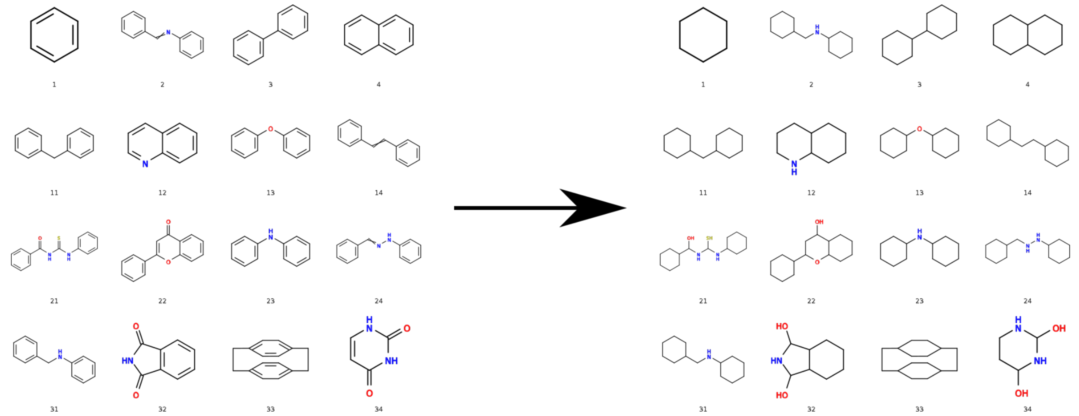
1 Background

The Bemis-Murcko scaffold[1] provided by `DataWarrior`[2] retains information about bond order and chirality. Sometimes, however, it suffices to retain only atom connectivity, like an assumption «there are only single bonds». Note `DataWarrior` equally offers the export of Bemis-Murcko skeleton, however this simplifies e.g. the scaffold about an imidazole into one of cyclopentane.



2 Typical use

The script runs from Python's CLI with a file listing SMILES to process as parameter. File `test_input.smi` (from sub-folder `test_data`) is an example:

```
python saturate_murcko_scaffolds.py [test_input.smi]
```

This generates `test_input_sat.smi` as permanent record; the addition of `_sat` is only a reminder of the performed saturation. The input file is preserved.

The file extension `.smi` of the input file is a suggestion, because it is frequently seen (e.g., around `OpenBabel`[3]). Internally, the script considers any character prior to the first period as part of the name of the input file. In favour of contemporary Python 3, earlier support for now legacy Python 2 was discontinued.

3 Example

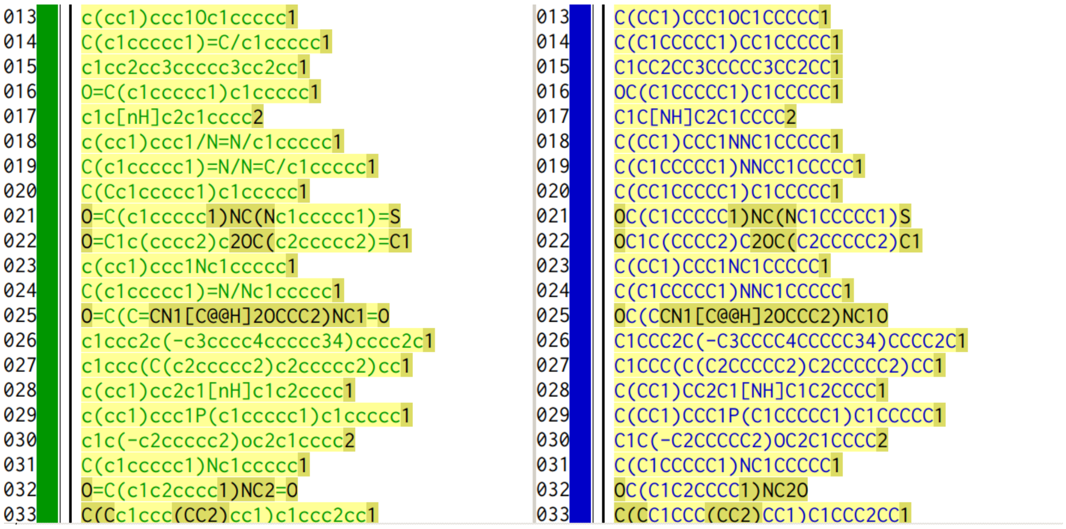
For a collection of organic materials, the Bemis-Murcko scaffolds were extracted with `DataWarrior` (then release 5.0.0 for Linux, January 2019) as listing `test_input.smi` including higher bond

[1] Bemis, G. W.; Murcko, M. A. *J. Med. Chem.* 1996, **39**, 2887-2893, doi 10.1021/jm9602928.

[2] Sander, T.; Freyss, J.; von Korff, M.; Rufener, C.; *J. Chem. Inf. Model.* 2015, **55**, 460-473, doi 10.1021/ci500588j. The program, (c) 2002–2022 by Idorsia Pharmaceuticals Ltd., is freely available under `http://www.openmolecules.org`. For the source code (GPLv3), see `https://github.com/thsa/datawarrior`.

[3] www.openbabel.org. The script initially was developed for and tested with OpenBabel (release 2.4.1; Nov 12, 2018) and Python 2.7.17 provided by Linux Xubuntu 18.04.2 LTS. Meanwhile, support for legacy Python 2 was dropped in favour of contemporary Python 3.

orders (see folder `test_data`). The effect of the «artificial saturation» is easy to recognize while comparing the scaffold lists (fig. 1) in a difference view of the two `.smi` files.



```
013 c(cc1)ccc1Oc1ccccc1                 013 C(CC1)CCC1OC1CCCCC1
014 C(c1ccccc1)=C/c1ccccc1              014 C(C1CCCCC1)CC1CCCCC1
015 c1cc2cc3ccccc3cc2cc1                015 C1CC2CC3CCCCC3CC2CC1
016 O=C(c1ccccc1)c1ccccc1               016 OC(C1CCCCC1)C1CCCCC1
017 c1c[nH]c2c1cccc2                    017 C1C[NH]C2C1CCCC2
018 c(cc1)ccc1/N=N/c1ccccc1             018 C(CC1)CCC1NNC1CCCCC1
019 C(c1ccccc1)=N/N=C/c1ccccc1          019 C(C1CCCCC1)NNCC1CCCCC1
020 C(Cc1ccccc1)c1ccccc1                020 C(CC1CCCCC1)C1CCCCC1
021 O=C(c1ccccc1)NC(Nc1ccccc1)=S        021 OC(C1CCCCC1)NC(NC1CCCCC1)S
022 O=C1c(cccc2)c2OC(c2ccccc2)=C1       022 OC1C(CCCC2)C2OC(C2CCCCC2)C1
023 c(cc1)ccc1Nc1ccccc1                 023 C(CC1)CCC1NC1CCCCC1
024 C(c1ccccc1)=N/Nc1ccccc1             024 C(C1CCCCC1)NNC1CCCCC1
025 O=C(C=CN1[C@@H]2OCCC2)NC1=O         025 OC(CCN1[C@@H]2OCCC2)NC1O
026 c1ccc2c(-c3cccc4ccccc34)cccc2c1     026 C1CCC2C(-C3CCCC4CCCCC34)CCCC2C1
027 c1ccc(C(c2ccccc2)c2ccccc2)cc1       027 C1CCC(C(C2CCCCC2)C2CCCCC2)CC1
028 c(cc1)cc2c1[nH]c1c2cccc1            028 C(CC1)CC2C1[NH]C1C2CCCC1
029 c(cc1)ccc1P(c1ccccc1)c1ccccc1       029 C(CC1)CCC1P(C1CCCCC1)C1CCCCC1
030 c1c(-c2ccccc2)oc2c1cccc2            030 C1C(-C2CCCCC2)OC2C1CCCC2
031 C(c1ccccc1)Nc1ccccc1                031 C(C1CCCCC1)NC1CCCCC1
032 O=C(c1c2cccc1)NC2=O                 032 OC(C1C2CCCC1)NC2O
033 C(Cc1ccc(CC2)cc1)c1ccc2cc1          033 C(CC1CCC(CC2)CC1)C1CCC2CC1
```

Figure 1: Difference view of the SMILES strings of a Murcko scaffold *prior* (left hand column) and *after* an «artificial saturation» (right hand column). The processing affects explicit bond order indicators, e.g. double bond (equality sign, e.g., line #14), triple bond bond (octohorpe, not shown); or about implicit aromatization (lower case ⟶ upper case) for atoms of carbon, nitrogen, oxygen (depicted); or phosphorus, sulfur (not depicted). Stereochemical indicators about double bonds will be removed (e.g., slashes in lines #18 and #19). Descriptors of stereogenic centers (@-signs, e.g., line #25) and charges (not shown) are copied verbatim.

Subsequently, `OpenBabel`[3] was used to illustrate the work performed. While eventually automated (cf. script `test_series.py`, deposit in folder `test_data`), instructions issued to `OpenBabel` on the command line followed the pattern of

```
obabel -ismi test_input.smi -O test_input_color.svg -xc10 -xr12 -xl --addinindex
```

to generate a `.svg` file (vector representation), or

```
obabel -ismi test_input_sat.smi -O test_input_sat_color.png -xc10 -xr12 -xl
↪  --addinindex -xp 3000
```

to generate a bitmap `.png` with structure formulae depicted in a grid of 10 columns by 12 rows.

It is remarkable how well `OpenBabel`'s displays the molecular structures with advanced motifs. In addition to those shown in the first illustration of this guide, see sub-folder `test_data` for a more extensive survey (e.g., the scaffold of cyclophane [entry #33], sparteine [#38], or adamantane [#50]).

4 Known peculiarities

The script neither removes, nor newly assigns SMILES descriptors about the absolute configuration of stereogenic centers (`@`). Thus, the «reduction» of double bonds e.g., ketones to secondary alcohols may yield new stereogenic centers with an explicit description of configuration.

For a selection of elements (C, N, O, P, S), the script recognizes their use in aromatic systems (e.g., as `c1ccncc1` in pyridine) with an implicit bond order. To offer a "saturation", these characters returned as upper case characters which transforms e.g., pyridine into piperidine (`C1CCNCC1`). The script provides additional "saturation" by dropping explicit information related to double and triple bonds which SMILES encode (`=`, `#` regarding bond order; `/` (forward slash), `\` (backward slash) regarding (*cis*)-(*trans*) relationship around double bonds).

The capitalization of characters however is not applied to atoms enclosed in square brackets. This shall prevent e.g., the transformation of `[sn]` which were a valid description of tin (Sn) into `[SN]`. Instead, the pair of square brackets, including their content enclosed, is copied verbatim into the newly written SMILES string about the reduced compound, which – in addition to the element symbol(s) – equally accounts for the stereochemical descriptor (like in `[S@]`) and charges (like in `[Fe3+]`).

So far, the underlying algorithm accepts at maximum one pair of square brackets per SMILES string only. Instances like imidazole (`c1ncc[nH]1`) are known to resolve as imidazolidine with `C1NCC[nH]1` instead of the anticipated alternative `C1NCCN1`.

The script will not actively alter a charge assigned to an atom. If present (e.g., quaternary ammonium, carboxylate), this information will be carried over to the newly written SMILES string. Given the reduction of bond orders, depending on the substrate submitted and context, this approach may be sensible (e.g., about N in cetyltrimethylammonium bromide), or not. Other libraries than the current script (e.g., RDKit[4]) might offer help to sanitize the processed SMILES strings.

If the input SMILES string describes more than exactly one molecule by the concatenating `"."` (period character), this special sign equally is the newly written SMILES string. This permits working with SMILES about e.g., co-crystals, like about 1,4-benzoquinone and hydroquinone, `C1=CC(=O)C=CC1=O.c1cc(ccc1O)O` resolved as `C1CC(O)CCC1O.C1CC(CCC1O)O`.

5 License

Norwid Behrnd, 2019–22, GPLv3.

[4] For an overview about the freely available RDKit library, see www.rdkit.org. An introduction into the topic of «molecular sanitization» is provided in the section of this very title in the on-line RDKit Book.