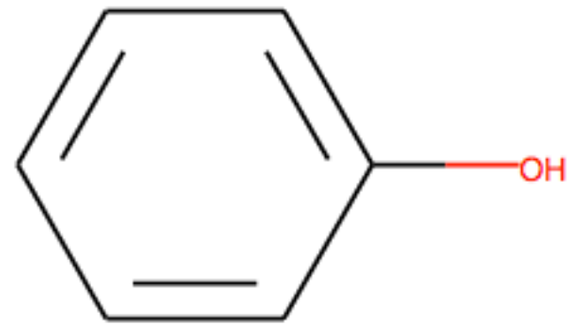
MMPDB: Taming chiral structures for MMP prediction

Andrew Dalke <dalke@dalkescientific.com>

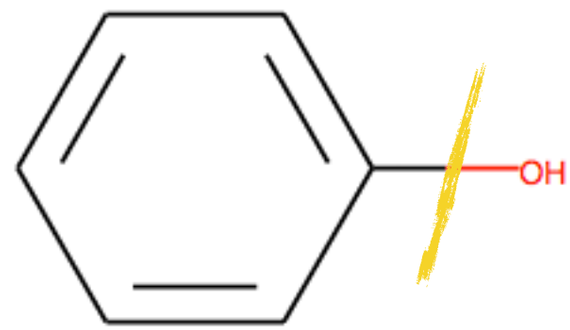
2017 RDKit User's Group Meeting
21 September 2017

MMP Basics

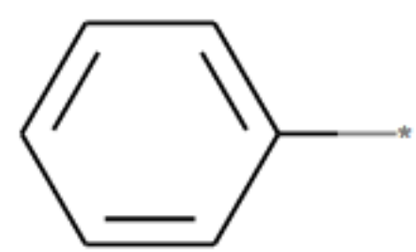
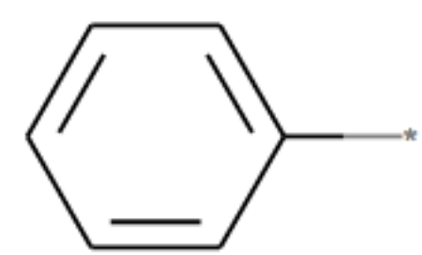
Input structure



Identify bonds to fragment

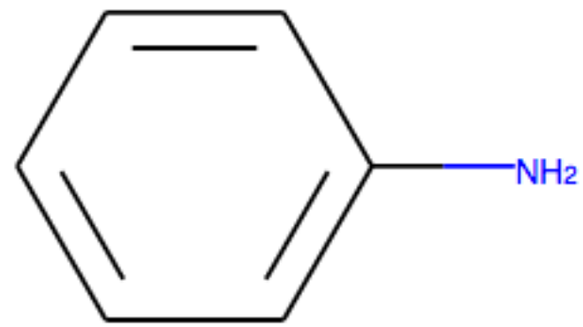


Generate fragmentations

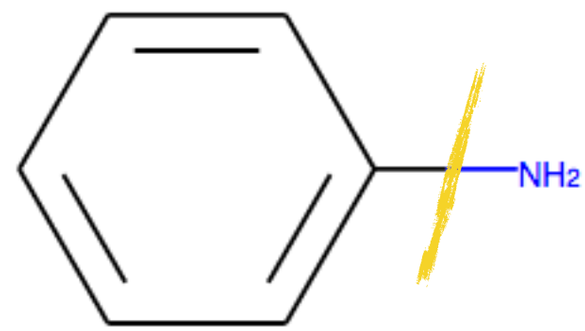
	variable part	constant part
1.		
2.		

phenylamine

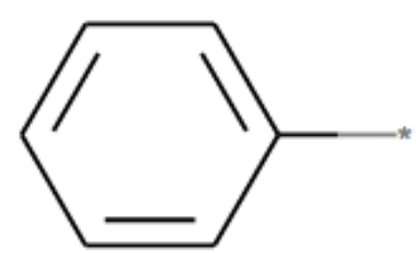
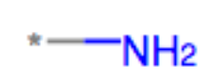
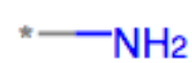
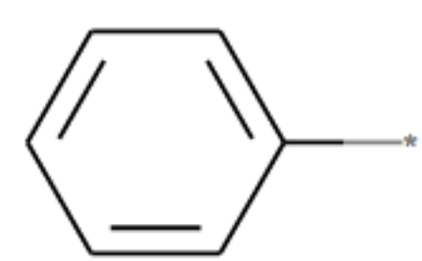
Input structure



Identify bonds to fragment

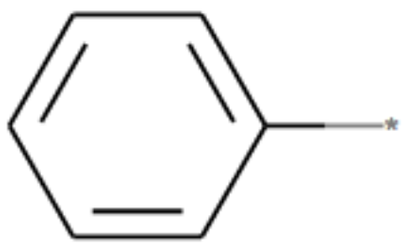
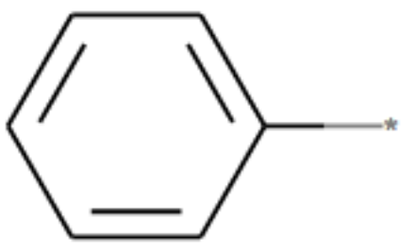
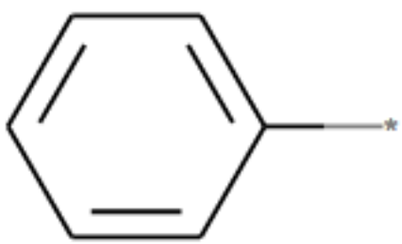


Generate fragmentations

	variable part	constant part
1.		
2.		

Index

constant part → list of variable parts

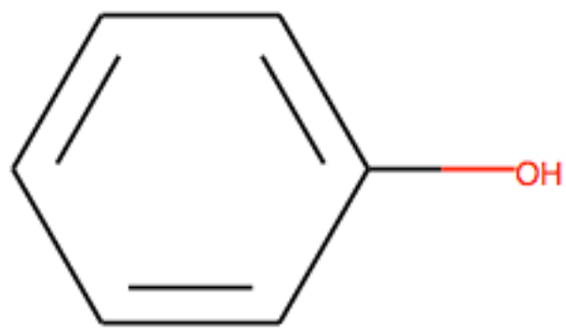
1.		 , 
2.		
3.		

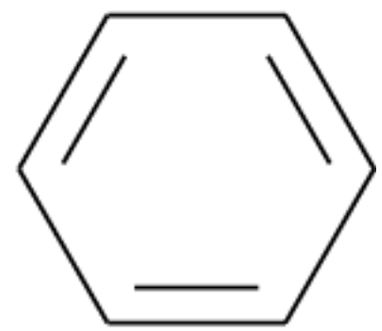
Result: [*]N>>[*]O is a matched molecular pair.

Hydrogen substitutions

Wait, what about [*]O>>[*]H or [*]N>>[*]H?

c1ccccc1O





c1ccccc1

For the 1-cut cases, create an additional constant with the attachment point replaced with "[H]"

	variable part	**constant part**	**constant-with-H**
1.	[*]c1ccccc1	[*]O	O
2.	[*]O	[*]c1ccccc1	c1ccccc1

Hydrogen pairs

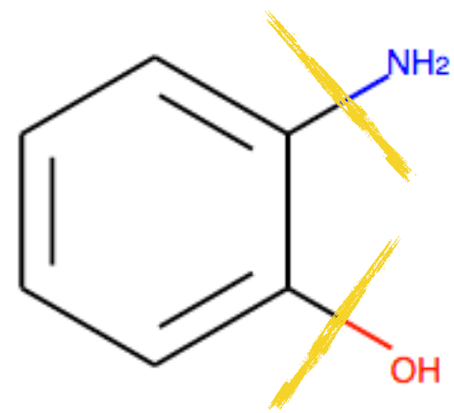
If one of the "constant-with-H" structures is an exact match to a canonicalized input structure then it's part of a hydrogen substitution.

	variable part	constant part	constant-with-H
1.	[*]c1ccccc1	[*]O	O
2.	[*]O	[*]c1ccccc1	c1ccccc1
3.	[*]c1ccccc1	[*]N	N
4.	[*]N	[*]c1ccccc1	c1ccccc1

Resulting hydrogen matched pairs:

[*]O>>[*]H because of Oc1ccccc1 to c1ccccc1
[*]N>>[*]H because of Nc1ccccc1 to c1ccccc1

Max two cuts: 2-aminophenol



	variable part	constant part	constant-with-H
1.	[*]c1ccccc1N	[*]O	O
2.	[*]O	[*]c1ccccc1N	Nc1ccccc1
3.	[*]c1ccccc1O	[*]N	N
4.	[*]N	[*]c1ccccc1O	Oc1ccccc1
5.	[*]c1ccccc1[*]	[*]N.[*]O	-

All fragmentations

		variable part	constant part	constant-with-H
2-aminophenol	**1.**	[*]c1ccccc1N	[*]O	O
	2.	[*]O	[*]c1ccccc1N	Nc1ccccc1
	5.	[*]c1ccccc1O	[*]N	N
	4.	[*]N	[*]c1ccccc1O	Oc1ccccc1
	3.	[*]c1ccccc1[*]	[*]N.[*]O	-
phenol	**6.**	[*]O	[*]c1ccccc1	c1ccccc1
	7.	[*]c1ccccc1	[*]O	O
phenylamine	**8.**	[*]N	[*]c1ccccc1	c1ccccc1
	9.	[*]c1ccccc1	[*]N	N

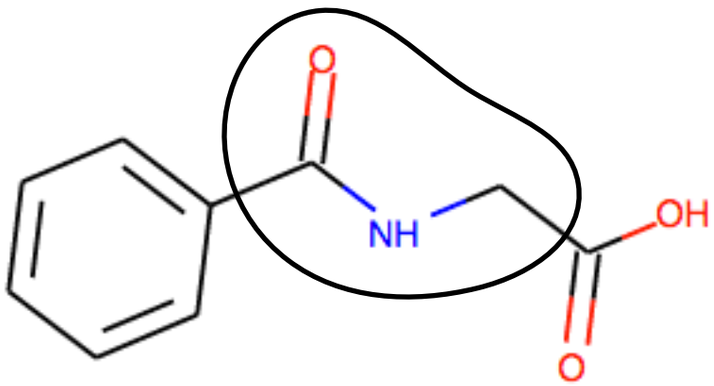
[*]N>>[*]O - phenyl amine to phenol

[*]N>>[*][H] - 2-aminophenol to phenol,
phenylamine to benzene

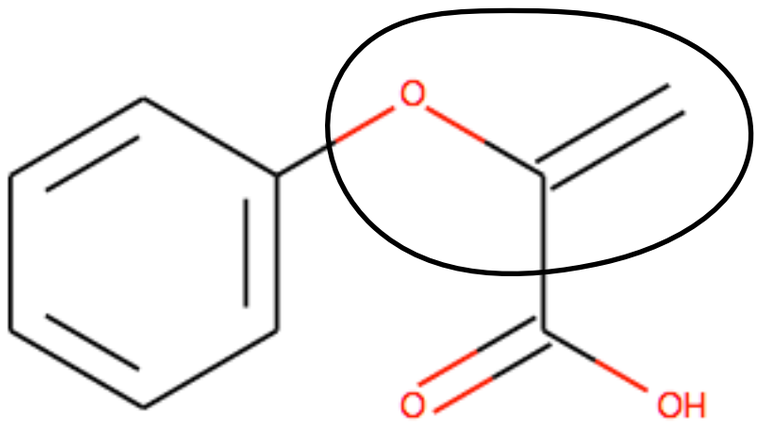
[*]O>>[*][H] - 2-aminophenol to phenylamine,
phenol to benzene

Indexing 2 or 3 cuts is harder

Labeled attachments



HIPPAC01



PASZIS

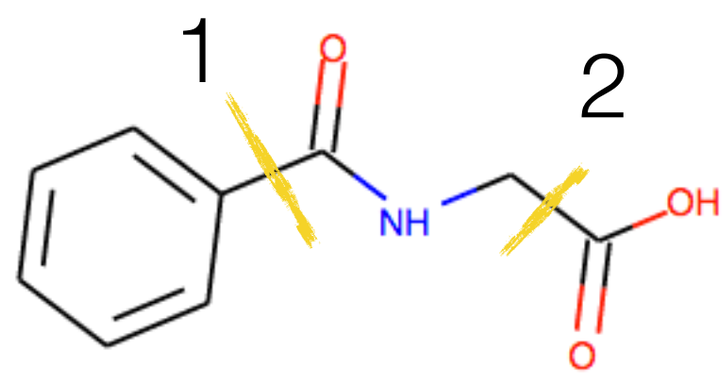
Variable Transform: [*:1]CNC([*:2])=O >> [*:2]OC([*:1])=C

Constant: [*:1]C(=O)O.[*:2]c1ccccc1

But which is [*:1] and which is [*:2]?

Hussain and Rea

1. Label the bonds arbitrarily



2. Cut and canonicalize

Variable: [*:2]CNC([*:1])=O

Constant: [*:2]C(=O)O.[*:1]c1ccccc1

3. Relabel so the constant part is in numeric order

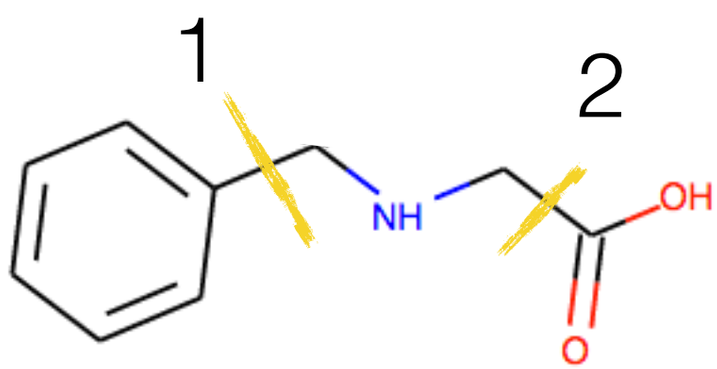
Variable: [*:1]CNC([*:2])=O

Constant: [*:1]C(=O)O.[*:2]c1ccccc1

Assumes the label does not affect canonicalization order.
This was also before Nadine's canonicalization improvements.

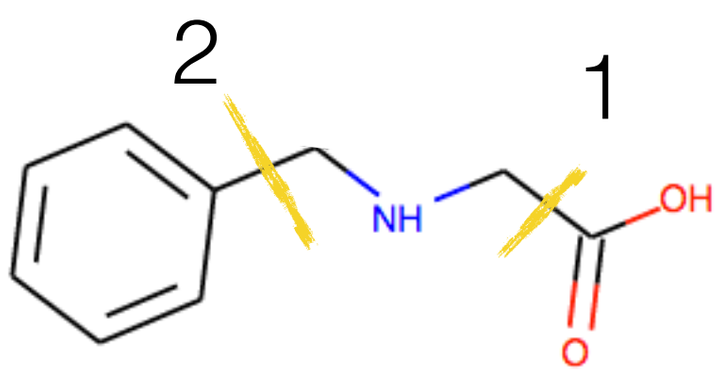
Symmetry Nuance

There can be several equivalent fragmentations.



Variable: [*:2]CNC[*:1]

Constant: [*:1]C(=O)O.[*:2]c1ccccc1



Variable: [*:1]CNC[*:2]

Constant: [*:1]C(=O)O.[*:2]c1ccccc1

But that's okay.
Don't need a canonical variable part.
Index, then canonicalize the transform.

"cansmirks"

Keep track of the symmetry classes of the attachment points in each variable part.

1-cut: 1
2-cuts: 11 or 12
3-cuts: 111, 112, 122, 121, 123,

[*:2]CNC[*:1] with symmetry class "11"
[*:1]OC([*:2])=C with symmetry class "12"

create transform	[*:2]CNC[*:1] >> [*:1]OC([*:2])=C
order LHS labels	[*:1]CNC[*:2] >> [*:2]OC([*:1])=C
use symmetry so smallest RHS labels come first	[*:1]CNC[*:2] >> [*:1]OC([*:2])=C

Not globally canonical!

Order depends the canonicalization order of the terms in the constant part

Constant: [*:1]C(=O)O.[*:2]c1ccccc1

Transform: [*:1]CNC([*:2])=O >> [*:2]OC([*:1])=C

In a different pair with the same transform:

Constant:[*:1]C(=O)S.[*:2]C(N)=O

Transform: [*:2]CNC([*:1])=O >> [*:1]OC([*:2])=C

Difficult to use

Can't get full statistics for a given transform.
(Could re-canonicalize all n! ways to label a cut. Slow!)

Would like to apply all applicable transforms to a given input structure. (Apply the fragment algorithm. If the variable part is in a known transform, apply the transform.) Need to generate all n! ways to represent the variable.

Given two structures, what's the predicted difference? Again, need to generate all n! representations for the two variable parts.

Need a fully canonical fragmentation

Problem 1: The constant part is canonical. How do we canonicalize the variable part?

Problem 2: The labels from the initial arbitrary bond index assignment may affect the canonicalization order.

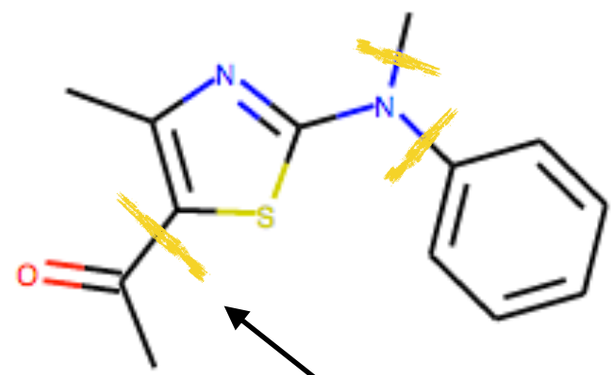
_smilesAtomOutputOrder

```
>>> mol = Chem.MolFromSmiles("O=C(O)C(N)C")
>>> Chem.MolToSmiles(mol)
'CC(N)C(=O)O'
>>> mol.GetProp("_smilesAtomOutputOrder")
'[5,3,4,1,0,2,]'
```

	0 1 2 3 4 5
Input SMILES	O=C(O)C(N)C
Canonical SMILES	CC(N)C(=O)O
_smilesAtomOutputOrder	5 3 4 1 0 2

Structure to fragment

Input structure

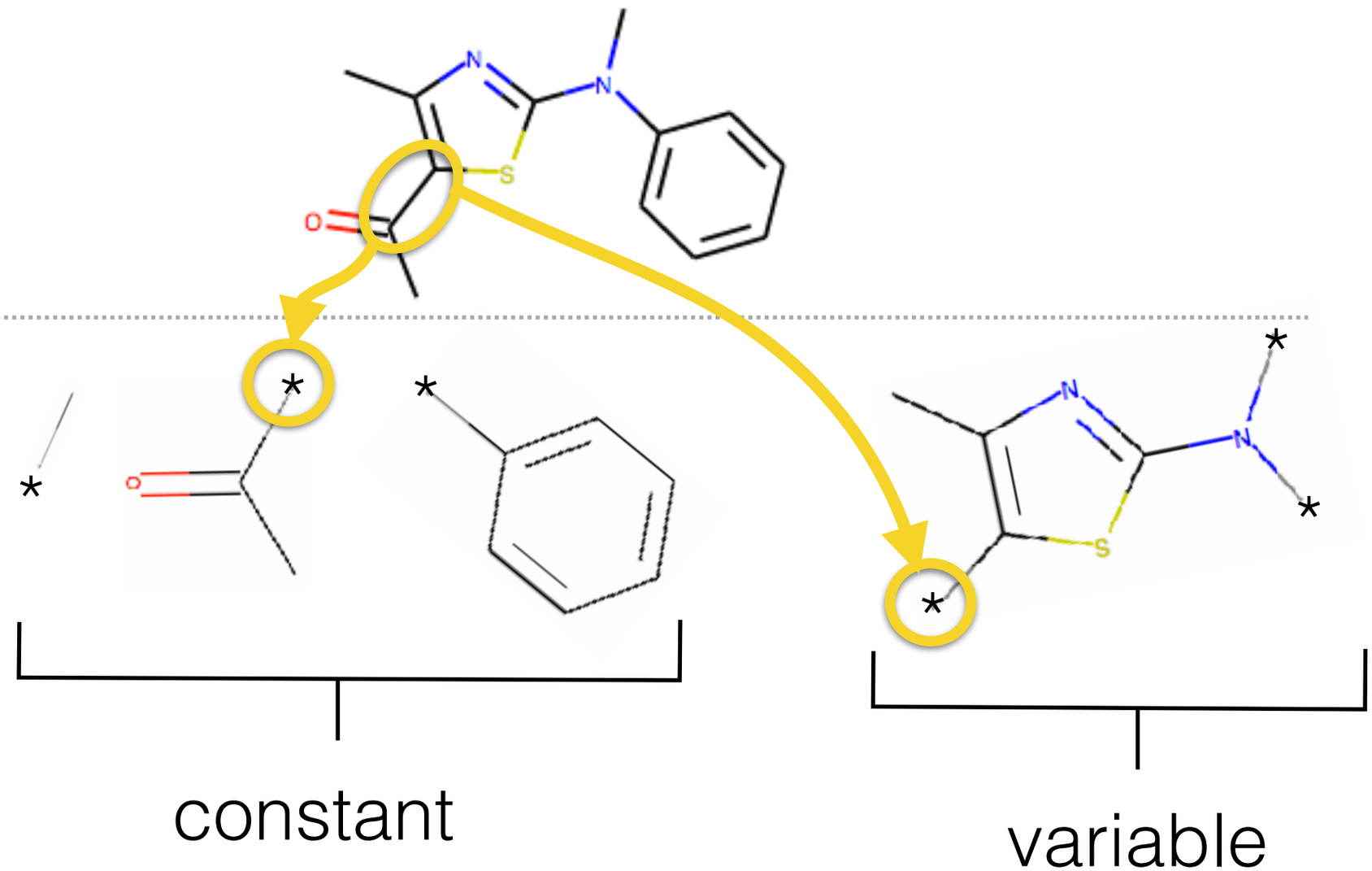


3 cuts, but focus on this one

Track attachment points

Input structure

Fragmented (3-cuts)

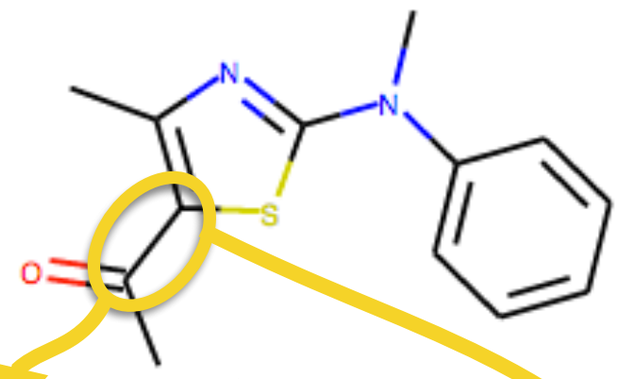


constant

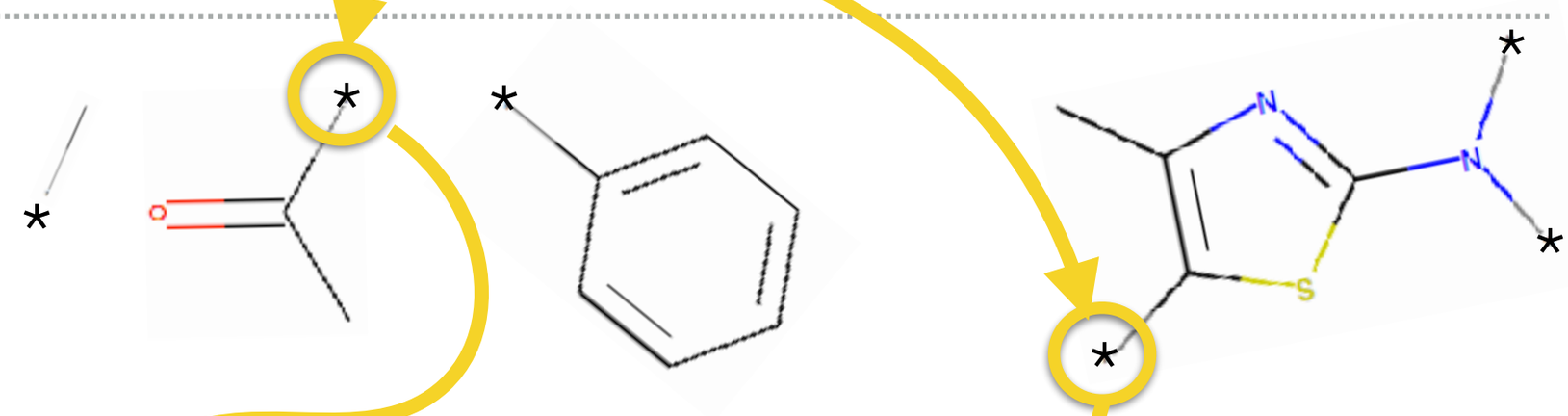
variable

Canonicalize fragments

Input structure



Fragmented (3-cuts)



constant part

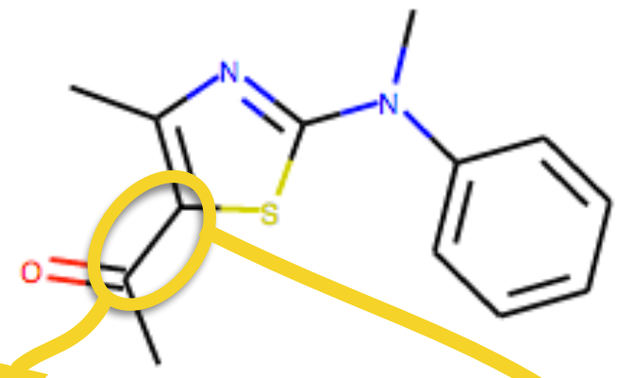
variable part

Canonical SMILES

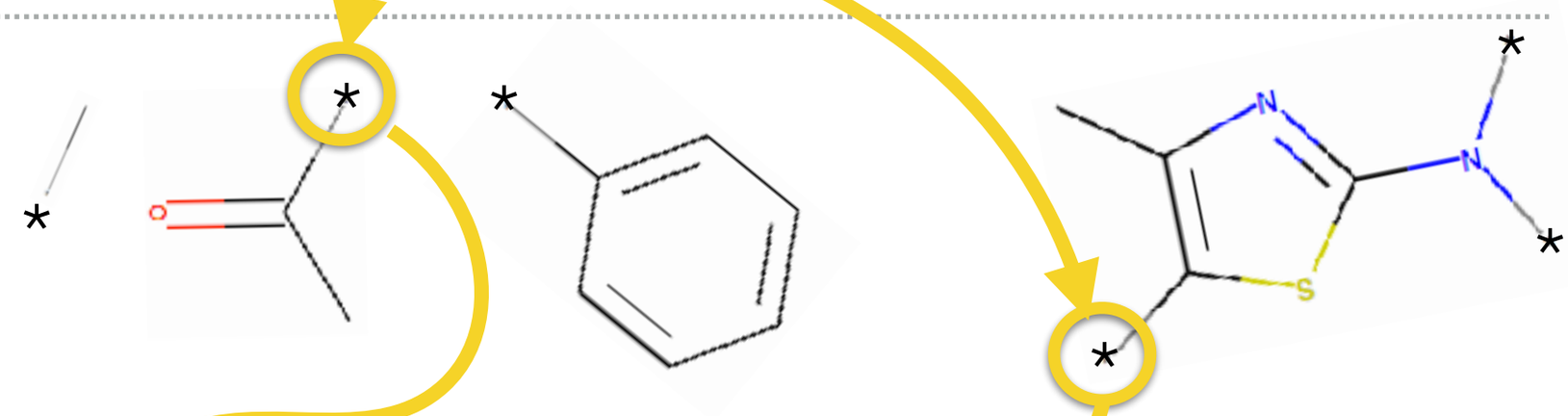
[*]C.[*]C(C)=O.[*]c1ccccc1 [*]c1sc(N([*])[*])nc1C

Use the canonical order of the constant part

Input structure



Fragmented (3-cuts)



constant part **variable part**

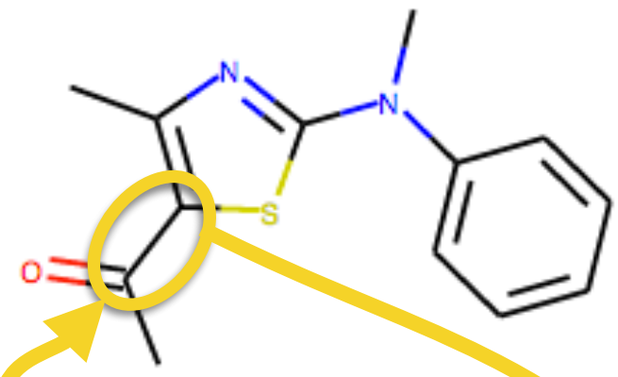
Canonical SMILES

[*]C.[*]C(C)=O.[*]c1ccccc1 [*]c1sc(N([*])[*])nc1C

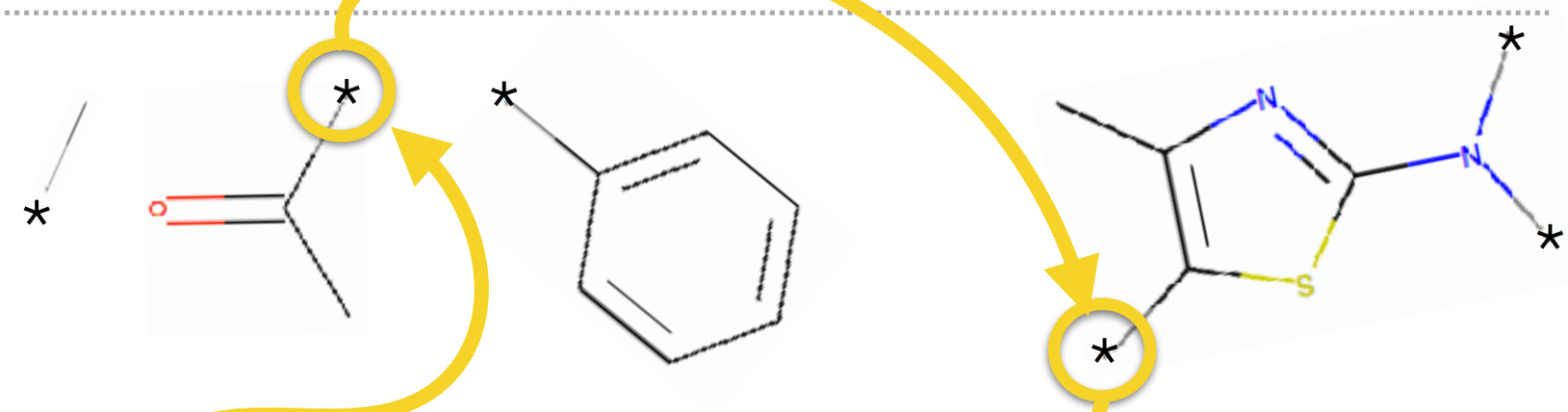
0 1 2

Determine the variable attachment order

Input structure



Fragmented (3-cuts)



Canonical SMILES

constant part

[*]C.[*]C(C)=O.[*]c1ccccc1

0 1 2

variable part

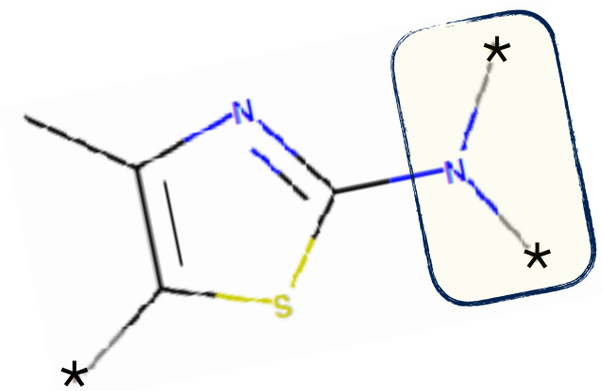
[*]c1sc(N([*])[*])nc1C

1 0 2

Order: 012 → 102

Symmetry (again)

The variable part contains two symmetric attachment points.



constant part **variable part**

[*]C.[*]C(C)=O.[*]c1ccccc1 [*]c1sc(N([*])[*])nc1C

0 1 2

Two possible choices

Order: 102	1	0	2
Order: 120	1	2	0

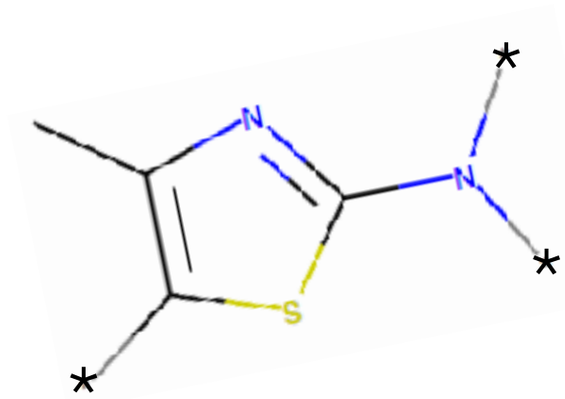
Choose the order which is numerically smallest.

<u>102</u> < 120

Fragmentation record

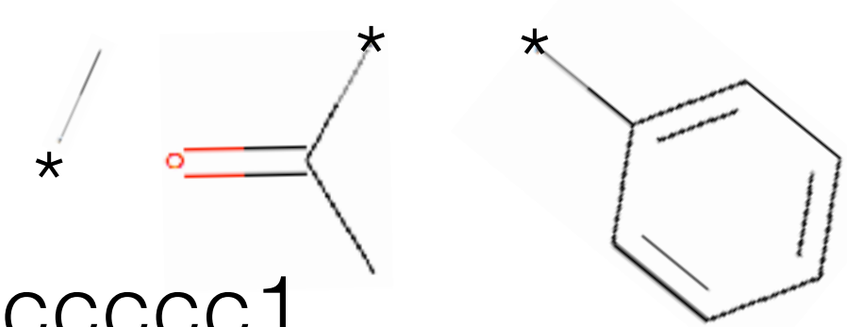
id: FUHJAT
input SMILES: Cc1c(sc(n1)N(C)c2ccccc2)C(=O)C
#cuts: 3

variable:
#heavies: 7
SMILES: [*]c1sc(N([*])[*])nc1C
symmetry class: 122



order: 102

constant:
#heavies: 10
SMILES: [*]C.[*]C(C)=O.[*]c1ccccc1
symmetry class: 123



JSON-Lines format

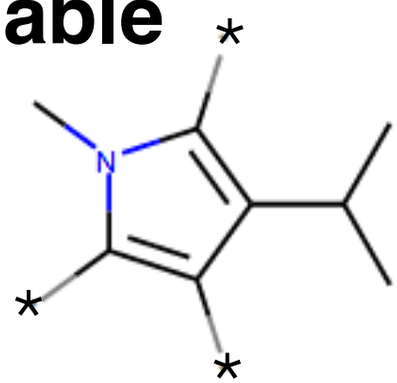
```
["VERSION", "mmpdb-fragment/2"]
["SOFTWARE", "mmpdb-2.1b6"]
["OPTION", "cut_smarts", "[#6+0;!$(*=,#[!#6])]!@!=!#[!#0;!#1;!$([CH2]);!$([CH3][CH2])]"]
["OPTION", "max_heavies", "100"]
["OPTION", "max_rotatable_bonds", "10"]
["OPTION", "method", "chiral"]
["OPTION", "num_cuts", "3"]
["OPTION", "rotatable_smarts", "[!$([NH]!@C(=O))&!D1&!$(*#*)]-&!@[!$([NH]!@C(=O))&!D1&!$(*#*)]"]
["OPTION", "salt_remover", "<default>"]
```

```
[ "RECORD", "FUHJAT",
  "Cc1c(sc(n1)N(C)c2ccccc2)C(=O)C",
  17,
  "CC(=O)c1sc(N(C)c2ccccc2)nc1C", [
    [ 1,
      "N",
      1,
      "1",
      "[*]C",
      "0",
      16,
      "1",
      "[*]C(=O)c1sc(N(C)c2ccccc2)nc1C",
      "Cc1nc(N(C)c2ccccc2)sc1C=O"
    ],
  ....
```

```
    [ 3,
      "N",
      7,
      "122",
      "[*]c1sc(N([*])[*])nc1C",
      "102",
      10,
      "123",
      "[*]C.[*]C(C)=O.[*]c1ccccc1",
      null
    ],
...
]
```

Complicates "cansmirks"

LHS variable

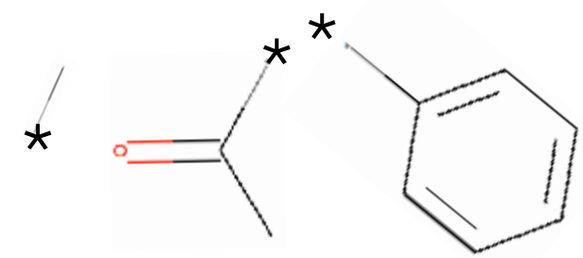


[*]c1c(C(C)C)c([*])n(C)c1[*]

symmetry class 123

order 120

Constant



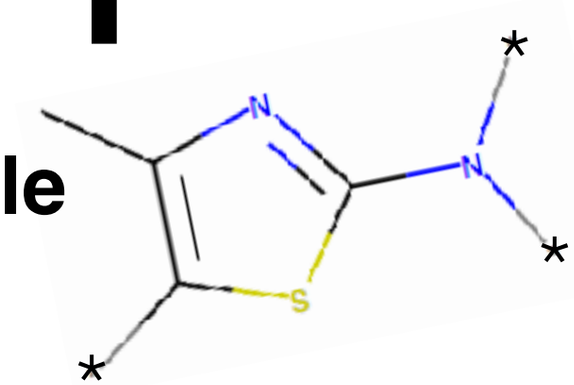
[*]C.[*]C(C)=O.[*]c1ccccc1

symmetry class 123

order 102

The LHS will be numbered
[*:1], [*:2], [*:3].
What is the smallest possible
RHS numbering?

RHS variable



[*]c1sc(N([*])[*])nc1C

symmetry class 122

Lookup table

- 4532 possible combinations of symmetry classes and orderings.
- If multiple solutions, choose the smallest.

LHS variable symmetry class and order: 123, 120
constant symmetry class: 123
RHS variable symmetry class and order: 122, 102

$$(123, 120, 123, 122, 102) \Rightarrow \begin{cases} \text{RHS variable order} = 012 \\ \text{constant order} = 132 \end{cases}$$

Transform: [*:1]c1c(C(C)C)c([*:2])n(C)c1[*:3] >>
[*:1]c1sc(N([*:2])[*:3])nc1C

Constant: [*:3]C.[*:1]C(C)=O.[*:2]c1ccccc1

Advantages

The unlabeled constant and variables are all canonical.
The labeled variable transform is globally canonical.

Use string match to find identical transforms.

To apply possible transforms to a given structure: fragment, match on the unlabeled constant, apply the new cansmirks to get the order for the new variable part, then "weld" the constant to the new variable part.

"Welding" two fragments

How do you re-connect two "*"-labeled fragments?

[*:1]CN + [*:1]O → OCN

SMILES syntax manipulation!

Convert "[*:1]" to "%91", "[*:2]" to "%92", "[*:3]" to "%93".

→ C<u>%91</u>N.O<u>%91</u>

Then re-canonicalize to give OCN.

I found it easier than using the RDKit graph API.

Welding details

In the RDKit, the [*] only appears in a few places.

The first atom term, or after a dot-disconnect:
[*:1]C.... → C%91...

The last atom term: ...C[*:2]→ ...C%92

A branch directly after an atom: ...C([*:3])C... →...C%93C

Complex example: [*:3]P([*:1])([*:2])N → P%93%91%92N

In principle, C(CC)[*] is possible.
I check for that case, but it's not
supported as RDKit doesn't generate it.

Welding and chirality

SMILES chirality is based on the configuration order of the bonds

[*:1][C@](N)(O)S → [C@]%91(N)(O)S

- but -

[*:1][C@H](O)S → [C@@H]%91(O)S

The configuration order around the C changed from
"*"=1, "implicit H"=2, "O"=3, "S"=4
to
"implicit H"=1, "*"=2, "O"=3, "S"=4
so the chirality needs to be inverted.

How do you fragment?

1. Roll your own code

```
rwmol = Chem.RWMol(mol)
rwmol.RemoveBond(atom1, atom2)
wildcard1 = rwmol.AddAtom(Chem.Atom(0))
wildcard2 = rwmol.AddAtom(Chem.Atom(0))
rwmol.AddBond(atom1, wildcard1, Chem.BondType.SINGLE)
rwmol.AddBond(atom2, wildcard2, Chem.BondType.SINGLE)
return rwmol.GetMol()
```

   Don't use! Can invert chirality and drop double-bond stereochemistry F/C=C/F!

2. FragmentOnBonds() - mmpdb uses this with a workaround for issue #1039

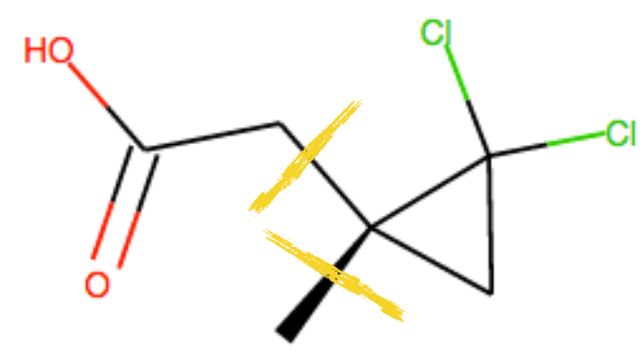
3. Something else?

Have you verified it?

input molecule → fragment → weld → output molecule

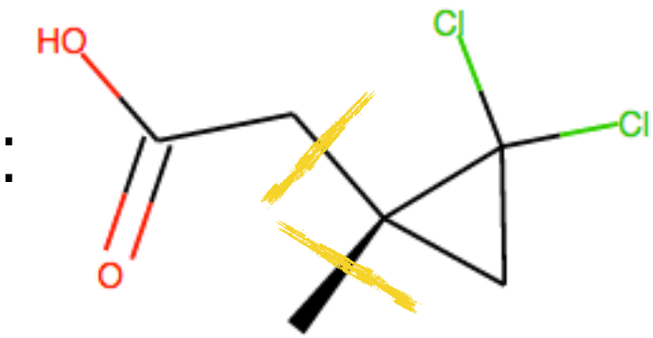
The input and output molecules should match.

GANKIP
C[C@@]1(CC1(Cl)Cl)CC(=O)O

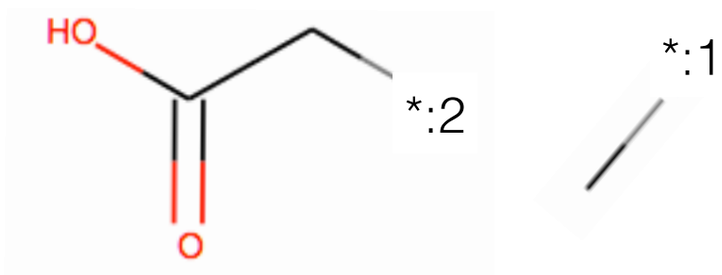


Labeled vs. Unlabeled

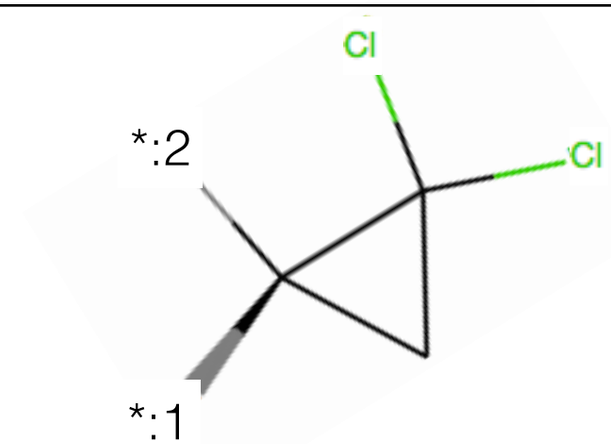
Input:



Labeled

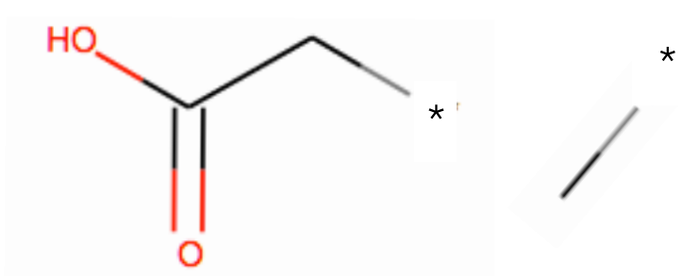


[*:1]C.[*:2]CC(=O)O



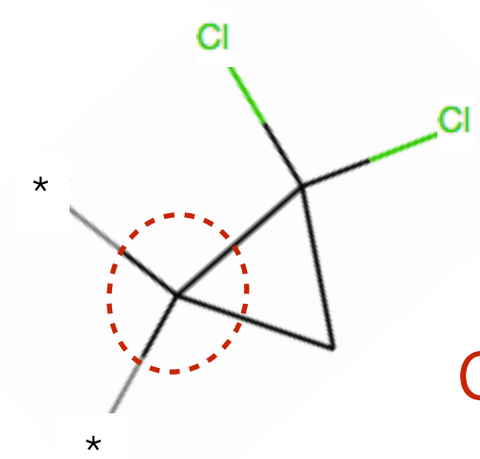
[*:1][C@]1([*:2])CC1(Cl)Cl

Unlabeled



[*]C.[*]CC(=O)O

Order 01



Lost chirality!

[*]C1([*])CC1(Cl)Cl

Recover Chirality

Identify the atoms in the output SMILES which are no longer chiral. (Tokenize the SMILES and use "_smilesAtomOutputOrder".)

C[C@@]1(CC1(Cl)Cl)CC(=O)O

[*]C.[*]CC(=O)O.[*]C1([*])CC1(Cl)Cl

Figure out if it needs an implicit hydrogen and convert to bracket form.

[*]C.[*]CC(=O)O.[*][C]1([*])CC1(Cl)Cl

Chiral enumeration

For each atom there two possible chiralities.

[*]C.[*]CC(=O)O.[*][C@]1([*])CC1(Cl)Cl ← This one!

[*]C.[*]CC(=O)O.[*][C@@]1([*])CC1(Cl)Cl

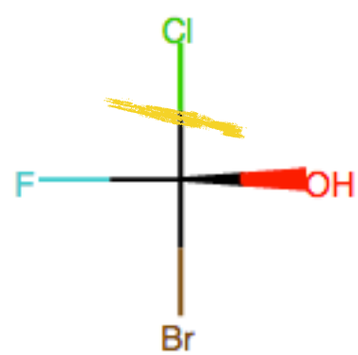
Weld them together and canonicalize to see which matches the input.

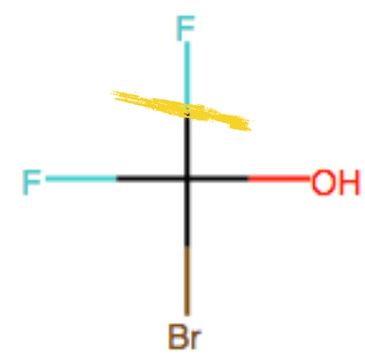
In general, there are *n* such atoms so enumerate all 2^n possibilities from [@][@][@]... to [@@][@@][@@].

Use the first enumeration which matches.
The result is canonical.

Incomplete chirality

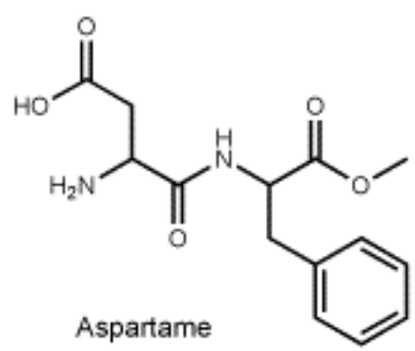
Can these form a matched pair?



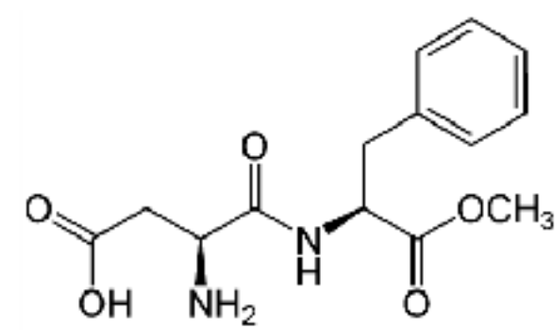


[*:1]Cl>>[*:1]F

Often the chirality is not fully specified.



http://www.chem.umass.edu/people/cmartin/Courses/Chem250/Sugars/index.html



https://en.wikipedia.org/wiki/Aspartame

"Up enumeration"

During fragmentation, find atoms which were not chiral during input but which became a possible stereocenter.

Generate the 3^n-1 possible chiral assignments.
(CHI_UNSPECIFIED, CHI_TETRAHEDRAL_CW, CHI_TETRAHEDRAL_CCW)

Canonicalize and remove duplicates.

Generate the fragmentions for:
(constant, variable),
all (constant, up-enumerated variable)
all (up-enumerated constant, variable)
and label them "N", "V", and "C", respectively.

Up-enumerated indexing

During the indexing phase, find all variables with the same constant SMILES.

One of the pair must not be up-enumerated.

Help on mmpdb

https://github.com/rdkit/mmpdb

The README (shown on the project page) walks through the steps of how to fragment, index and search.

Each of the subcommands has help:
mmpdb fragment --help

There are also special help subcommands:

help-analysis	overview on how to use mmpdb for structure analysis
help-admin	overview on how to use administer an mmpdb database
help-smiles-format	description of the SMILES files parsing options
help-fragments-format	description of the fragments file format
help-property-format	description of the property file format

Thanks

Roche funded the project and contributed it to RDKit.

Developed with Jérôme Hert and Christian Kramer@Roche.

GSK contributed Hussain and Rea's mmpa code to RDKit, which became the starting point for mmpdb.

Greg Landrum for RDKit, answering my questions about how to work with fragments correctly, and dealing with the issues I submitted.