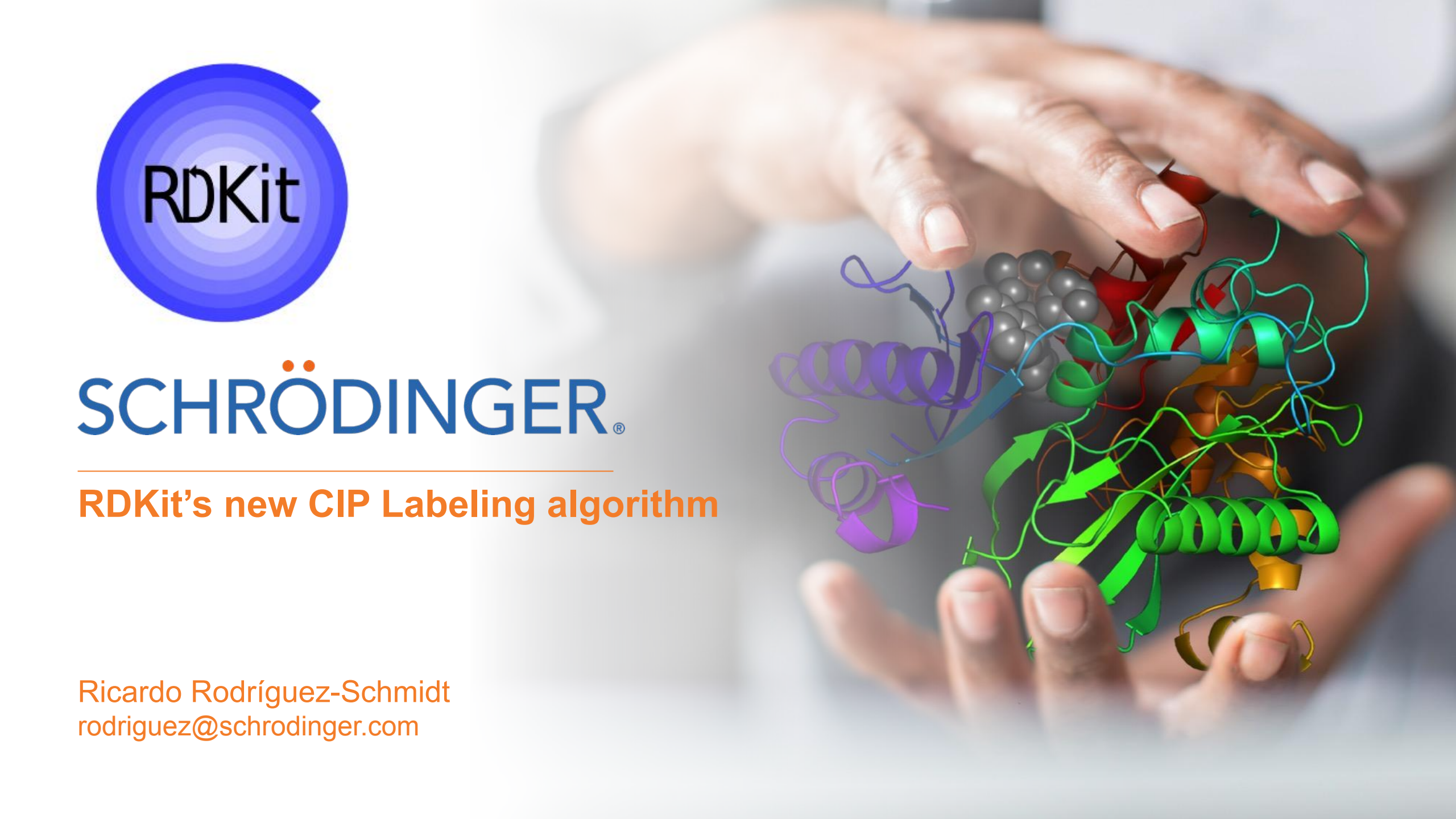


RDKit's new CIP Labeling algorithm

Ricardo Rodríguez-Schmidt
rodriguez@schrodinger.com

Why is this important?

Chirality (R, S):

- General chemistry, IUPAC standard
- Based on non-trivial rules
- Requires full molecule information
- Computationally slow

- Useful in depiction, communication

Parity (CW, CCW):

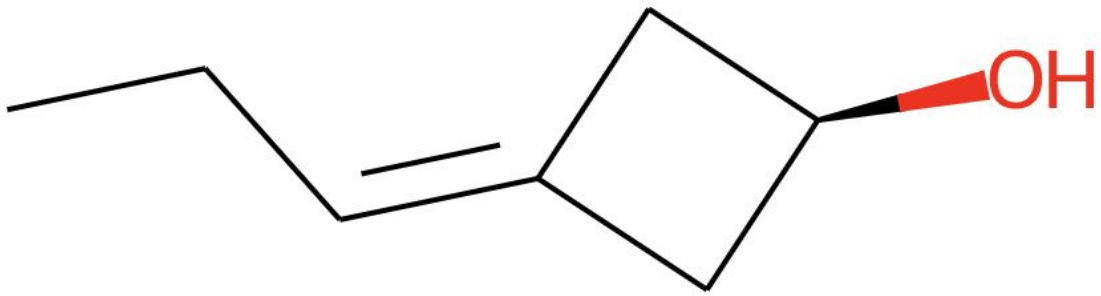
- Computational chemistry concept
- Only local info involved
- Requires references
- Computationally fast

- Not great for depiction or communication

- There is no direct relationship between Parity and Chirality: R ≠ CW and S ≠ CCW !!
- The current RDKit CIP labeling algorithm is just an approximation, and should not be trusted, since it produces wrong results.

CIP labeling is not always easy!

How would you label this molecule?



- At least one stereo center
- R/r/S/s E/e/Z/z ??
- Answer later in the talk

The new algorithm - The source

- This new code is a C++ port of “centres”, originally developed by John Mayfield:

  https://github.com/SiMolecule/centres

- It consists of a rigorous implementation of Cahn-Ingold-Prelog priority rules in

  Algorithmic Analysis of Cahn–Ingold–Prelog Rules of Stereochemistry: Proposals for Revised Rules and a Guide for Machine Implementation.
  Robert M. Hanson, Sophia Musacchio, John W. Mayfield, Mikko J. Vainio, Andrey Yerin, and Dmitry Redkin. Journal of Chemical Information and Modeling 2018 *58* (9), 1755-1765
  DOI: 10.1021/acs.jcim.8b00324

The new algorithm - What does it do?

- Labels tetrahedral and sp2 bond stereo centers* with the accurate CIP stereo labels, setting the “_CipCode” property of atoms and bonds.

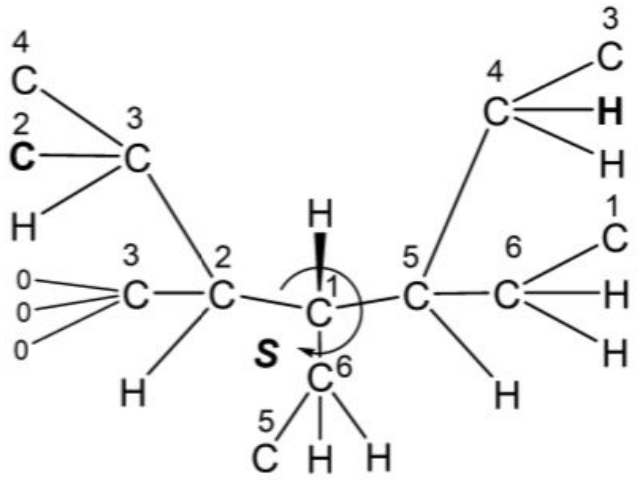
  (*: helicenes are not supported by the original code; allene-like, octahedral and other stereochemistry types depend on some hooks RDKit doesn’t have yet, but might be implemented in the future)

- The code is focused on accuracy, and has not been optimized for performance

  (but we might optimize it if see a need for it).

The new algorithm - How does it work?

- It focuses on each stereo center and progressively “unrolls” the molecule into graphs, sequentially applying the CIP rules to resolve the priorities around the center.



- Once the priorities are known, it uses chiral and stereo bond tags to find the appropriate stereochemical label for the center.

Validation

- Initial validation was done using the CIP Validation Suite, which is based on the examples in the IUPAC's Blue Book.

  https://github.com/CIPValidationSuite/ValidationSuite

- Greg and Dan Nealschneider ran additional validation by using the labeler on a large volume of molecules.

- Most of the stereo labels matched the references. All mismatches were explained by missing features in the labeler (e.g. helicity) or RDKit's stereo center detection.

Benchmarking

- Based on the CIP Validation Suite, discarding helicity M/P labels.
- Pessimistic benchmark: many of these structures have been chosen to be tricky!
- RDKit's old algorithm doesn't support pseudo asymmetry, but still labels some pseudo asymmetric centers, so we made all references uppercase.

"Old" algorithm:

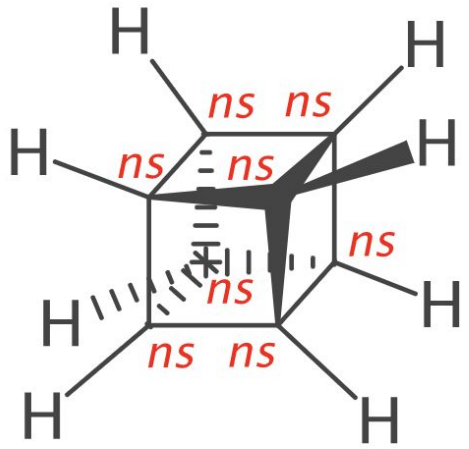
- Failed 99 / 300 structures.
- Ran in ~ 0.7 seconds.
- All structures processed in under 5 ms.

New labeler:

- Failed 7 / 300 structures.
- Took ~ 6.7 seconds.
- 81 % of the structures ran in <= 5 ms
- 17 % was between 5 and 100 ms
- 1.5 % took between 100 and 300 ms
- Only 2 structures took > 300 ms

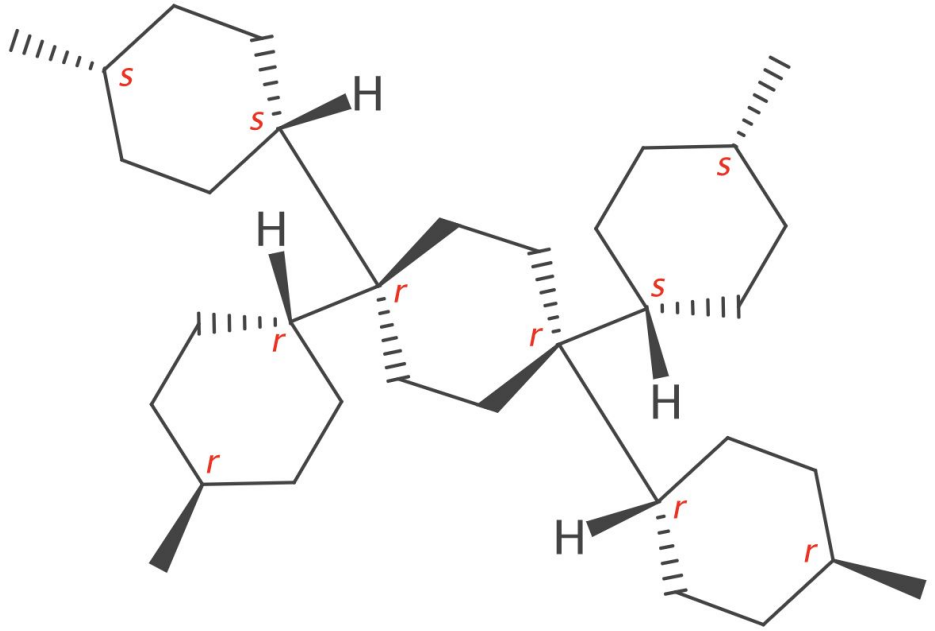
Benchmarking

VS009:



Run time: ~ 2.7 s

VS226:



Run time: ~ 1.2 s

Slower than the old algorithm, so don’t use it if you don’t require accurate stereo labels.

How to use it

- C++:

```
RDKIT_CIPLABELER_EXPORT void assignCIPLabels(ROMol &mol);

RDKIT_CIPLABELER_EXPORT void
assignCIPLabels(ROMol &mol, const boost::dynamic_bitset<> &atoms,
                const boost::dynamic_bitset<> &bonds);
```

  - Include `Graphmol/CIPLabeler/CIPLabeler.h`
  - Link against `GraphMol CIPLabeler`

- Python:

```
AssignCIPLabels( (Mol)mol [, (object)atomsToLabel=None [, (object)bondsToLabel=None]]) -> None :
    New implementation of Stereo assignment using a true CIP ranking
```

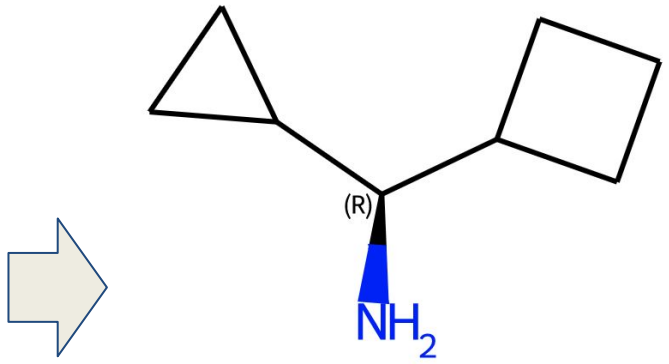
  - Import `rdkit.Chem.rdCIPLabeler`

Some examples

Github Issue #2252:

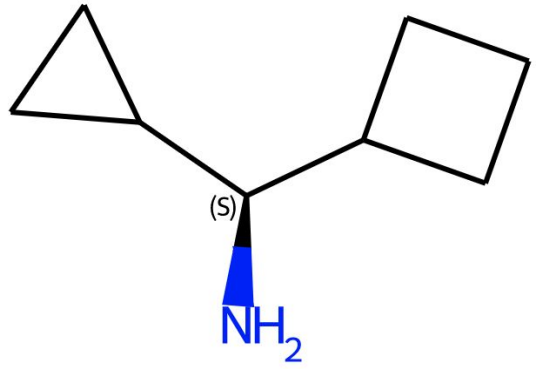
```
In [2]: from rdkit import Chem
        m = Chem.MolFromSmiles('N[C@H](C1CCC1)C1CC1')
        at = m.GetAtomWithIdx(1)

In [3]: at.SetProp('atomNote', at.GetProp('_CIPCode'))
        m
```



Wrong!!!

```
In [4]: Chem.rdCIPLabeler.AssignCIPLabels(m)
        at = m.GetAtomWithIdx(1)
        at.SetProp('atomNote', at.GetProp('_CIPCode'))
        m
```



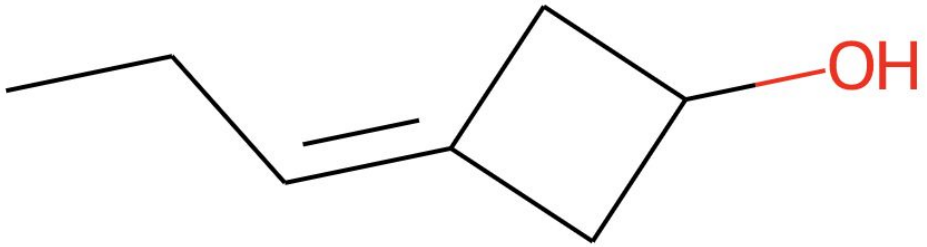
Right!!!

Some examples

Github Issue #2984:

```
In [28]: m = Chem.MolFromSmiles(r'CC/C=C1\C[C@@H](O)C1')
         m
```

Out[28]:




```
In [29]: Chem.FindMolChiralCenters(m)
Out[29]: []

In [30]: Chem.FindMolChiralCenters(m, useLegacyImplementation=False)
Out[30]: []
```

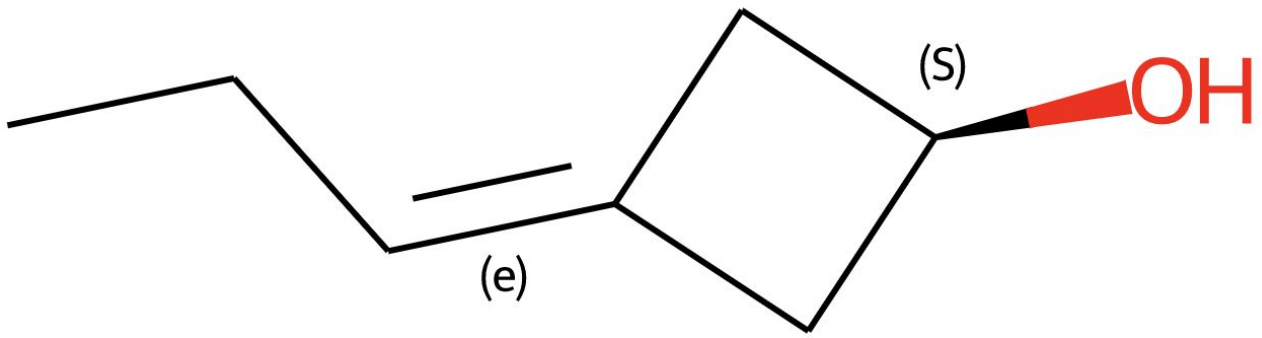
RDKit code does not see any stereo centers here (not even the new algorithm)

Some examples

In [9]:
```
m = Chem.MolFromSmiles(r'CC/C=C1\C[C@@H](O)C1', sanitize=False)
Chem.SanitizeMol(m)
Chem.SetBondStereoFromDirections(m)

Chem.rdCIPLabeler.AssignCIPLabels(m)
m
```

Out[9]:



Once we are able to find these centers, we will be able to resolve them!

To summarize

- CIP rules are not easy!!
- This provides accurate CIP labeling for tetrahedral and sp2 bond stereo centers.
- Support for other types of stereochemistry might be added in the future.
- Definitely slower than the old algorithm so don’t use it if you don’t need absolute stereo labels, but definitely do if you need them.

Thanks for your attention!

We have some open positions! Have a look at:
https://www.schrodinger.com/careers