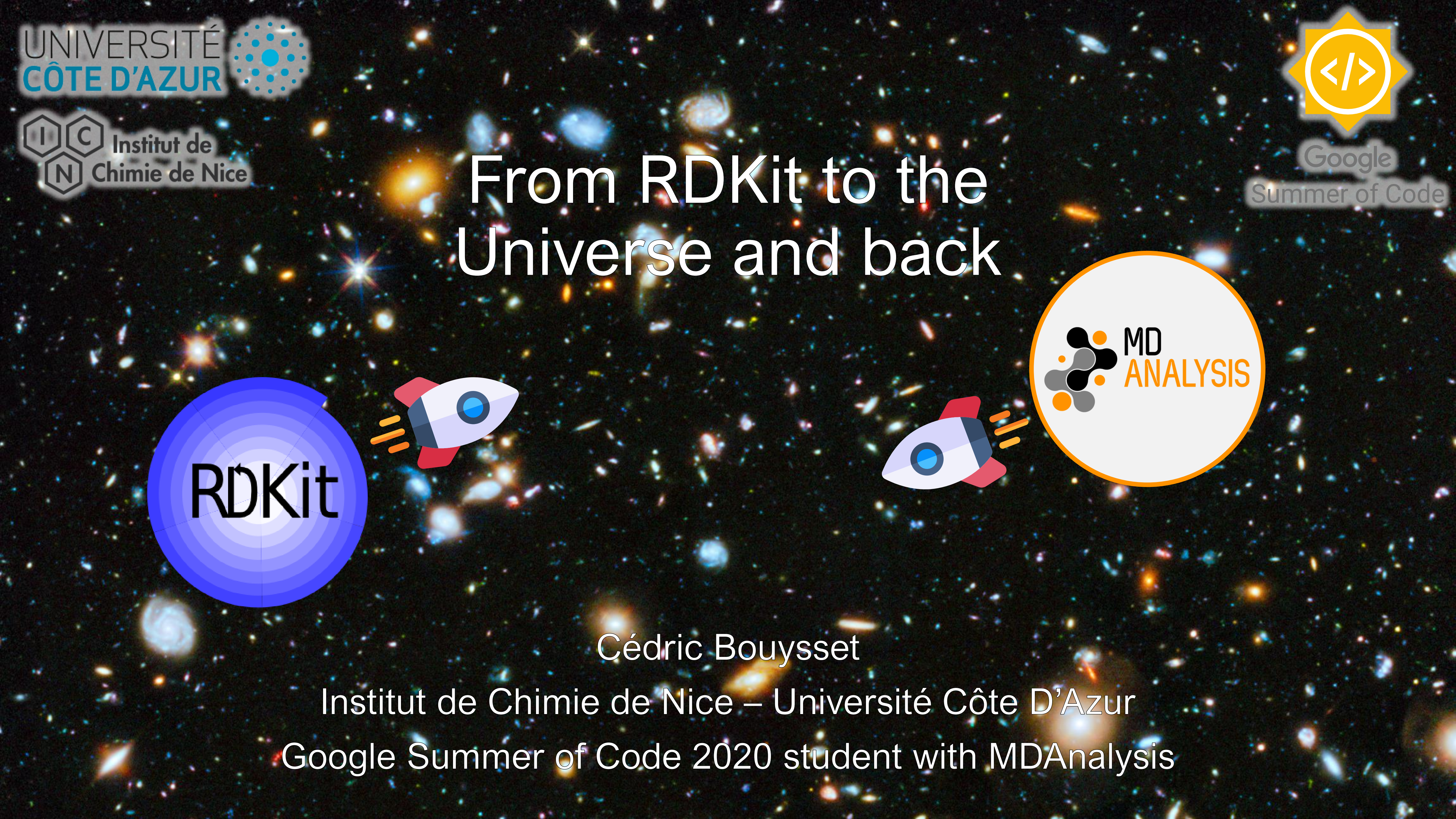


From RDKit to the Universe and back

Cédric Bouysset

Institut de Chimie de Nice – Université Côte D'Azur

Google Summer of Code 2020 student with MDAnalysis

MD ANALYSIS

- Analyze trajectories from molecular dynamics simulations
- Support for reading most computational chemistry file formats (Gromacs, Amber, NAMD, XYZ, PDBQT…*etc.*)
- Trajectory manipulation (alignment, centering…*etc.)*
- Extensive atom selection commands
- Core objects: Universe and AtomGroup

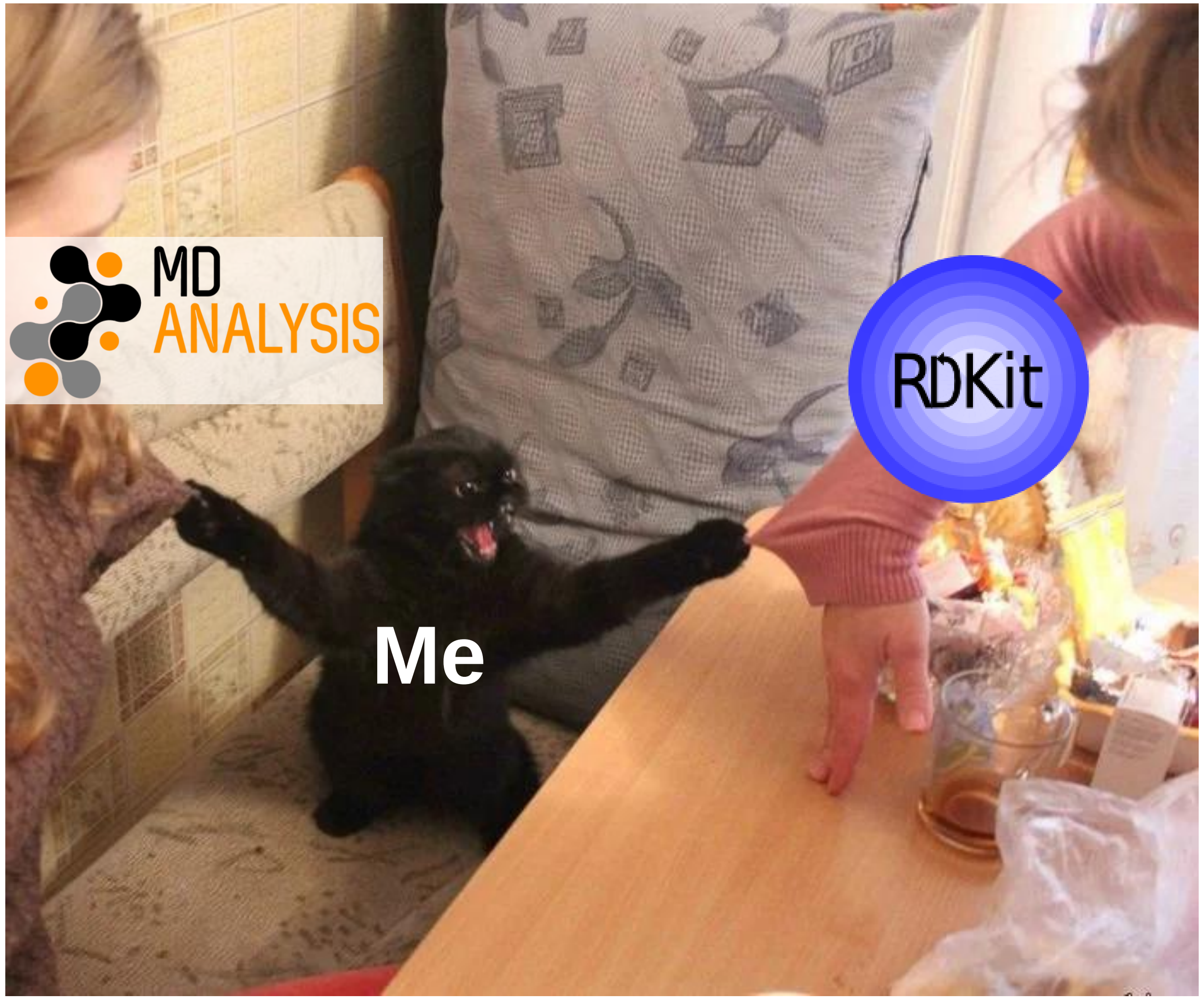
https://www.mdanalysis.org/

Motivation

	Knows chemistry	Reads MD trajectories
RDKit	✓	✗
MDAnalysis	✗	✓

→ Both packages would complement each other

Summary of the project



From RDKit to the Universe

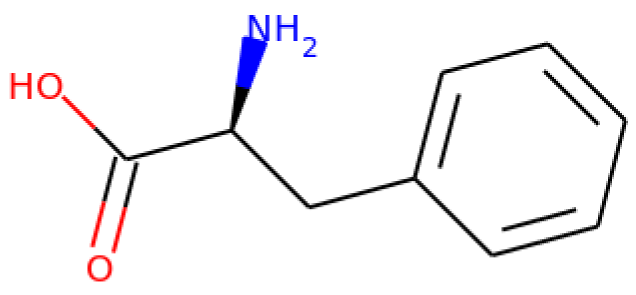
RDKit $\rightarrow$ MD ANALYSIS

From RDKit to the Universe

- Brings indirect support for SMILES, SDF and more
- Atom properties are kept *(residues, chains, charges…etc.)*
- Introduces aromaticity to MDAnalysis

```
In [6]:  u = mda.Universe.from_smiles("CCO", numConfs=10)
          u.trajectory

Out[6]:  <RDKitReader with 10 frames of 9 atoms>
```

```
In [7]:  mol = Chem.MolFromSequence("F")
          mol

Out[7]:
```



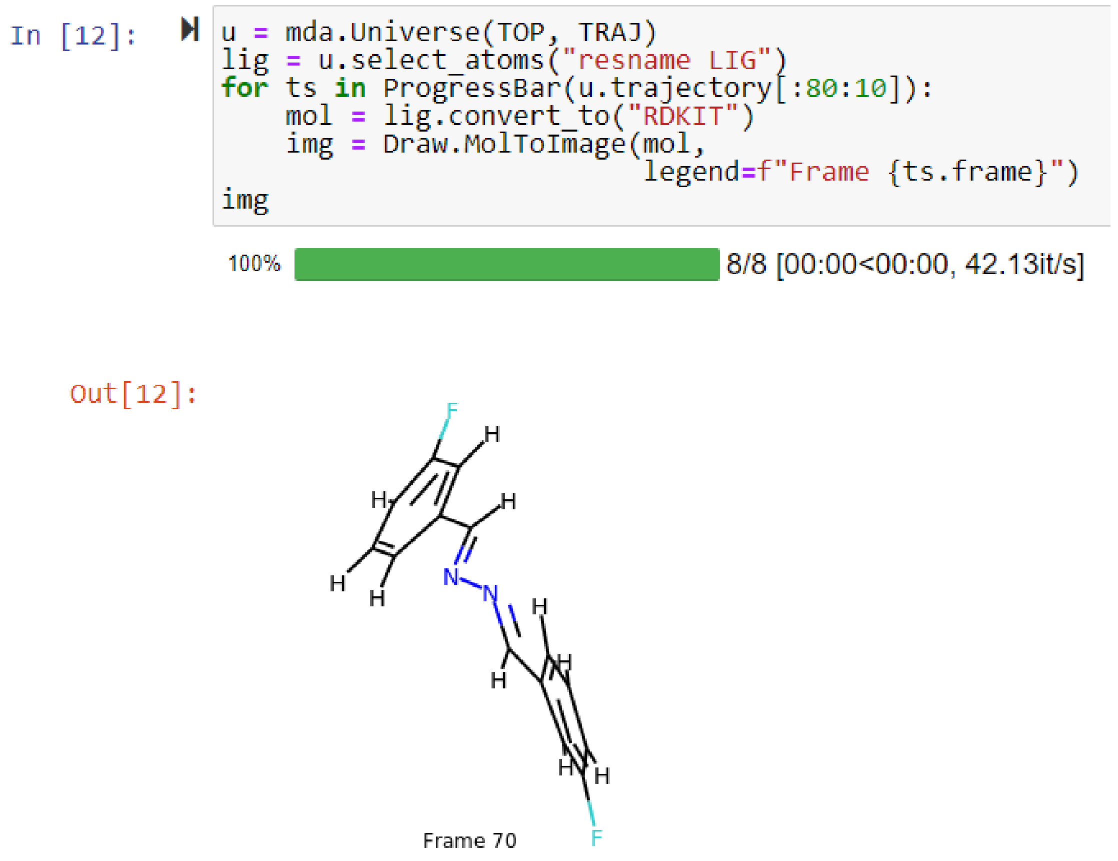
```
In [8]:  u = mda.Universe(mol)
          u.select_atoms("aromatic")

Out[8]:  <AtomGroup with 6 atoms>
```

From the Universe to RDKit

MD ANALYSIS $\rightarrow$ RDKit

From the Universe to RDKit

- Brings indirect support for MD trajectories, PDBQT and more
- AtomGroup properties (residues, chains…*etc.*) are kept



```
In [12]:  u = mda.Universe(TOP, TRAJ)
          lig = u.select_atoms("resname LIG")
          for ts in ProgressBar(u.trajectory[:80:10]):
              mol = lig.convert_to("RDKIT")
              img = Draw.MolToImage(mol,
                                    legend=f"Frame {ts.frame}")
          img

          100% ████████████████ 8/8 [00:00<00:00, 42.13it/s]

Out[12]:
```

Requirements

- Elements*
- Bonds*
- Explicit hydrogens

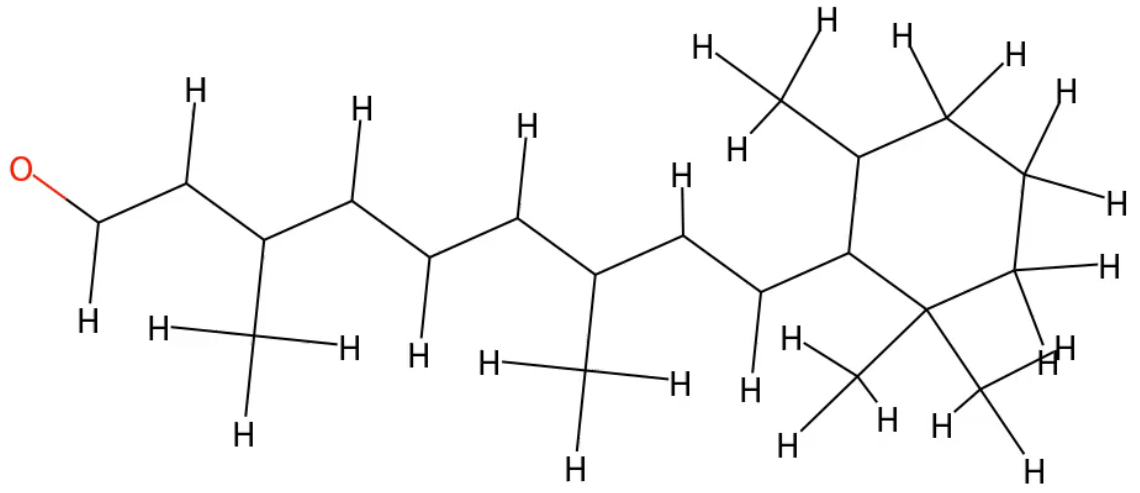
→Infers bond orders and formal charges from there

* can be guessed by MDAnalysis

Inferring bond orders and charges

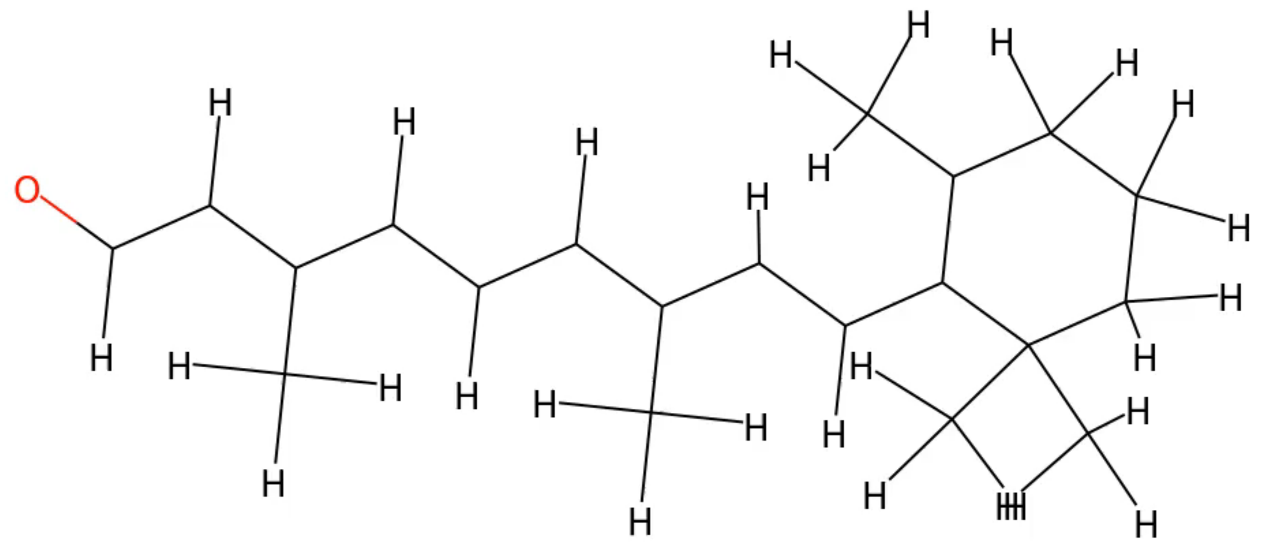
Pseudo code:

```
For each atom:
    Compare expected vs current valence
    If all valence electrons are paired:
        Continue to next atom
    Else if atom exceeds expected valence:
        Assign (+) charge
    Else:
        If a neighbor can accept a double/triple bond:
            Increase bond order
        Else:
            Assign (-) charge
```



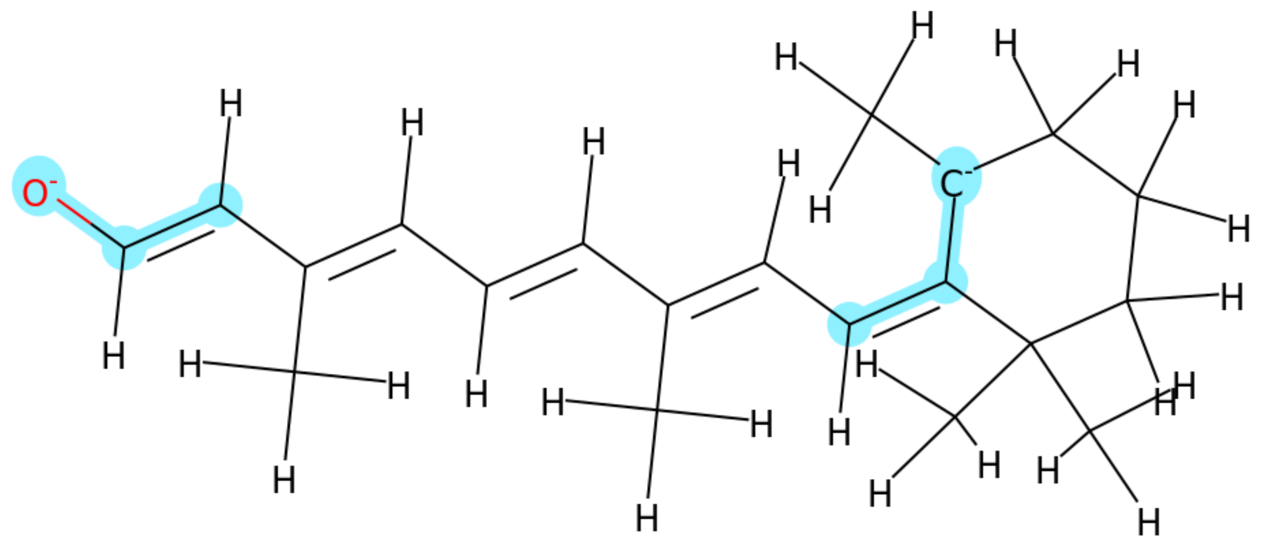
Inferring bond orders and charges

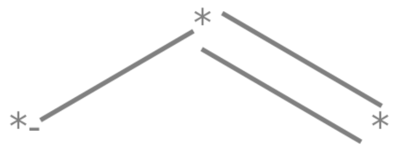
Problem 1: order dependency



Inferring bond orders and charges

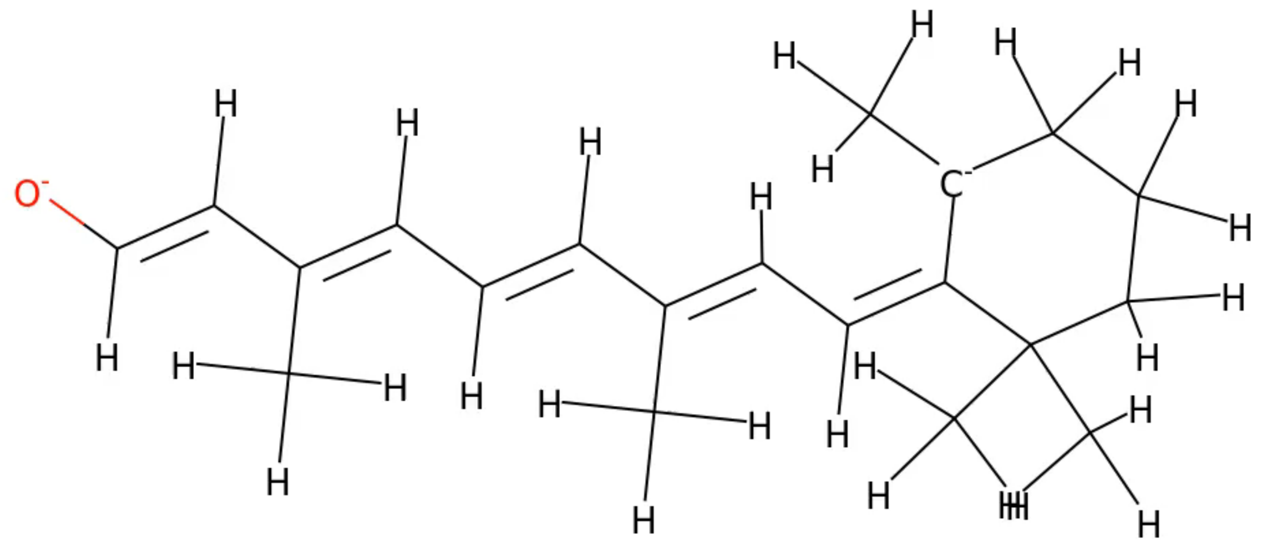
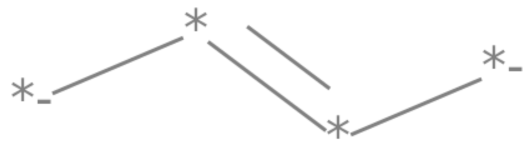
Problem 1: order dependency





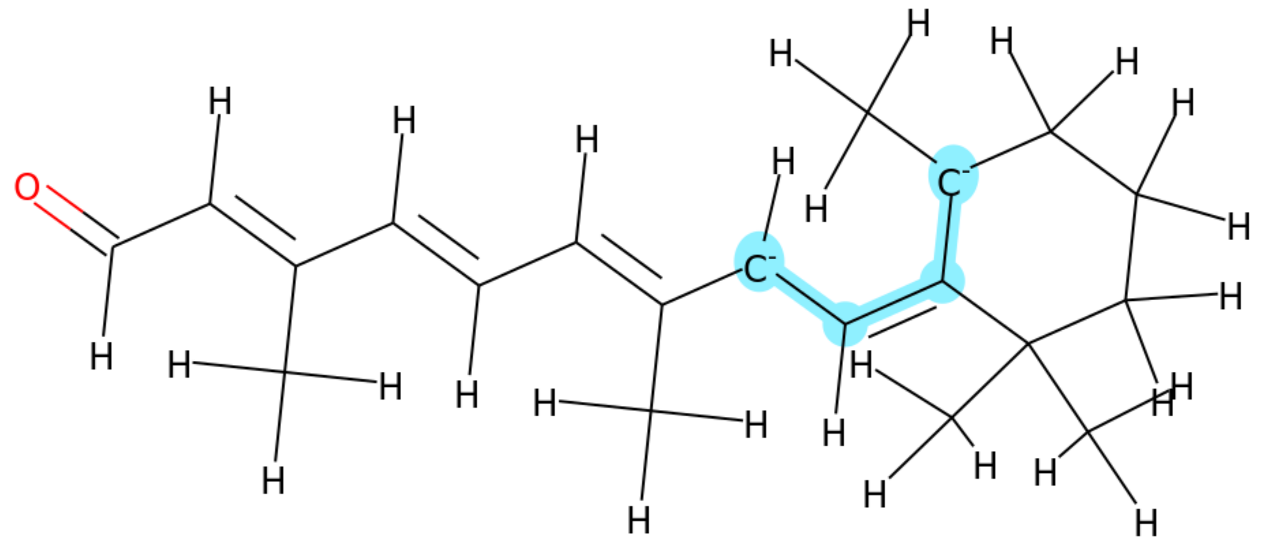
Inferring bond orders and charges

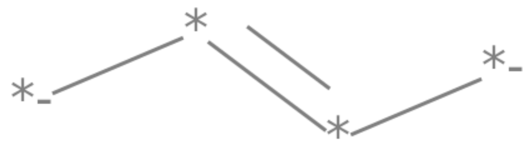
Problem 1: order dependency

Inferring bond orders and charges

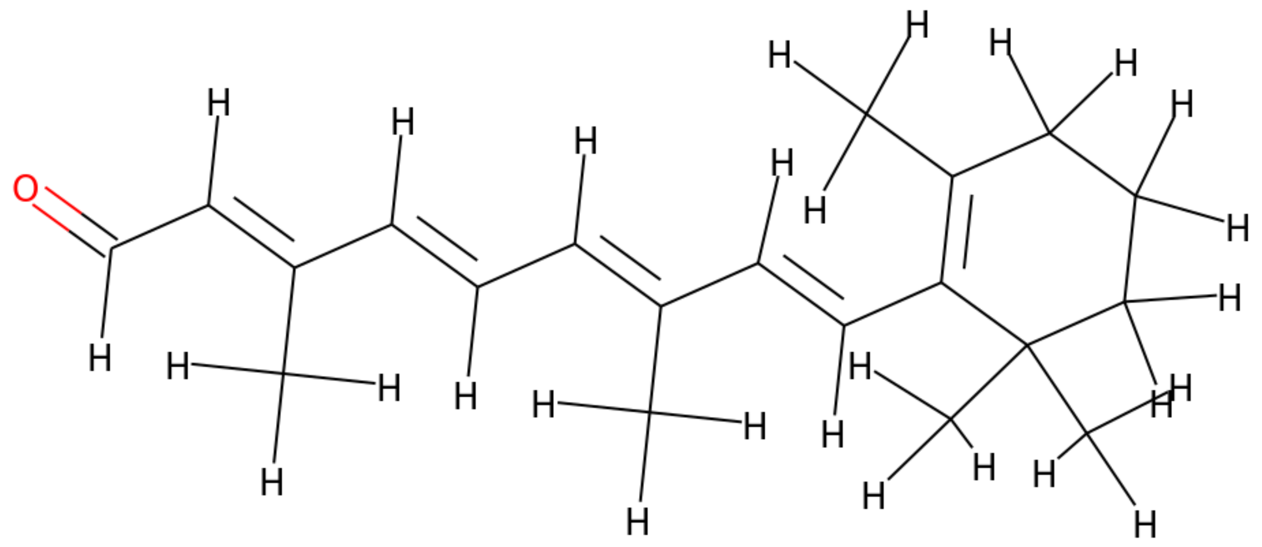
Problem 1: order dependency





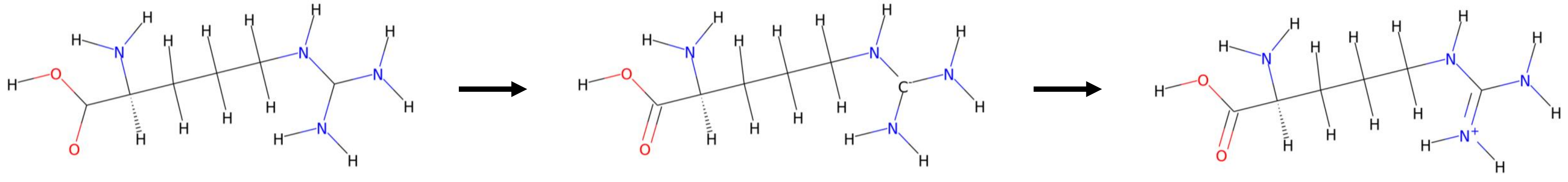
Inferring bond orders and charges

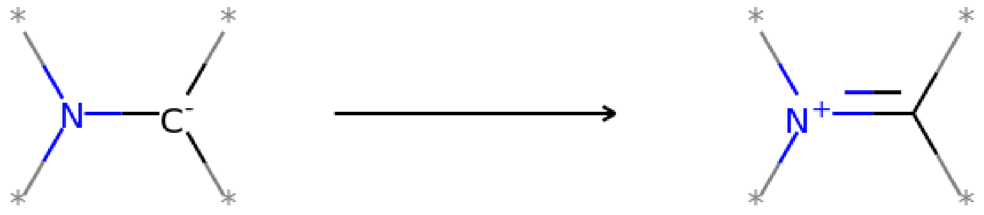
Problem 1: order dependency



Inferring bond orders and charges

Problem 2: logical pitfalls

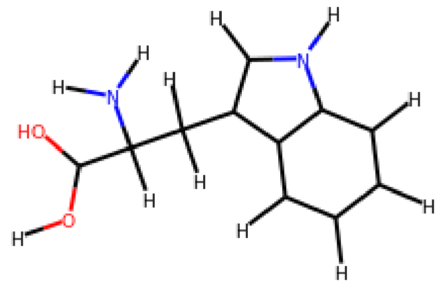




Fallback option

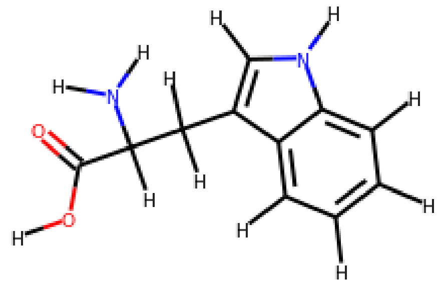
```
In [25]:  mol = u.atoms.convert_to("RDKIT", NoImplicit=False)
          mol
```

Out[25]:



```
In [26]:  ref = Chem.MolFromSequence("W")
          ref = Chem.AddHs(ref)
          AllChem.AssignBondOrdersFromTemplate(ref, mol)
```

Out[26]:



Leveraging the interoperability

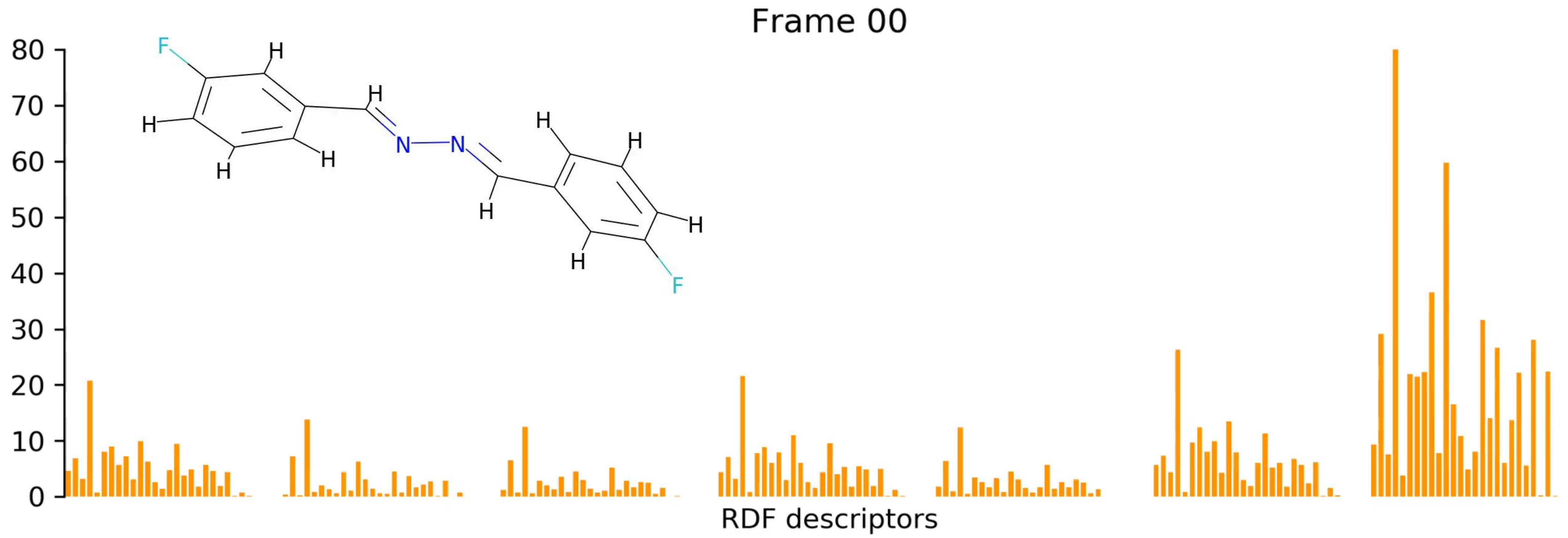
MD ANALYSIS

RDKit

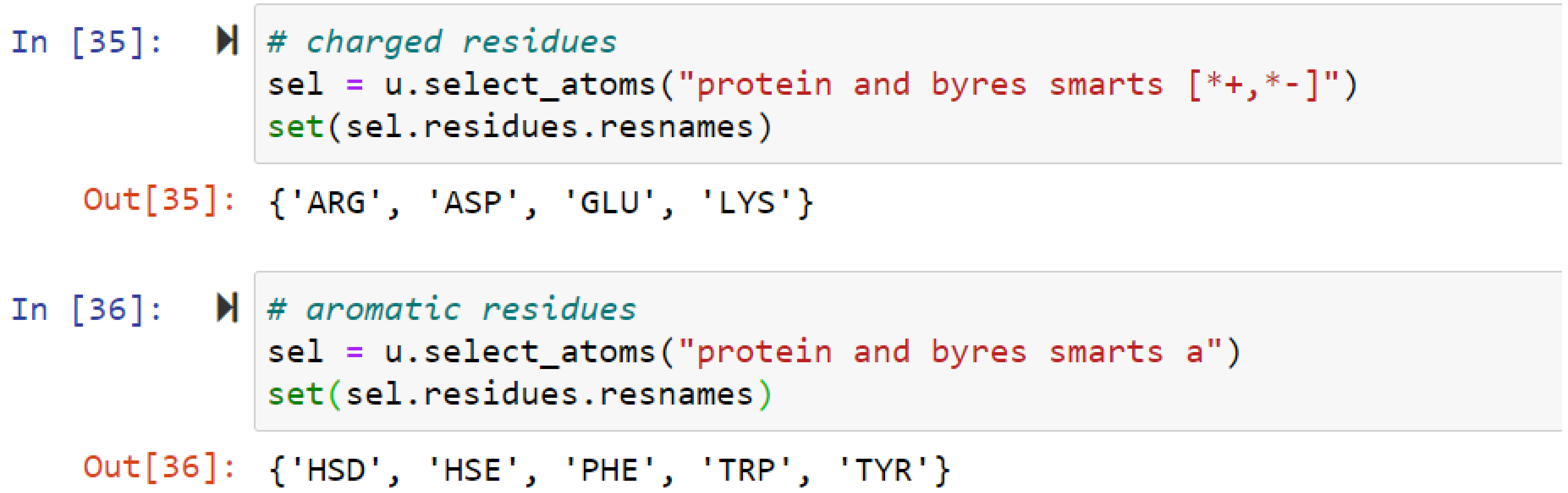
3D descriptors from MD simulations

In [5]:

```
from MDAnalysis.analysis.RDKit import RDKitDescriptors
u = mda.Universe(TOP, TRAJ)
lig = u.select_atoms("resname LIG")
desc = RDKitDescriptors(lig, "RDF").run()
```

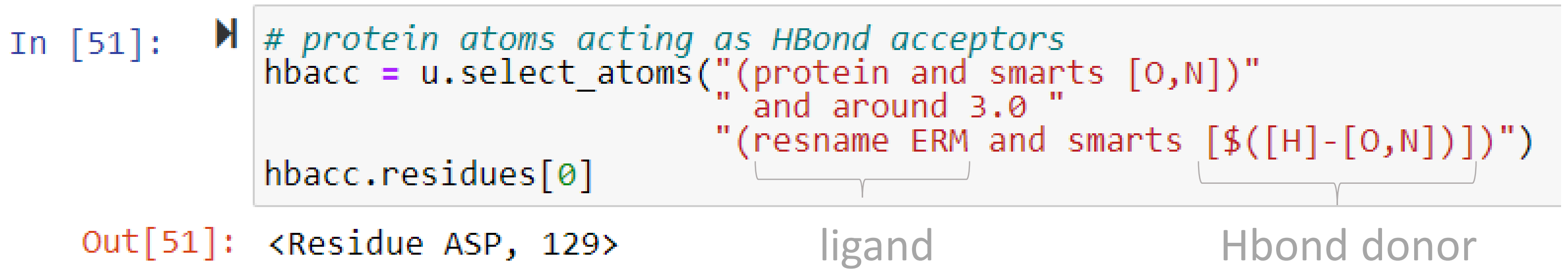


Smarter SMARTS selection



```
In [35]:  # charged residues
          sel = u.select_atoms("protein and byres smarts [*+,*-]")
          set(sel.residues.resnames)

Out[35]:  {'ARG', 'ASP', 'GLU', 'LYS'}

In [36]:  # aromatic residues
          sel = u.select_atoms("protein and byres smarts a")
          set(sel.residues.resnames)

Out[36]:  {'HSD', 'HSE', 'PHE', 'TRP', 'TYR'}
```

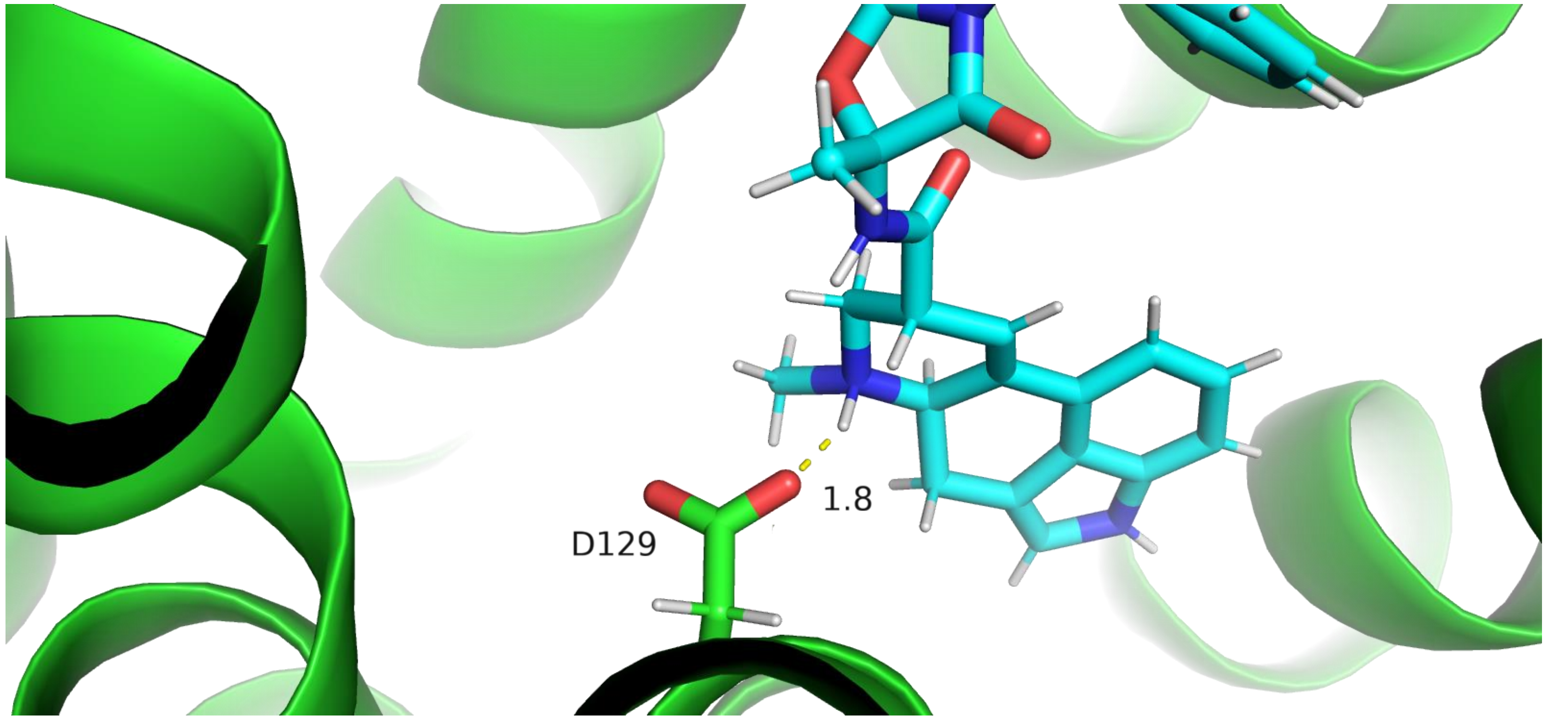
Smarter SMARTS selection



```
In [51]:  # protein atoms acting as HBond acceptors
          hbacc = u.select_atoms("(protein and smarts [O,N])"
                                 " and around 3.0 "
                                 "(resname ERM and smarts [$([H]-[O,N])])")
          hbacc.residues[0]

Out[51]:  <Residue ASP, 129>
```

ligand

Hbond donor



But wait, there's more!

- Fingerprints

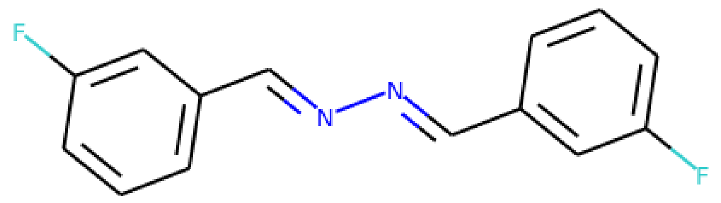
```
In [34]:  from MDAnalysis.analysis.RDKit import get_fingerprint
          lig = u.select_atoms("resname LIG")
          fp = get_fingerprint(lig, "Morgan",
                               radius=2, hashed=True, nBits=1024)
          # number of activated bits
          len(fp.GetNonzeroElements())

Out[34]:  20
```

- Drawing AtomGroups

```
In [7]:  # before
         u.select_atoms("resname LIG")

Out[7]:  <AtomGroup with 28 atoms>
```

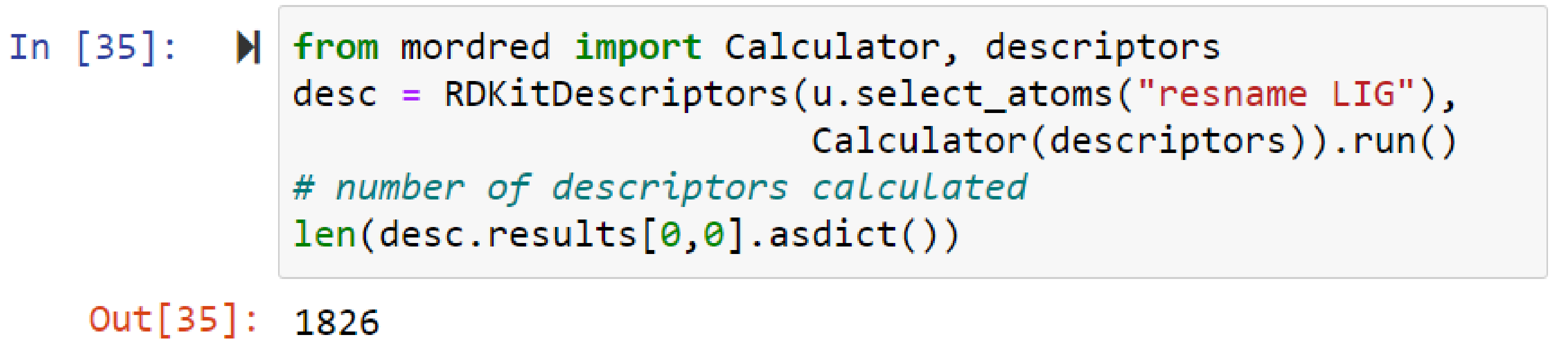
$\longrightarrow$

```
In [8]:  from MDAnalysis.visualization.RDKit import RDKitDrawer
         u.select_atoms("resname LIG")

Out[8]:
```



But wait, there's more!

- More features through RDKit-compatible packages:
  - Mordred (descriptors)



```
In [35]:  from mordred import Calculator, descriptors
          desc = RDKitDescriptors(u.select_atoms("resname LIG"),
                                  Calculator(descriptors)).run()
          # number of descriptors calculated
          len(desc.results[0,0].asdict())

Out[35]:  1826
```

  - ODDT (docking and fingerprints)
  - …

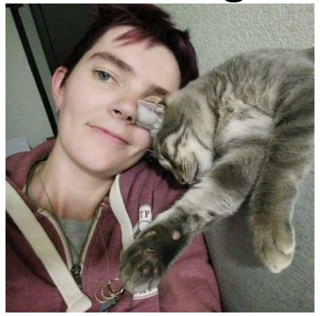
Acknowledgements

Google Summer of Code

NUMFOCUS [FISCALLY SPONSORED PROJECT]

- Mentors and MDAnalysis core devs

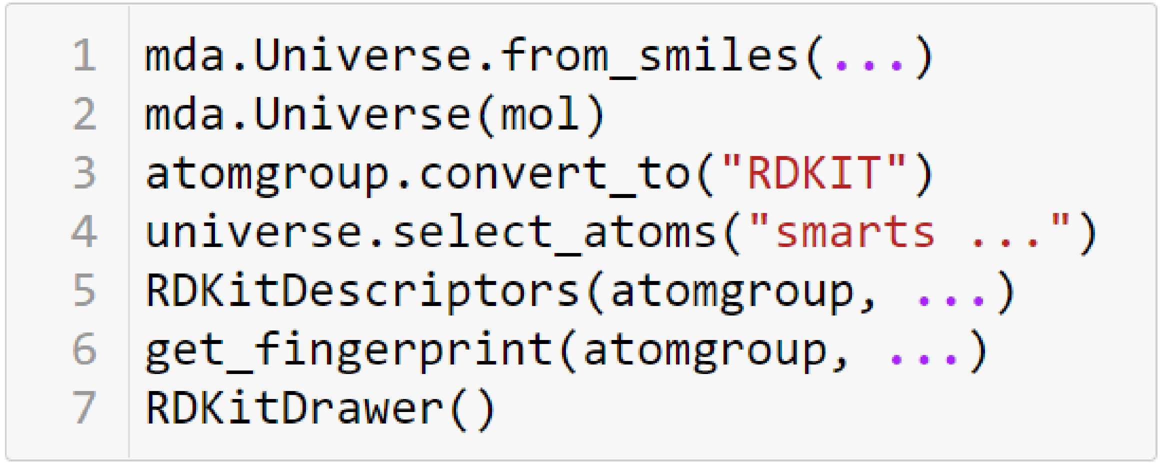
  Irfan Alibay, Richard Gowers, Fiona Naughton, Oliver Beckstein



- RDKit and the community
- Chemosim Lab

Summary

- New features in MDAnalysis (will be available in version 2.0)



```
mda.Universe.from_smiles(...)
mda.Universe(mol)
atomgroup.convert_to("RDKIT")
universe.select_atoms("smarts ...")
RDKitDescriptors(atomgroup, ...)
get_fingerprint(atomgroup, ...)
RDKitDrawer()
```

- Development of a valence-based and rule-based method to quickly infer bonds orders and formal charges