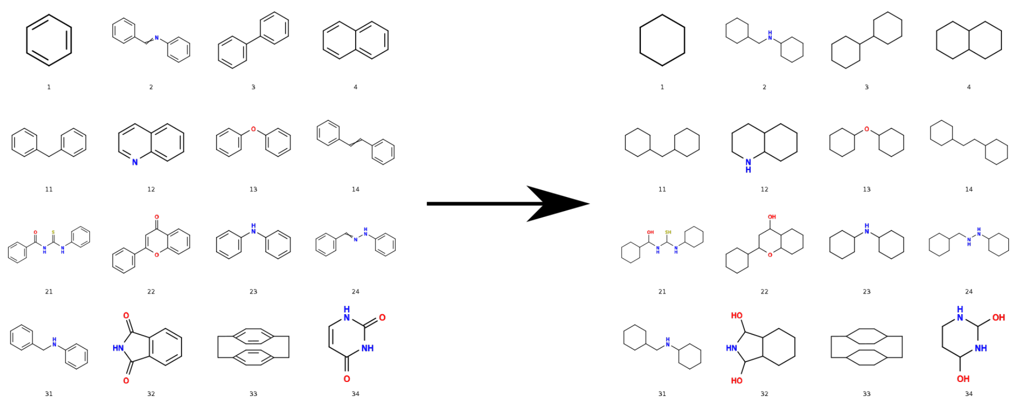
1 Background

The Bemis-Murcko scaffold[1] provided by `DataWarrior`[2] retains information about bond order and chirality. Sometimes, however, it suffices to retain only which atoms are connected with each other, equivalent to the 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 in the CLI of Python. File `listing_file.txt` contains the SMILES to work with:

```
python saturate_MurckoScaffolds.py [listing_file.txt]
```

This generates `saturated_Murcko_scaffold_sat.smi` as permanent record, which shares the file name with the input file *with the addition of* `_sat` *(like saturated).* Regardless of the three-character file extension of the input, the plain ASCII-list of SMILES is recorded in a file of extension `.smi`, accepted recognized e.g., by `openbabel` to contain this specific information.

While intended to work with the current branch of Python 3.6+, the script equally works well with the deprecated, legacy branch of Python 2, e.g., Python 2.7.16.

3 Example

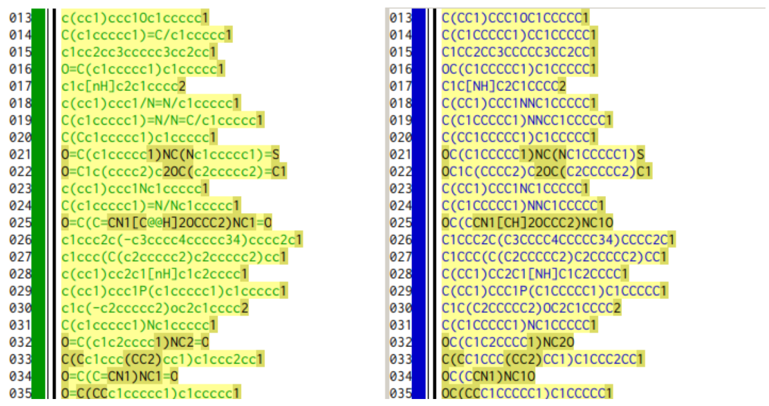
For a collection of organic materials, the Bemis-Murcko scaffolds was extracted with `DataWarrior` (then release 5.0.0 for Linux, January 2019) as listing `test_input.smi.txt` including higher bond orders (see folder `test_data`). The «artificial saturation» was triggered by

```
python saturate_MurckoScaffolds.py test_input.smi
```

to yield `test_input_sat.smi`. The effect of this operation is easy to recognize while comparing the scaffold lists (fig. 1) in a difference view of the two `.smi` files.

[1] Bemis GW, Murcko MA *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) 2002–2019 by Idorsia Pharmaceuticals Ltd., is freely available under `http://www.openmolecules.org`.



```
013 c(cc1)ccc1Oc1ccccc1
014 C(c1ccccc1)=C/c1ccccc1
015 c1cc2cc3ccccc3cc2cc1
016 O=C(c1ccccc1)c1ccccc1
017 c1c[nH]c2c1cccc2
018 c(cc1)ccc1/N=N/c1ccccc1
019 C(c1ccccc1)=N/N=C/c1ccccc1
020 C(Cc1ccccc1)c1ccccc1
021 O=C(c1ccccc1)NC(Nc1ccccc1)=S
022 O=C1c(cccc2)c2OC(c2ccccc2)=C1
023 c(cc1)ccc1Nc1ccccc1
024 C(c1ccccc1)=N/Nc1ccccc1
025 O=C(C=CN1[C@@H]2OCCC2)NC1=O
026 c1ccc2c(-c3cccc4ccccc34)cccc2c1
027 c1ccc(C(c2ccccc2)c2ccccc2)cc1
028 c(cc1)cc2c1[nH]c1c2cccc1
029 c(cc1)ccc1P(c1ccccc1)c1ccccc1
030 c1c(-c2ccccc2)oc2c1cccc2
031 C(c1ccccc1)Nc1ccccc1
032 O=C(c1c2cccc1)NC2=O
033 C(Cc1ccc(CC2)cc1)c1ccc2cc1
034 O=C(C=CN1)NC1=O
035 O=C(CCc1ccccc1)c1ccccc1
```

```
013 C(CC1)CCC1OC1CCCCC1
014 C(C1CCCCC1)CC1CCCCC1
015 C1CC2CC3CCCCC3CC2CC1
016 OC(C1CCCCC1)C1CCCCC1
017 C1C[NH]C2C1CCCC2
018 C(CC1)CCC1NNC1CCCCC1
019 C(C1CCCCC1)NNCC1CCCCC1
020 C(CC1CCCCC1)C1CCCCC1
021 OC(C1CCCCC1)NC(NC1CCCCC1)S
022 OC1C(CCCC2)C2OC(C2CCCCC2)C1
023 C(CC1)CCC1NC1CCCCC1
024 C(C1CCCCC1)NNC1CCCCC1
025 OC(CCN1[CH]2OCCC2)NC1O
026 C1CCC2C(C3CCCC4CCCCC34)CCCC2C1
027 C1CCC(C(C2CCCCC2)C2CCCCC2)CC1
028 C(CC1)CC2C1[NH]C1C2CCCC1
029 C(CC1)CCC1P(C1CCCCC1)C1CCCCC1
030 C1C(C2CCCCC2)OC2C1CCCC2
031 C(C1CCCCC1)NC1CCCCC1
032 OC(C1C2CCCC1)NC2O
033 C(CC1CCC(CC2)CC1)C1CCC2CC1
034 OC(CCN1)NC1O
035 OC(CCC1CCCCC1)C1CCCCC1
```

Figure 1: Difference view of the SMILES strings of a Murcko scaffold *prior* (left hand column) and *after* an «artificial saturation» (right hand column). Note the removal of explicit bond order indicators, e.g. double bond (equality sign), triple bond bond (octohorpe), or about implicit aromatization (lower case → upper case for atoms of carbon, nitrogen, oxygen (depicted); or sulfur (not depicted). At the same time, stereochemical indicators are removed, too (e.g., slashes in lines #018 and #019, @-signs line #025).

Subsequently, `openbabel`[3] was used to illustrate the work performed. While eventually automated (cf. script `test_series.py`, deposit in folder `test_data`), instructions delivered to `openbabel` on the command line follow 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`. The instructions share the definition of data input (`-ismi`) – eventually *a list of SMILES*, rather than only one SMILES string – and the definition of the output format based on the extension of the file to be written (`.svg`, or `.png`). Both put the entries into an array of 10 columns and 12 rows (`-xc`, `xr`) and label them in the consecution of their appearance in the input files (`--addinindex`) as inner index. Contrasting to the vector output, in case of `.png`, an explicit scaling by a factor of 10 (`-xp` parameter) was required to yield an intelligible illustration.

It is remarkable how well `openbabel`'s algorithm displays the molecular structures with advanced motifs (e.g., the scaffold of cyclophane [entry #33 in the synopses, `test_data` folder], sparteine [#38], or adamantane [#50]), some equally depicted in the first illustration of this guide.

4 License

[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. It equally works with Python 3.8.2 (recommended current branch, released April 1, 2020) and openbabel (release 3.0.0 by April 6, 2020) as provided in Debian 10.