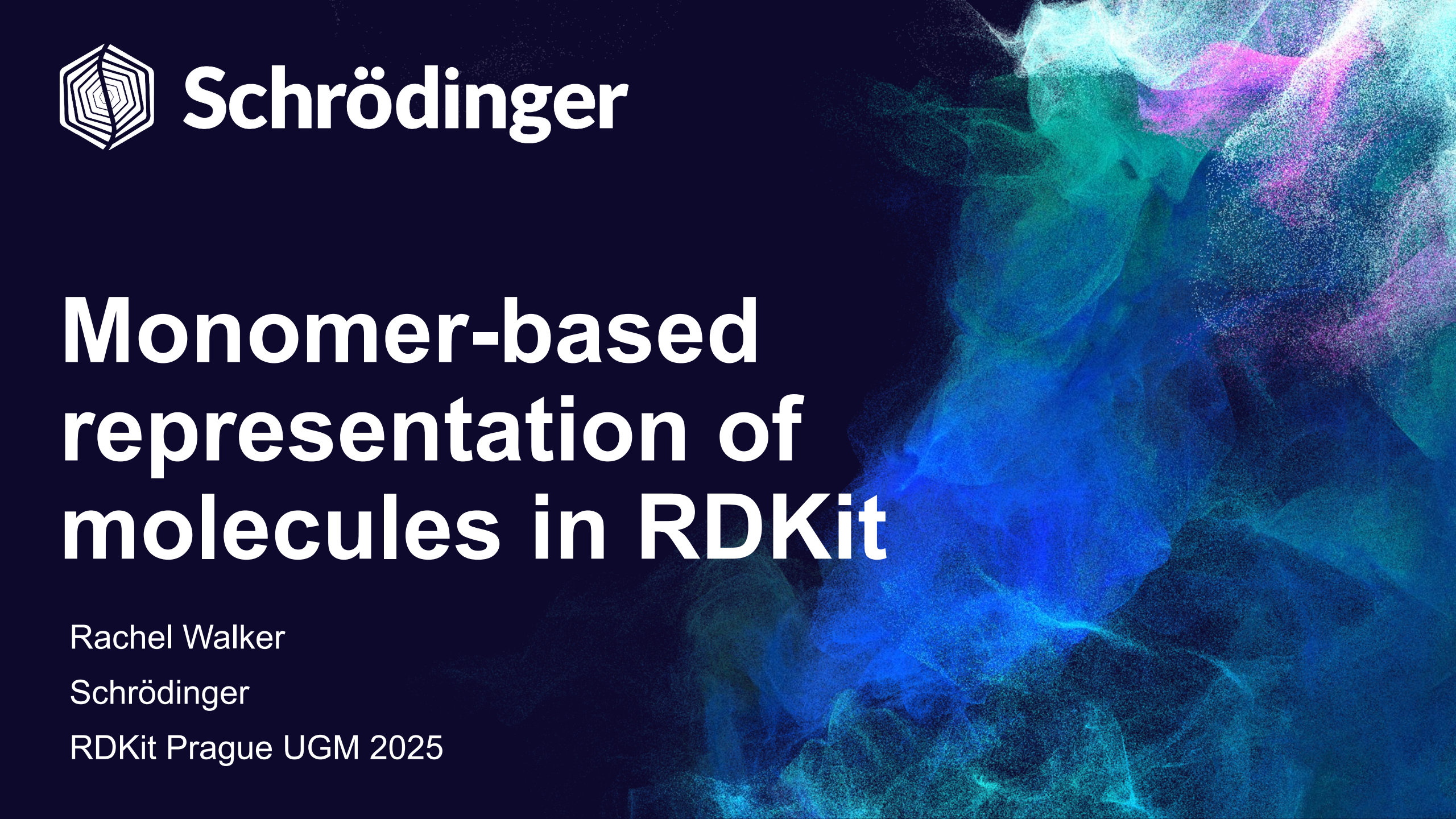


Schrödinger

Monomer-based representation of molecules in RDKit

Rachel Walker

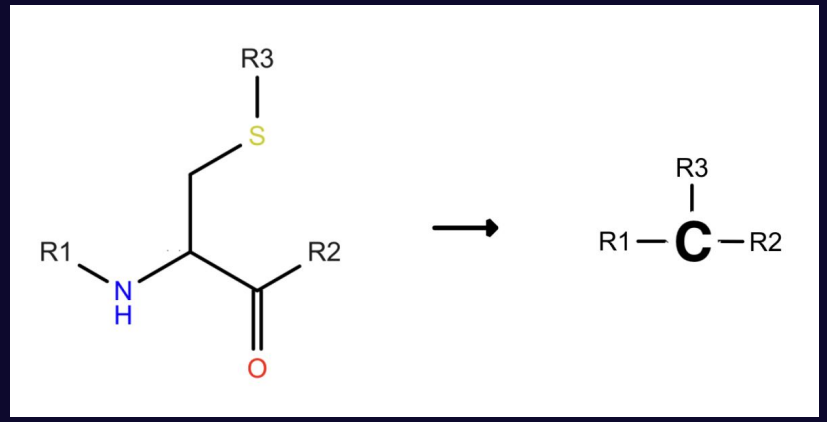
Schrödinger

RDKit Prague UGM 2025

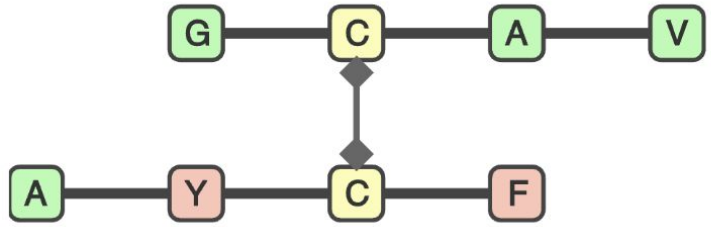
MonomerMol Proposal

See this PR https://github.com/rdkit/rdkit/pull/8218

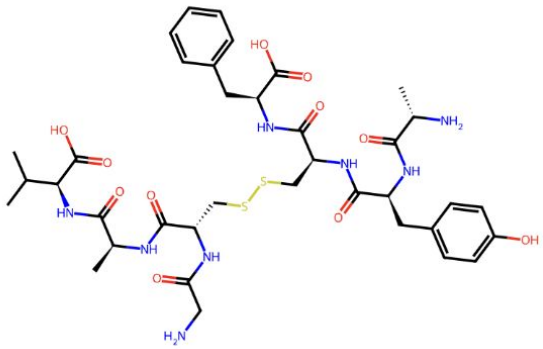
Objective: Have a native RDKit data structure that represents molecules using monomers to allow for sequence-based serialization, manipulation, and depiction of molecules.



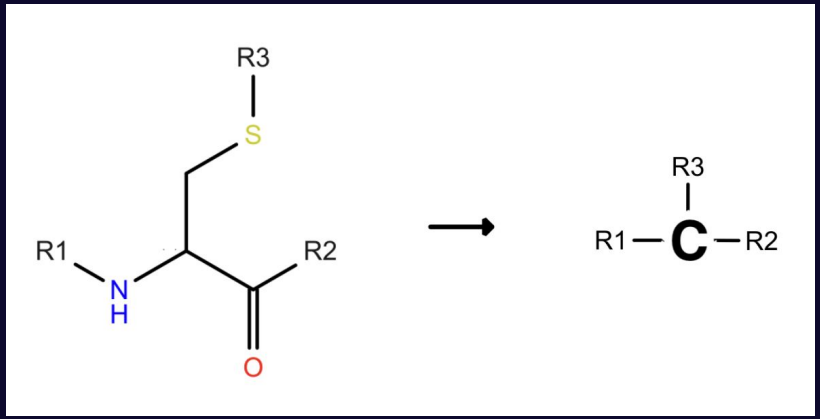

```
In [2]: helm_str = "PEPTIDE1{G.C.A.V}|PEPTIDE2{A.Y.C.F}$PEPTIDE1,PEPTIDE2,2:R3-3:R3$$$"
        monomer_mol = rdkit_extensions.to_rdkit(helm_str)
        drawMonomeristic(monomer_mol)
Out[2]:
```

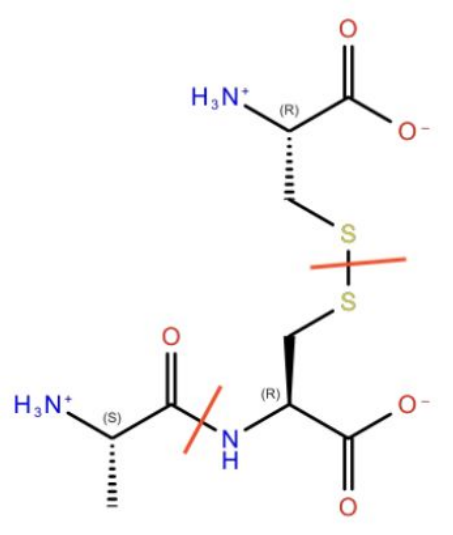



```
In [3]: rdkit_extensions.cg_to_atomistic(monomer_mol)
Out[3]:
```



Background: Sequence-based text formats





SCSR

```
M  V30 COUNTS 3 2 0 0 0
M  V30 BEGIN ATOM
M  V30 1 Ala 5.775 -4.775 0.0 0 CLASS=AA ATTCHORD=(2 2 Br)
SEQID=1
M  V30 2 Cys 6.641 -5.275 0.0 0 CLASS=AA ATTCHORD=(4 1 Al 3
Cx) SEQID=1
M  V30 3 Cys 6.641 -6.275 0.0 0 CLASS=AA ATTCHORD=(2 2 Cx)
SEQID=1
M  V30 END ATOM
M  V30 BEGIN BOND
M  V30 1 1 1 2
M  V30 2 1 2 3
M  V30 END BOND
M  V30 END CTAB
```

HELM

```
PEPTIDE1{A.C}|PEPTIDE2{C}$PEPTIDE1,PEPTIDE2,2:R3-1:R3
```

BILN

```
A-C(1,3).C(1,3)
```

Schrödinger

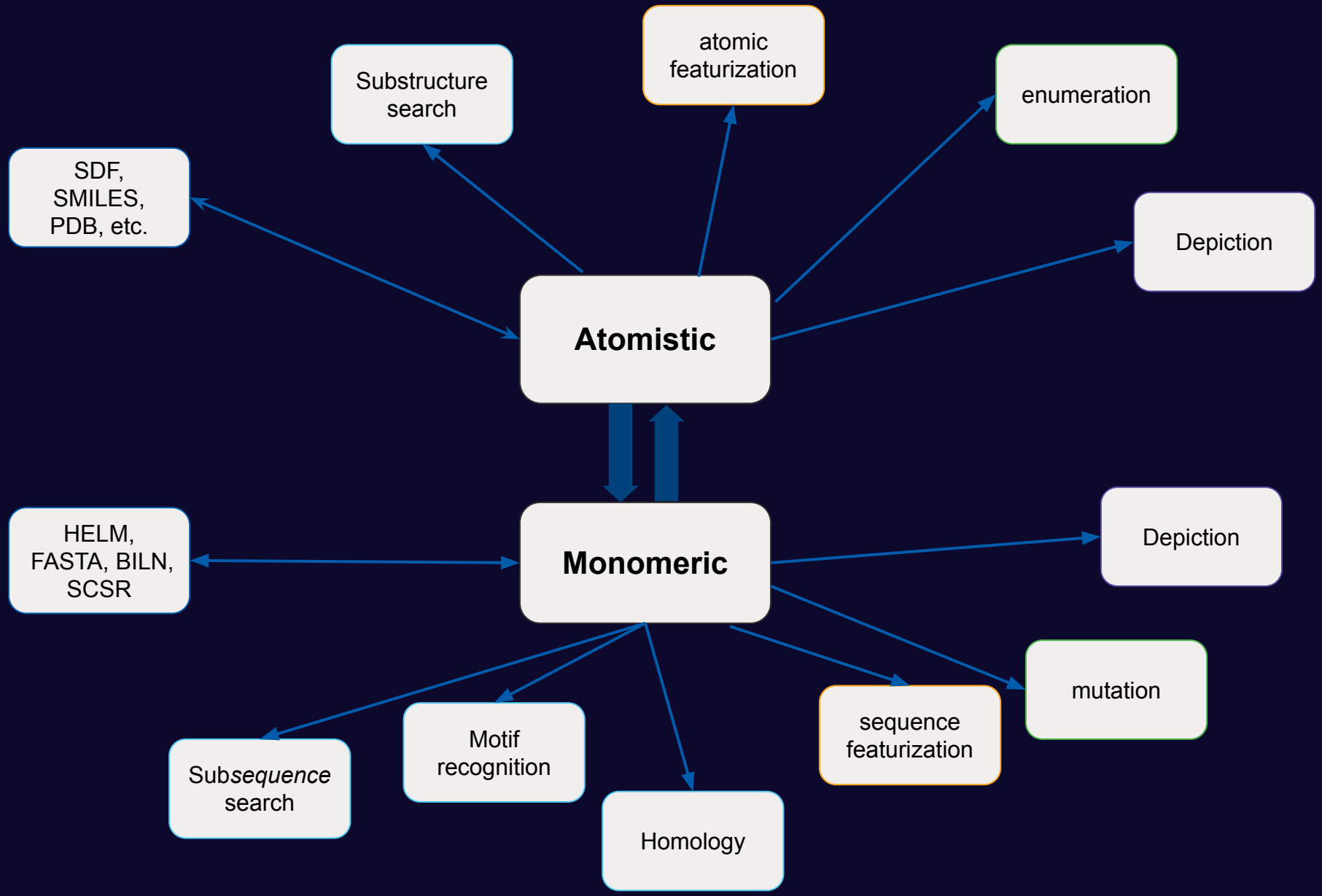
Existing monomer-ish abilities in RDKit

- atomLabel or smilesSymbol
- Abbreviations
- Biovia Super atoms
- SCSR (https://github.com/rdkit/rdkit/pull/8147 from tad@cdd)
- Reading/Writing FASTA and HELM v1 (roger@nextmove)
- MonomerInfo

Representing a Monomer-based Molecule using an ROMol

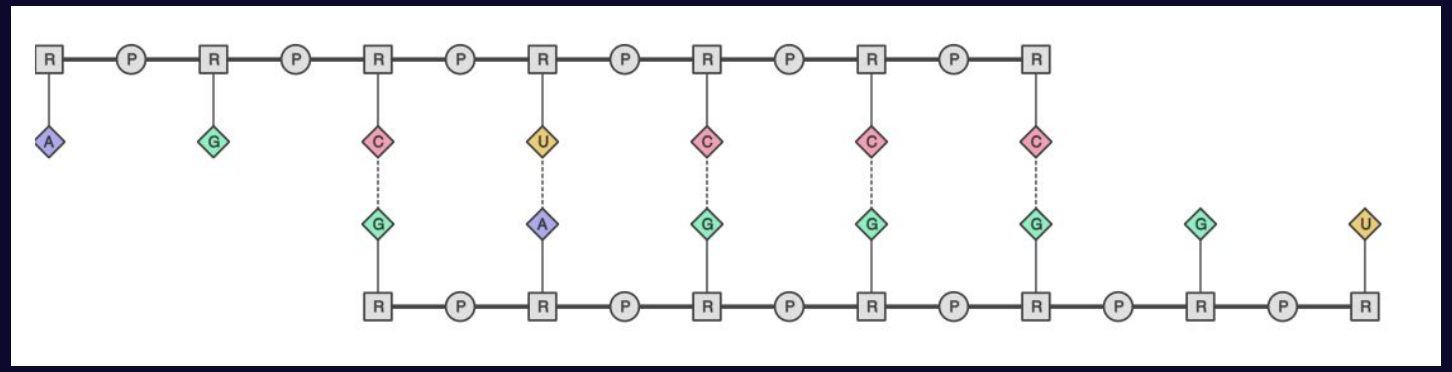
- **Atoms/Monomers:** Dummy atoms with `RDKit::AtomPDBResidueInfo`
  - `residueName, residueNumber, chainID`
- **Bonds/Linkages:** Property `LINKAGE` set to `RX-RY` , where `X` is the attachment point from the `beginAtom()` monomer and `Y` is the attachment point in the `endAtom()` monomer
  - Direction matters
- Customizable monomer database contains atomistic information and PDB names

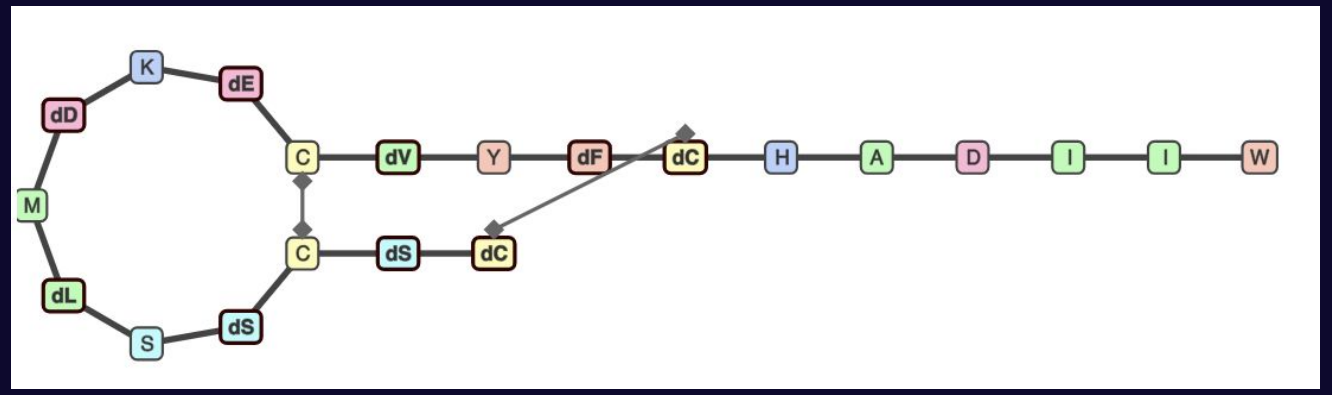
How would MonomerMol be used?

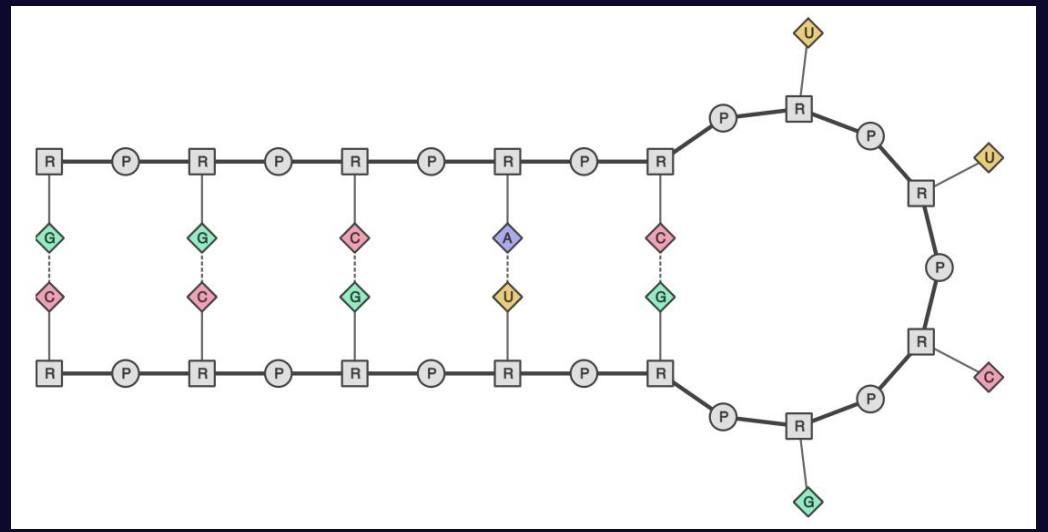


Schrödinger

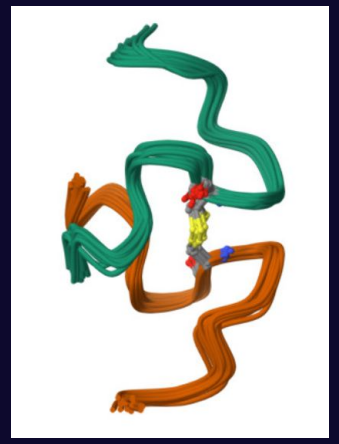
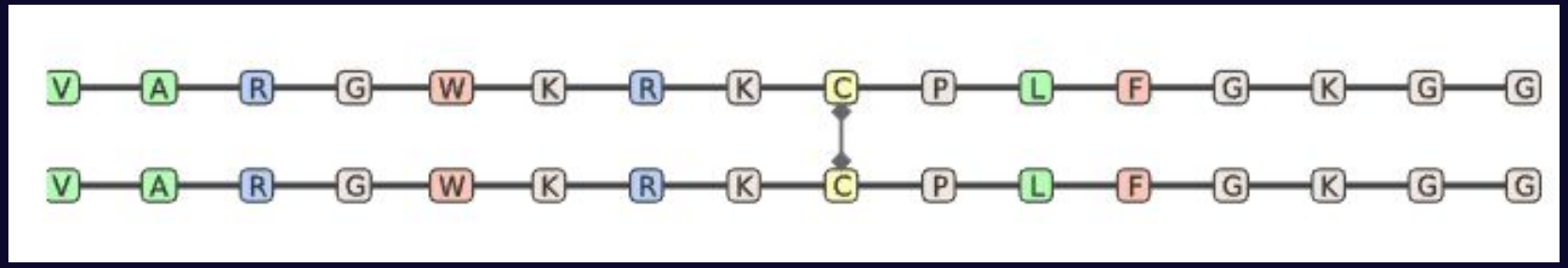
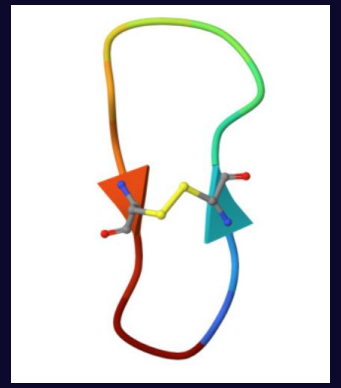
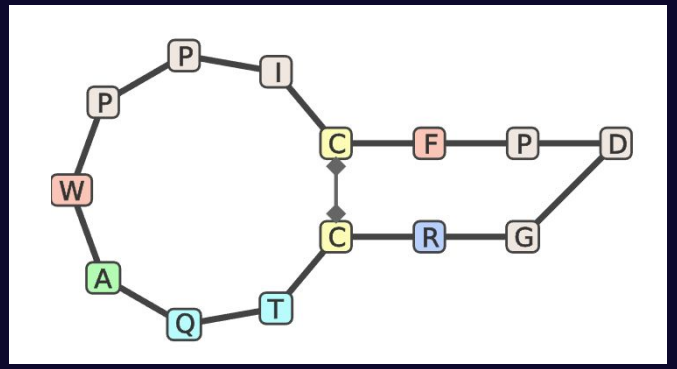
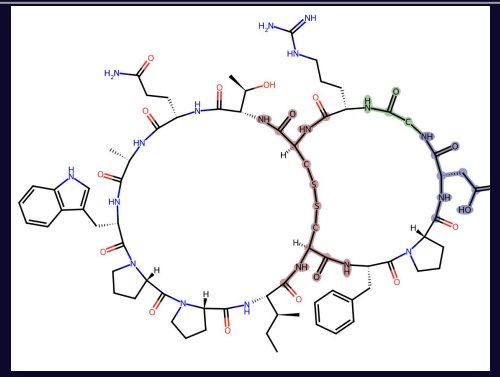
Depiction


R P R P R P R P R P R P R
A G C U C C C
G A G G G G U
R P R P R P R P R P R P R P R


K dE dD C dV Y dF dC H A D I I W
M C dS dC
dL S dS


U R P P R U
R P R P R P R P R P R P R P
G G C A C P
C C G U G R
R P R P R P R P R P R P R P C
P R P
G

Convert between atomistic and monomeric

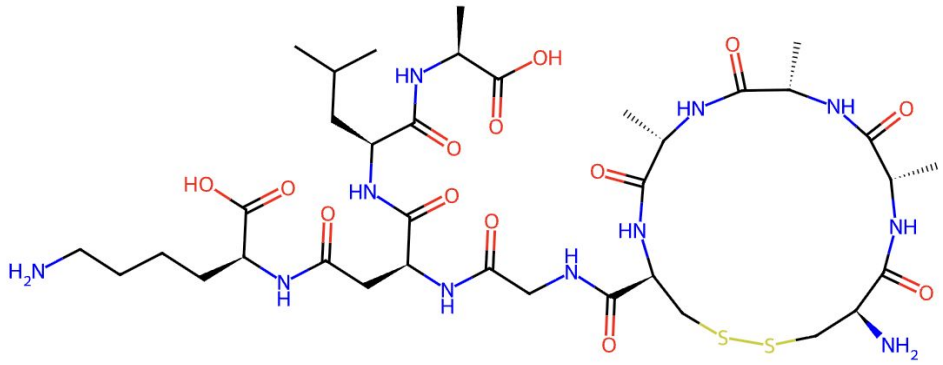
2N65 from RCSB				
5VAV from RCSB				

Convert between atomistic and monomeric

In [12]:
```
smiles = "CC(C)C[C@H](NC(=O)[C@H](CC(=O)N[C@@H](CCCCN)C(=O)O)NC(=O)CNC(=O)[C@@H]1CSSC[C@H](N)C(=O)N[C@@H](C)C(=O)N[C
atomistic_mol = Chem.MolFromSmiles(smiles)
monomer_mol = rdkit_extensions.toMonomeric(atomistic_mol)
print(f"HELM is: {rdkit_extensions.to_string(monomer_mol, rdkit_extensions.Format.HELM)}")
atomistic_mol
```

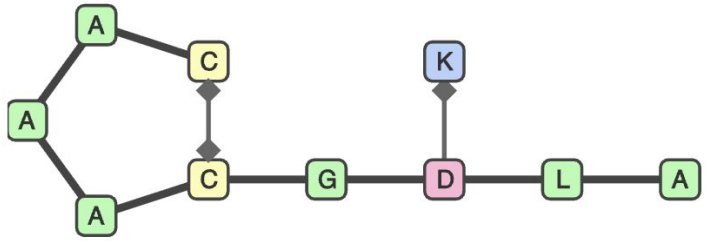
```
HELM is: PEPTIDE1{C.A.A.A.C.G.D(K)L.A}$PEPTIDE1,PEPTIDE1,5:R3-1:R3$$$V2.0
```

Out[12]:



In [13]:
```
SVG(bbchem_endpoints.to_image(to_binary(monomer_mol)))
```

Out[13]:

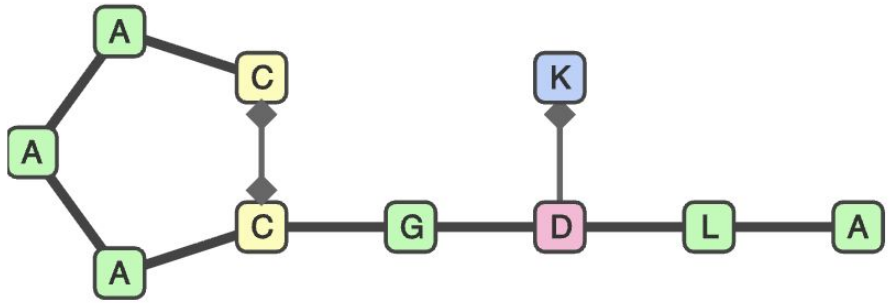


Schrödinger

Subsequence Searching

```
In [13]: SVG(bbchem_endpoints.to_image(to_binary(monomer_mol)))
```

Out[13]:

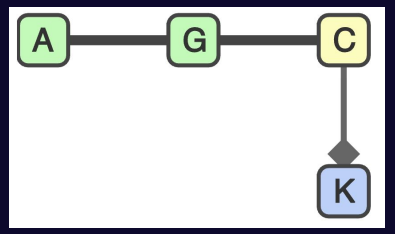



```
In [16]: monomer_query1 = rdkit_extensions.to_rdkit("PEPTIDE1{C.G.D(K)}$$$$V2.0")
         monomer_query2 = rdkit_extensions.to_rdkit("PEPTIDE1{C.G.D.K}$$$$V2.0")
         print(has_subsequence_match(monomer_mol, monomer_query1))
         print(has_subsequence_match(monomer_mol, monomer_query2))
```

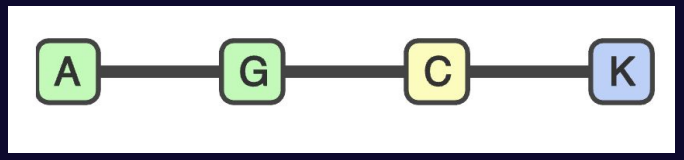
```
True
False
```

MonomerMol Hash

The queries from the last slide have different hashes because the linkage between C and K is different



≠



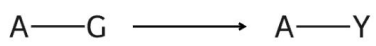

```python
monomer_mol_hash_test_cases = [
    # Same monomers, opposite chain direction
    ("PEPTIDE1{G.P}$$$$V2.0", "PEPTIDE1{P.G}$$$$V2.0", False),
    ("PEPTIDE1{A.G.C.K}$$$$V2.0", "PEPTIDE1{K.C.G.A}$$$$V2.0", False),
    # R2-R1 cyclic peptide with different starting residues
    ("PEPTIDE1{A.A.G.F.P.V.F.F}$PEPTIDE1,PEPTIDE1,8:R2-1:R1$$$V2.0",
     "PEPTIDE1{F.F.A.A.G.F.P.V}$PEPTIDE1,PEPTIDE1,8:R2-1:R1$$$V2.0", True),
    # Disulfide bonded cyclic peptide with different starting residues,
    # different because E moves from end of chain to beginning of chain
    ("PEPTIDE1{C.E.C.C.E}$PEPTIDE1,PEPTIDE1,1:R3-3:R3$$$V2.0",
     "PEPTIDE1{E.C.C.E.C}$PEPTIDE1,PEPTIDE1,2:R3-5:R3$$$V2.0", False),
    # Branched monomer vs backbone monomer are different
    ("PEPTIDE1{A.G.C(K)}$$$$V2.0", "PEPTIDE1{A.G.C.K}$$$$V2.0", False),
    # Cycle closures using different attachments are different
    # (disulfide bond vs backbone bond)
    ("PEPTIDE1{C.F.C.C.C}$PEPTIDE1,PEPTIDE1,1:R3-5:R3$$$V2.0",
     "PEPTIDE1{C.F.C.C.C}$PEPTIDE1,PEPTIDE1,1:R2-5:R1$$$V2.0", False),
    # Same sequence but broken into chains, same topology
    ("PEPTIDE1{A.G.C.K}|PEPTIDE2{A.G.C.K}$PEPTIDE1,PEPTIDE2,4:R2-1:R1$$$V2.0",
     "PEPTIDE1{A.G.C.K.A.G.C.K}$$$$V2.0", True)
]

@pytest.mark.parametrize('helm1, helm2, expected_equivalency',
                         monomer_mol_hash_test_cases)
def test_monomer_mol_hash(helm1, helm2, expected_equivalency):
    mol1 = rdkit_extensions.to_rdkit(helm1, Format.HELM)
    mol2 = rdkit_extensions.to_rdkit(helm2, Format.HELM)
    hash1 = get_monomer_mol_hash(mol1)
    hash2 = get_monomer_mol_hash(mol2)
    result = (hash1 == hash2)
    assert result is expected_equivalency
```

Some more possible uses

- Canonical ordering (using SMILES output order)
- MCS
- Serialization
- “Reaction” enumeration
- Mutations
- Featurization
- Sketcher

In [98]:
```
from rdkit.Chem import rdChemReactions
from schrodinger.rdkit_extensions import to_rdkit, to_string, Format

reactant = to_rdkit('PEPTIDE1{A.G}$$$$')
reactant.GetAtomWithIdx(0).SetAtomMapNum(1)
reactant.GetAtomWithIdx(1).SetAtomMapNum(2)
product = to_rdkit('PEPTIDE1{A.Y}$$$$')
product.GetAtomWithIdx(0).SetAtomMapNum(1)
product.GetAtomWithIdx(1).SetAtomMapNum(2)
rxn = rdChemReactions.ChemicalReaction()
rxn.AddReactantTemplate(reactant)
rxn.AddProductTemplate(product)
rxn.Initialize()

rxn
```

Out[98]:

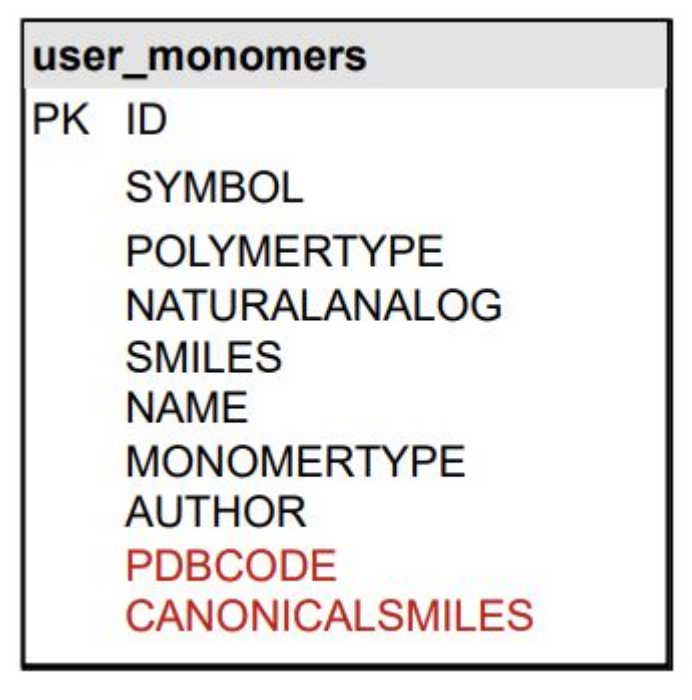


In [99]:
```
from rdkit.Chem.Draw import IPythonConsole

mol = to_rdkit('PEPTIDE1{A.G.K}$$$$')
print("reaction:")
print('', to_string(mol, Format.HELM))
print("products:")
for (product, ) in rxn.RunReactant(mol, 0):
    product.SetBoolProp('HELM_MODEL', True)
    print('', to_string(product, Format.HELM))

IPythonConsole.ShowMols((mol, product))
```

```
reaction:
 PEPTIDE1{A.G.K}$$$$V2.0
products:
 PEPTIDE1{A.Y.K}$$$$V2.0
```

Out[99]:

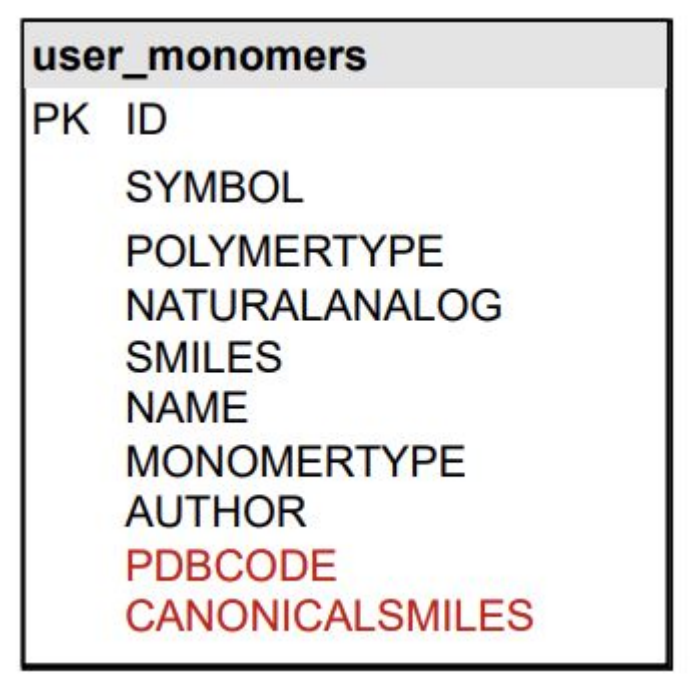
Monomer Database Schema




```
{
    "AUTHOR": "Schrödinger, Inc.",
    "ID": 9,
    "MONOMERTYPE": "Backbone",
    "NAME": "Cysteine",
    "NATURALANALOG": "C",
    "PDBCODE": "CYS",
    "POLYMERTYPE": "PEPTIDE",
    "SMILES": "O=C([C@H](CS[H:3])N[H:1])[OH:2]",
    "SYMBOL": "C"
}
```


https://github.com/PistoiaHELM/HELMMonomerSets

Monomer Database Schema




```
{
    "AUTHOR": "Schrödinger, Inc.",
    "ID": 9,
    "MONOMERTYPE": "Backbone",
    "NAME": "Cysteine",
    "NATURALANALOG": "C",
    "PDBCODE": "CYS",
    "POLYMERTYPE": "PEPTIDE",
    "SMILES": "O=C([C@H](CS[H:3])N[H:1])[OH:2]",
    "SYMBOL": "C"
}
```


Backbone
Branch
Undefined
Endcap?

PEPTIDE
RNA
CHEM

https://github.com/PistoiaHELM/HELMMonomerSets

Current Proposal

Included in https://github.com/rdkit/rdkit/pull/8218

- C++ interface to build a MonomerMol
- Atomistic ↔ Monomeric conversions for peptides (when residue information is present)
- Hard-coded monomer definitions

Future contributions

- Python wrappers
- Monomer DB support
- SMILES monomers — used when monomer is not identified in monomer DB
- HELM reader and writer
- FASTA reader and writer
- Coordinate and image generation
- MonomerMol hash
- Residue identification via SMARTS matching

Challenges & Open questions for the RDKit community

- Storing monomer definitions
- Residue identification
- Broader nonstandard nucleotide support
- Support for mixed-state structure (atoms and monomers)