EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

BIO-4372
CHEMINFORMATICS
Winter Semester 2022-23

Dr. Philipp Thiel
Institute for Bioinformatics
and Medical Informatics

Assignment Sheet 4

Submission deadline: November 21, 2022, 9 a.m.

A4.1: SMARTS (to be solved by hand) [5 points]

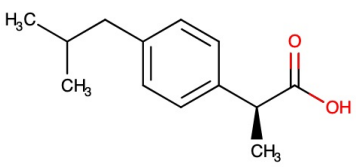
SMARTS are a powerful technique for specifying structural patterns. It can be used to filter compound databases in order to remove molecules containing substructures with undesirable properties such as toxicity or to search for molecules of particular interest.

1. Draw the 2D structural patterns described by the following SMARTS:
   - `a[C&H1&R0](CC(=O)C)[r6&!a]O`
   - `c1([N+](=O)O)ccc(A)cc1`
   - `C1=Cc2c([c&H1][c&H1][c&H1][c&H1]2)OC1`
   - `O=[!N]AA[#8]`
2. Now, please draw a minimal and well defined molecule that contains all these substructures you just have drawn. **Minimal** means that the target molecular graph has the smallest possible order and **well defined** means that no dangling primitives are left in the final molecule. Hints: (1) you do not need to increase the number of atoms over the ones already explicitly specified by the SMARTS patterns. (2) always consider at which positions you cannot extend the structure(s).

   Sounds like a puzzle? It is a puzzle!

A4.2: Morgan's Algorithm (to be solved by hand) [5 points]

As you already know, SMILES in general are not unique but canonical SMILES try to solve that problem by using a pre-generated atom ordering. One of the most basic methods for canonical atom numbering is Morgan's algorithm. Here you can practice the application of this algorithm as described in the lecture on actual molecules. Please generate the canonical atom enumeration for the following molecule:



Draw each iteration step as a labeled molecular graph. Feel free to resolve ambiguities as you like but document your procedure.

A4.3: Implementation of Morgan's Algorithm [10 points]

Now that you have collected some experience with Morgan's algorithm, please implement it in Python using RDKit and canonize your SMILES implementation from A3.2 or take the provided solution as starting point.

1. Despite being cyclic, use the molecule in `aspirin.sdf` to implement the required Morgan steps and make sure your algorithm works correctly. The solution for this molecule can be found in the slides.

   Requirement: implement for debugging a switch to optionally print the mapping of RDKit atom index to the current EC number or canonical label after every iteration of the relaxation and enumeration phase, respectively. Use one line per atom and separate iterations by a blank line. The corresponding functions in the provided solution already have a boolean argument that can be used to switch to debug mode.

2. Now use `smiles_01.sdf` to test and possibly improve your implementation.
3. Finally, use the numbering you generated to produce canonical SMILES for `smiles_01.sdf`. Thus, if you have not already an ordering-guided SMILES generation, this is the point to do so!

Submission

1. Your solutions to A4.1 and A4.2.
2. Your properly documented Python implementation.

Please bundle the files in a tar-archive to simplify uploading to ILIAS.

▷ Please use Slack to discuss problems in the first place
▷ If you have confidential questions, don't hesitate to drop by or write an e-mail