- Length of unbranched chains
- Hetero-atom fraction
- Halide fraction
- Formal charges
- Rotatable bonds

It also includes slightly more complex algorithms for hydrogen-bond donors and acceptors as well as chiral centers.

See Also:

The *Basic Properties* section in the Filter Files chapter.

6.2 Functional Groups

Functional group removal remains at the heart of the filter algorithm. Functional groups fall into several categories including:

- Reactive or labile groups
- Undesirable groups
- Generally acceptable groups
- Protecting groups
- User-derived groups

While reasonable defaults are provided for each functional group in all of the above categories, it is unusual to find an experienced computational or medicinal chemist who agrees with all of the default values. Examining the filter file at least once is strongly encouraged. Developing a custom filter to fit a desired design is also strongly encouraged. The filter files were designed as basic guides.

See Also:

The filter files contain only functional group names. If there is any confusion regarding the definitions, please refer to the *Functional Group Rules* section for complete descriptions.

6.2.1 Dyes

Years ago, many high-throughput assays could be interfered with by colored molecules. While assay technology has continued to advance and often this is not a problem, it is recognized that most molecules that are dyes are not the type of molecules that are commonly carried forward in lead-development projects. Therefore, a pattern-based filter for dye molecules is included. While these patterns occasionally identify molecules that are considered acceptable by some users, in general, they identify molecules that the majority of chemists would rather not see at the top of their virtual-screening hitlists.

6.3 LogP

The XLogP algorithm ([Wang-1997]) is provided because its atom-type contribution allows calculation of the XLogP contribution of any fragment in a molecule and allows minimal corrections in a simple additive form to calculate the LogP of any molecule made from combinations of fragments. Further, although the method contains many many free parameters, its simple linear form allows for ready interpretation of the model and most of the parameters in the model make rational sense.

Unfortunately, the original algorithm is difficult to implement as published. First, the internal-hydrogen bond term was calculated using a single 3D conformation. It was found that this was both arbitrary and unnecessary. This arbitrary 3D calculation has been replaced with a 2D approach to recognize common internal-hydrogen bonds. In tests, this 2D method worked comparably to the published 3D algorithm. Next, the training set had a few subtle atom-type inconsistencies.

Both of these problems were corrected and refit to the original XLogP training data. This implementation gives results that are quite similar to the original XLogP algorithm, so it is called OEXLogP to distinguish it from the original method.

See Also:

The *XLogP parameter* in the Filter Files chapter.

6.4 LogS

The work of [Yalkowsky-1980] at Arizona has resulted in what is now called the “generalized solvation equation.” It states that the solubility of a compound can be broken into two steps, first the melting for the pure solid to pure liquid and second, the phase transfer from pure liquid into water. For many small organic molecules, this second step is somewhat related to LogP.

Because of this relation we choose to explore the use of the XLogP atom-types in solubility prediction. The expectation was that this might provide an approximate though robust and fast method for calculating solubility. We fit the XLogP atom-types to a training set of nearly 1000 public solubilities. From this we derived a linear model for solubility. The model is extremely fast and is useful for classifying compounds as insoluble, poorly soluble, slightly soluble, moderately, soluble or very soluble. The model is notable for the difficulty it has predicting solubilities for compounds with ionizable groups. Further, it is not suitable for the PK predictions that come late in a project. However, it is useful for eliminating compounds with severe solubility problems early in the virtual-screening process.

See Also:

The *solubility parameter* in the Filter Files chapter.

6.5 Polar Surface Area

Topological polar-surface area (TPSA) is based on the algorithm developed by Ertl *et al* [Ertl-2000]. In Ertl’s publication, use of TPSA both with and without accounting for phosphorus and sulfur surface-area is reported. However, evidence shows that in most PK applications one is better off not counting the contributions of phosphorus and sulfur atoms toward the total TPSA for a molecule. This implementation of TPSA allows either inclusion or exclusion of phosphorus and sulfur surface area with the default being to not include it.

See Also:

Including phosphorus and sulfur surface area:

- The `PSA_USE_SandP` parameter in the *filter file*

Warning: TPSA values are mildly sensitive to the protonation state of a molecule. If the `pKaNorm` parameter is false, the TPSA value is calculated using the input structure and if `pKaNorm` is true, the TPSA is calculated using the pKa normalized molecular structure.

See Also:

The *pKa normalization* section of the Filter Preprocessing chapter.

6.6 Lipinski and Hydrogen-bonds

The work of Lipinski ([Lipinski-1997]) introduced the application of simple filter-like rules to roughly predict late-stage PK properties, in particular oral bioavailability. Unfortunately, Lipinski’s “Rule-of-Five” has come into the common vernacular to such a large degree that some of the specific details are often lost in the commotion. There are two critical examples of this. First, Lipinski used “violation of 2 rules” to categorize compounds. In the subsequent analysis, a significant difference in the two populations of molecules was detected, however, little analysis of the importance of a single violation was done. However, many now consider a single violation to be bad and two violations to be worse. At this writing, there is no known evidence to support this. Second, Lipinski used well-codified and well-understood yet imprecise definitions of “hydrogen-bond donors” and “hydrogen-bond acceptors” in his classification

model. While this makes the algorithm quite understandable and easy to implement, it sometimes causes confusion with those who prefer more refined definitions of hydrogen-bond donors and acceptors.

To address the first problem, users are allowed to set the number of Lipinski failures required to reject a molecule. In keeping with the original publication, the default value is 2. To address the second problem, two kinds of hydrogen-bond donors and acceptors are calculated. In the Lipinski calculation, the published definitions are used (donor count is the number nitrogen or oxygen atoms with at least one hydrogen attached and acceptors are the number of nitrogens and oxygens). For calculation of the number of donors and acceptors in the molecule for the sake of chemical properties, a more complex algorithmic approach is taken. This approach identifies the donors and acceptors outlined in the work of Mills and Dean ([MillsDean-1996]) and also in the book by Jeffrey ([Jeffrey-1997]).

See Also:

The *Lipinski violations* and *hydrogen-bond acceptors* sections in the Filter Files chapter.

6.7 Aggregators

The Shoichet lab ([McGovern-2003], [Seidler-2003]) has demonstrated the importance of small-molecule aggregation in medium and high-throughput assays. Because these small-molecule aggregates can sequester some proteins, they give the appearance of being active inhibitors. There are now several hundred published aggregators in addition to a published QSAR model for predicting aggregation propensity. The user has the ability to eliminate any of the known aggregators as well as the ability to eliminate compounds that are predicted to be aggregators using the QSAR model. In OpenEye's experience, the published QSAR model for predicting aggregators is quite aggressive. It occasionally identifies compounds that are known to be genuine small-molecule inhibitors of a specific protein. While in most cases this model can be useful, until more definitive work is published, we feel you should gain some experience with the model and judge its performance for yourself.

More recent work in industry indicates that in some cases, aggregation properties are specific to the particular experimental conditions being used. Thus we recommended caution in the interpretation of these predictions. Nevertheless, aggregation remains an important issue in HTS hit follow-up and, short of experimental validation, this flag may be the best-available.

See Also:

The *aggregators parameter* in the Filter Files chapter.

FILTER FILES

There are two parameter files a user can provide if they would like to override or augment the default parameter sets. We recommend that you do this by modifying the provided default (*filter_example* and *newrule_example*) files rather than starting from scratch.

There are two primary files a user may choose to provide. The first is the “filter file”. It provides acceptable limits for all of the physical properties and functional groups in the default filter. The second is the “newrule file”. If you have a filter you like, but would like to augment it with a set of additional rules, these can be added with a newrule file.

There are four types of statements that can occur in a filter file:

- physical property limits
- rules
- new rules
- selections

The statements should occur one-per-line in the filter file.

Note: If the appropriate line is not in the filter file, or the value is false, the respective measure will not be used in filtering and its value will not be included in any table-based output.

7.1 Physical Property Limits

There are a large number of physical property limits. They occur as three fields on a line. For example:

```
MIN_HETEROATOMS 2 "Minimum number of heteroatoms"
```

The first field is the property keyword, the second field is the value assigned to that keyword, and the third field is a brief informational message. There are a fixed number of physical property keywords. No additional physical property keywords can be added by the user. The current keywords and brief definitions of each are listed below.

Hint: The values listed below are those found in the default *BlockBuster* filter.

7.1.1 Basic Properties

Molecular Weight

Isotopic molecular weight

MIN_MOLWT 130 "Minimum molecular weight"
MAX_MOLWT 781 "Maximum molecular weight"

Heavy Atom Count

Number of non-hydrogen atoms

MIN_NUM_HVY 9 "Minimum number of heavy atoms"
MAX_NUM_HVY 55 "Maximum number of heavy atoms"

Carbon Count

Number of carbons

MIN_CARBONS 3 "Minimum number of carbons"
MAX_CARBONS 41 "Maximum number of carbons"

Hetero-Count

Number of non-carbon and non-hydrogen atoms

MIN_HETEROATOMS 1 "Minimum number of heteroatoms"
MAX_HETEROATOMS 14 "Maximum number of heteroatoms"

Hetero-Atom to Carbon Ratio

Hetero-count/carbon-count

MIN_Het_C_Ratio 0.04 "Minimum heteroatom to carbon ratio"
MAX_Het_C_Ratio 4.0 "Maximum heteroatom to carbon ratio"

Chiral Count

Number of chiral atoms

MIN_CHIRAL_CENTERS 0 "Minimum chiral centers"
MAX_CHIRAL_CENTERS 21 "Maximum chiral centers"

Hydrogen-bond Acceptors

Number of atoms which match any of the following:

- degree 2, aromatic, non-positive nitrogens
- electron rich or negative, valence less than 4, non-aromatic nitrogens
- negatively charged or not electron withdrawn and neutral oxygens
- degree 1, double bonded, electron rich sulfur

MIN_HBOND_ACCEPTORS 0 "Minimum number of hydrogen-bond acceptors"
MAX_HBOND_ACCEPTORS 13 "Maximum number of hydrogen-bond acceptors"

Hydrogen-bond Donors

Number of hydrogen atoms on nitrogen, oxygen, or sulfur atoms

MIN_HBOND_DONORS 0 "Minimum number of hydrogen-bond donors"
MAX_HBOND_DONORS 9 "Maximum number of hydrogen-bond donors"

Lipinski Acceptors

Number of nitrogens or oxygens

MIN_LIPINSKI_ACCEPTORS 1 "Minimum number of oxygen & nitrogen atoms"
MAX_LIPINSKI_ACCEPTORS 14 "Maximum number of oxygen & nitrogen atoms"

Lipinski Donors

Number of hydrogens attached to nitrogen or oxygen

MIN_LIPINSKI_DONORS 0 "Minimum number of hydrogens on O & N atoms"
MAX_LIPINSKI_DONORS 6 "Maximum number of hydrogens on O & N atoms"

Halide Fraction

Percent of molecular weight from halides

MIN_HALIDE_FRACTION 0.0 "Minimum Halide Fraction"
MAX_HALIDE_FRACTION 0.66 "Maximum Halide Fraction"

Formal Count

Number of atoms with a formal charge (excludes dative)

MIN_COUNT_FORMAL_CRG 0 "Minimum number formal charges"
MAX_COUNT_FORMAL_CRG 4 "Maximum number of formal charges"

Formal Sum

Total formal charge

MIN_SUM_FORMAL_CRG -2 "Minimum sum of formal charges"
MAX_SUM_FORMAL_CRG 2 "Maximum sum of formal charges"

Connected Non-Ring

Considers sets of contiguous (bonded) non-ring atoms

```
MIN_CON_NON_RING 0 "Minimum number of connected non-ring atoms"  
MAX_CON_NON_RING 19 "Maximum number of connected non-ring atoms"
```

Unbranched Chains

The size of unbranched non-ring chains

```
MIN_UNBRANCHED 1 "Minimum number of connected unbranched non-ring atoms"  
MAX_UNBRANCHED 13 "Maximum number of connected unbranched non-ring atoms"
```

Total Functional Group Count

Total number of functional groups. Does not count any ring-systems as functional groups. Degree 1 heteroatoms, particularly those with double bonds or dative bonds are considered part of ring systems and do not count as a functional group.

```
MIN_FCNGRP 0 "Minimum number of functional groups"  
MAX_FCNGRP 7 "Maximum number of functional groups"
```

Note: This is different than the functional group rules.

Ring Systems

Number of ring systems (contiguous systems of ring atoms and bonds)

```
MIN_RING_SYS 0 "Minimum number of ring systems"  
MAX_RING_SYS 5 "Maximum number of ring systems"
```

Ring Size

Maximum size of any single ring system

```
MIN_RING_SIZE 0 "Minimum atoms in any ring system"  
MAX_RING_SIZE 20 "Maximum atoms in any ring system"
```

Rotor Count

Number of rotatable bonds. Allows optional adjustment for aliphatic rings following the method of [Oprea-2000].

```
MIN_ROT_BONDS 0 "Minimum number of rotatable bonds"  
MAX_ROT_BONDS 16 "Maximum number of rotatable bonds"  
ADJUST_ROT_FOR_RING true "BOOLEAN for whether to estimate degrees of freedom in rings"
```

Rigid Count

Number of rigid bonds (non-rotatable bonds)

```
MIN_RIGID_BONDS 4 "Minimum number of rigid bonds"  
MAX_RIGID_BONDS 55 "Maximum number of rigid bonds"
```

7.1.2 LogP

The logP calculation is a derivative of the published XLogP algorithm [Wang-1997], but reparameterized without the dependence on 3D coordinates or the SYBYL/Mol2 aromaticity model.

XLogP

Calculated LogP

```
MIN_XLOGP -3.0 "Minimum XLogP"  
MAX_XLOGP 6.85 "Maximum XLogP"
```

7.1.3 Solubility

The solubility predictions are based on using the atom-types from the XLogP algorithm, [Wang-1997], and reparameterizing them based on available solubility data. Rather than a quantitative cutoff, solubility uses categories. The 6 allowable categories are:

1. insoluble
2. poorly
3. moderately
4. soluble
5. very
6. highly

These categories are keywords used in the filter files as follows.

Solubility

Calculated solubility class

```
MIN_SOLUBILITY insoluble "Minimum solubility"
```

7.1.4 Pharmacokinetic Predictors

Several secondary filters that are built upon published combinations of simpler properties are available.

Note: All of these properties are used for filtering in the default filters.

Lipinski Violations

Number of allowable Lipinski violations. A single Lipinski violation is considered acceptable. The published work, [Lipinski-1997], allows compounds to pass with a single violation but not multiple violations.

```
MAX_LIPINSKI 3 "Maximum number of Lipinski violations"
```

See Also:

The *Lipinski theory* section in the Molecular Properties and Predictors chapter.

PSA

Peter Ertl's, [Ertl-2000], topological polar surface area (phosphorus and sulfur area is optional).

```
PSA_USE_SandP false "Count S and P as polar atoms"  
MIN_2D_PSA 0.0 "Minimum 2-Dimensional (SMILES) Polar Surface Area"  
MAX_2D_PSA 205.0 "Maximum 2-Dimensional (SMILES) Polar Surface Area"
```

See Also:

The *PSA theory* section in the Molecular Properties and Predictors chapter.

GSK/Veber

Veber's measure of bioavailability (PSA > 140 or Rotatable bonds >10). [Veber-2002].

```
GSK_VEBER false "PSA>140 or >10 rot bonds"
```

Abbott/Martin

Yvonne Martin's Abbott Bioavailability Score. This is reported as a probability that F>10% in rats. [Martin-2005]

```
MIN_ABS 0.11 "Minimum probability F>10% in rats"
```

Pharmacopia/Egan

Egan egg measure of bioavailability (LogP >5.88 or PSA > 131.6). [Egan-2000]

```
PHARMACOPIA false "LogP > 5.88 or PSA > 131.6"
```

7.1.5 Aggregators

Aggregators are small molecules that can interfere with assay results by sequestering protein in an aggregation of small molecules in solution. They appear to have activity in many assays, but in fact are usually not specific inhibitors of the protein in question. Includes two measures of whether a molecule is one of the aggregators defined by Shoichet et al. [McGovern-2003] [Seidler-2003] The first measure, AGGREGATORS, is whether the molecule is an exact match to one of the approximately 400 published aggregators. The second measure, PRED_AGG, is whether the molecule hits in Shoichets QSAR model for predicting aggregators.

Aggregators

Whether a compound is known or predicted to aggregate in concentrations common in virtual screening.

```
AGGREGATORS true "Eliminate known aggregators"
PRED_AGG false "Eliminate predicted aggregators"
```

7.1.6 Elemental Filters

The elemental filters are applied in this order:

1. Test for the existence of any of the metals in the `ELIMINATE_METALS` filter in the molecule.
2. Remove salts by stripping away all the disconnected components except for the largest.
3. Test to make sure only atoms specified in `ALLOWED_ELEMENTS` filter are in the molecule.

See Also:

- [Metal Removal](#)
- [Salt Removal](#)

The format of the two elemental filter fields is the keyword followed by a comma delimited list of atomic symbols.

Eliminate Metals

Any molecule with the atoms indicated in `ELIMINATE_METALS` fail to pass the filter.

```
ELIMINATE_METALS Sc, Ti, V, Cr, Mn, Fe, Co, Ni, Cu, Zn, Y, Zr, Nb, Mo, Tc, Ru, Rh, Pd, Ag, Cd
```

Allowed Elements

Molecules with atoms other than those specified by `ALLOWED_ELEMENTS` fail to pass the filter.

```
ALLOWED_ELEMENTS H, C, N, O, F, P, S, Cl, Br, I
```

7.2 Functional Group Rules

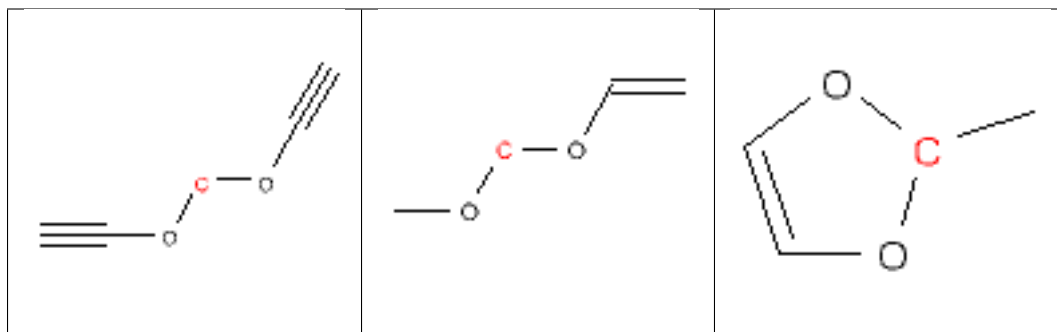
Rules statements set the limits for the maximum number of the specified type of functional group that may be allowed in the molecule.

The first field of a rule statement is the word `RULE` in all capital letters. The second field is a number indicating the maximum number of the group allowed in a molecule. The third field is the functional group keyword. Functional-group keywords are case sensitive.

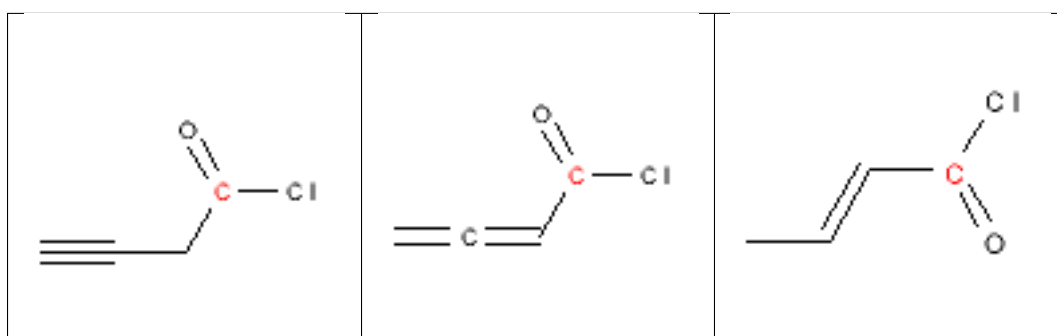
```
RULE 0 acid_halide
```

The following is a list of functional groups which filter recognizes by default. Three example matches are provided with the atoms that correspond to each other highlighted. Due to the highly complex nature of the patterns, in particular recursive SMARTS, it is not possible to fully highlight every atom that was included as part of the match.

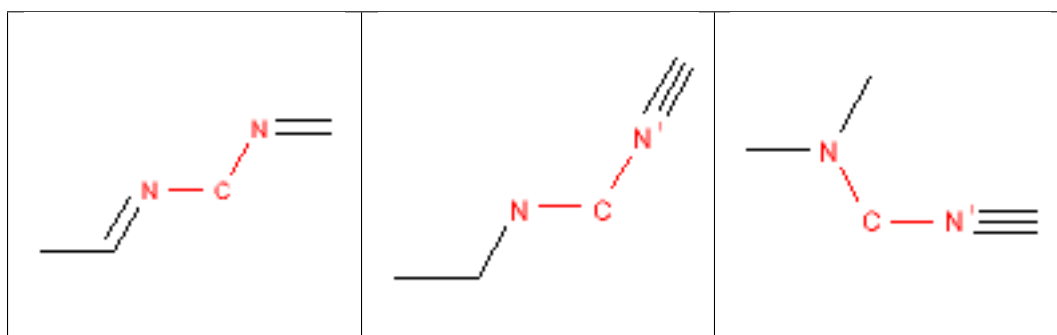
7.2.1 acetal



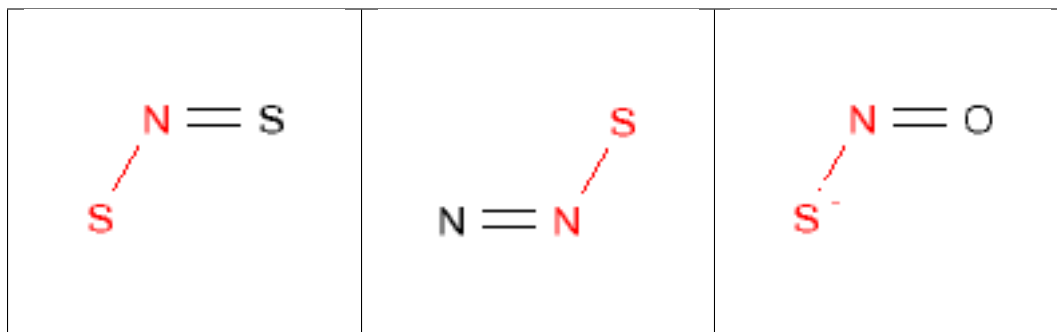
7.2.2 acid_halide



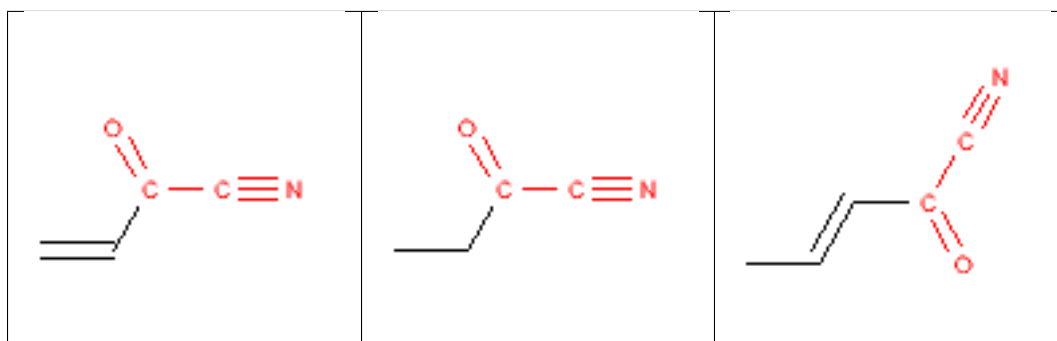
7.2.3 acyclic_NCN



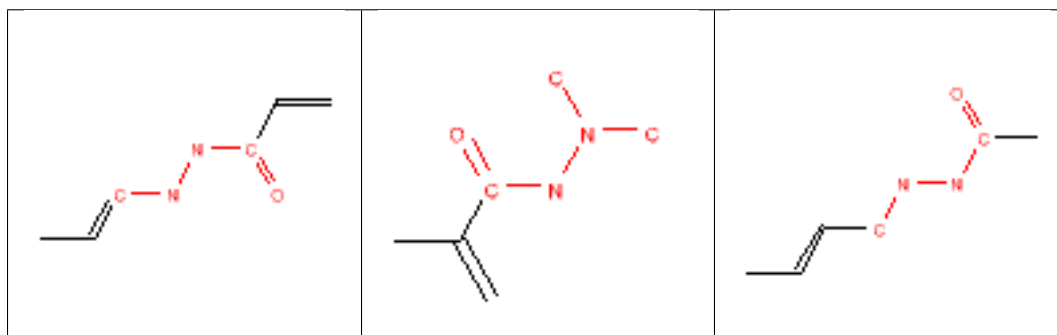
7.2.4 acyclic_NS



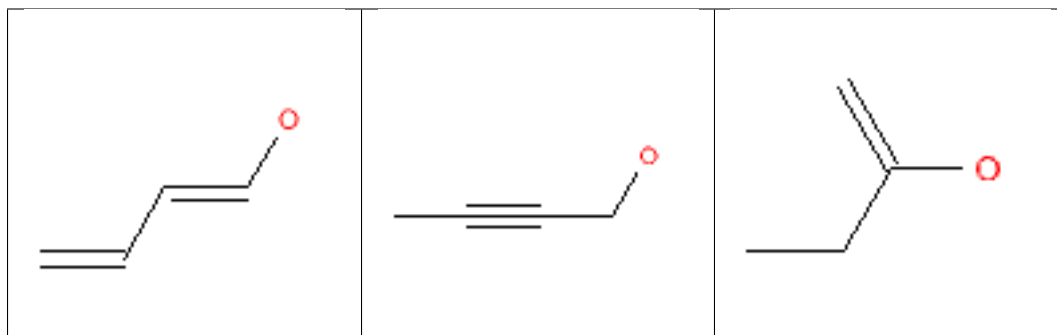
7.2.5 acyl_cyanides



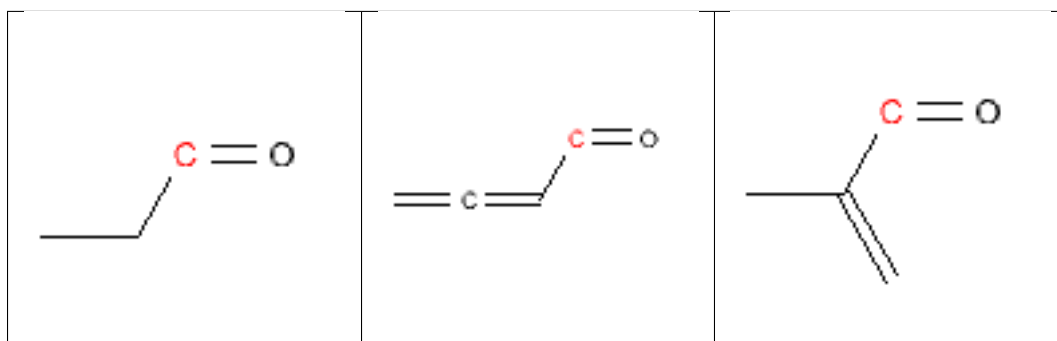
7.2.6 acylhydrazide



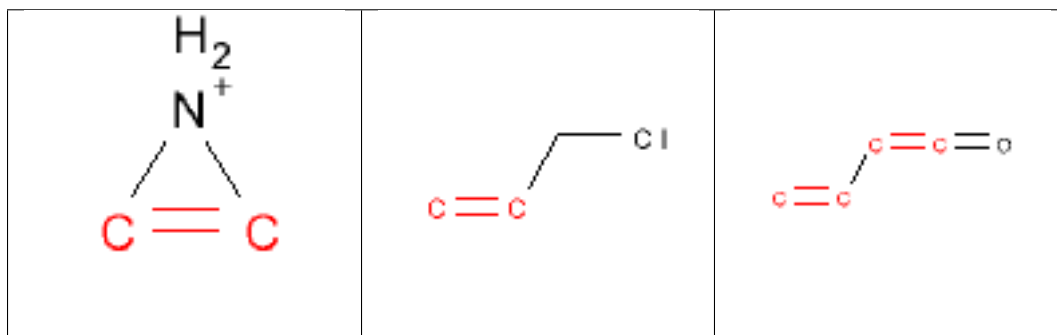
7.2.7 alcohol



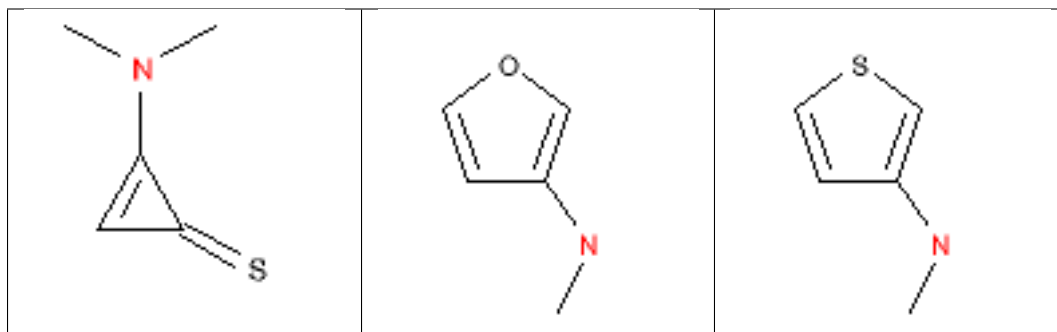
7.2.8 aldehyde



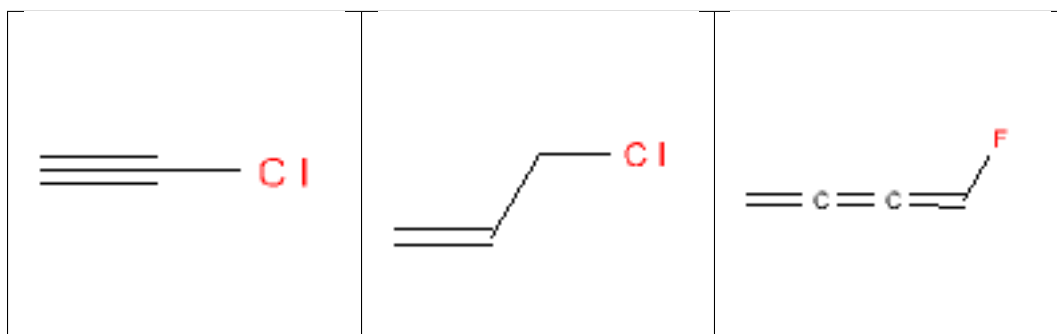
7.2.9 alkene



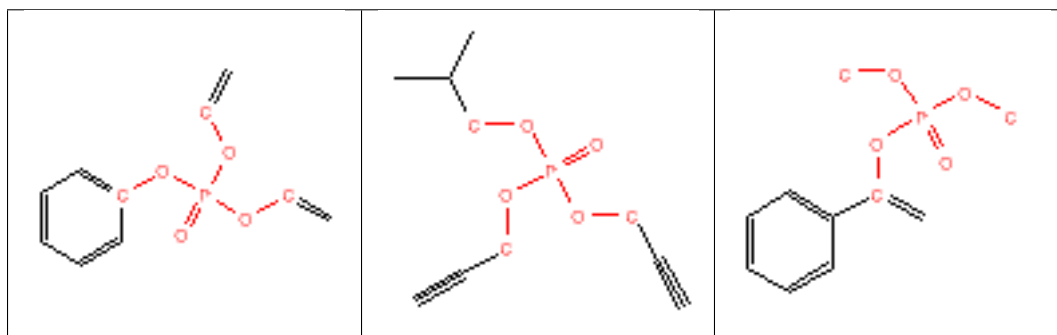
7.2.10 alkylaniline



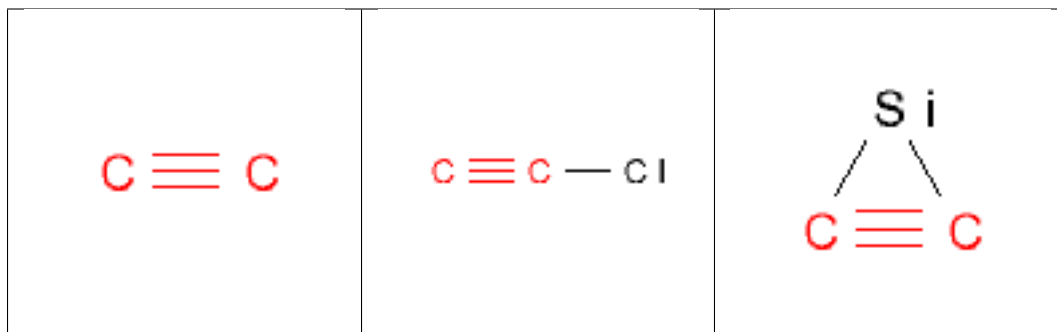
7.2.11 alkyl_halide



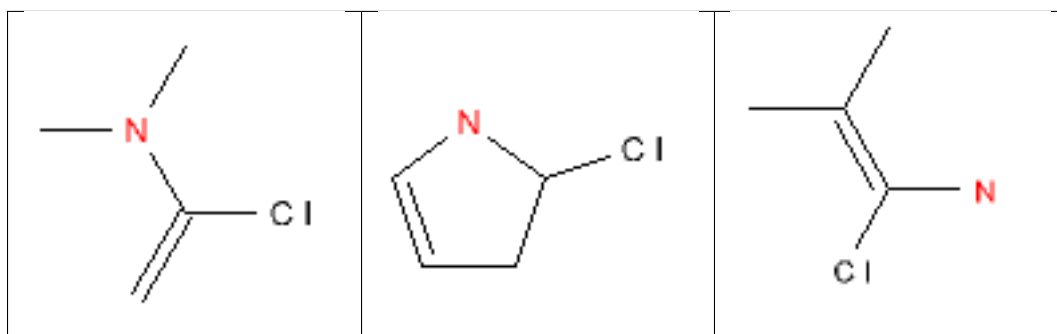
7.2.12 alkyl_phosphate



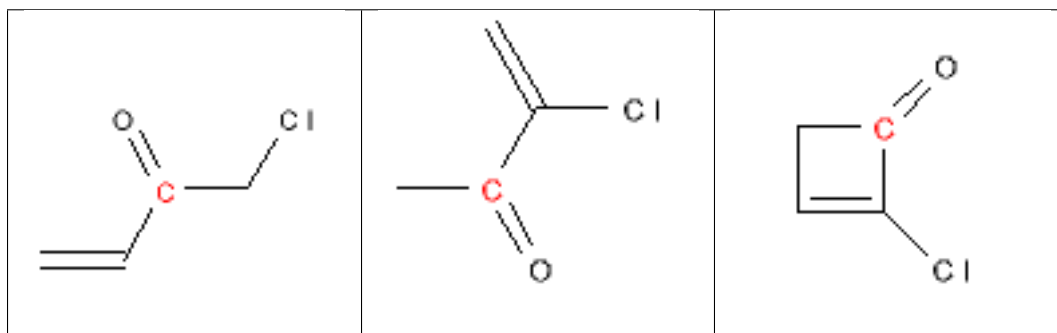
7.2.13 alkyne



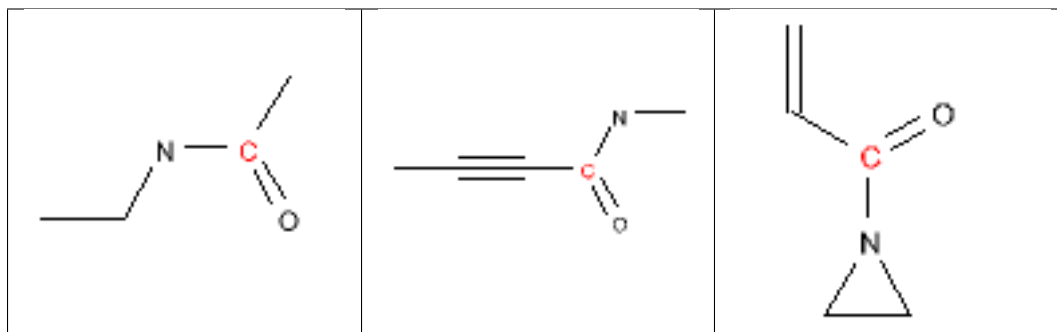
7.2.14 alphahalo_amine



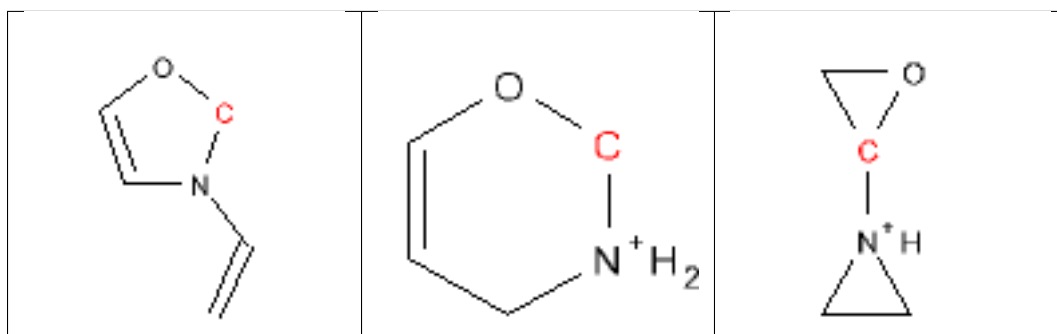
7.2.15 alphahalo_ketone



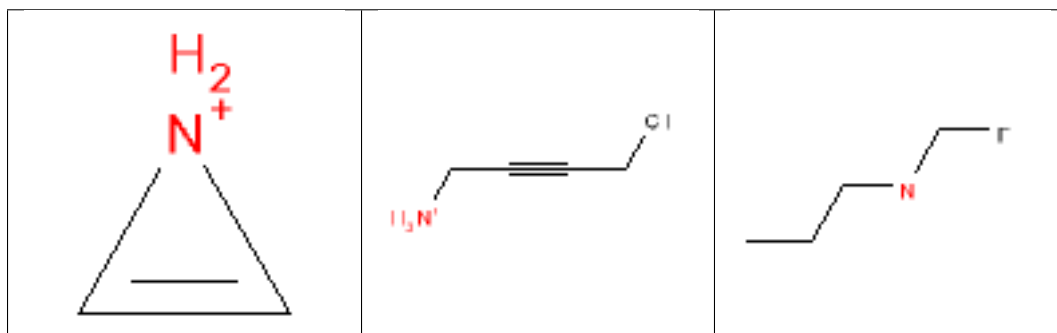
7.2.16 amide



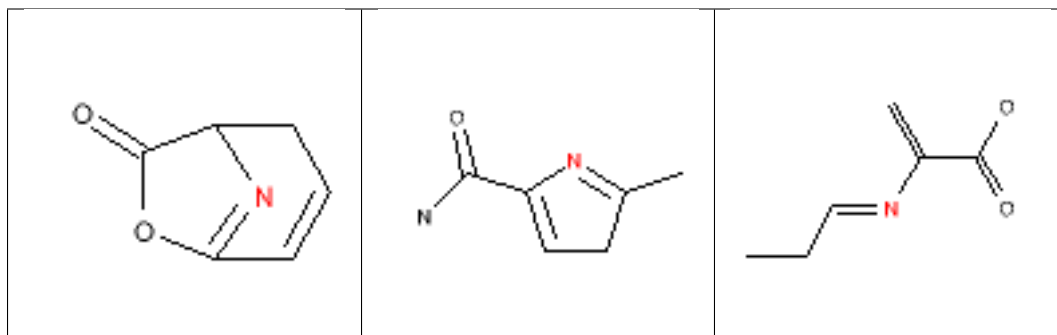
7.2.17 aminal



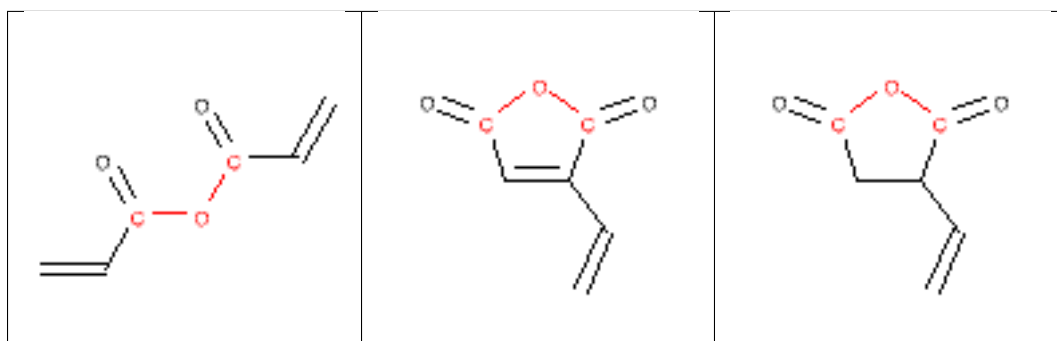
7.2.18 amine



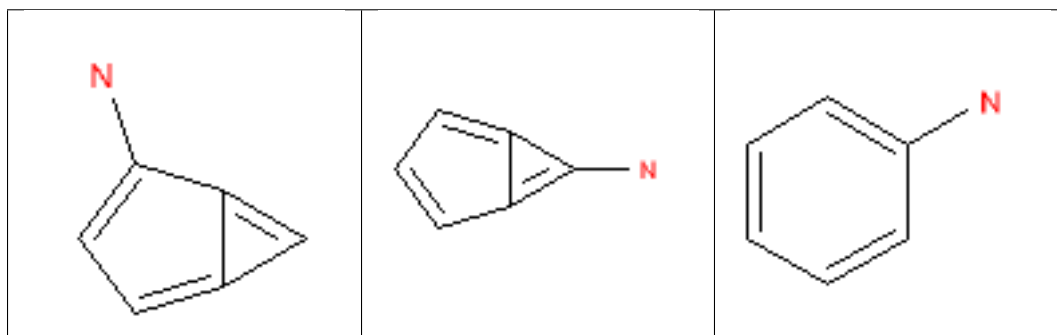
7.2.19 amino_acid



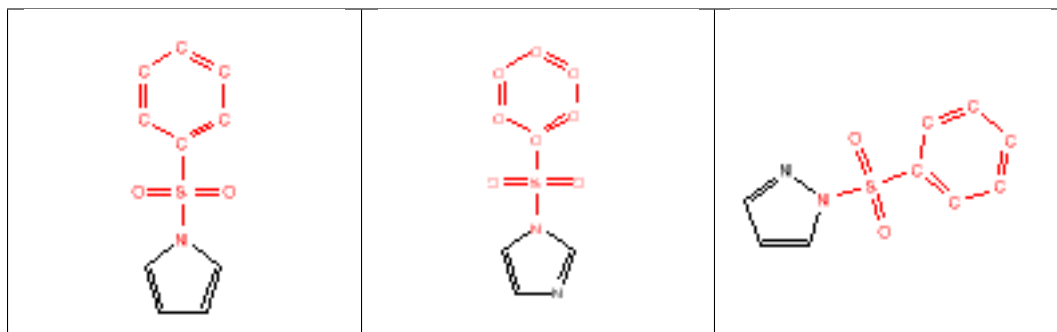
7.2.20 anhydride



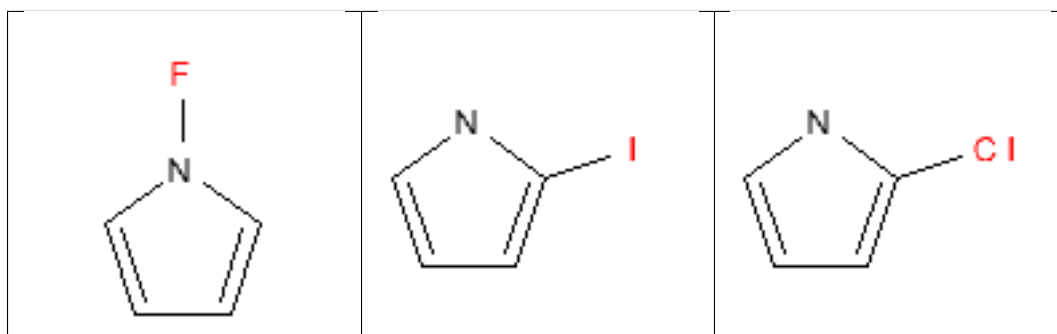
7.2.21 aniline



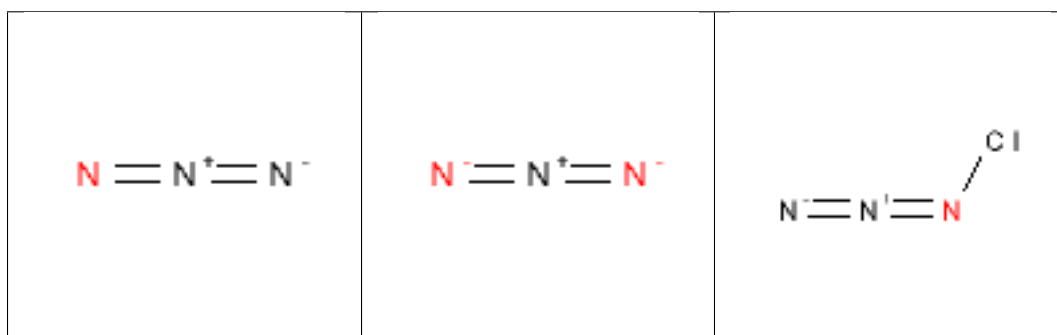
7.2.22 arenesulfonyl



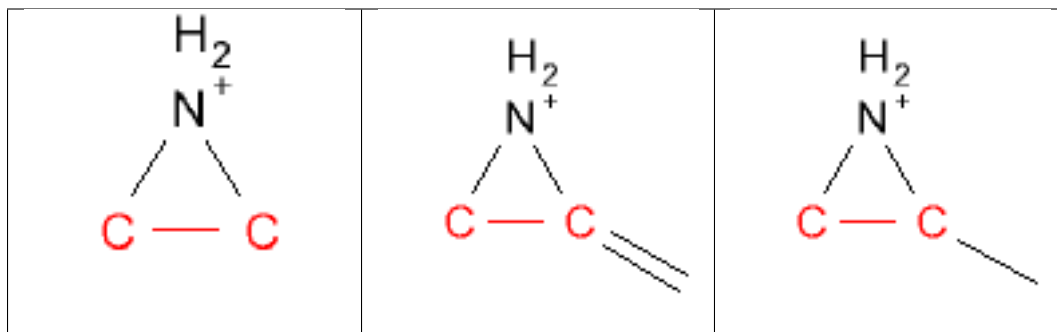
7.2.23 aryl_halide



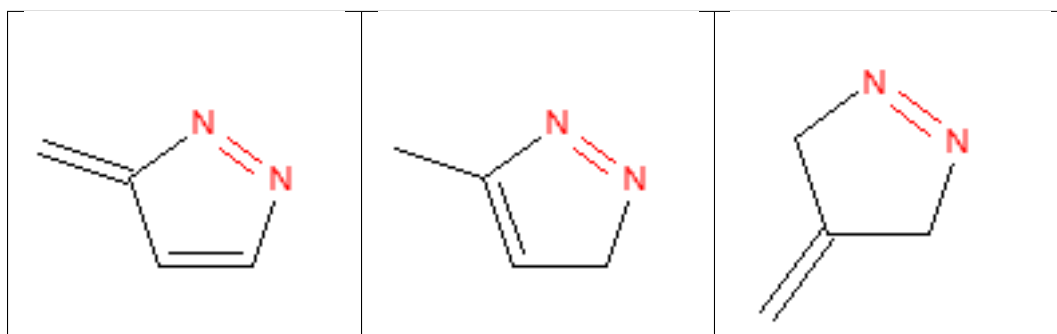
7.2.24 azide



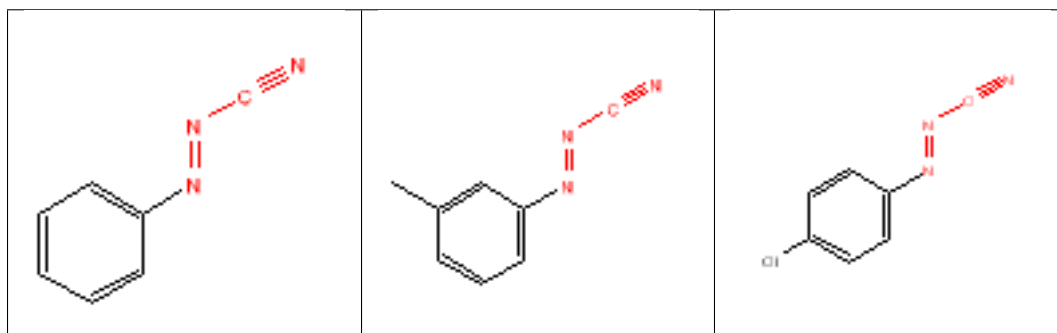
7.2.25 aziridine



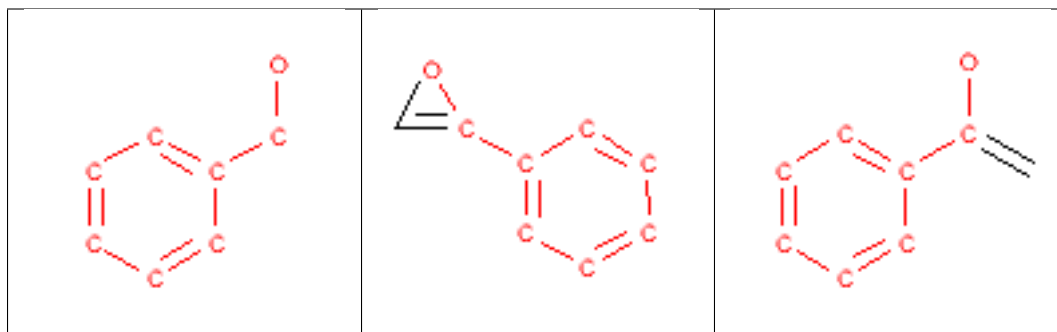
7.2.26 azo



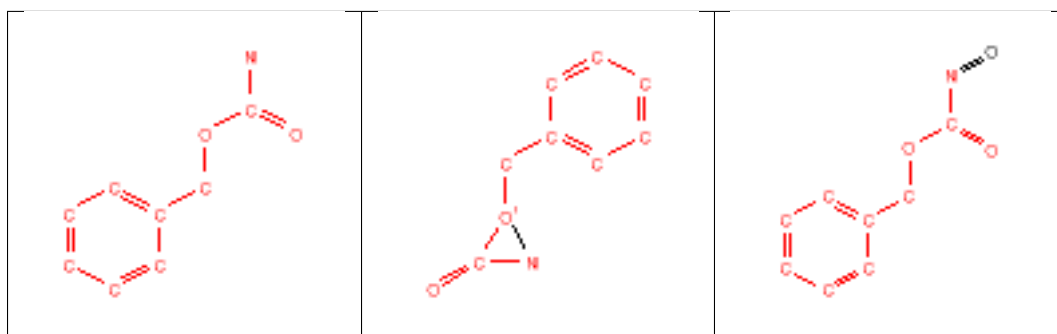
7.2.27 azocyanamides



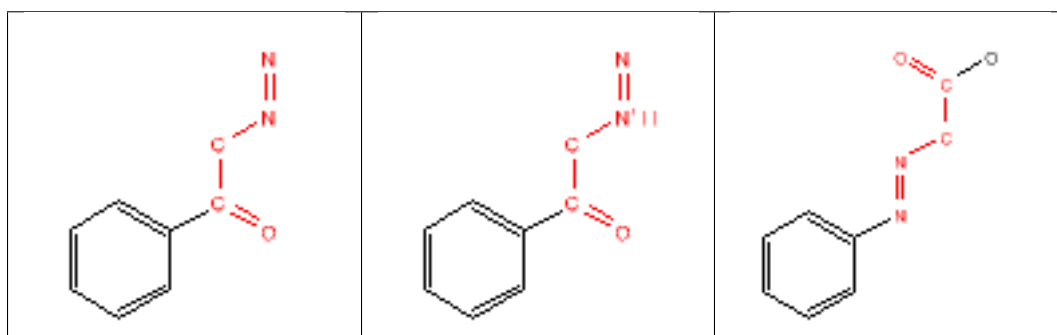
7.2.28 benzyl_ether



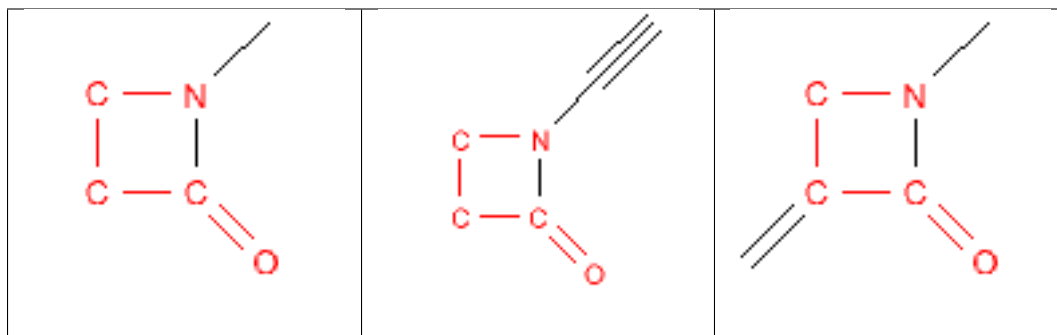
7.2.29 benzyloxycarbonyl_CBZ



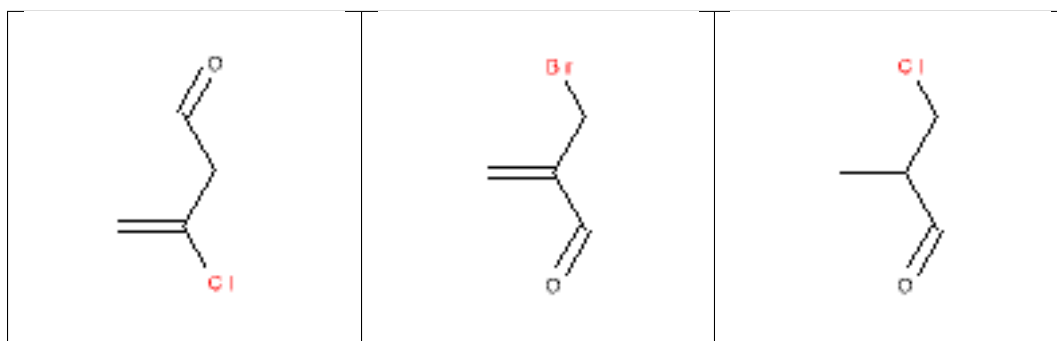
7.2.30 beta_azo_carbonyl



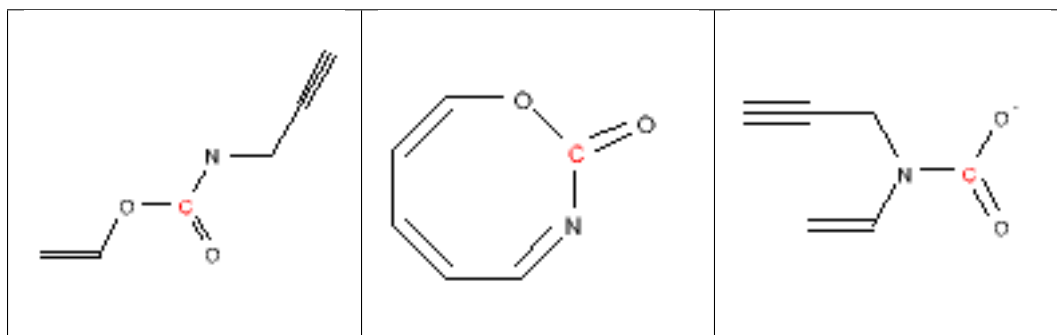
7.2.31 beta_carbonyl_quat_nitrogen



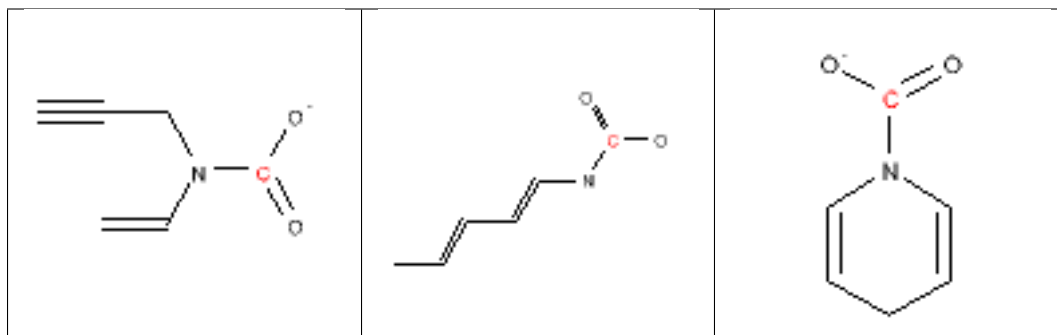
7.2.32 beta_halo_carbonyl



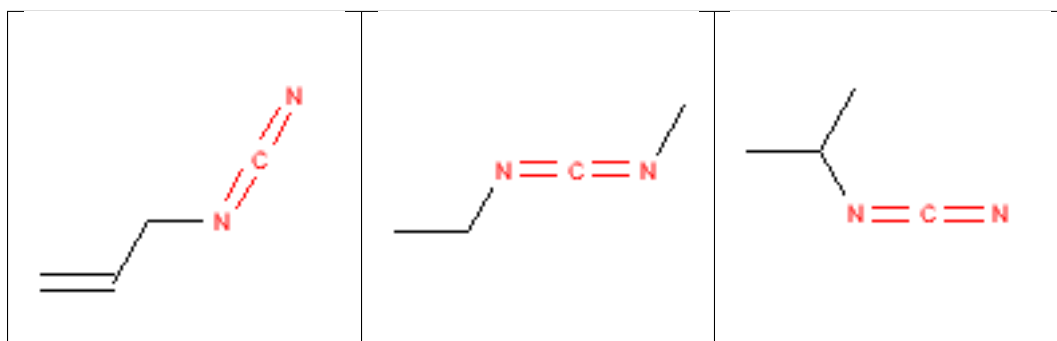
7.2.33 carbamate



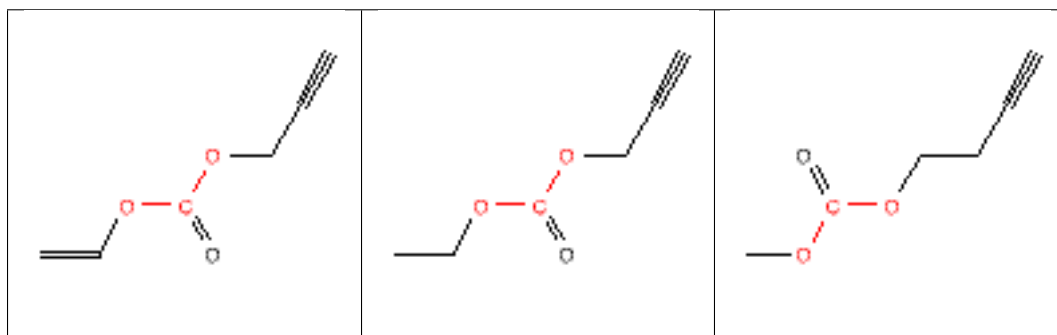
7.2.34 carbamic_acid



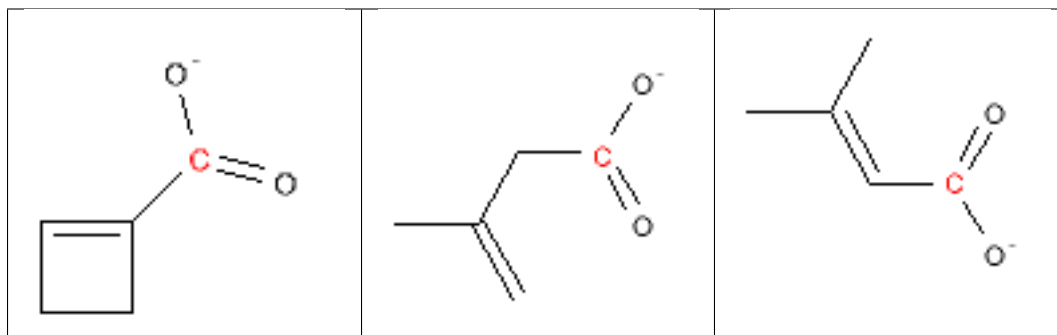
7.2.35 carbodiimide



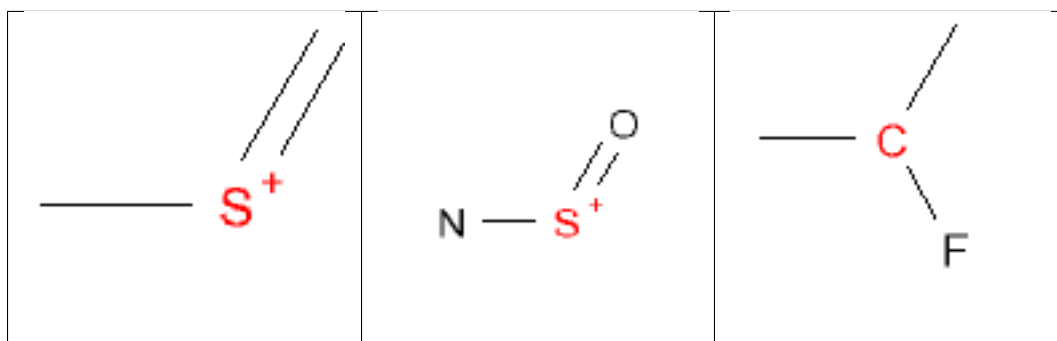
7.2.36 carbonate



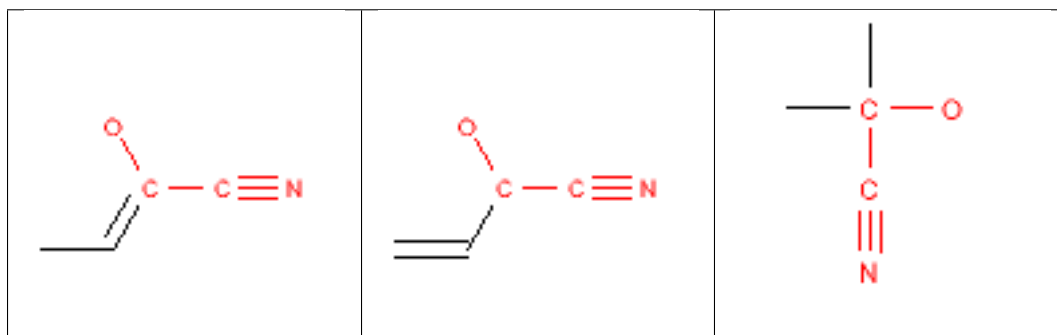
7.2.37 carboxylic_acid



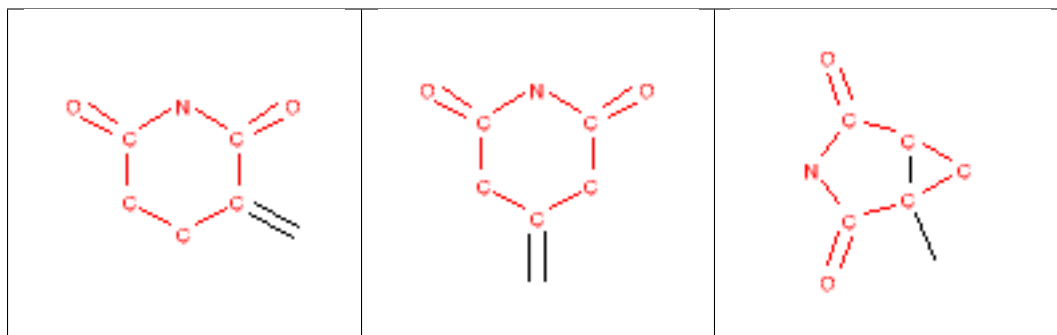
7.2.38 cation_C_Cl_I_P_or_S



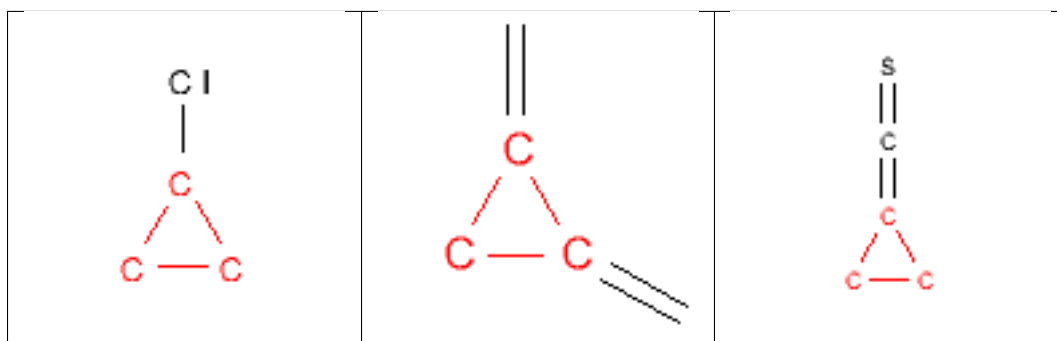
7.2.39 cyanohydrins



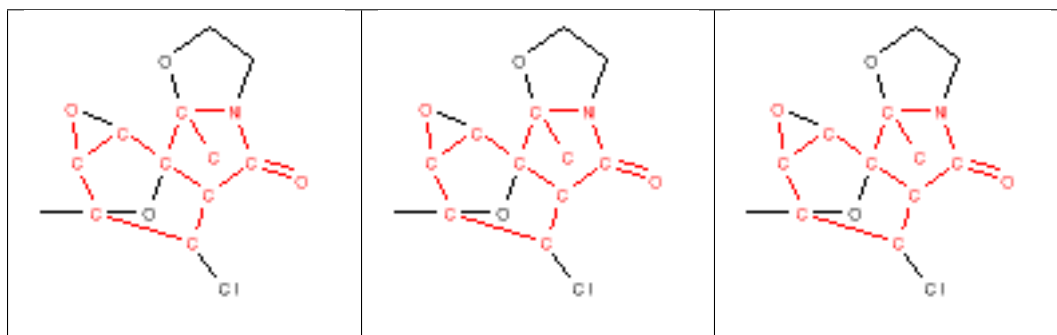
7.2.40 cycloheximide_derivatives



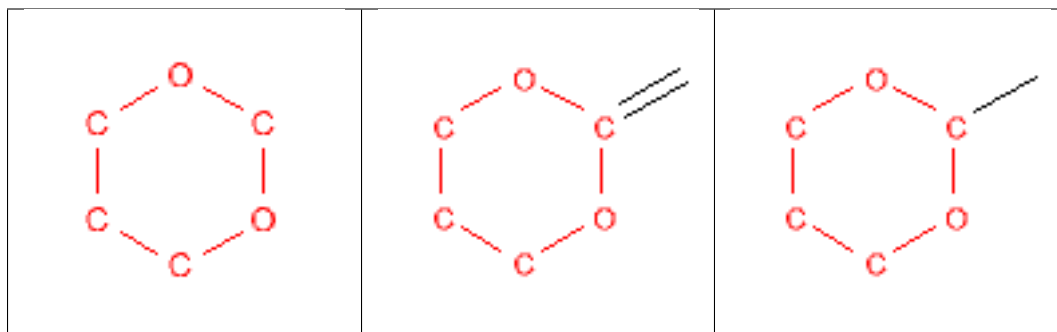
7.2.41 cyclopropyl



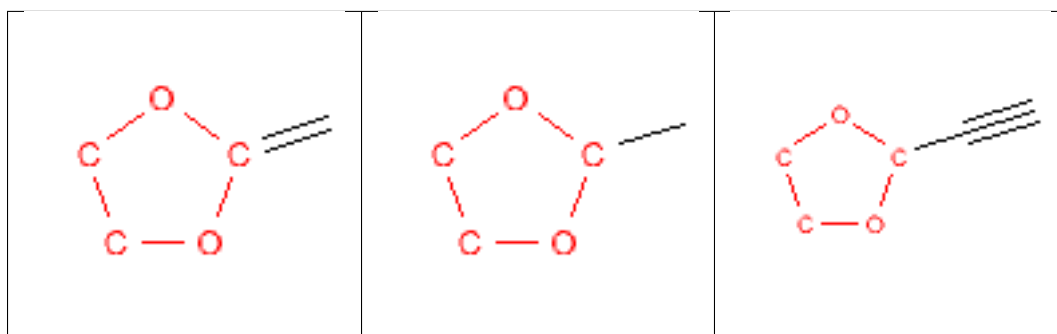
7.2.42 cytochalasin_derivatives



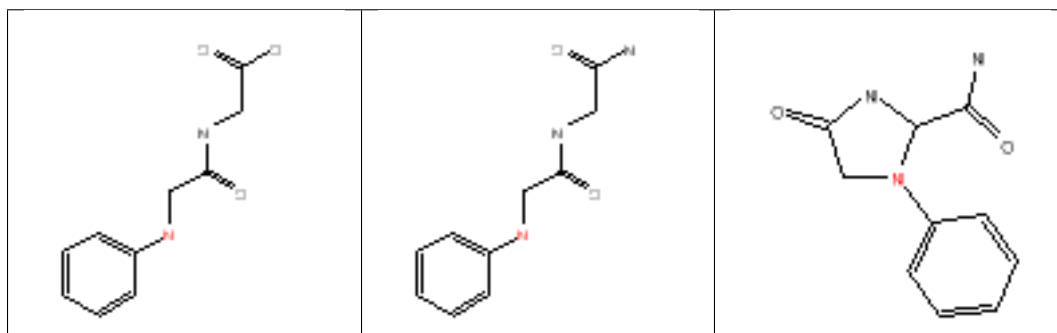
7.2.43 dioxane_6MR



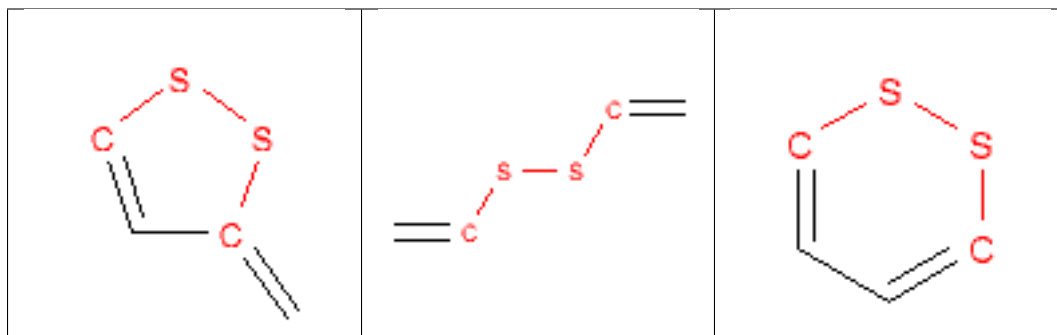
7.2.44 dioxolane_5MR



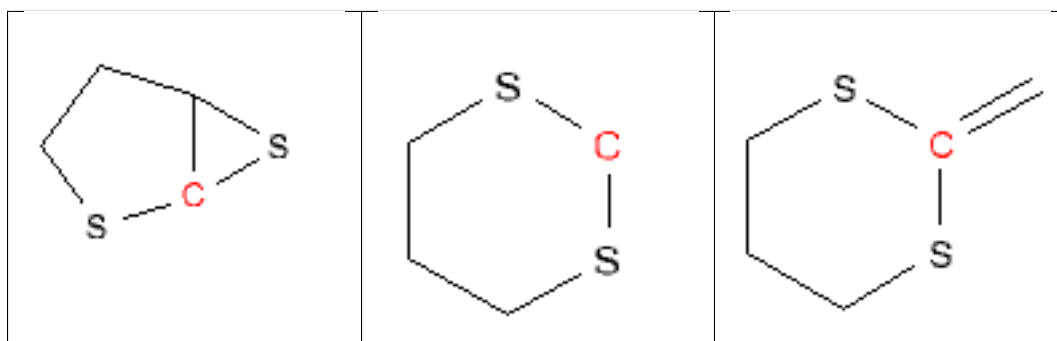
7.2.45 di_peptide



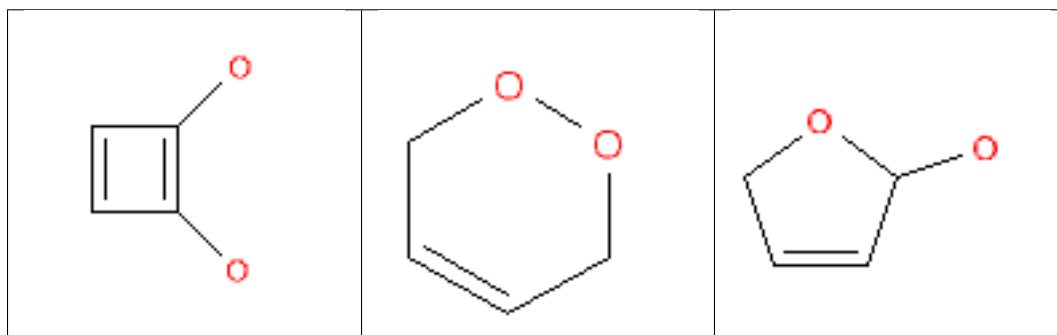
7.2.46 disulfide



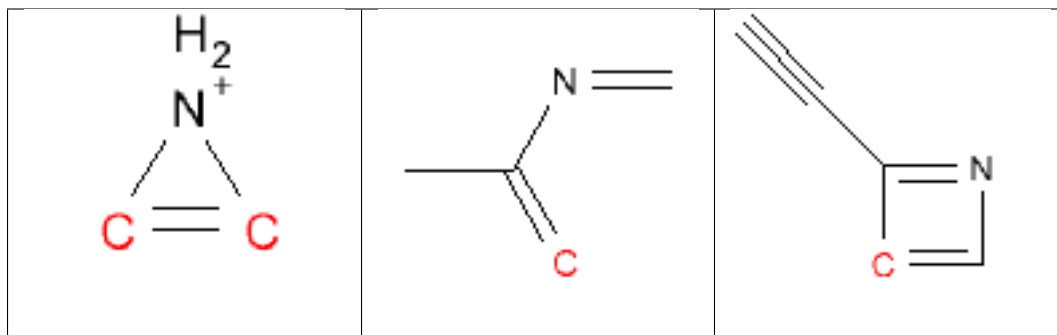
7.2.47 dithioacetal



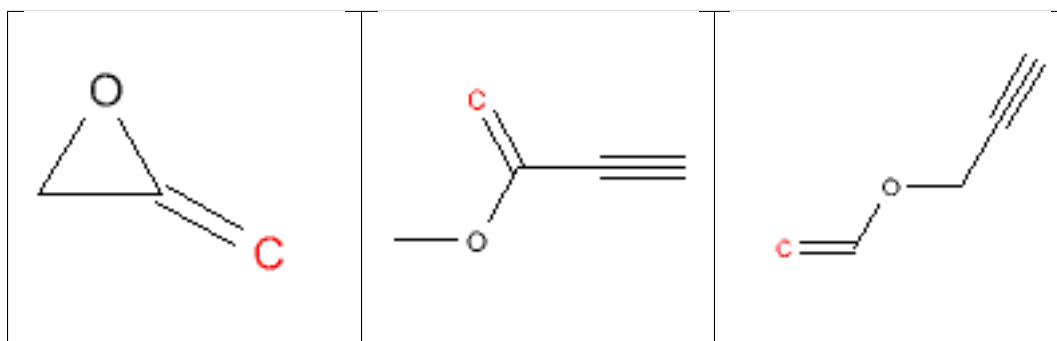
7.2.48 dye



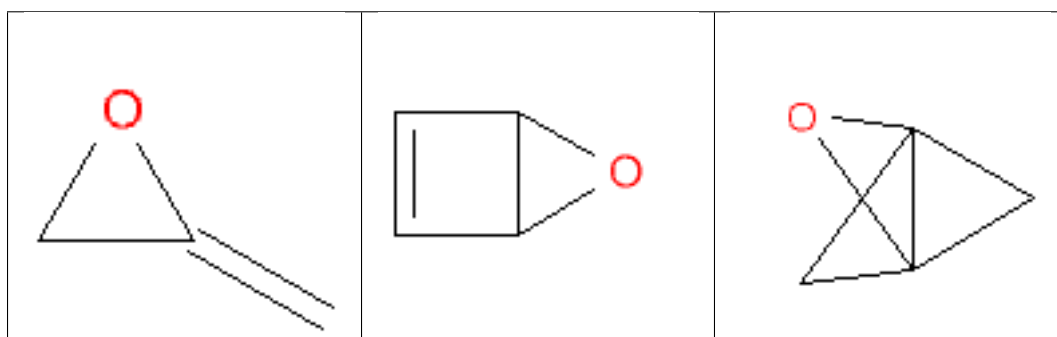
7.2.49 enamine



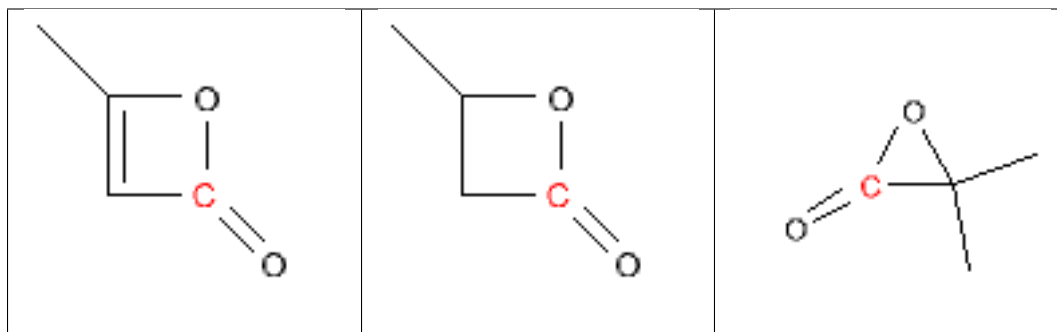
7.2.50 enol_ether



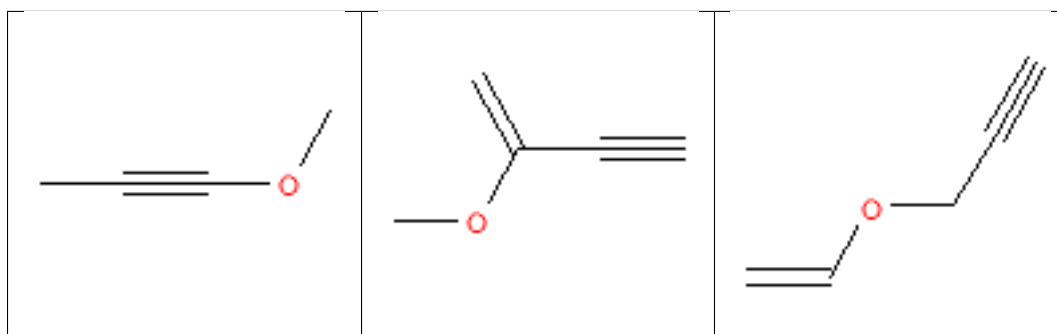
7.2.51 epoxide



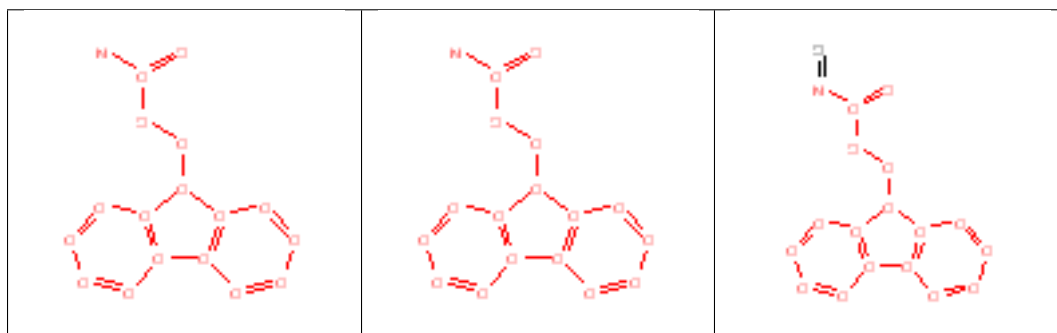
7.2.52 ester



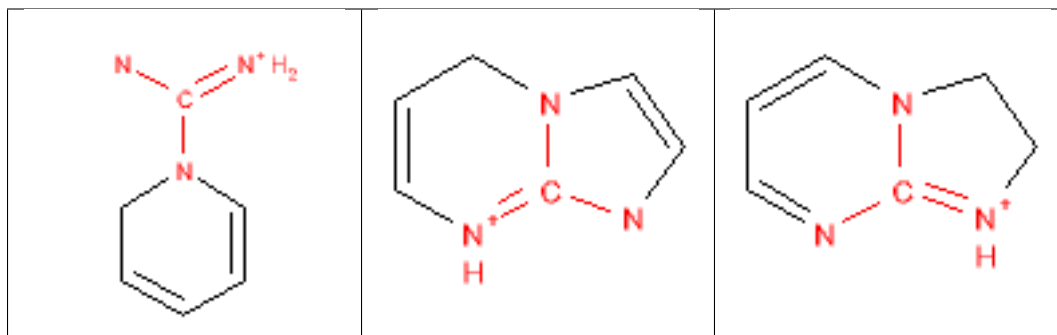
7.2.53 ether



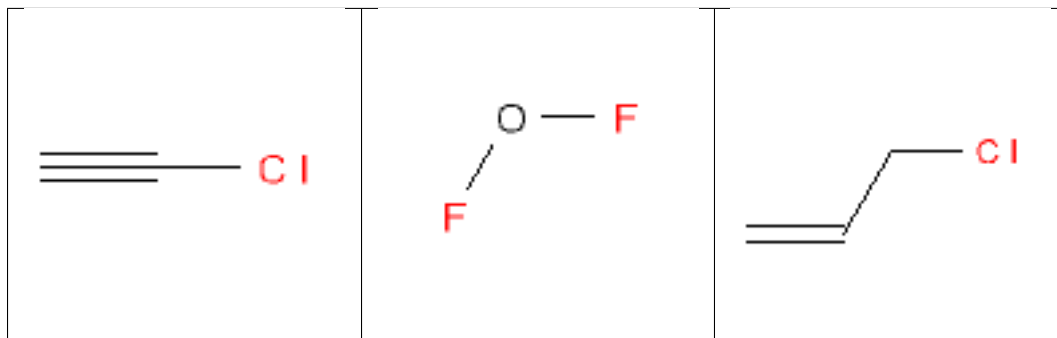
7.2.54 fluorenylmethoxycarbonyl_Fmoc



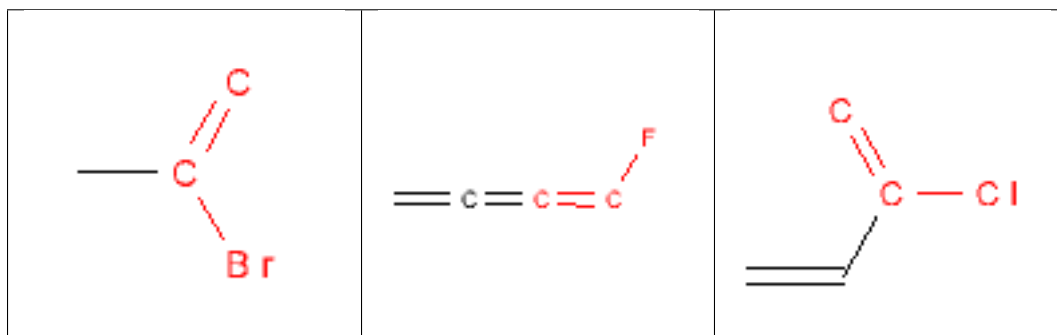
7.2.55 guanidine



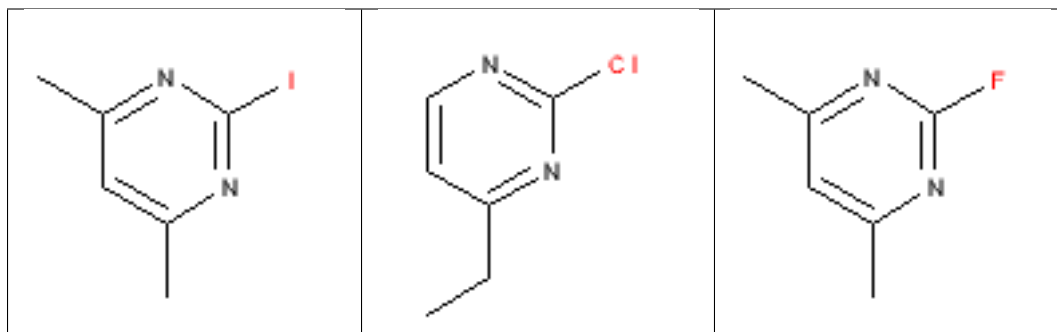
7.2.56 halide



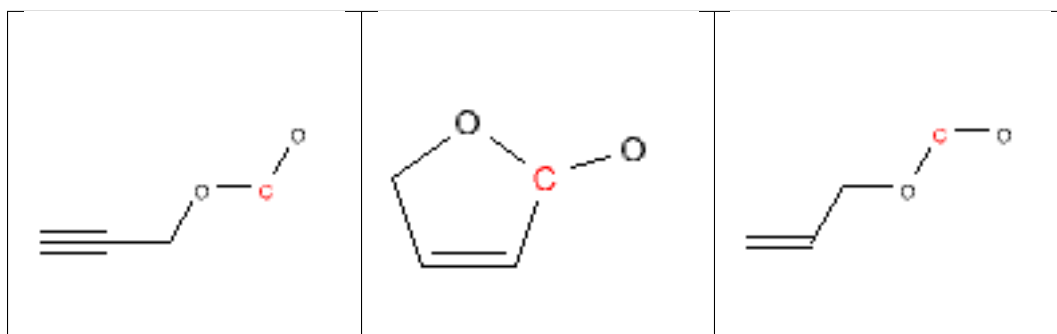
7.2.57 halo_alkene



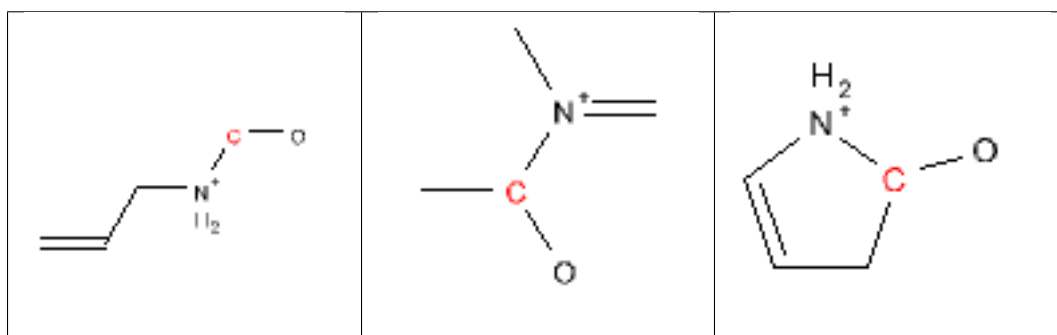
7.2.58 halopyrimidine



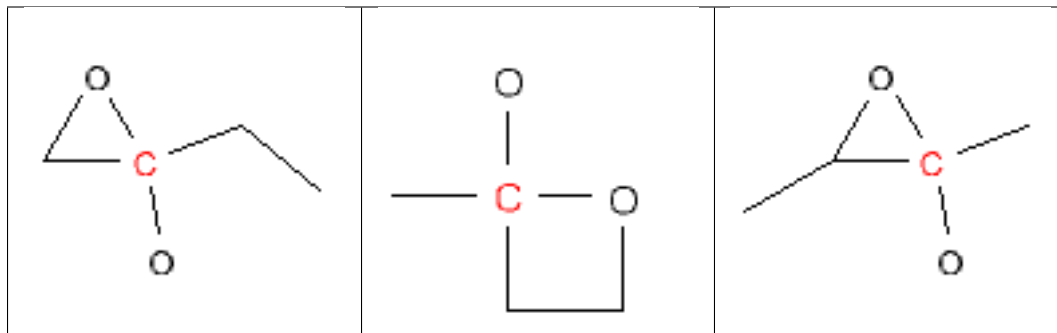
7.2.59 hemiacetal



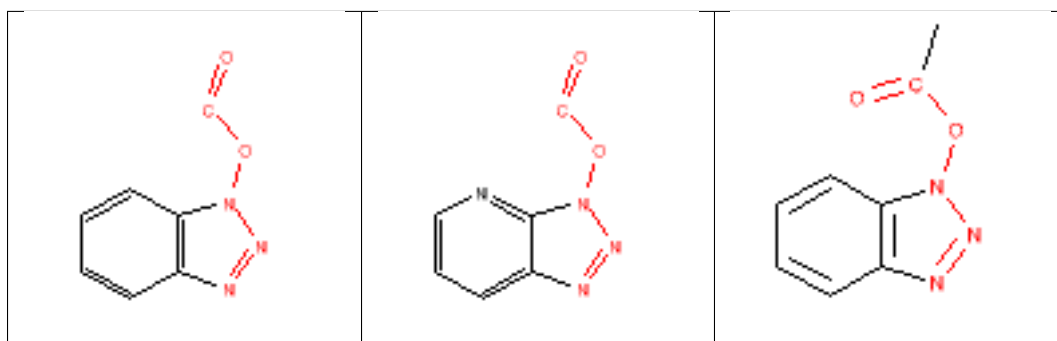
7.2.60 hemiaminal



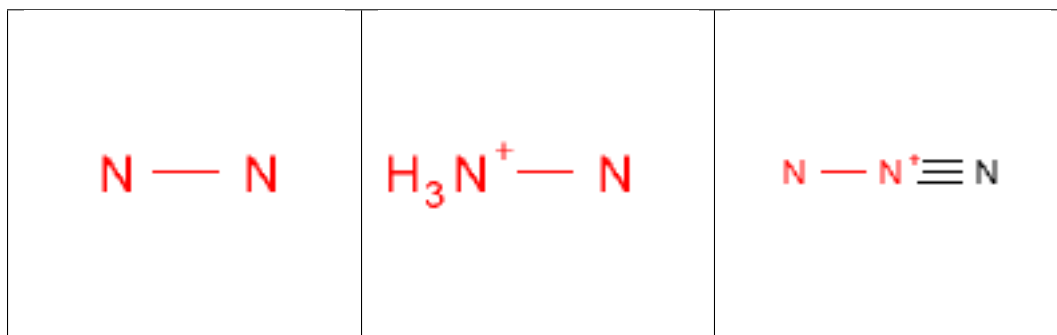
7.2.61 hemiketal



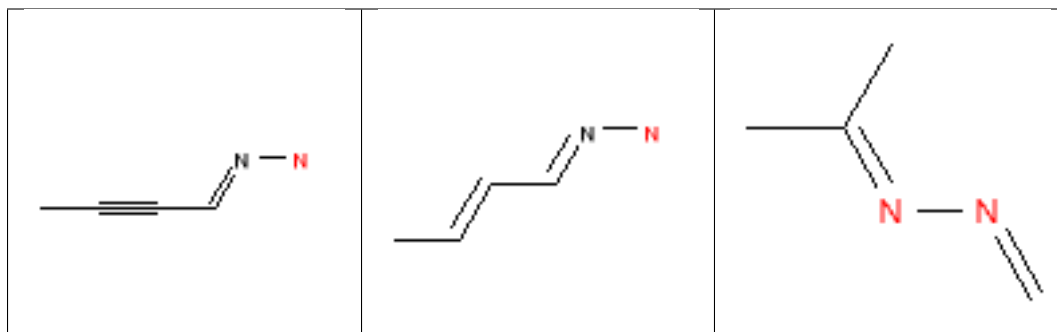
7.2.62 HOBT_esters



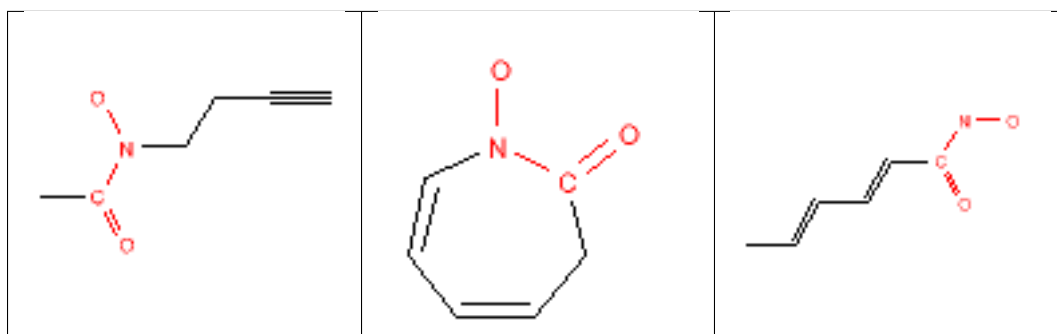
7.2.63 hydrazine



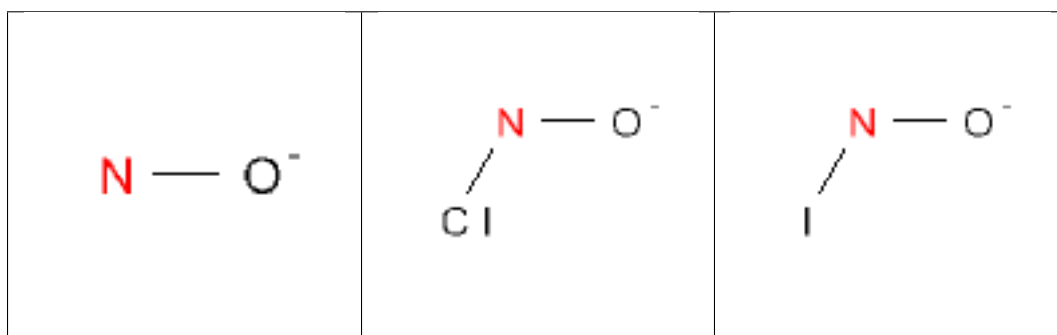
7.2.64 hydrazone



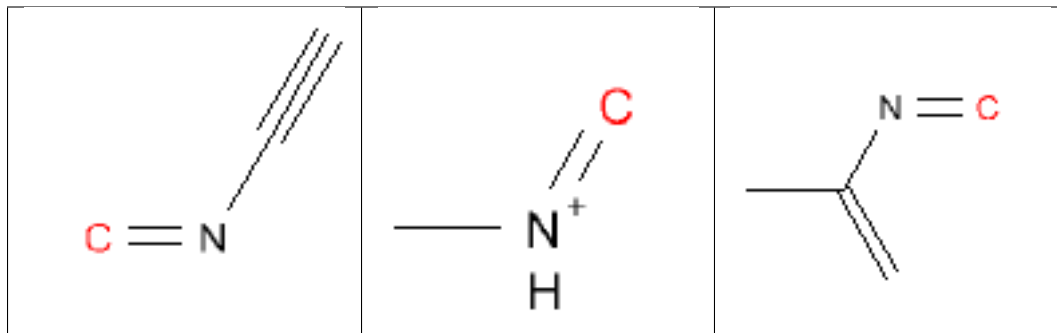
7.2.65 hydroxamic_acid



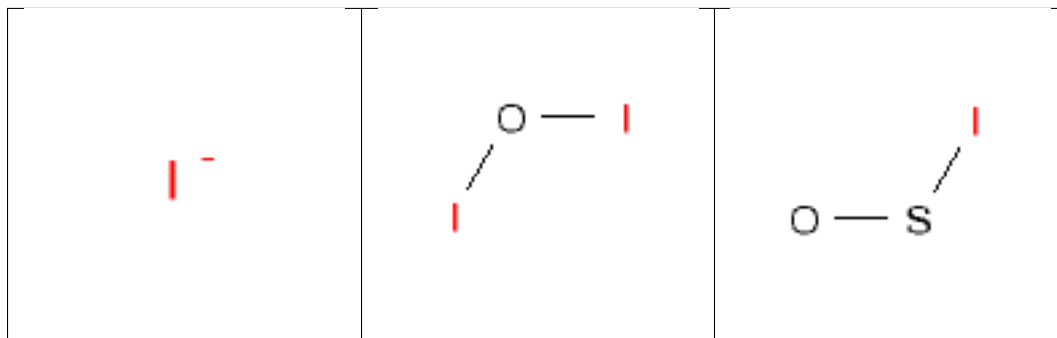
7.2.66 hydroxylamine



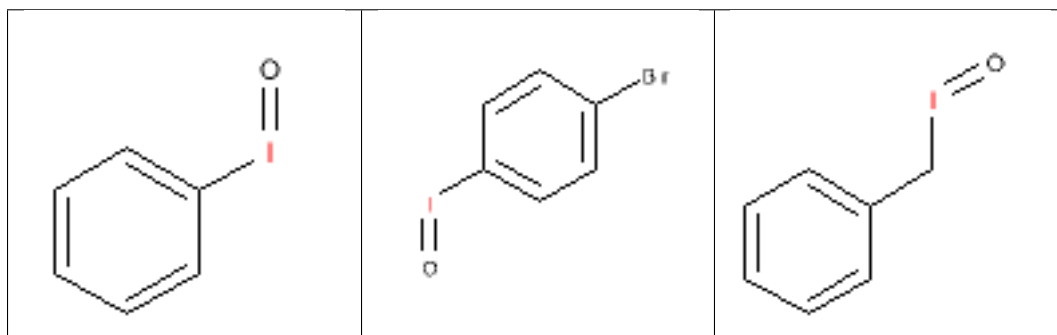
7.2.67 imine



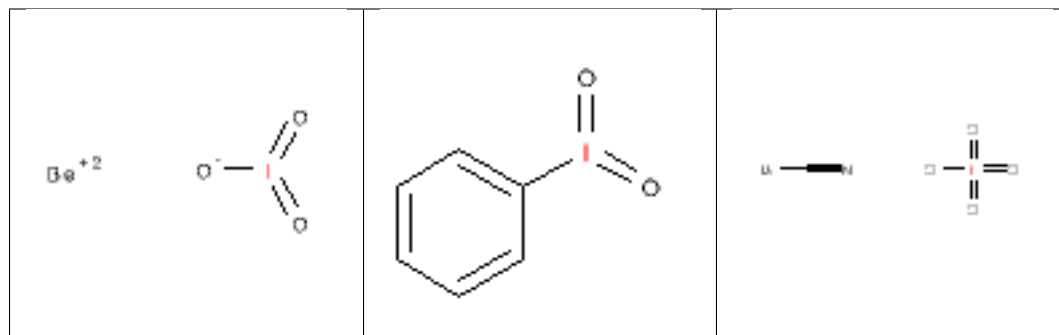
7.2.68 iodine



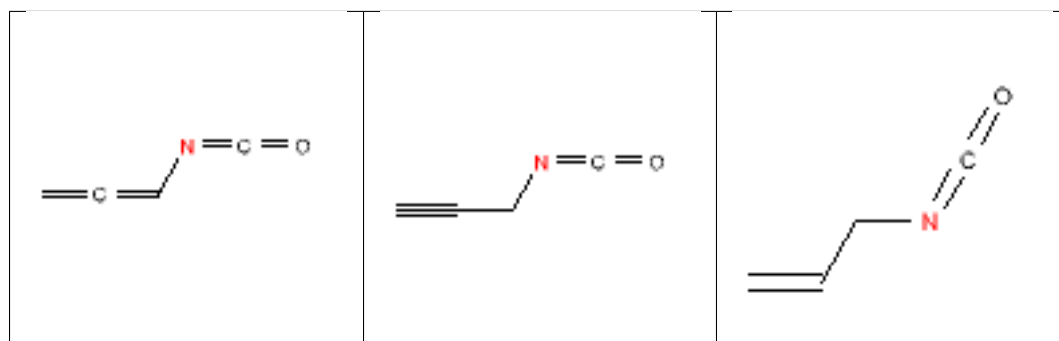
7.2.69 iodoso



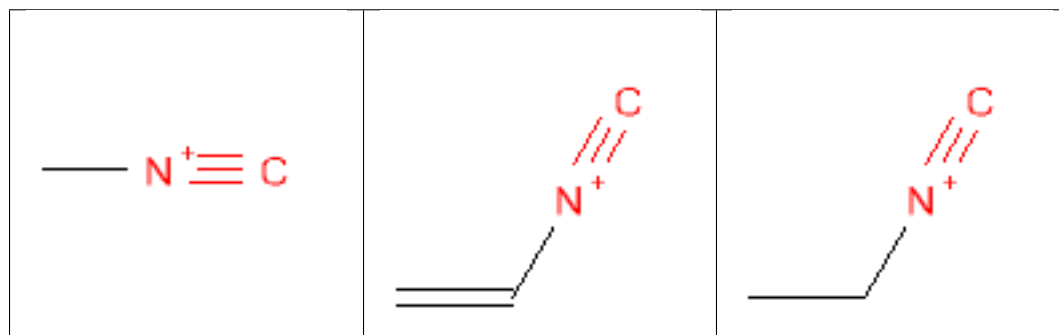
7.2.70 iodoxy



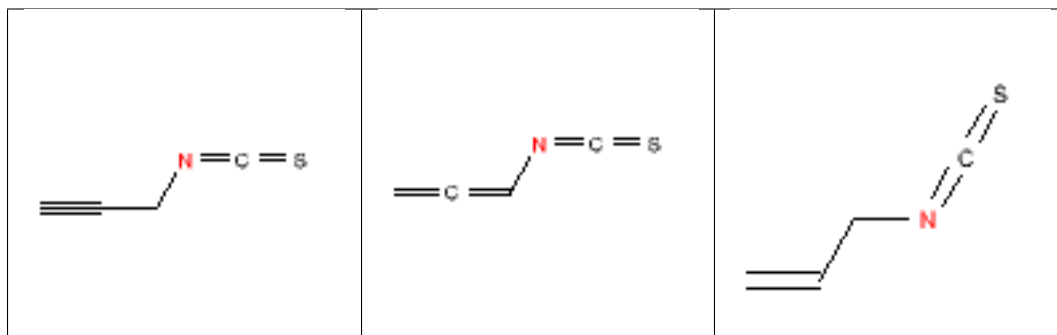
7.2.71 isocyanate



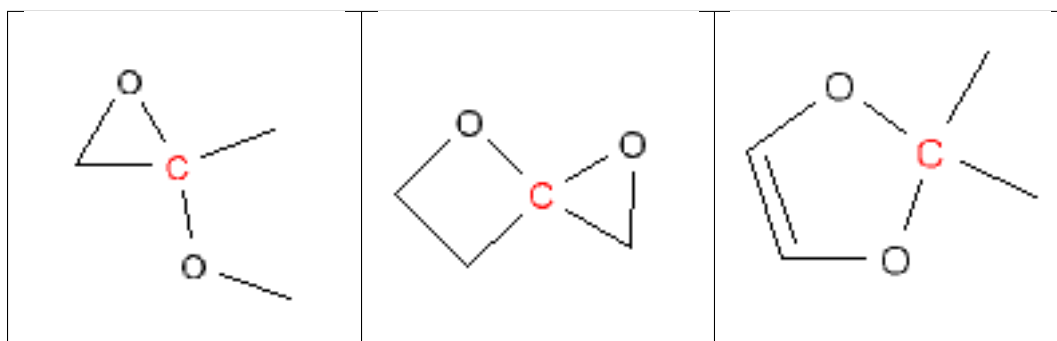
7.2.72 isonitrile



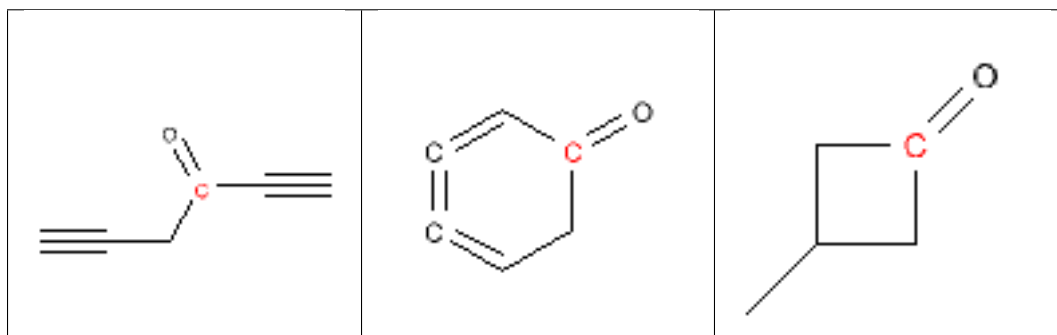
7.2.73 isothiocyanate



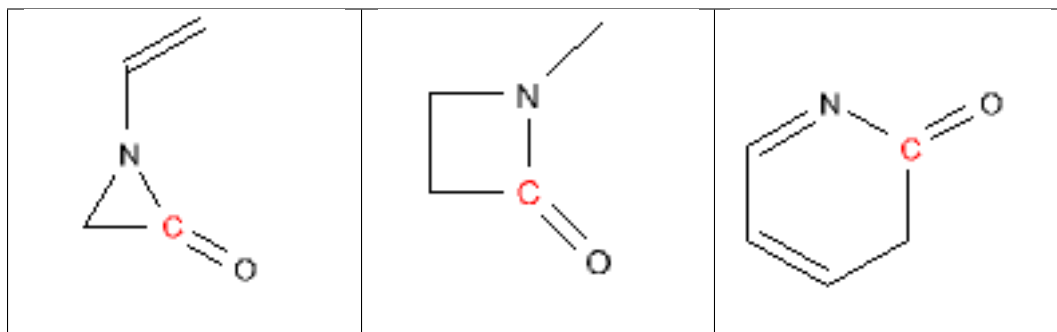
7.2.74 ketal



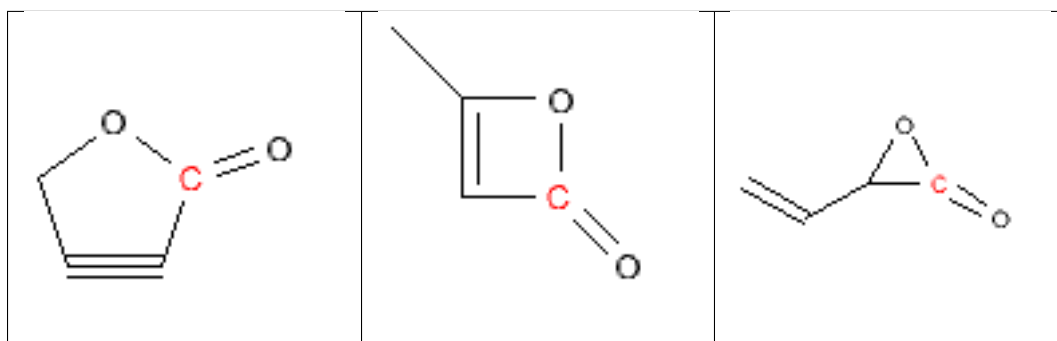
7.2.75 ketone



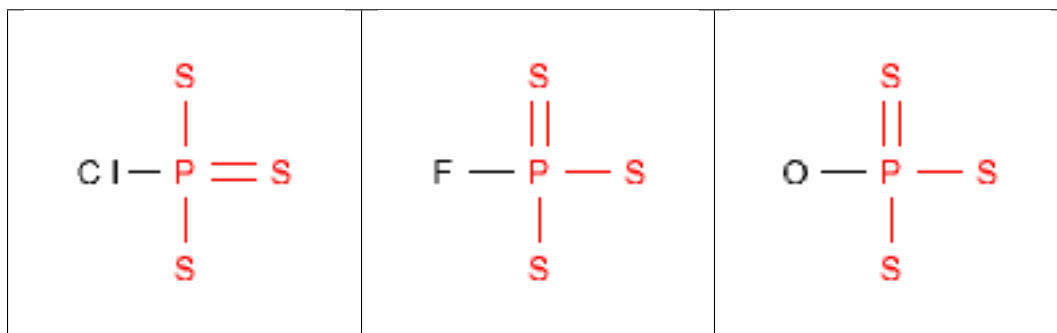
7.2.76 lactam



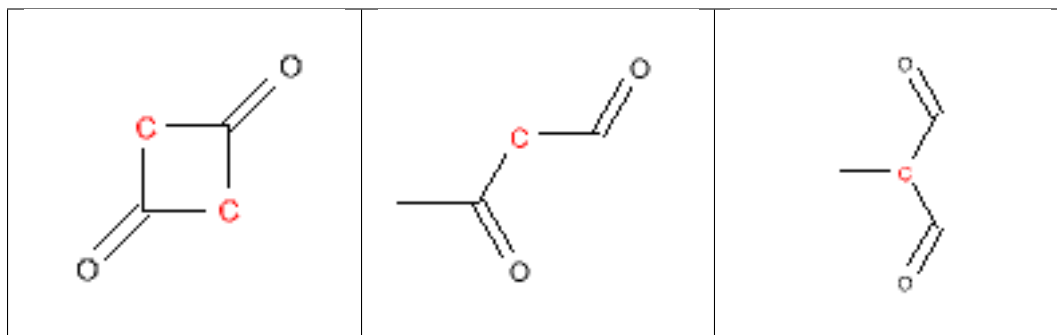
7.2.77 lactone



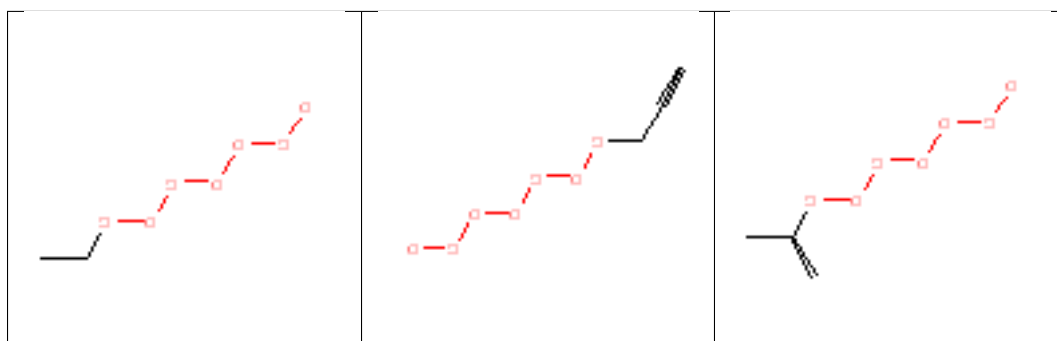
7.2.78 lawesson_s_reagent



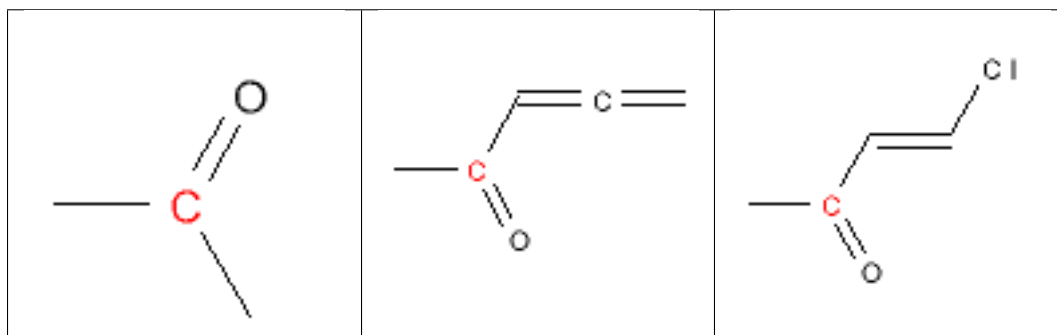
7.2.79 malonic



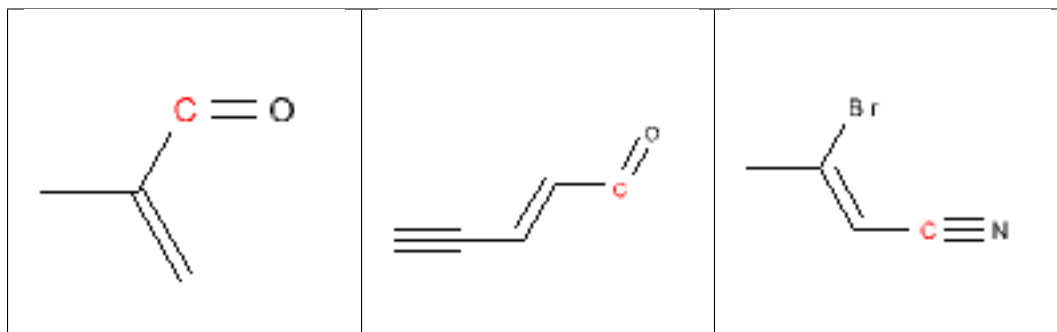
7.2.80 methoxyethoxymethyl_MEM



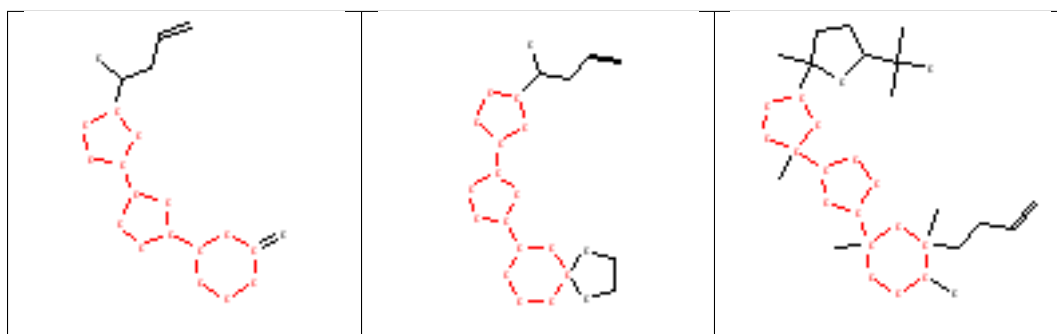
7.2.81 methyl_ketone



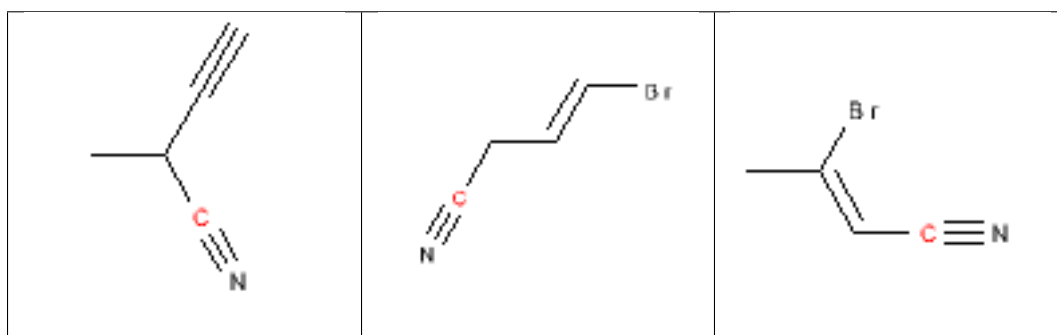
7.2.82 michael_acceptor



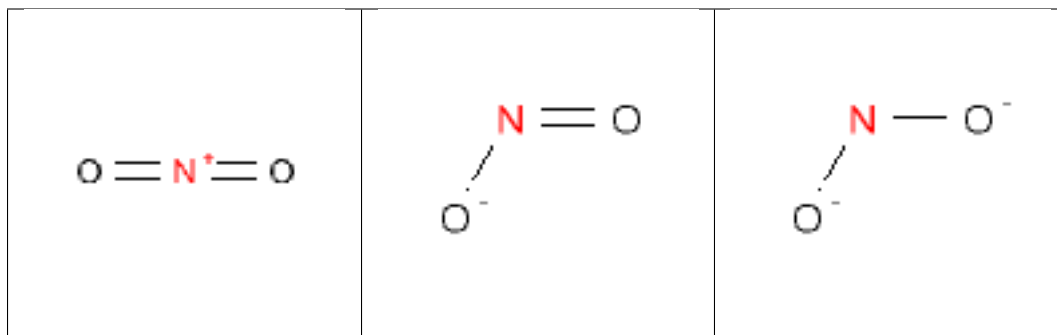
7.2.83 monensin_derivatives



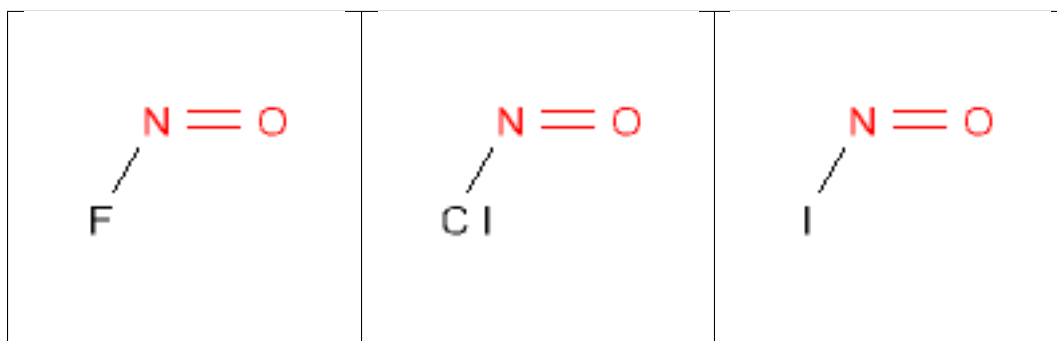
7.2.84 nitrile



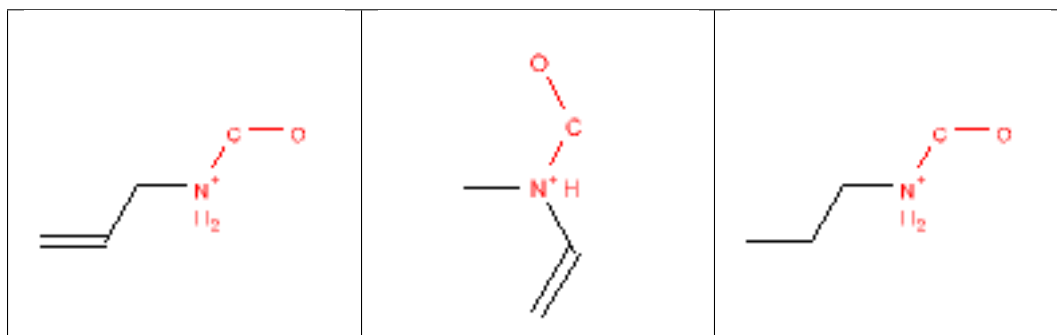
7.2.85 nitro



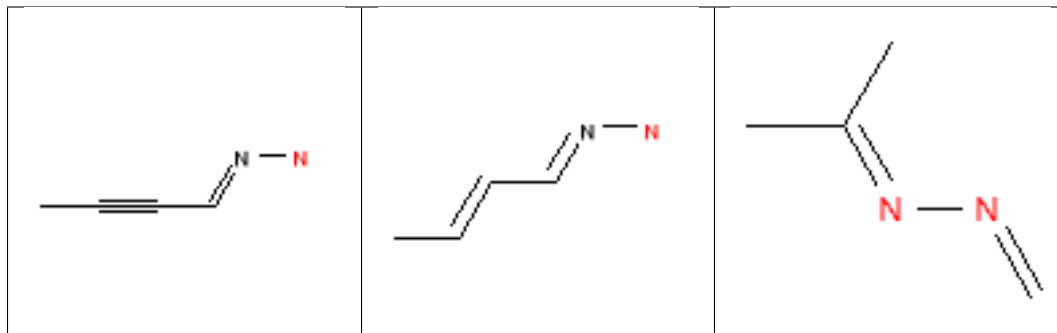
7.2.86 nitroso



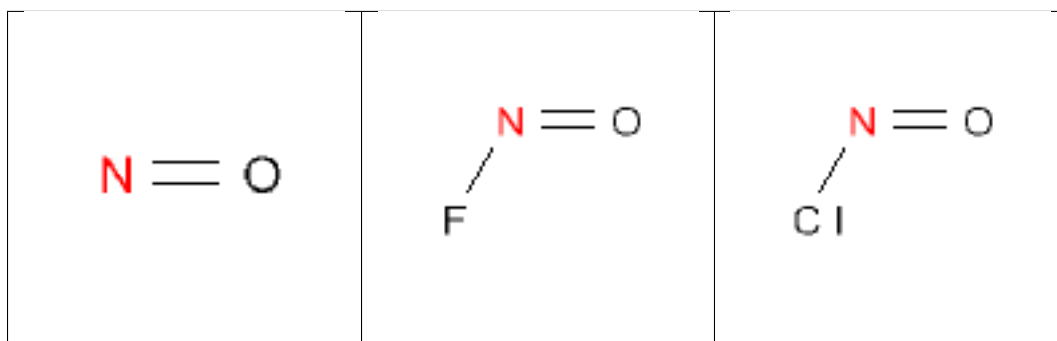
7.2.87 N_methoyl



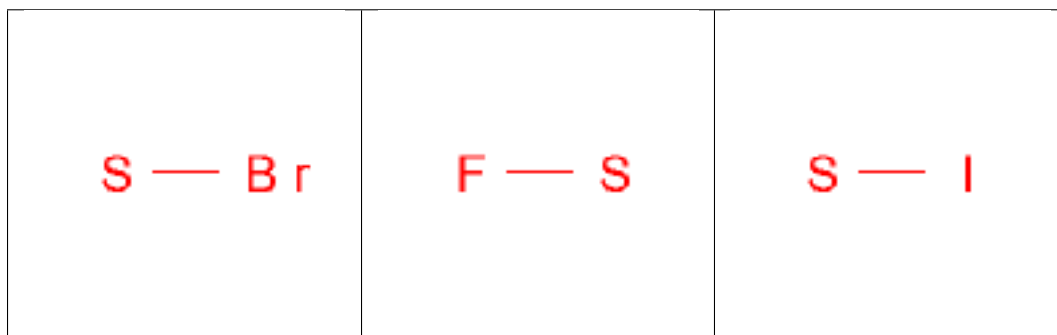
7.2.88 nonacylhydrazone



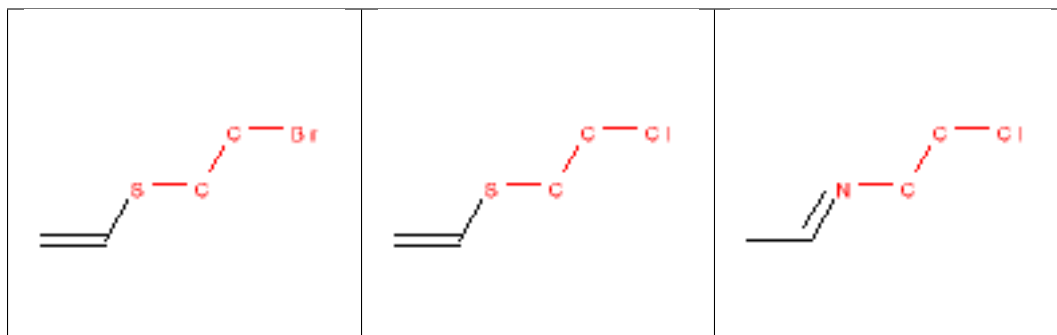
7.2.89 noxide



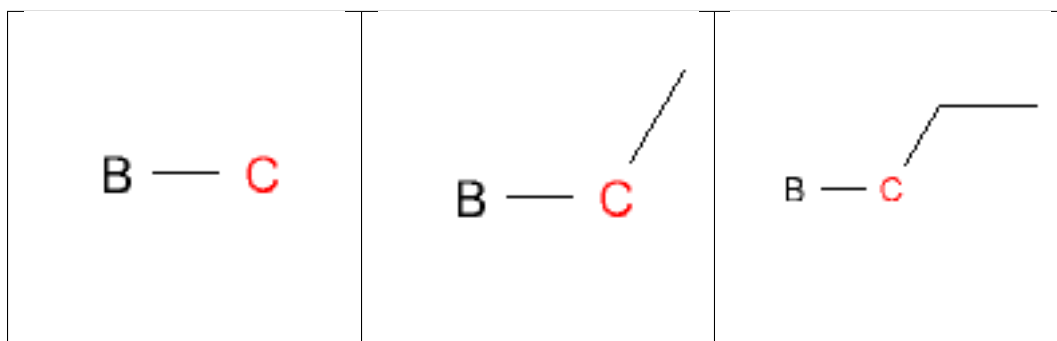
7.2.90 N_P_S_Halides



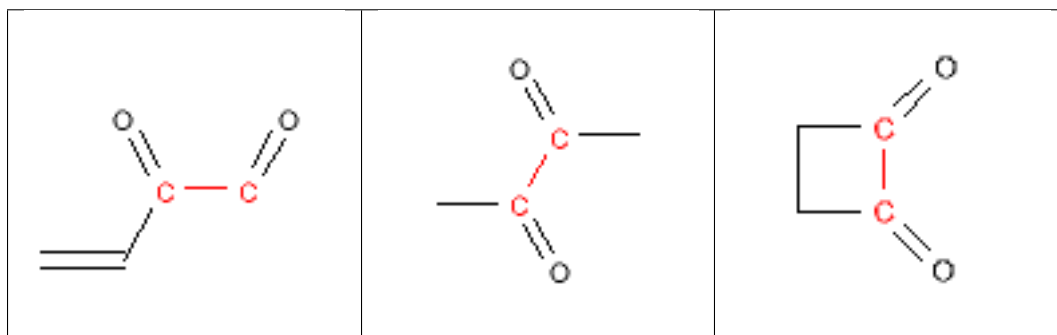
7.2.91 NS_beta_halothyl



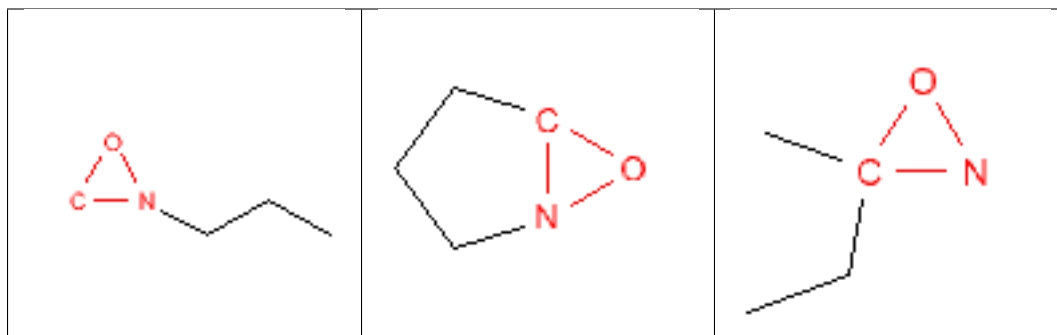
7.2.92 organometallic



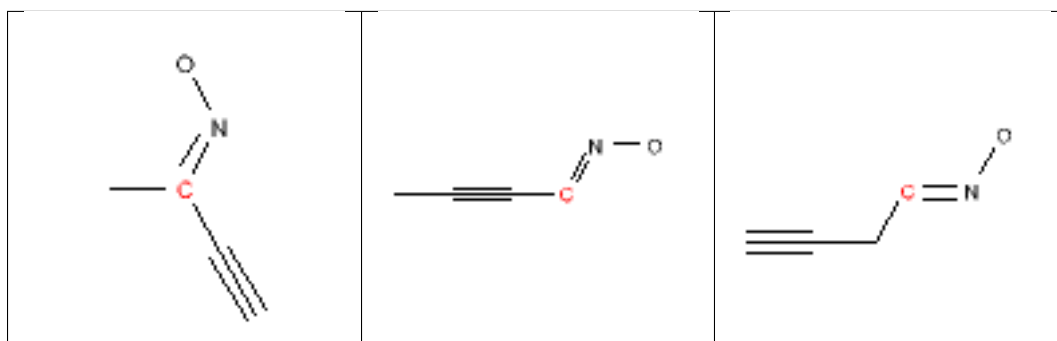
7.2.93 oxalyI



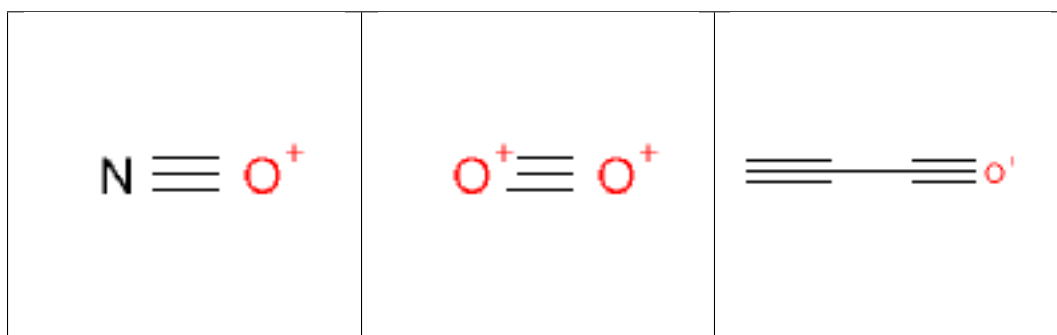
7.2.94 oxaziridine



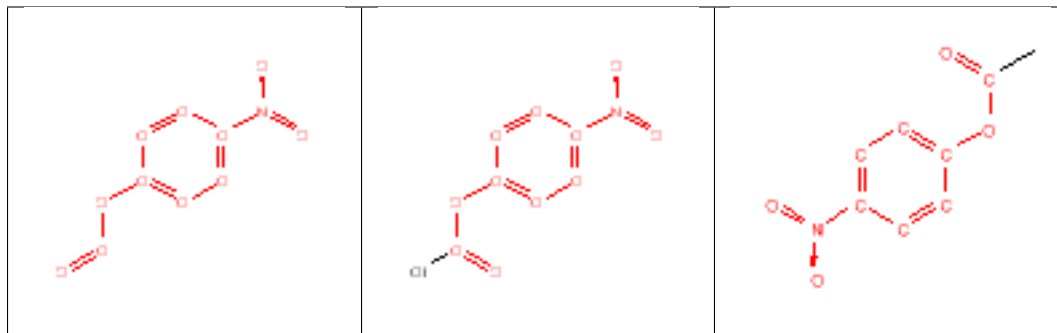
7.2.95 oxime



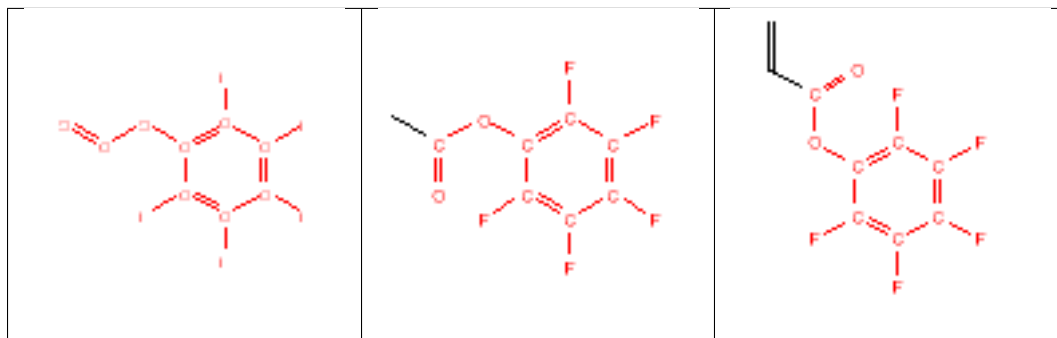
7.2.96 oxygen_cation



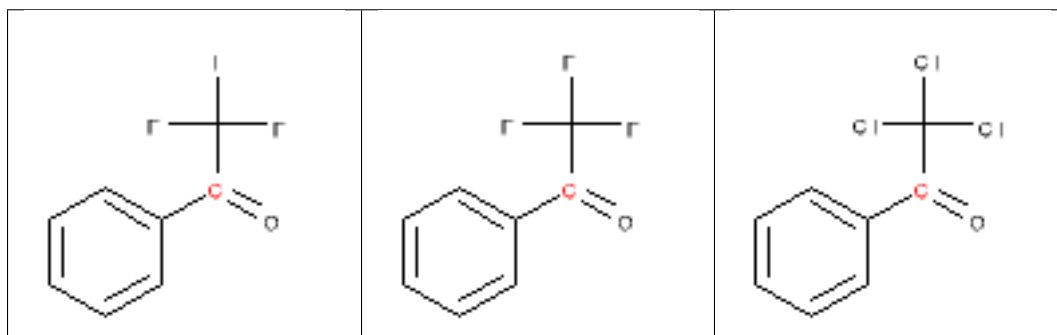
7.2.97 paranitrophenyl_esters



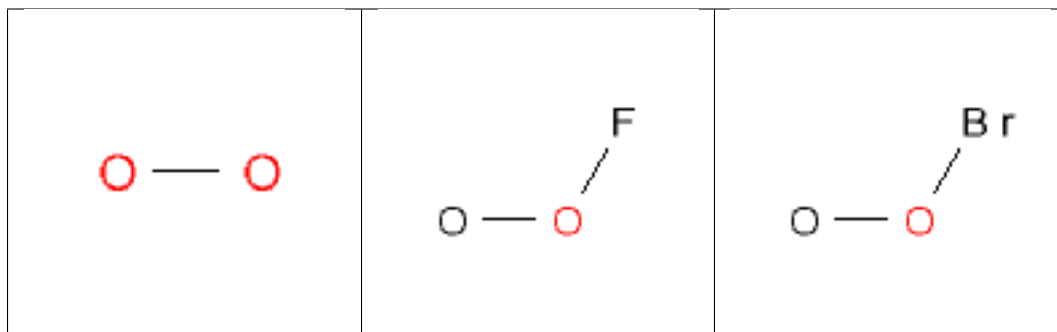
7.2.98 pentafluorophenyl_esters



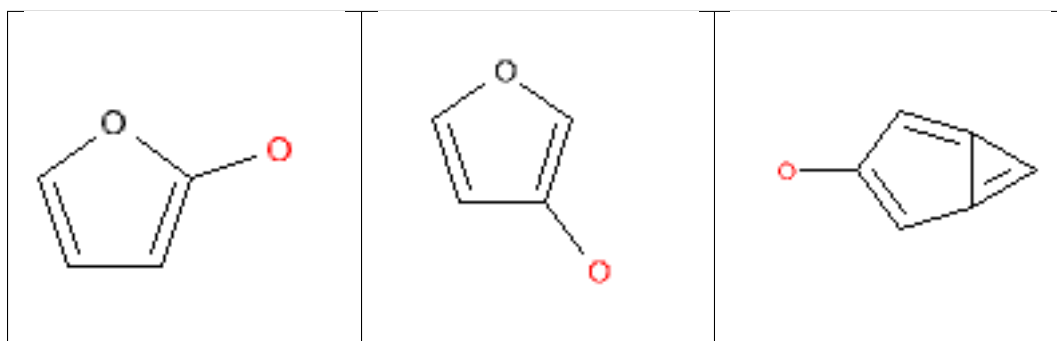
7.2.99 perhalo_ketone



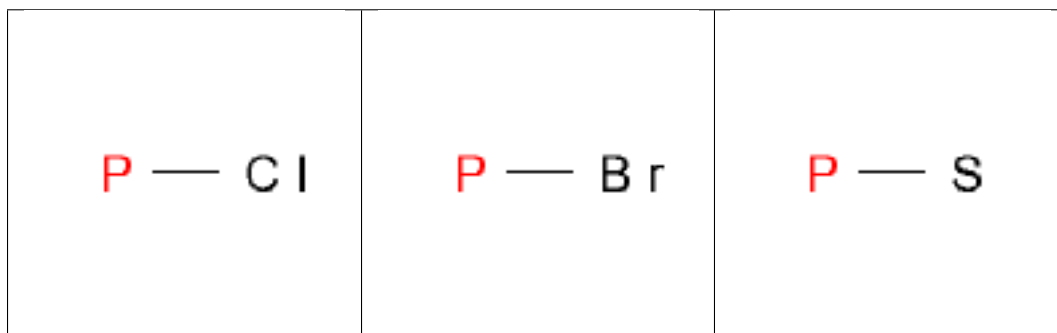
7.2.100 peroxide



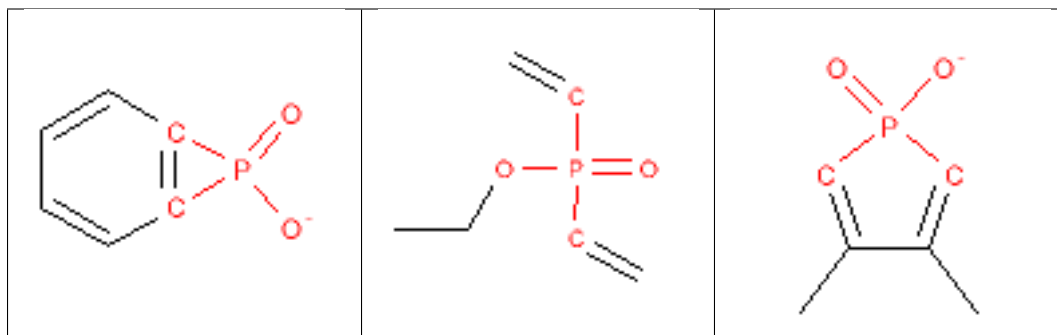
7.2.101 phenol



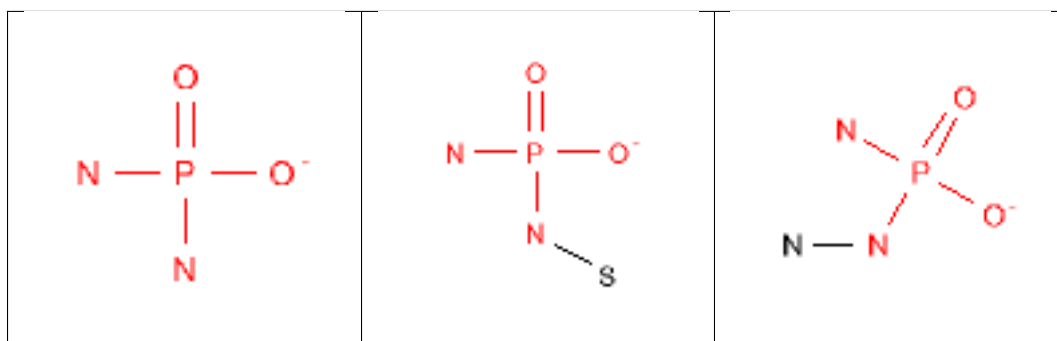
7.2.102 phosphanes



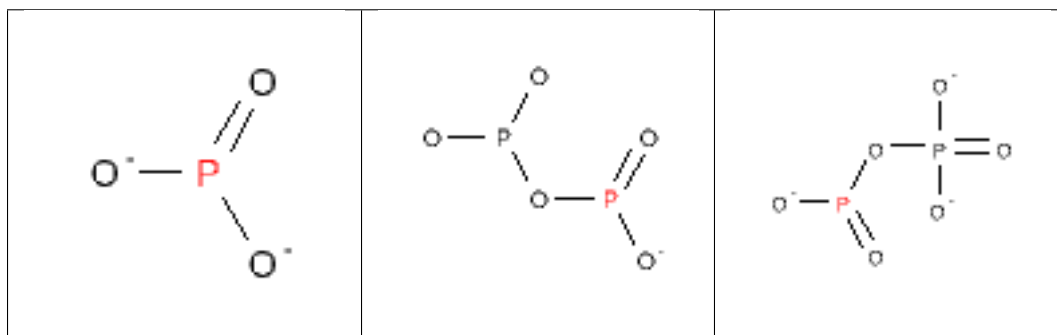
7.2.103 phosphinic_acid



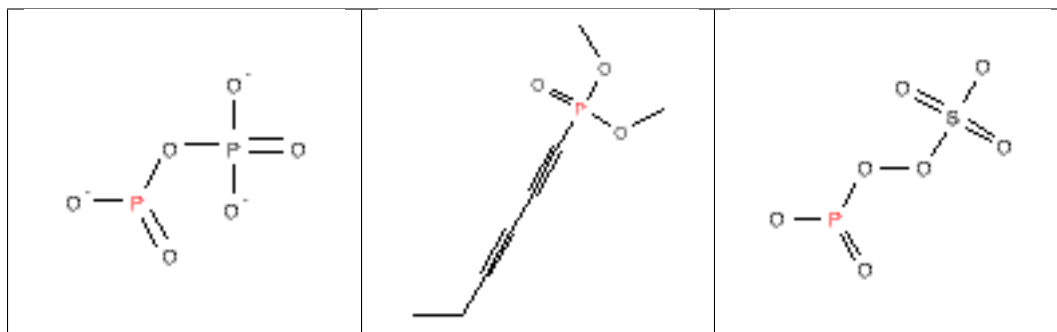
7.2.104 phosphonamide



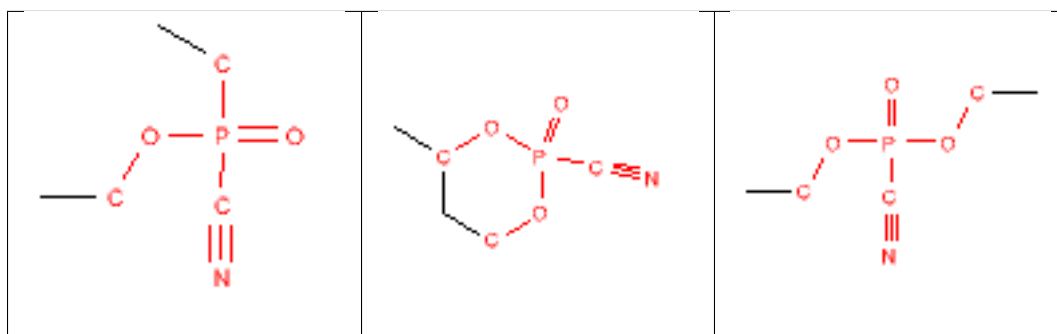
7.2.105 phosphonic_acid



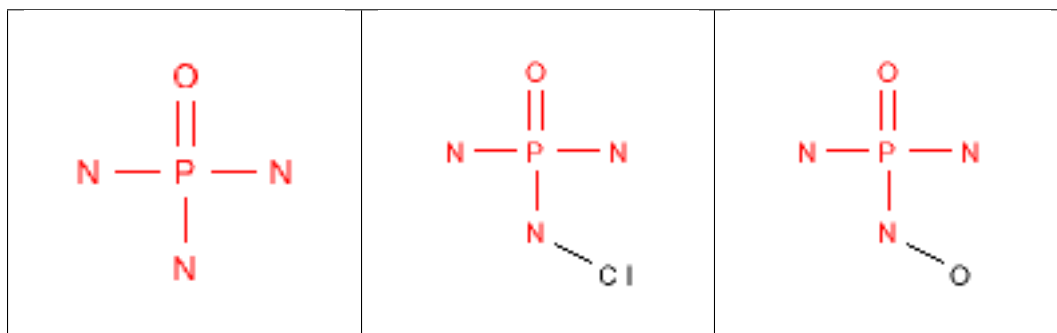
7.2.106 phosphonic_ester



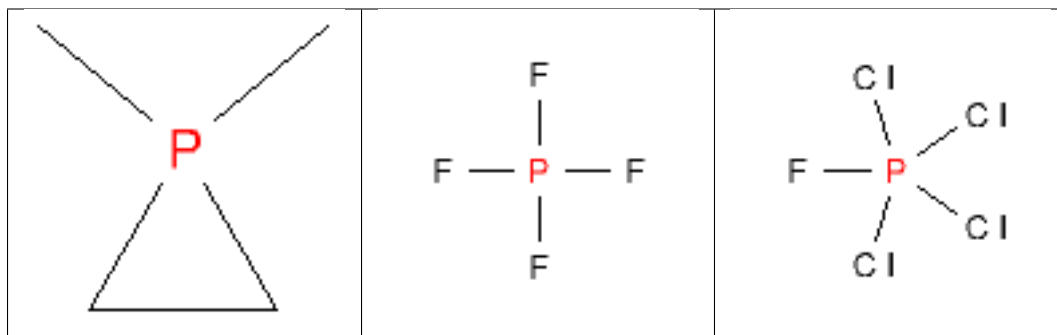
7.2.107 phosphonylnitrile



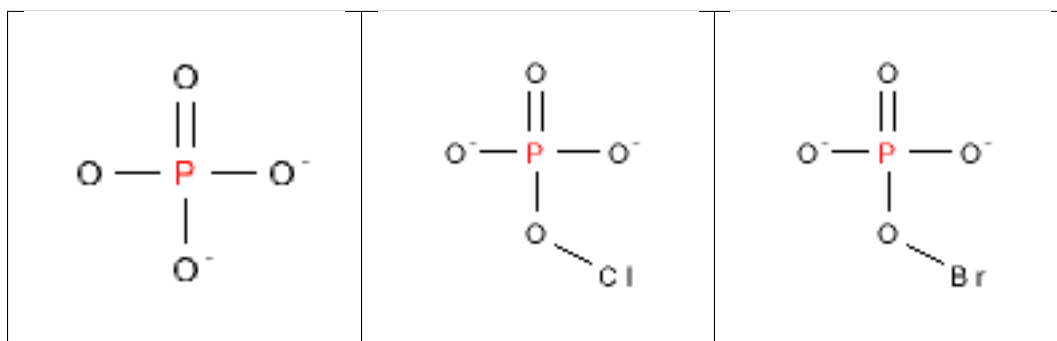
7.2.108 phosphoramides



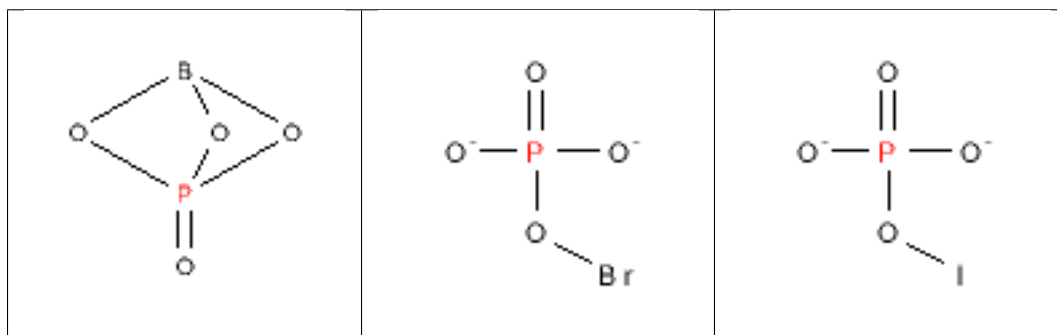
7.2.109 phosphoranes



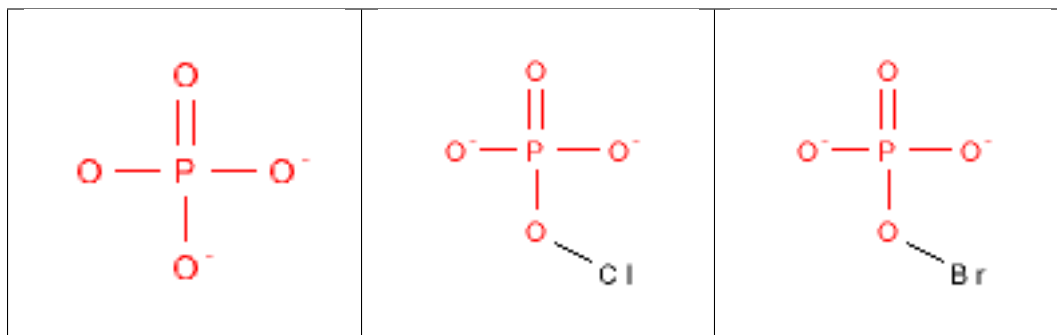
7.2.110 phosphoric_acid



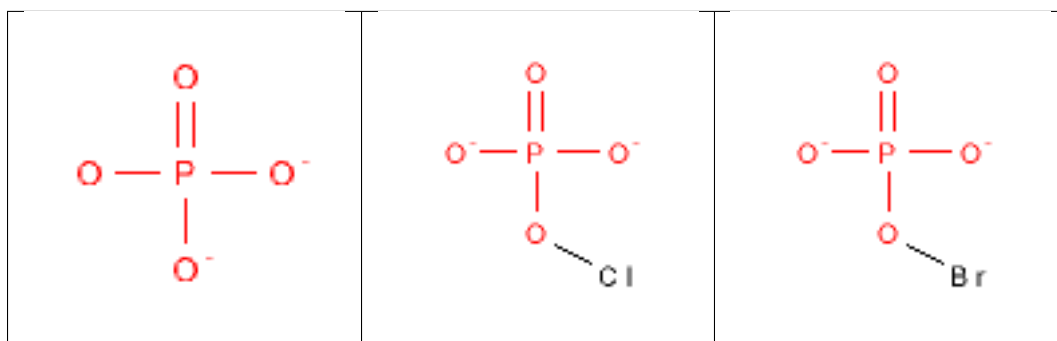
7.2.111 phosphoric_ester



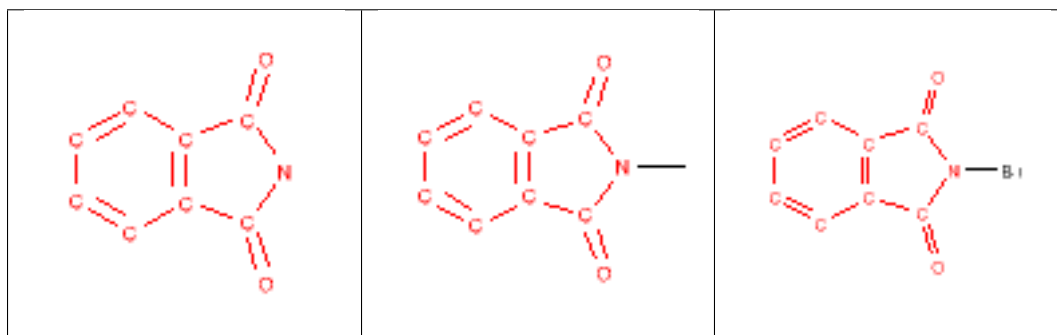
7.2.112 phosphoryl



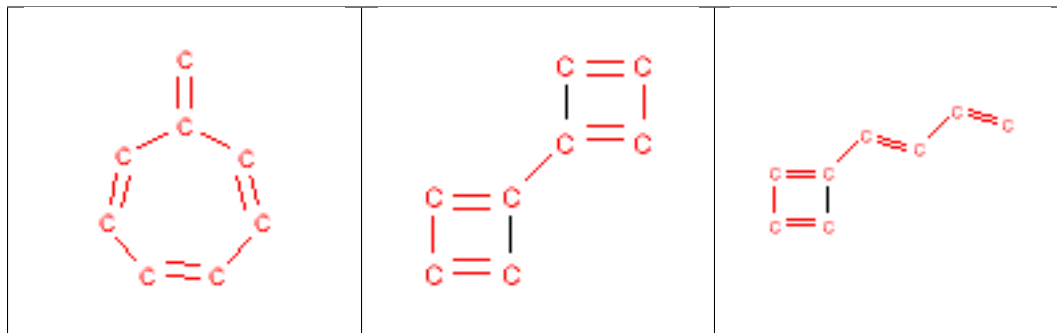
7.2.113 phosphoryl



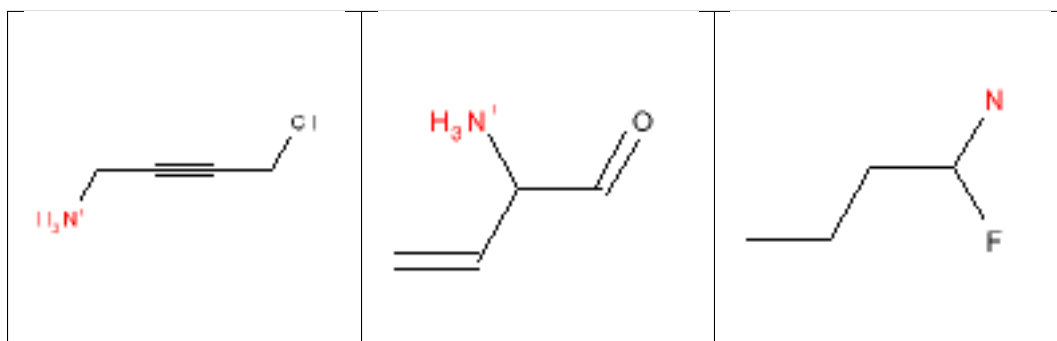
7.2.114 phthalimides_PHT



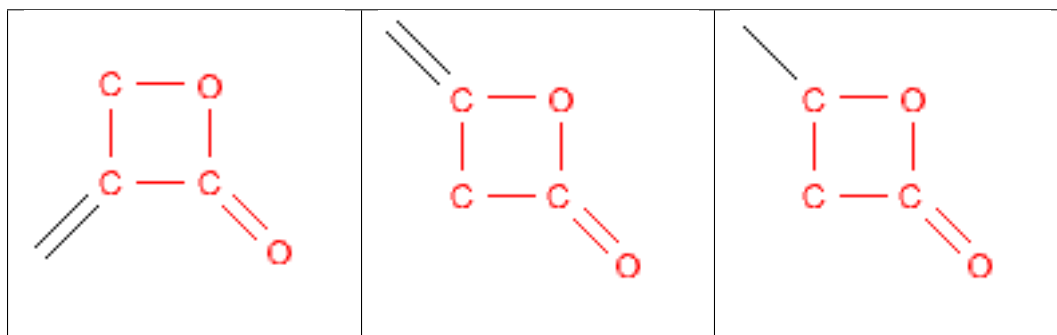
7.2.115 polyenes



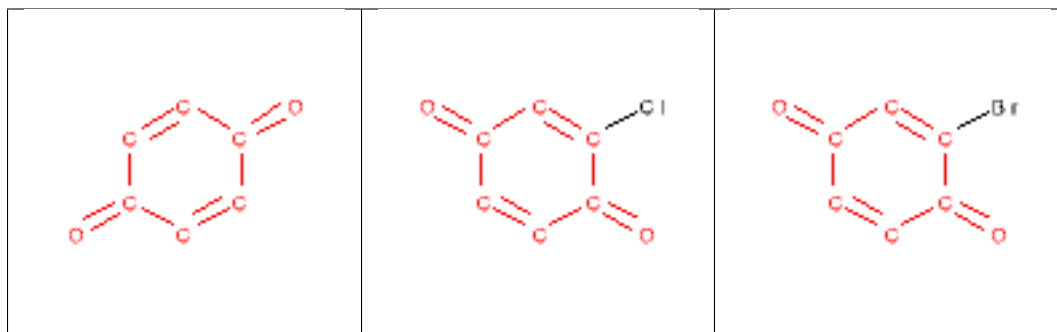
7.2.116 primary_amine



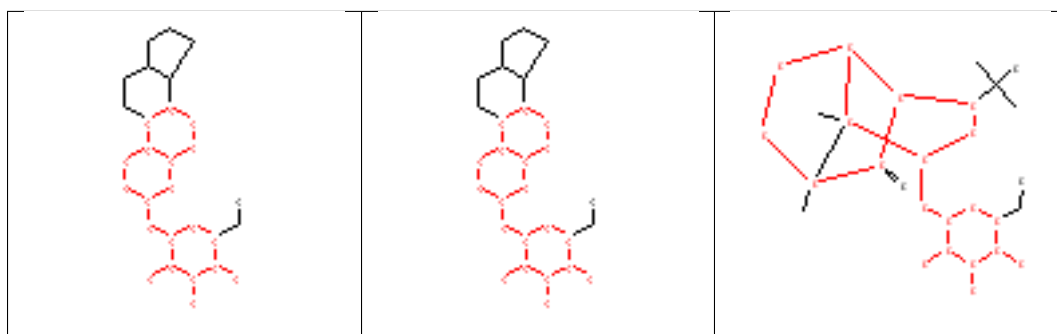
7.2.117 propiolactones



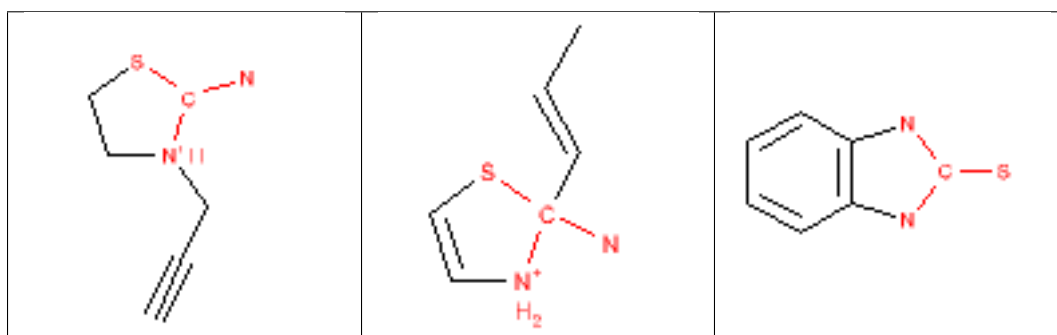
7.2.118 quinone



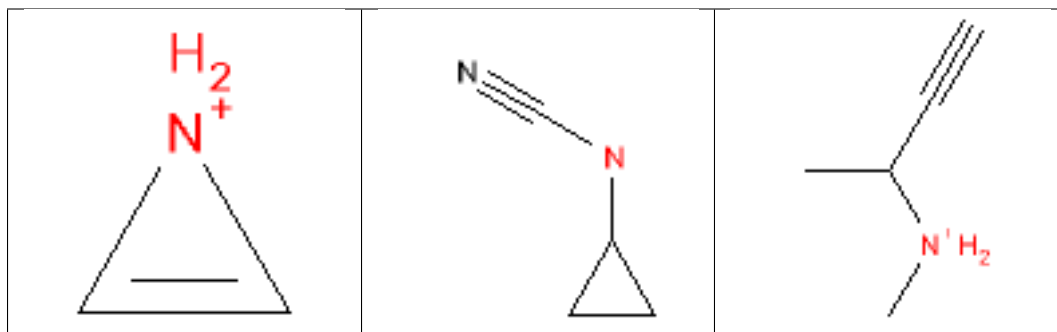
7.2.119 saponin_derivatives



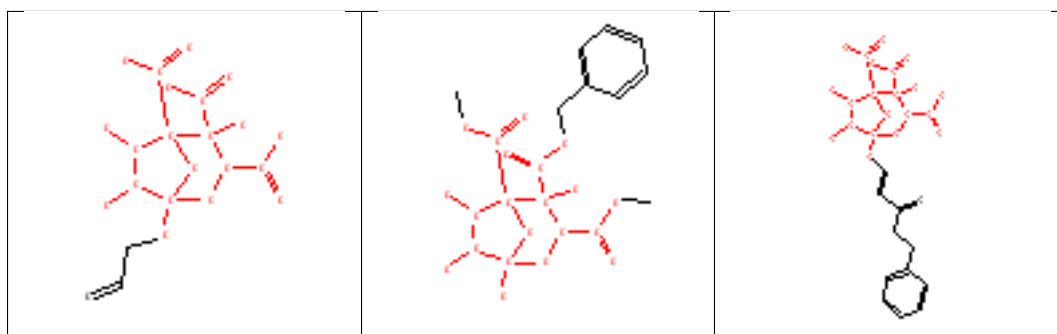
7.2.120 SCN2



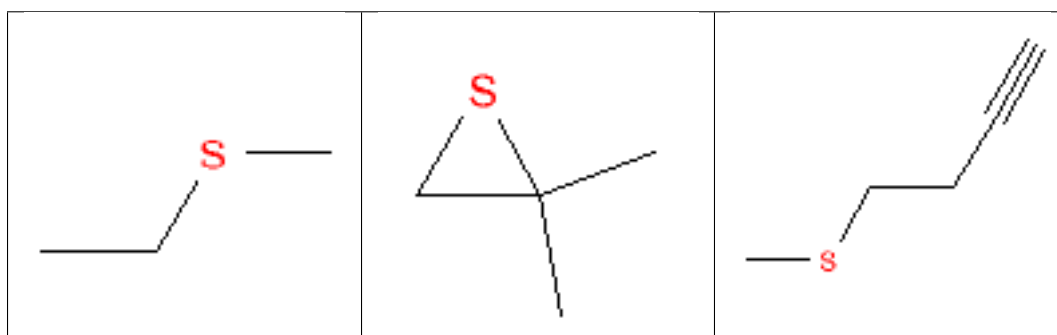
7.2.121 secondary_amine



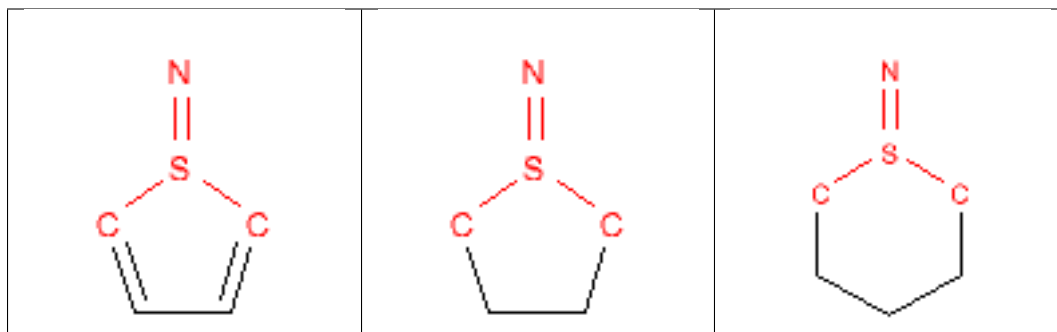
7.2.122 squalestatin_derivatives



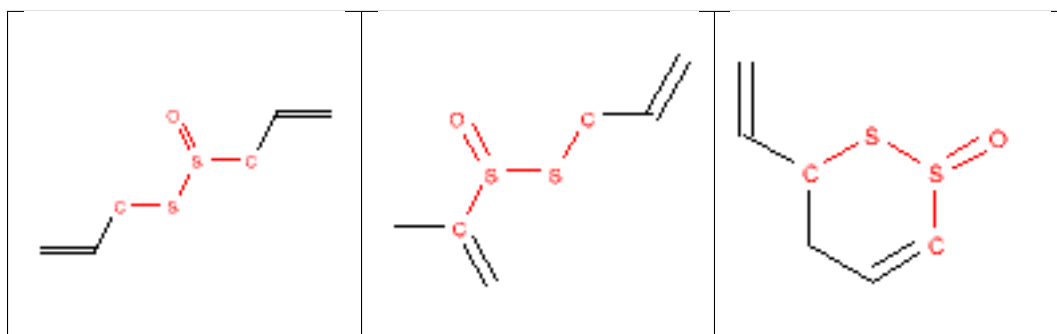
7.2.123 sulfide



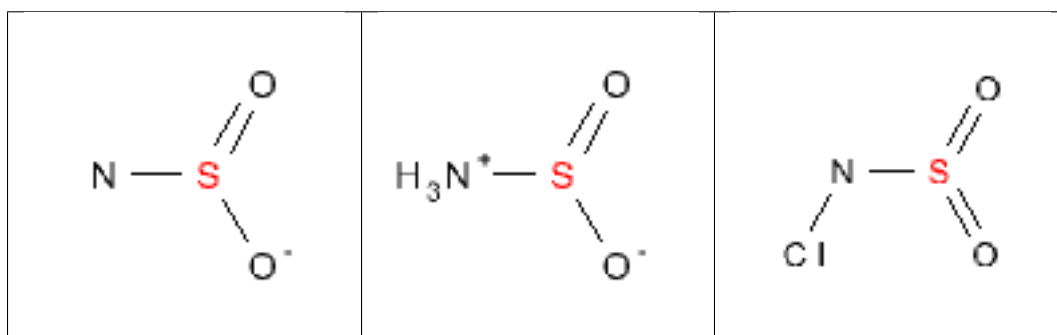
7.2.124 sulfinimine



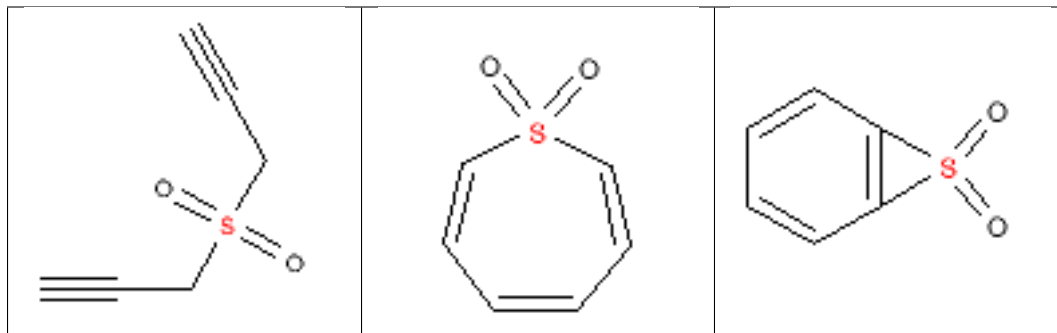
7.2.125 sulfinylthio



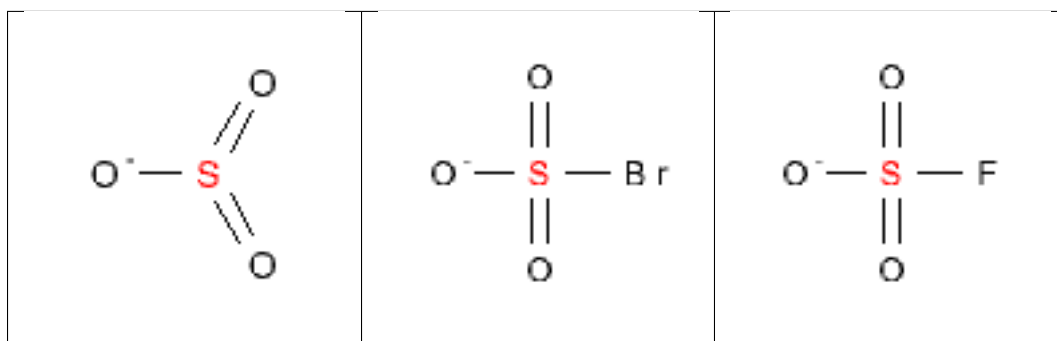
7.2.126 sulfonamide



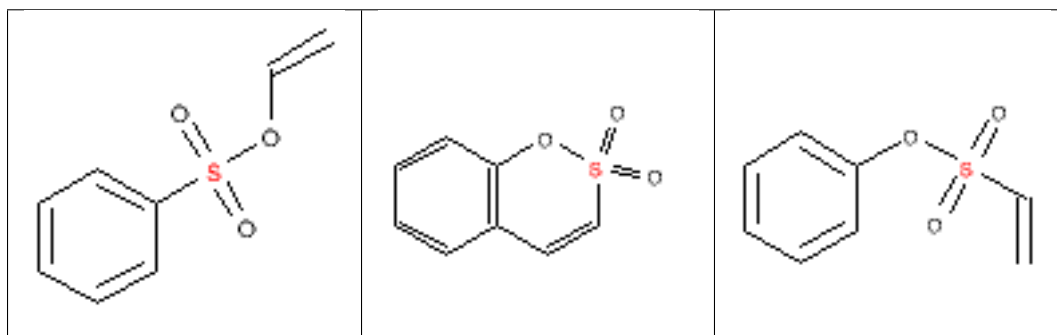
7.2.127 sulfone



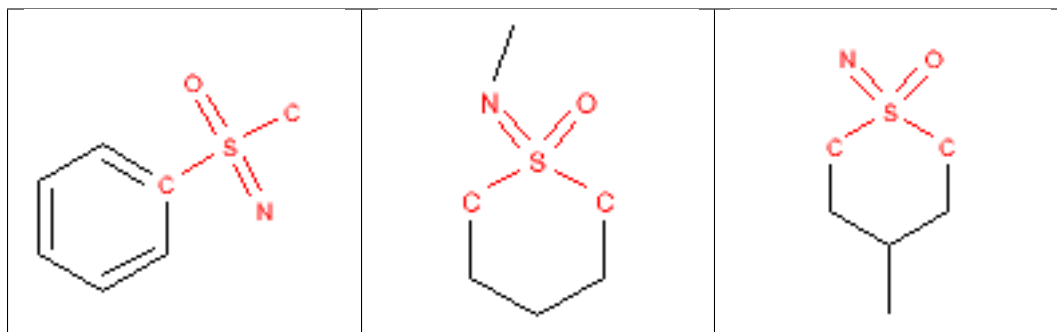
7.2.128 sulfonic_acid



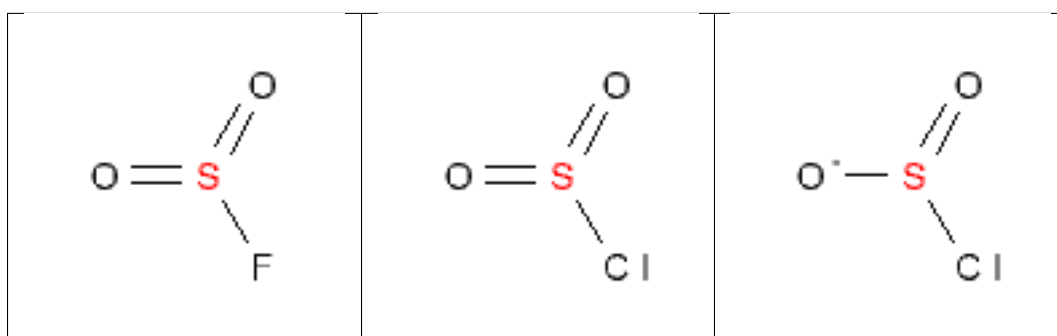
7.2.129 sulfonic_ester



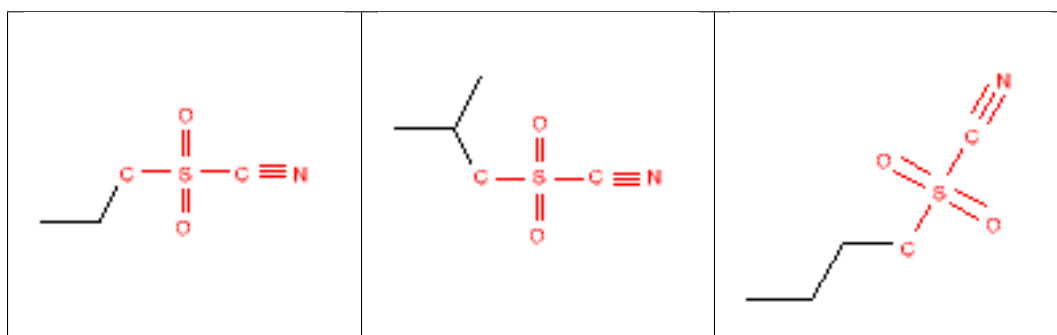
7.2.130 sulfonimine



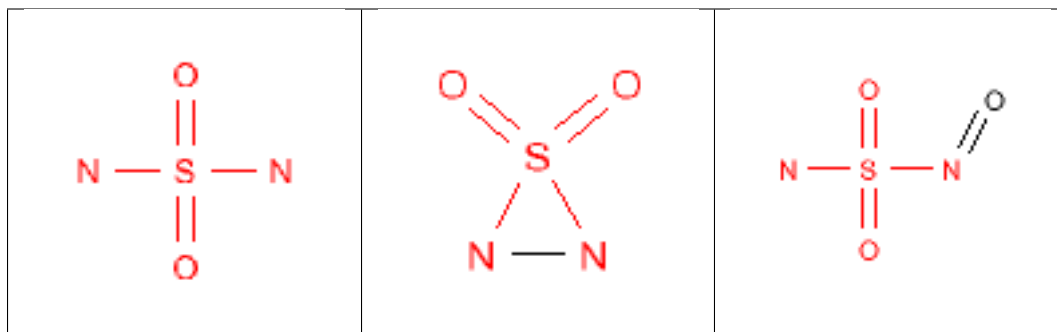
7.2.131 sulfonyl_halide



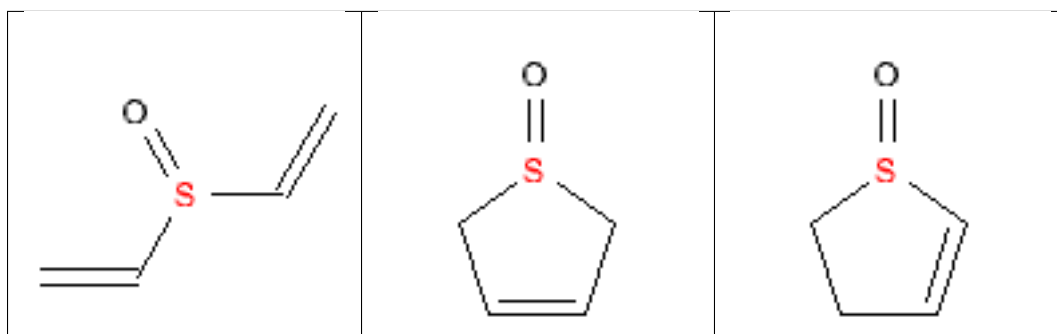
7.2.132 sulfonylnitrile



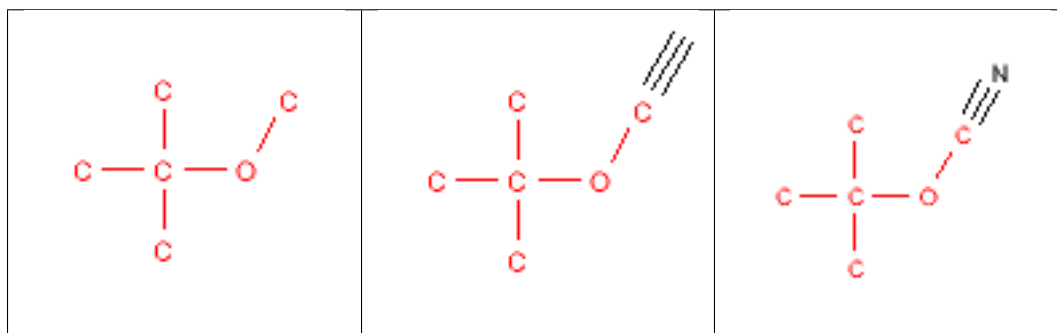
7.2.133 sulfonylurea



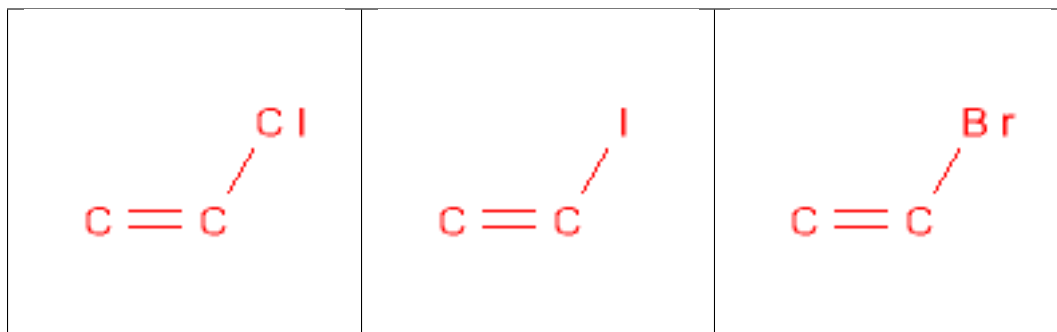
7.2.134 sulfoxide



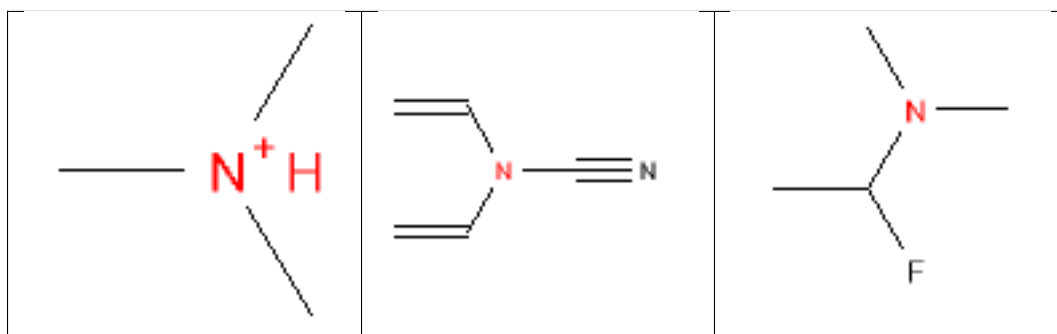
7.2.135 t_butyl_ether



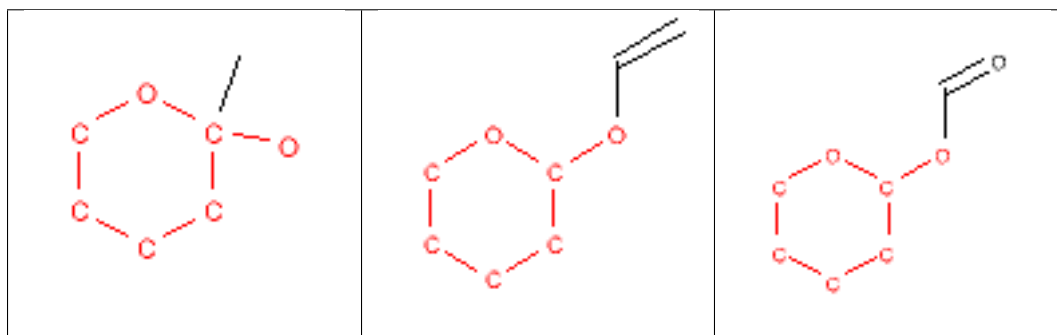
7.2.136 terminal_vinyl



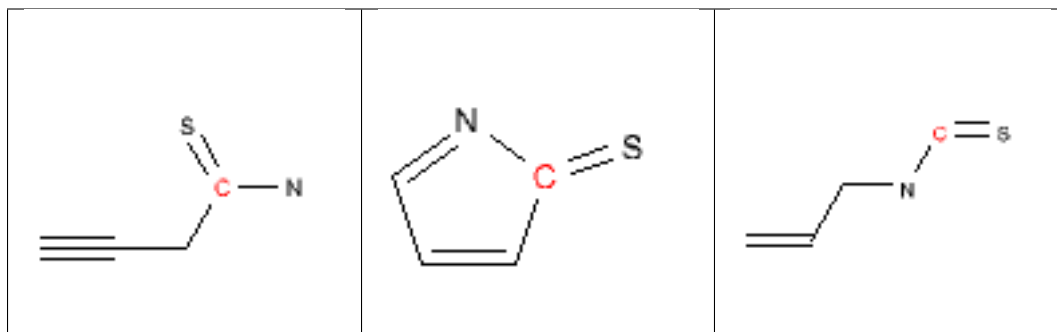
7.2.137 tertiary_amine



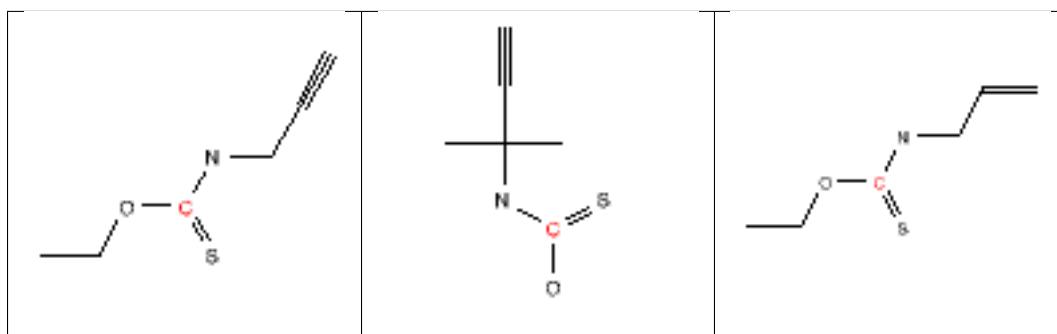
7.2.138 tetrahydropyran_THP



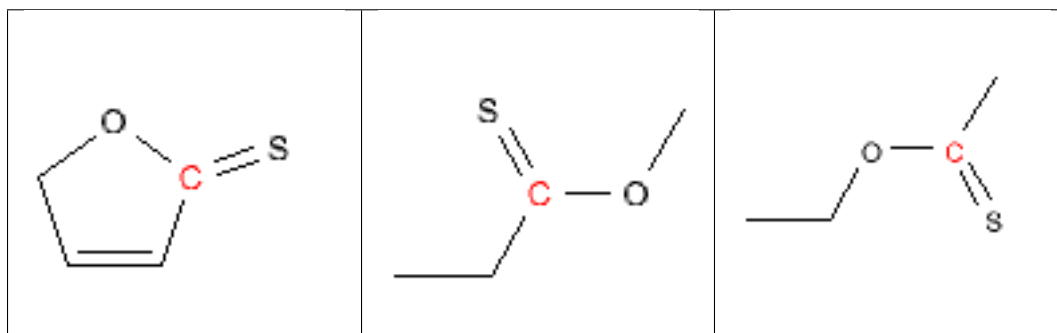
7.2.139 thioamide



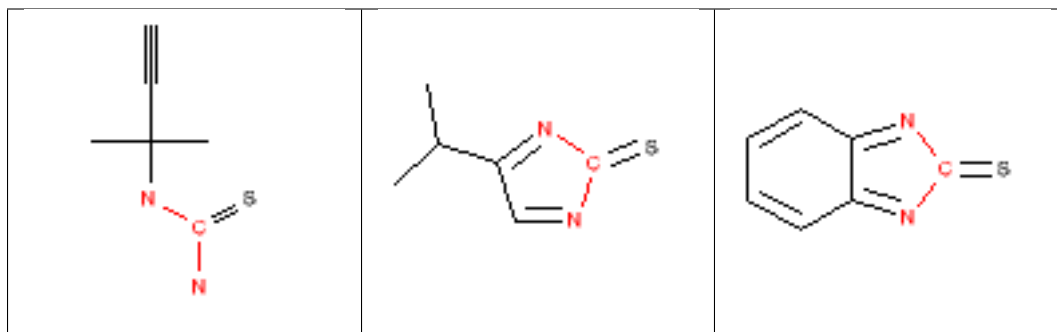
7.2.140 thiocarbamate



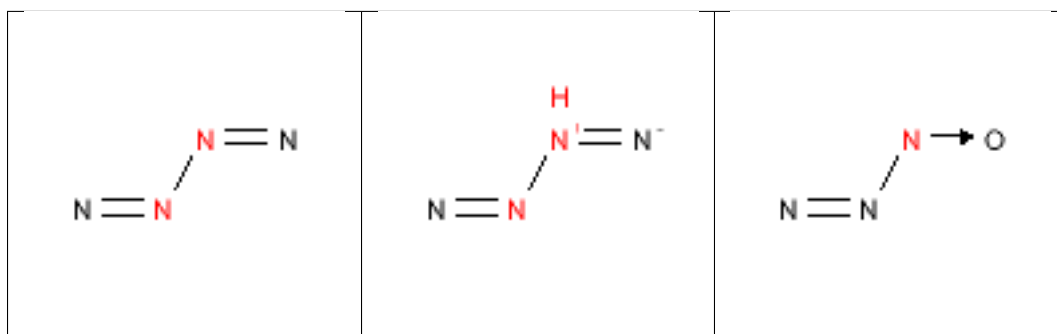
7.2.141 thioester



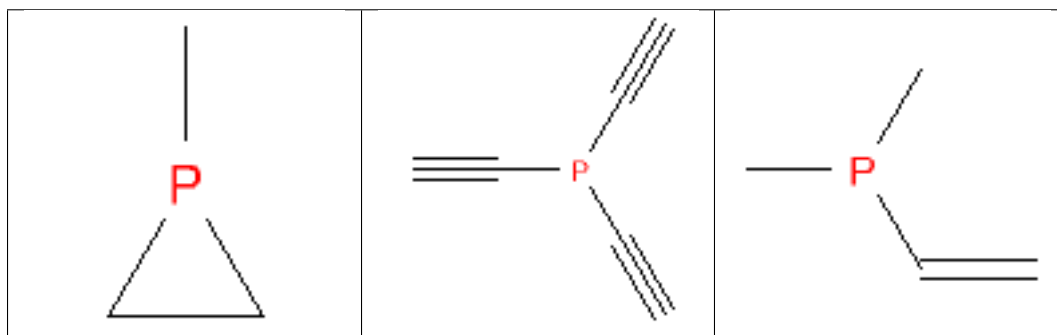
7.2.142 thiourea



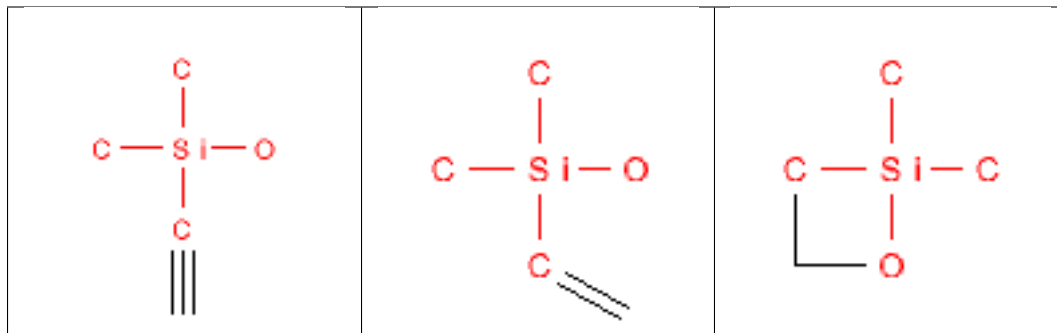
7.2.143 triazine



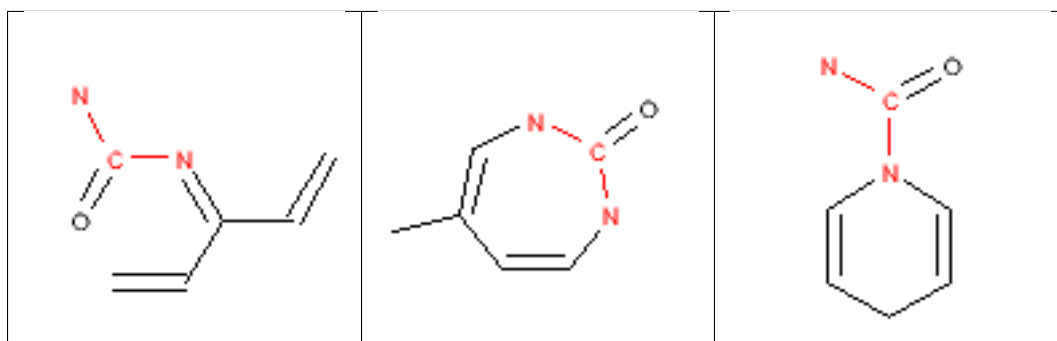
7.2.144 tricarb_o_phosphene



7.2.145 trimethylsilyl_TMS



7.2.146 urea



7.3 New Rules

New rules specify additional functional groups or substructures that may be used. They must specify a substructure definition in the form of a SMARTS in addition to the substructure name and maximum limit. For example:

```
NEWRULE norborane 1 C1CC2CCC1C2
```

The first field is the **NEWRULE** keyword. The second field defines the name associated with the substructure (primarily for logging purposes). The third field indicates the maximum number of the substructure that can be allowed. The fourth field is the SMARTS string for the substructure, norborane in this case. This example rule would indicate that molecules with a single norborane substructure would be allowable, but that those with 2 or more norboranes would be eliminated.

New rules that have a name that is identical with one of the original rules take precedence over the original rule.

7.4 Selection Statements

The select statement allows a filter file to specify the required number of substructures in order to be able to pass the filter. These statements are similar to new rules except that they list a required range for passing the filter rather than the range for failing to pass the filter. For example:

```
SELECT amine 1 1 [N;!$(*-[*!#6;!#1]);!$(*-a);!$(*=, #*)]
```

The first field is the `SELECT` keyword. The second field indicates the name for the selection (again for logging purposes). The third field is the minimum number of substructures required to be in the molecule. The fourth field is the maximum number of substructures allowed in the molecule. The fifth field is the substructure defined by a SMARTS pattern. The example requires that molecules contain exactly one amine. Currently, only a single `SELECT` statement is allowed in the filter file. Any complex boolean substructure statements can be incorporated directly into the SMARTS. If multiple `SELECT` statements occur in a filter file, only the final one will be applied.

RELEASE NOTES

8.1 FILTER 2.1.1

8.1.1 New features

- Added several halide that are more strict to the fragment filter.
- Added several rules to remove bad ring systems to the fragment filter.
- FILTER will now throw an error if duplicate RULE lines are found in the filter file.

Note: The proper way to override a RULE line is to use a NEWRULE line.

8.1.2 Minor bug fixes

- Removing duplicate nitroso and phosphoryl filter file RULES from the default filters.

8.2 FILTER 2.1.0 (*May 2010*)

8.2.1 New Features

- FILTER now will run with a valid OMEGA license

8.3 FILTER 2.0.3 (*November 2009*) (Toolkit only)

This is a minor bug fix release that includes some missing filter definition files that were missing from FILTER 2.0.2.

8.3.1 Bug fixes

- FILTER distribution now includes filter_alldrug.txt filter definition file.
- FILTER distribution now includes filter_blockbuster.txt filter definition file.
- FILTER distribution now includes filter_2.5_blockbuster.txt filter definition file.

8.4 FILTER 2.0.2 (*March 2009*)

This is a minor bug fix release that updates FILTER to OpenEye's standardized distribution system.

8.4.1 Bug fixes

- FILTER installations now support having multiple versions and platform architectures in the same directory structure.
- FILTER installations on Microsoft Windows platforms now have documentation available in the start menu.
- FILTER installations on Microsoft Windows platforms now have OpenEye Command Prompts available from the start menu. These prompts set the appropriate environment variables for running OpenEye software.

8.5 FILTER 2.0.1 (*January 2007*)

This release is a broad-sweeping bug fix release with a dramatically improved introduction in the documentation. It includes many simple and complex bug fixes including; conceptual clarifications of several algorithms, fixes and improvements to the data representation, several advances in the pKa model, and significantly more documentation. While this is only a bug-fix release, it represents a significant step-forward in the refinement of the product. We hope you find it suits your needs.

8.5.1 New features

- Added three additional filters that are designed around the best-selling prescription drugs
- Predicted aggregator QSAR model introduced to further enhance aggregator recognition capability
- Implemented an algorithm for "total functional group count" that takes account of all functional groups rather than only those included in the filter patterns
- Added -tableFlag parameter to mark all failure values in the table file with an asterix
- Extended the unique flag to allow values of "true", "false" or a filename used to pre-load a file of uniques
- Added pKa for phenol groups with significant electron withdrawing groups

8.5.2 Bug fixes

- Fixed two bugs in the algorithm for recognizing hydrogen-bond donors and acceptors
- Abbott Bioavailability Score corrected to be reported as a probability
- Aggregator function augmented to include more than 250 additional known aggregators
- XLogP and LogS corrected to always examine the hypervalent form of molecules (as are used in the training set)
- Improved data reporting and filter control for the pharmacokinetic properties
- Fixed category reporting for solubility calculation
- Changed the default TPSA calculation to ignore rather than include P and S surface area
- Renamed and clarified aliphatic_group filter to "maximum connected non-ring atoms"
- Fixed typos in names beta_carbonyl_quat_nitrogen and t_butoxycarbonyl_tBOC

- Updated chloramidine filter name to the more common imidoyl_chloride
- Fixed error in beta_azo_carbonyl pattern
- Made patterns for cytochalasin, monensin, squalastatin and saponin more specific
- Clarified the exclusion patterns for acyclic_NCN and SCN2
- Dramatically improved the specificity and range of the michael_acceptor pattern
- Fixed error in alkyl_halide pattern that limited it to bromine and iodine
- Fixed multiple patterns that matched their function group multiple times because of symmetry
- Limited enamine definition to exclude exceedingly delocalized instances
- Fix t_butyl_ether pattern to avoid matching t_butyl_alcohols
- Removed dependence of rigid bond count upon implicit or explicit hydrogens
- Cleaned-up table data for the select parameter
- Corrected problem that MIN_HALIDE_FRACTION and ALLOWED_ELEMENTS could not properly be removed from the filter file
- Fixed bug so NEWRULE lines with trailing white space are not ignored
- Fixed crash bug that occurred when a filter file was specified that was not a valid file
- Improved pKa value for barbituate-like compounds
- Clarified alpha-acyl carbon deprotonation
- Removed 1,2,3 triazole pKa rule
- Clarified tertiary amine rule
- Improved efficiency for applying pKa rules to multi-conformer molecules

8.6 FILTER 2.0.0 (October 2005)

This is the first official “2.0” version of FILTER. It includes many bug-fixes and feature expansions based on the long beta period. Some of these include higher level filtering flags such as Lipinski and other similar measures. This release also includes a pKa model that was missed in the beta release as well as non-pH normalization capabilities. This version makes exporting filtering data to other programs facile with both table and sdtg data export.

8.6.1 New features

- Neutral pKa model optionally applied to all molecules
- Support for molecular graph normalization
- Optional salt file can be used to remove salts from database
- Calculated properties can be attached to molecules as SD tag data
- All filtering data can be saved in tab-separated value format
- Filter on fractional halide weight
- Removal of known aggregator molecules
- Veber bioavailability filtering

- Lipinski violation filtering
- Abbott/Martin bioavailability filtering
- Pharmacopia bioavailability filtering
- Solubility class filtering
- Chiral center count filtering
- Continuous Medicinal Value

BIBLIOGRAPHY

- [Lipinski-1997] Lipinski, C.A., Lombardo, F., Dominy, B.W. and Feeney, P.J., **Experimental and Computational Approaches to Estimate Solubility and Permeability in Drug Discovery and Development Settings**, *Advanced Drug Delivery Reviews*, Vol. 23, pp. 3-25, **1997**
- [McGovern-2003] McGovern S.L., Helfand, B.T., Feng, B. and Shoichet, B.K., **A Specific Mechanism of Nonspecific Inhibition**, *Journal of Medicinal Chemistry*, Vol. 46, pp. 4265-4272, **2003**
- [Seidler-2003] Seidler J., McGovern, S.L., Doman, T.N. and Shoichet, B.K., **Identification and Prediction of Promiscuous Aggregating Inhibitors Among Known Drugs**, *Journal of Medicinal Chemistry*, Vol. 46, pp. 4477-4486, **2003**
- [Oprea-2000] Oprea, T., **Property Distribution of Drug-Related Chemical Databases**, *Journal of Computer-Aided Molecular Design*, Vol. 14, pp. 251-264, **2000**
- [Wang-1997] Wang, R., Ying, F., and Lai, L., **A New Atom-Additive Method for Calculating Partition Coefficients**, *Journal of Chemical Information and Computer Science*, Vol. 37, pp. 615-621, **1997**
- [Ertl-2000] Ertl, P., Rohde, B., and Selzer, P., **Fast Calculation of Molecular Polar Surface Area as a Sum of Fragment-Based Contributions and its Application to the Prediction of Drug Transport Properties**, *Journal of Medicinal Chemistry*, Vol. 43, pp. 3714-3717, **2000**
- [Clark-1999] Clark, D.E., **Rapid Calculation of Polar Molecular Surface Area and its Application to the Prediction of Transport Phenomena. 1. Prediction of Intestinal Absorption**, *Journal of Pharmaceutical Sciences*, Vol. 88, pp. 807-814, **1999**
- [Martin-2005] Martin, Y.C., **A Bioavailability Score**, *Journal of Medicinal Chemistry*, Vol. 48, pp. 3164-3170, **2005**
- [Veber-2002] Veber, D.F., Johnson, S.R., Cheng, H.Y., Smith, B.R., Ward, K.W. and Kopple, K.D., **Molecular Properties that Influence the Oral Bioavailability of Drug Candidates**, *Journal of Medicinal Chemistry*, Vol. 45, pp. 2615-2623, **2002**
- [Egan-2000] Egan, W.J., Merz, K.M. and Baldwin, J.J., **Prediction of Drug Absorption Using Multivariate Statistics**, *Journal of Medicinal Chemistry*, Vol. 43, pp. 3867-3877, **2000**
- [Rishton-2003] Rishton, G.M., **Nonleadlikeness and Leadlikeness in Biochemical Screening**, *Drug Discovery Today*, Vol. 8, pp. 86-96, **2003**
- [Yalkowsky-1980] Yalkowsky, S.H. and Valvani, S.C., **Solubility and Partitioning 1: Solubility of Nonelectrolytes in Water**, *Journal of Pharmaceutical Sciences*, Vol. 69, pp. 912-922, **1980**
- [MillsDean-1996] Mills, J.E.J and Dean, P.M., **Three-Dimensional Hydrogen-Bond Geometry and Probability Information from a Crystal Survey**, *Journal of Computer-Aided Molecular Design*, Vol. 10, pp. 607-622, **1996**
- [Jeffrey-1997] George A. Jeffrey, **An Introduction to Hydrogen Bonding**, *Oxford University Press*, **1997**

- [Wishart-2006] Wishart, D.S., **DrugBank: A Comprehensive Resource For In Silico Drug Discovery and Exploration**, *Nucleic Acids Research*, Vol. 34, pp. D668-672, **2006**
- [Lovering-2009] Frank Lovering, Jack Bikker, and Christine Humblet, **Escape from Flatland: Increasing Saturation as an Approach to Improving Clinical Success**, *Journal of Medicinal Chemistry*, 52 (21):6752-6756
- [Ritchie-2009] Timothy J. Ritchie and Simon J.F. Macdonald, **The impact of aromatic ring count on compound developability - are too many aromatic rings a liability in drug design?**, *Drug Discovery Today*, 14(21):1011

INDEX

Symbols

- dots
filter command line option, 15
- fail
filter command line option, 14
- filter
filter command line option, 13
- in
filter command line option, 13
- info
filter command line option, 14
- interval
filter command line option, 14
- log
filter command line option, 14
- newrule
filter command line option, 14
- normalize
filter command line option, 15
- out
filter command line option, 13
- param
filter command line option, 13
- pkanorm
filter command line option, 15
- prefix
filter command line option, 14
- salt
filter command line option, 15
- sdtag
filter command line option, 15
- select
filter command line option, 14
- table
filter command line option, 14
- tableFlag
filter command line option, 14
- typecheck
filter command line option, 14

A

APPNAME_OE_ARCH, 4

E

environment variable

APPNAME_OE_ARCH, 4

OE_ARCH, 4

OE_LICENSE, 3

PATH, 4, 5

F

filter command line option

-dots, 15

-fail, 14

-filter, 13

-in, 13

-info, 14

-interval, 14

-log, 14

-newrule, 14

-normalize, 15

-out, 13

-param, 13

-pkanorm, 15

-prefix, 14

-salt, 15

-sdtag, 15

-select, 14

-table, 14

-tableFlag, 14

-typecheck, 14

O

OE_ARCH, 4

OE_LICENSE, 3

P

PATH, 4, 5