Stripper

- Version 1.0.0 -
Last update: December 16, 2010

silicos

Table of contents

1.   Introduction

Copyright (C) 2005-2010 by Silicos NV

This file is part of the Open Babel project. For more information, see http://openbabel.sourceforge.net/

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published the Free Software Foundation version 2 of the License.

This program is distributed in the hope that it will be useful, but WITH-OUT ANY WARRANTY; without even the implied warranty of MERCHANT-ABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

STRIPPER is a program that extracts predefined scaffolds from organic molecules. The program is written on top of the open source C++ API of Open Babel.

The program comes with a number of predefined molecular scaffolds for extraction. These scaffolds include, amongst others:

- o   Molecular frameworks as originally described by Bemis and Murcko;[1]

- o   Molecular frameworks and the reduced molecular frameworks as described by Ansgar Schuffenhauer and coworkers;[2]

- o   Scaffold topologies as described by Sara Pollock and coworkers.[3]

All of these scaffolds are explained in the following sections.

STRIPPER is a command line-driven program that is instructed by means of command line options and a 'scaffold' file. It is by means of the 'scaffold' file that the user defines one or more of the predefined scaffolds to be extracted. Figure 1 describes the actual data flow:

---

[1] Bemis, G.W. & Murcko, M.A. (1996) 'The properties of known drugs. 1. Molecular frameworks', *J. Med. Chem.* **39**, 2887-2893.

[2] Schuffenhauer, A.; Ertl, P.; Roggo, S.; Wetzel, S.; Koch, M.A. & Waldmann, H. (2007) 'The scaffold tree - Visualization of the scaffold universe by hierarchical scaffold classification', *J. Chem. Inf. Model.* **47**, 47-58.

[3] Pollock, S.N.; Coutsias, E.A.; Wester, M.J. & Oprea, T.I. (2008) 'Scaffold topologies. 1. Exhaustive enumeration up to eight rings', *J. Chem. Inf. Model.* **48**, 1304-1310.

Figure 1. The input and output flow of molecules and associated files in the program STRIPPER. Command line options are also indicated.
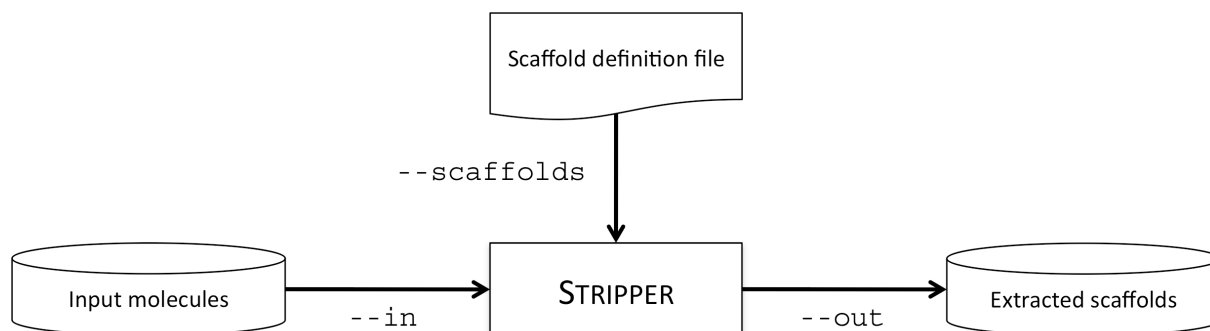
2.   Usage

2.1.   Command line interface

STRIPPER is run from the command line as follows:

```
> stripper [options]
```

Options can be required or optional. If one or more of the required options is missing, the program stops and displays an error message:

```
> stripper

ERROR: Command line option '--in' is missing.
       Please use -h or --help for more help.
```

Specifying the '-h' or '--help' option generates a help message:

```
> stripper -h

TASK:

  STRIPPER is a tool to extract predefined scaffolds from input molecules.

REQUIRED OPTIONS:

  --in <file>
    Specifies the file containing the input molecules. The format of the
    file is specified by the file extension. Gzipped files are also read.

OPTIONAL OPTIONS:

  --out <file>
    Specifies the file to which the extracted scaffolds are written.
    The file is a text file with on each row the original molecule
    and the generated scaffolds in smiles format. If not provided,
    then the output is written to stdout.

  --scaffolds <file>
    Specifies the file in which the required scaffolds have been defined.
    If not provided then all scaffolds are calculated.

  --noLog
    Suppresses the output of additional information to standard error.

  -h  --help

  -v  --version
```

Specifying the '-v' or '--version' option displays the release version of STRIPPER and the version of Open Babel that was used to write the program:

```
> stripper -v

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  STRIPPER v1.0.0 | Dec 10 2010 11:57:01

  -> GCC:       4.0.1 (Apple Inc. build 5490)
  -> Open Babel: 2.3.0

  Copyright (C) 2005-2010 by Silicos NV

  This file is part of the Open Babel project.
  For more information, see <http://openbabel.sourceforge.net/>

  This program is free software; you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation version 2 of the License.
```

```
   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++]
```

2.2.   Required command line options

2.2.1.   *--in*

The filename containing the input molecules is specified using the '`--in`' option:

```
> stripper --in filename.ext [other_options]
```

The input file should contain one or more molecules specified as a set of connection tables according specific molecular formats. The format of these connection tables is specified by the input filename extension ('.ext'). The allowed input formats are those that are supported by Open Babel. Zipped-formats are also allowed.

2.3.   Optional command line options

2.3.1.   *--out*

The name of the file to which the generated scaffolds are written is specified using the '`--out`' option:

```
> stripper --out filename [other_options]
```

The output file contains the generated scaffolds in a specific format. The first line is a header that contains the names of the generated scaffolds. These names are identical to the keywords used to defined the desired scaffolds (see section 3). The subsequent lines in the output file are composed of the generated scaffolds, with the data for each input molecule on a separate line. Scaffolds are represented in a SMILES notation. The content of a typical output file is shown here:

```
> cat scaffolds.txt


NAME     ORIGINAL   RINGS_WITH_LINKERS_1 MURCKO_1   OPREA_1
Mol1     c1ccccc1CO c1ccccc1   C1CCCCC1   C1CC1
Mol2     c1ncncc1   c1ncncc1   C1CCCCC1   C1CC1
```

If the '`--out`' option is not provided, then all output is written to standard output.

In the current version of STRIPPER, stereochemistry, either implicitly or explicitly defined in the molecule, is removed before the actual scaffold generation takes place.

If a molecule cannot be converted to a scaffold, for example in the case when the molecule contains no rings, then a '`-`' character is written to the output file.

2.3.2.   *--scaffolds*

This option specifies the file in which the keywords are contained that de-scribe the scaffolds that need to be generated. The name of the file is specified using the '`--scaffolds`' option:

```
> stripper --scaffolds filename [other_options]
```

A detailed overview of the specific scaffold keywords that are recognised is provided in Section **Error! Reference source not found.**. If the '`--scaffolds`' option is not provided, then by default all scaffold types are calculated.

2.3.3.   *--noLog*

Specifying this optional command line option suppresses the generation of additional information during the calculation process.

3.   Scaffold definitions

The scaffold definition file ('`--scaffold`') contains the names of all the specific scaffolds that one want to generate from each input molecule. The program recognizes only the keywords that are defined in this section.

Each line should contain only a single scaffold keyword. Blanc lines, or lines starting with '#' of '//', are ignored. These latter can be used as comment lines.

Keywords may be multiple repeated in a single scaffold definition file. In this case the specific scaffold is generated and written out more than once.

Scaffold keyword definitions are case-insensitive. Hence, 'Scaffold_1' is interpreted as being the same as 'SCAFFOLD_1' or 'scaffold_1'. However, these keywords are not the same as 'scaffold1'.

Each scaffold keyword is composed out of a main part and a number. The main part describes the type of scaffold, and the number defines a specific flavor of the specific scaffold type. The type and the number are concatenated by a underscore ('_') character.

3.1.   RINGS_WITH_LINKERS

The RINGS_WITH_LINKERS scaffold type is probably the most simple of all. In its most basic form, the scaffold is generated by iteratively removing all atoms having a valence of ≤1, until no such atoms remain. A couple of flavors exist of the RINGS_WITH_LINKERS scaffold which mainly differ in the way how exocyclic and exolinker double bonds are treated.

3.1.1.   RINGS_WITH_LINKERS_1

The RINGS_WITH_LINKERS_1 scaffold is generated by iteratively removing all atoms with a valence of ≤1. What remains is a scaffold composed solely of the rings and linkers of the original molecule (Figure 2). Atom elements and bond orders of the original molecule are kept unchanged in the resulting fragment.
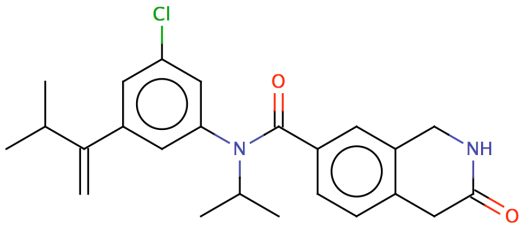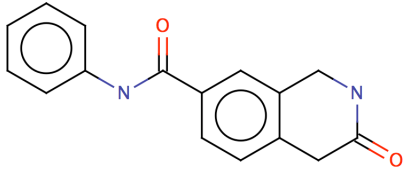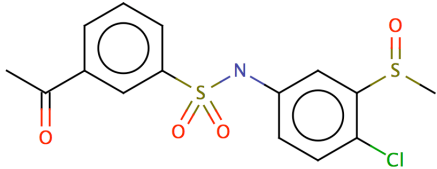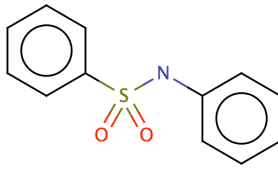
Figure 2. Example of the RINGS_WITH_LINKERS_1 scaffold type

Original molecule	**RINGS_WITH_LINKERS_1 scaffold**
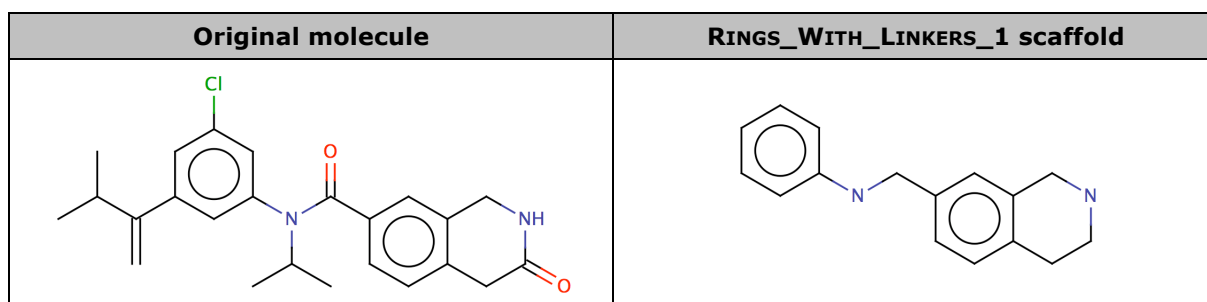
3.1.2. RINGS_WITH_LINKERS_2

The RINGS_WITH_LINKERS_1 scaffold differs from RINGS_WITH_LINKERS_1 by the fact that the exocyclic and exolinker =O functional groups in the original molecule are left intact in the generated scaffold (Figure 2).

Figure 3. Examples of the RINGS_WITH_LINKERS_2 scaffold type

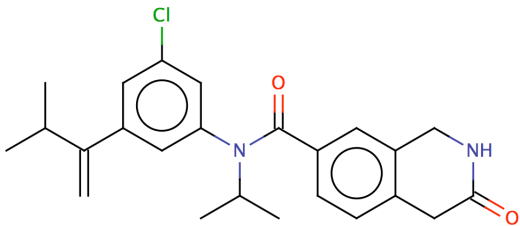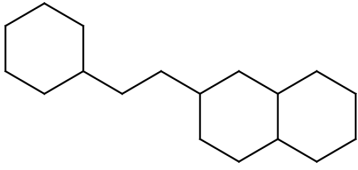| Original molecules	RINGS_WITH_LINKERS_2 scaffolds

3.2. MURCKO

The MURCKO scaffold type has been originally described by Bemis and Murcko.[1]

3.2.1. MURCKO_1

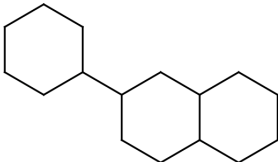An example of the MURCKO_1 scaffold as implemented in the current version of STRIPPER is given in Figure 4.
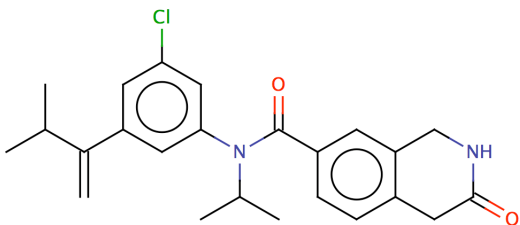
Figure 4. Example of the MURCKO_1 scaffold type

Original molecule	MURCKO_1 scaffold

3.2.2. MURCKO_2

The MURCKO_2 scaffold differs from the MURCKO_1 scaffold by the way linkers are represented. In the MURCKO_2 scaffold these linkers are all condensed to a single chain as shown in Figure 5.

Figure 5. Example of the MURCKO_2 scaffold type

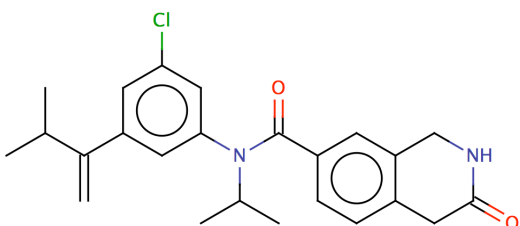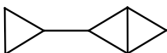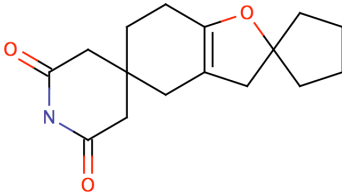| Original molecule	MURCKO_2 scaffold

3.3. OPREA

The OPREA scaffold type has been originally described by Pollock and co-workers.[3] Within the current version of Stripper, a number of flavours of this scaffold type have been implemented. In all cases, the minimal SMILES representation of the different scaffolds is returned.

3.3.1. OPREA_1

An example of the OPREA_1 scaffold type is given in Figure 7. From all implemented OPREA scaffolds in the current version of STRIPPER, this type is closest to the original description from the publication.
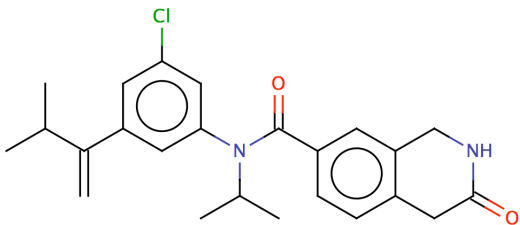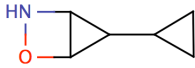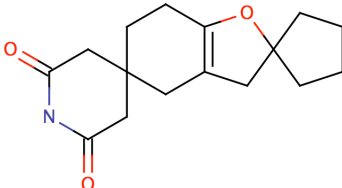
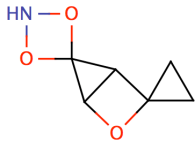*Figure 6. Examples of the OPREA_1 scaffold type*

Original molecule	OPREA_1 scaffold

3.3.2. OPREA_2

The OPREA_2 scaffold type differs from the OPREA_1 scaffold by the fact that all the existing hydrogen bonding acceptor and donor information of the ring atoms in the original molecule is kept intact and transferred into the resulting scaffold. Hydrogen bond acceptor atoms in the original molecule are represented by oxygen atoms in the final scaffold, while hydrogen bond donor atoms become represented by nitrogen atoms in the scaffold. Amide or sulfoxide ring oxygens are treated as being part of the ring and set to hydrogen bond acceptors. All other atoms are replaced by saturated carbon atoms and all bond orders are set to one. Neighbouring

ring atoms which are of the same type (hydrogen bond acceptor, hydrogen bond donor and carbon) are merged together into a single atom of the corresponding type. Linkers are replaced by a single bond (Figure 7).
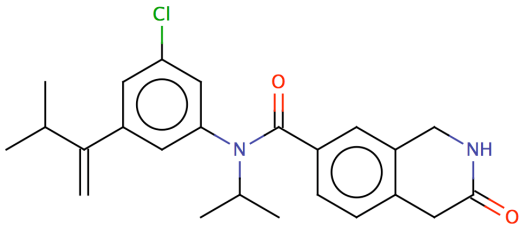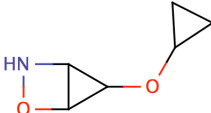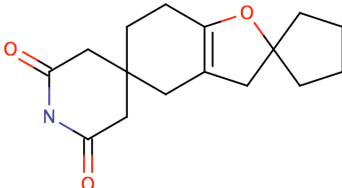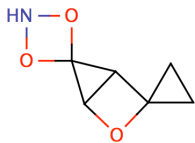
Figure 7. Examples of the OPREA_2 scaffold type

Original molecule	OPREA_2 scaffold

3.3.3. OPREA_3

The OPREA_3 scaffold type differs from the OPREA_2 scaffold by the fact that the hydrogen bonding acceptor and donor information of both the linker and ring atoms in the original molecule are kept and transferred into the resulting scaffold. Flanking atoms within rings and linkers which have the same type (hydrogen bond acceptor, hydrogen bond donor and carbon) are merged together into a single atom of the corresponding type (Figure 8).

Figure 8. Examples of the OPREA_3 scaffold type

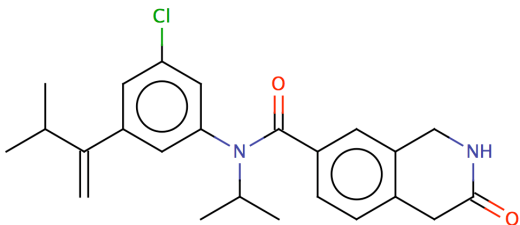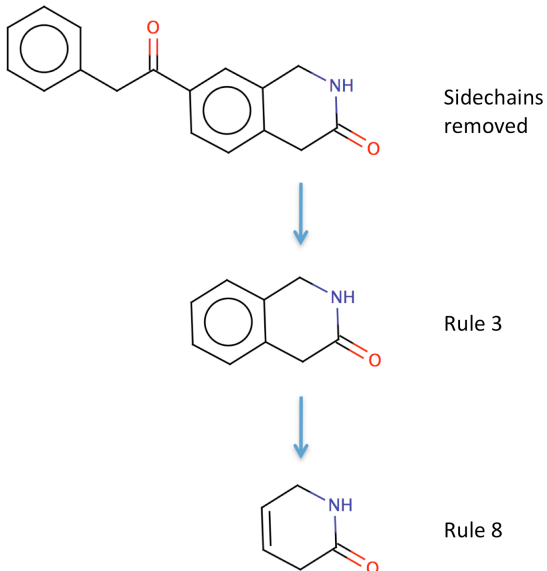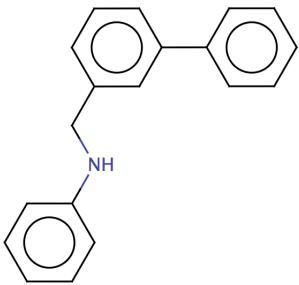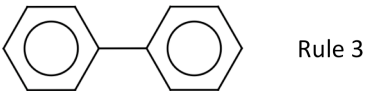| Original molecule	OPREA_3 scaffold

3.4.   SCHUFFENHAUER

The SCHUFFENHAUER scaffold type has been originally described by Ansgar Schuffenhauer and coworkers.[2]

3.4.1.   SCHUFFENHAUER_1

SCHUFFENHAUER_1 scaffolds are generated by removing in an iterative fashion all rings of the molecule until a single ring remains, unless it is not possible to tag the remaining rings in an unambigious way. In such cases where two or more choices are possible, the generated SCHUFFEN-HAUER_1 scaffold will consist of two or more rings. A couple of examples of the SCHUFFENHAUER_1 scaffold as implemented in the current version of STRIPPER is given in Figure 9.

Figure 9. Example of the SCHUFFENHAUER_1 scaffold type. The second example illustrates a typical case where none of the two remaining cycles can be uniquely be tagged for removal. In such case, the procedure stops at this stage.

Original molecule	SCHUFFENHAUER_1 scaffold

When additional logging is requested when running STRIPPER, the program will output each rule that is used in stripping each appropriate cycle. The rule numbering corresponds to the steps as described in the original Schuffenhauer publication.[2]

3.4.2. *SCHUFFENHAUER_2*

SCHUFFENHAUER_2 scaffolds are generated in the same way as SCHUFFEN-HAUER_1 scaffolds, but the procedure stops from the moment that two rings remain, unless it is not possible to tag the remaining rings in an un-ambigious way. In such cases, the generated SCHUFFENHAUER_2 scaffold will consist of three or more rings.

3.4.3. *SCHUFFENHAUER_3*

SCHUFFENHAUER_3 scaffolds are generated in the same way as SCHUFFEN-HAUER_1 scaffolds, but the procedure stops from the moment that three rings remain, unless it is not possible to tag the remaining rings in an un-ambigious way. In such cases, the generated SCHUFFENHAUER_3 scaffold will consist of four or more rings.

3.4.4. *SCHUFFENHAUER_4*

SCHUFFENHAUER_4 scaffolds are generated in the same way as SCHUFFEN-HAUER_1 scaffolds, but the procedure stops from the moment that four rings remain, unless it is not possible to tag the remaining rings in an un-ambigious way. In such cases, the generated SCHUFFENHAUER_4 scaffold will consist of five or more rings.

3.4.5. *SCHUFFENHAUER_5*

SCHUFFENHAUER_5 scaffolds are generated in the same way as SCHUFFEN-HAUER_1 scaffolds, but the procedure stops from the moment that five rings remain, unless it is not possible to tag the remaining rings in an un-ambigious way. In such cases, the generated SCHUFFENHAUER_5 scaffold will consist of six or more rings.

4.   Timings and number of scaffolds

The time needed for the calculation of the scaffolds depends strongly on the type of scaffold. In the following Table 1, a summary is given on the relative timings for each of the implemented scaffold types. These timings have been generated by performing the analysis on 100,000 randomly selected drug-like molecules. In addition to the calculation time for each scaffold, the table also gives the number of unique scaffolds that are generated from the 100,000 molecules in the input file.

Table 1. Summary of the relative timings to extract the different scaffolds on a set of 100,000 randomly selected drug-like molecules. In addition, the number of unique scaffolds for each calculation type is also given.

Scaffold	Relative time (compounds/sec)	Number of scaffolds (100,000 molecules)
RINGS_WITH_LINKERS_1	108	49,128
RINGS_WITH_LINKERS_2	82	52,620
MURCKO_1	106	16,124
MURCKO_2	108	5,349
OPREA_1	105	475
OPREA_2	94	7,328
OPREA_3	93	16,243
SCHUFFENHAUER_1	7	462
SCHUFFENHAUER_2	9	7,595
SCHUFFENHAUER_3	10	33,267
SCHUFFENHAUER_4	13	47,786
SCHUFFENHAUER_5	16	50,827

5. Revision history

5.1. Version 1.0

5.1.1. Version 1.0.0

This is the first release of STRIPPER.