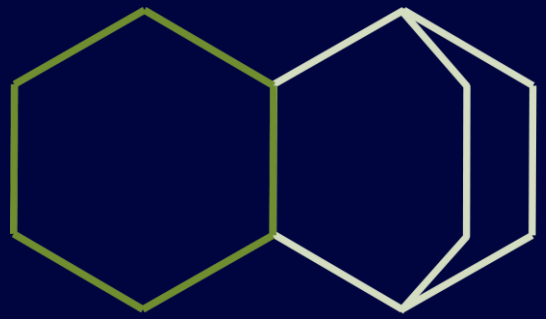
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN FACULTY
CENTER FOR BIOINFORMATICS



CHEMICALLY MEANINGFUL RING PERCEPTION: AN OPEN-SOURCE IMPLEMENTATION OF THE UNIQUE RING FAMILIES APPROACH

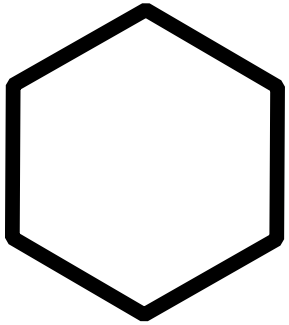
Florian Flachsenberg
Niek Andresen
Matthias Rarey

Ring Perception

- What is a ring?

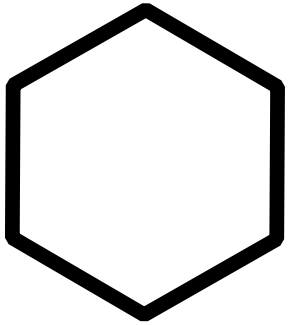
Ring Perception

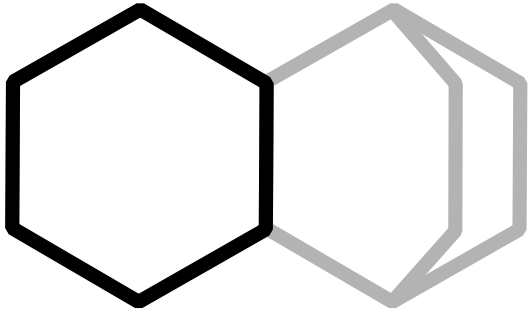
- What is a ring?



Ring Perception

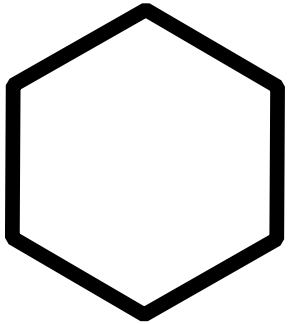
- What is a ring?

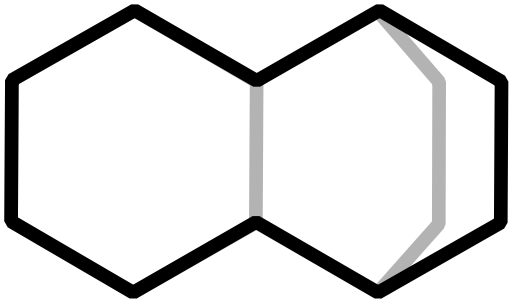


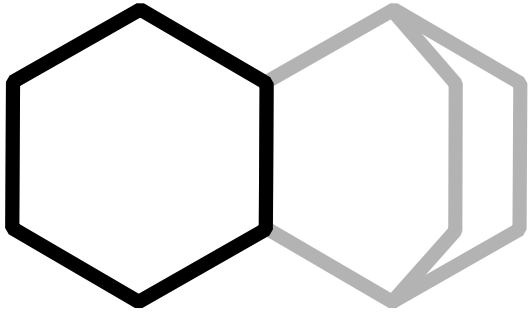


Ring Perception

- What is a ring?

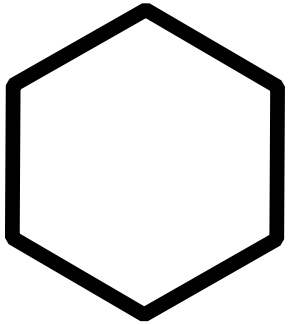


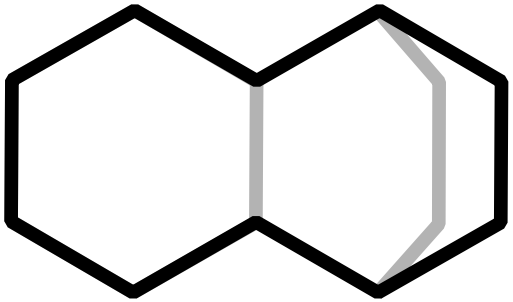


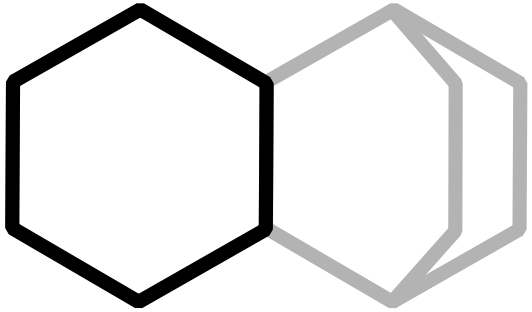


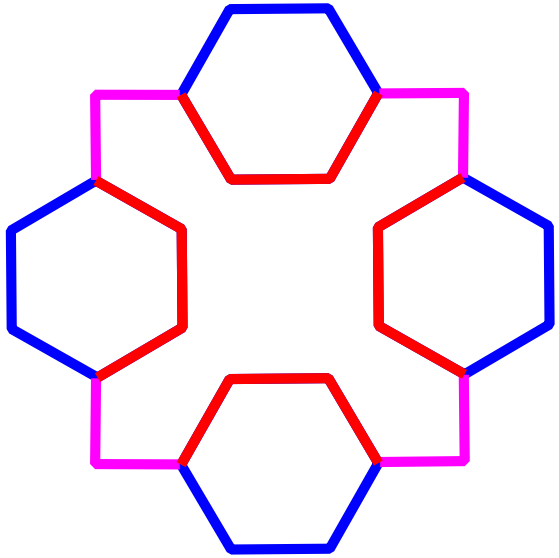
Ring Perception

- What is a ring?



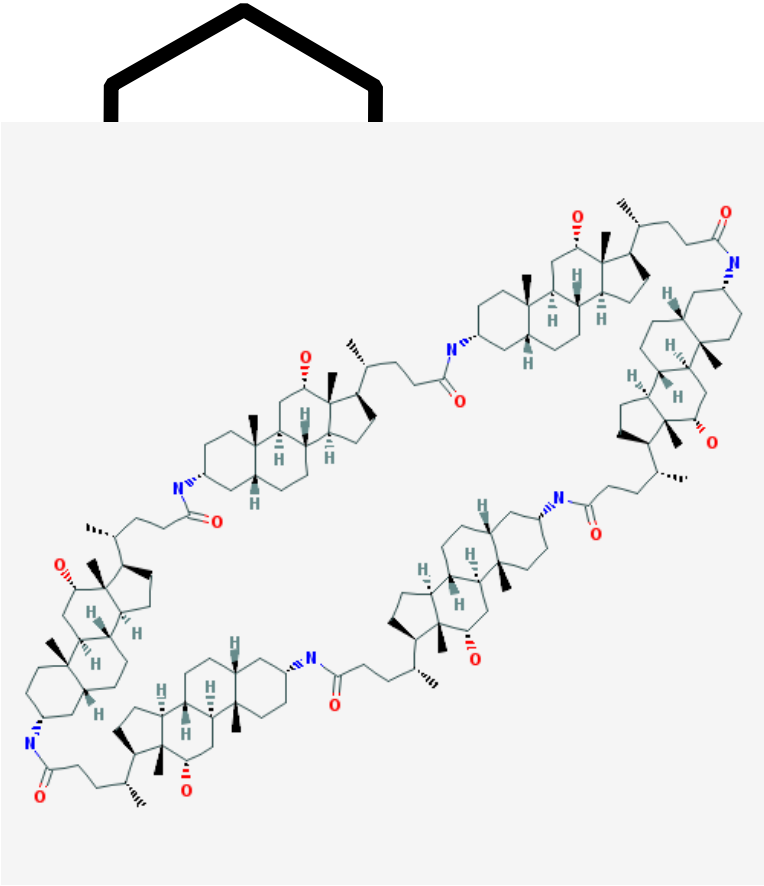


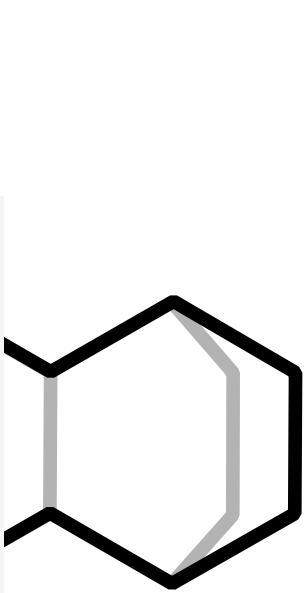


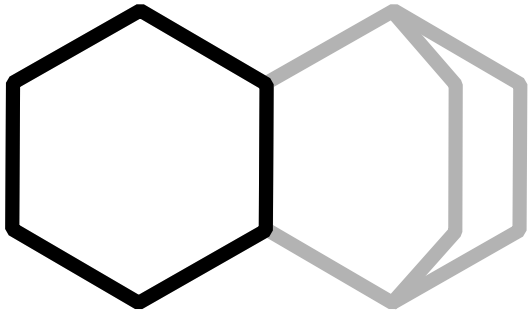


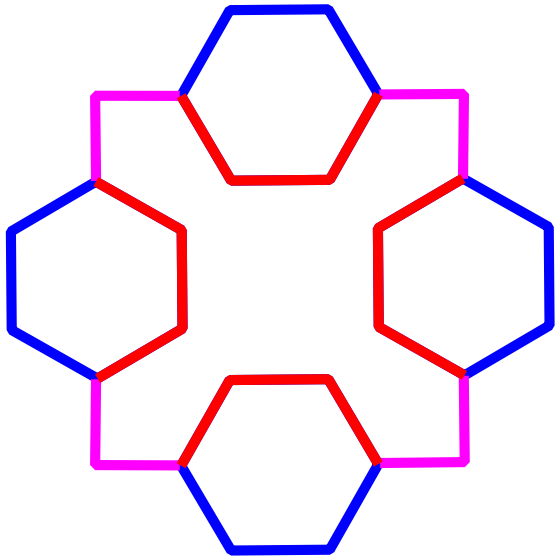
Ring Perception

- What is a ring?



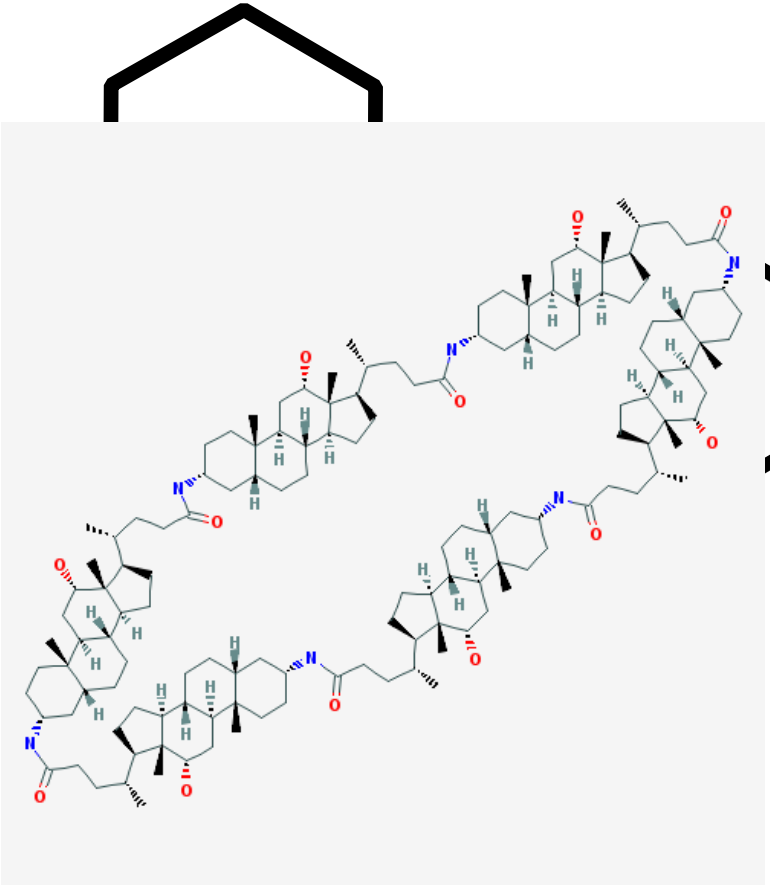


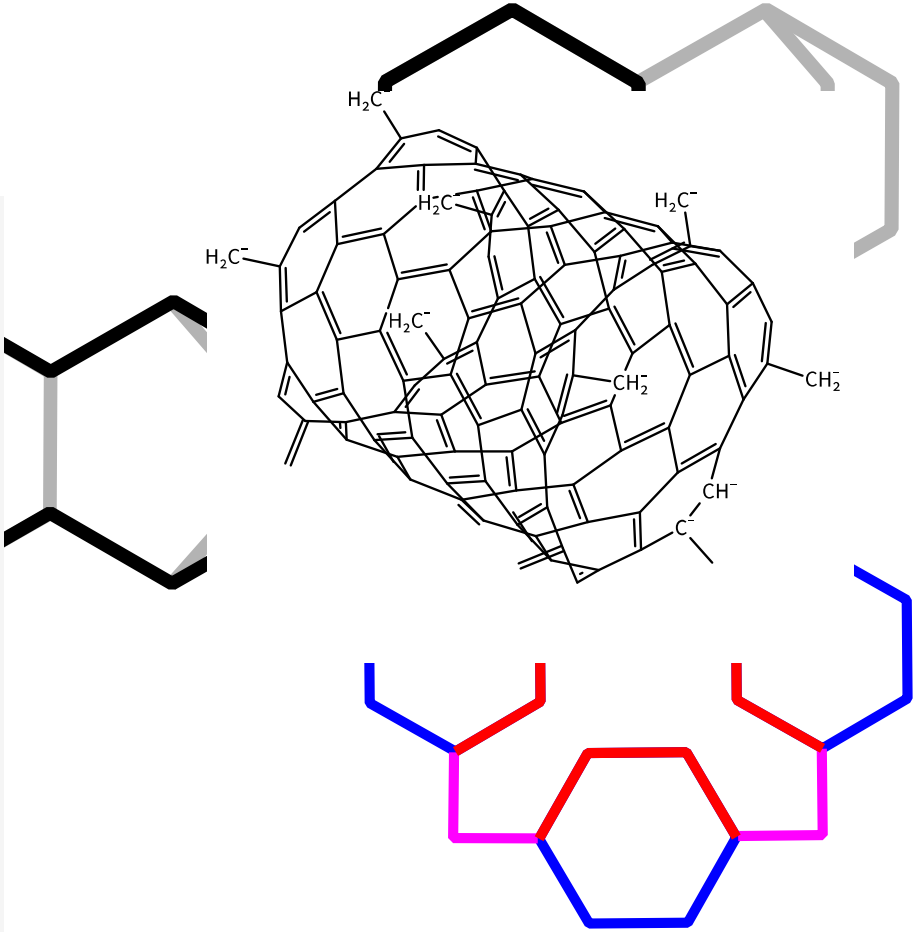




Ring Perception

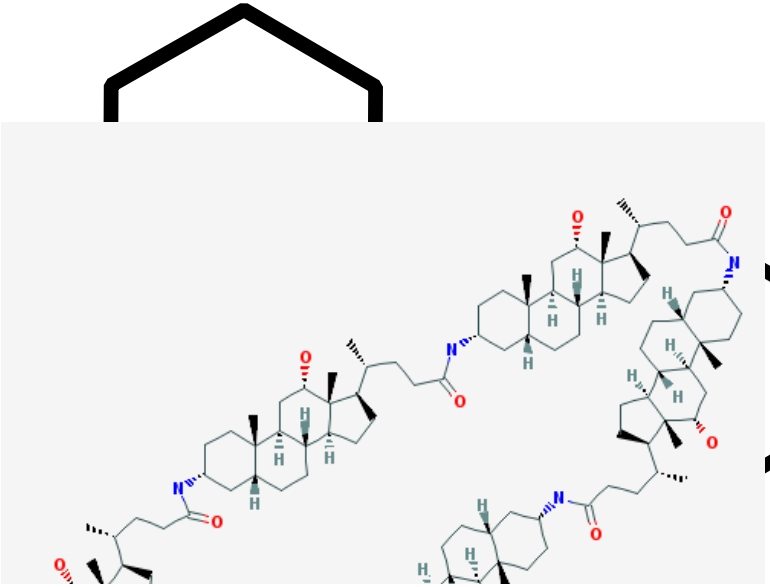
- What is a ring?

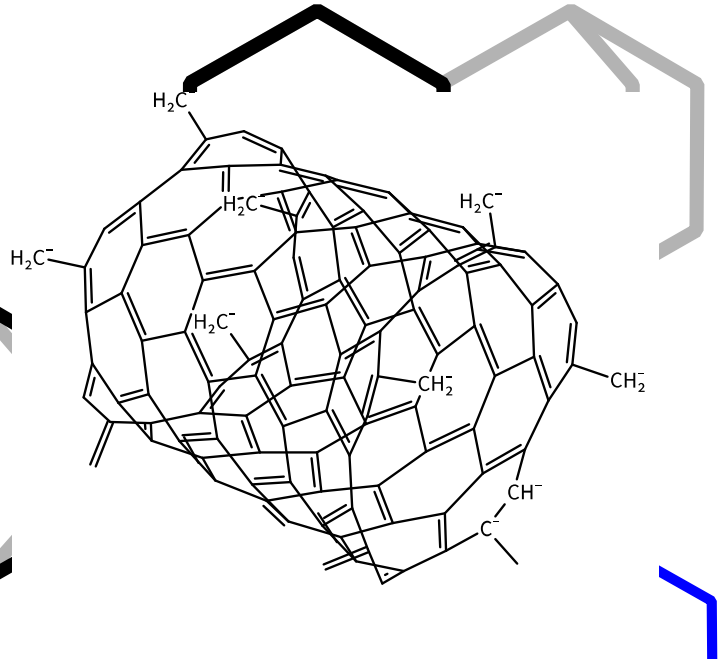




Ring Perception

- What is a ring?



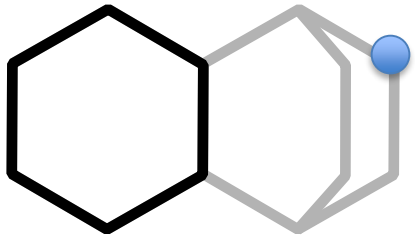


Connected cycle (subgraph where all vertices have a degree of two)

Ring Perception - Applications

- Calculation of atomic coordinates in 2D and 3D

- SMARTS[1] matching
  - To how many rings does an atom belong?



- Molecular descriptors
  - Similar to SMARTS, how many rings?

- Aromaticity detection

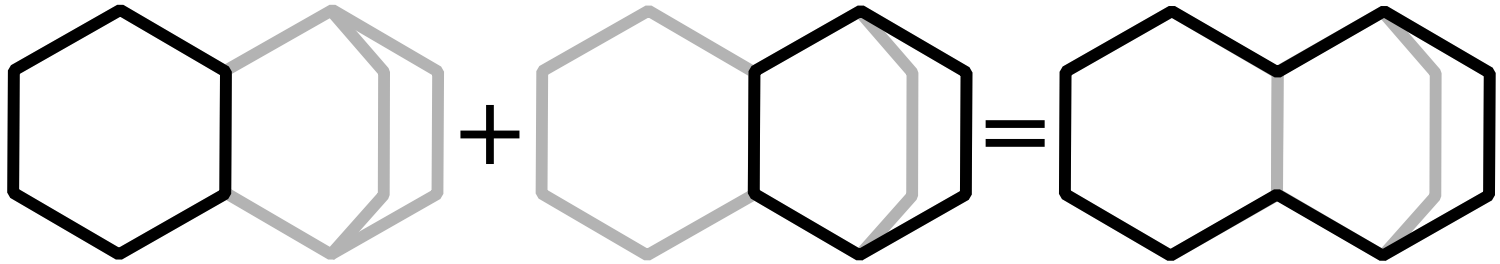
[1] http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html

Ring Perception - Requirements

- **Unique for each molecule**
- **Chemically meaningful**
- **Efficient to calculate**

Addition of Cycles

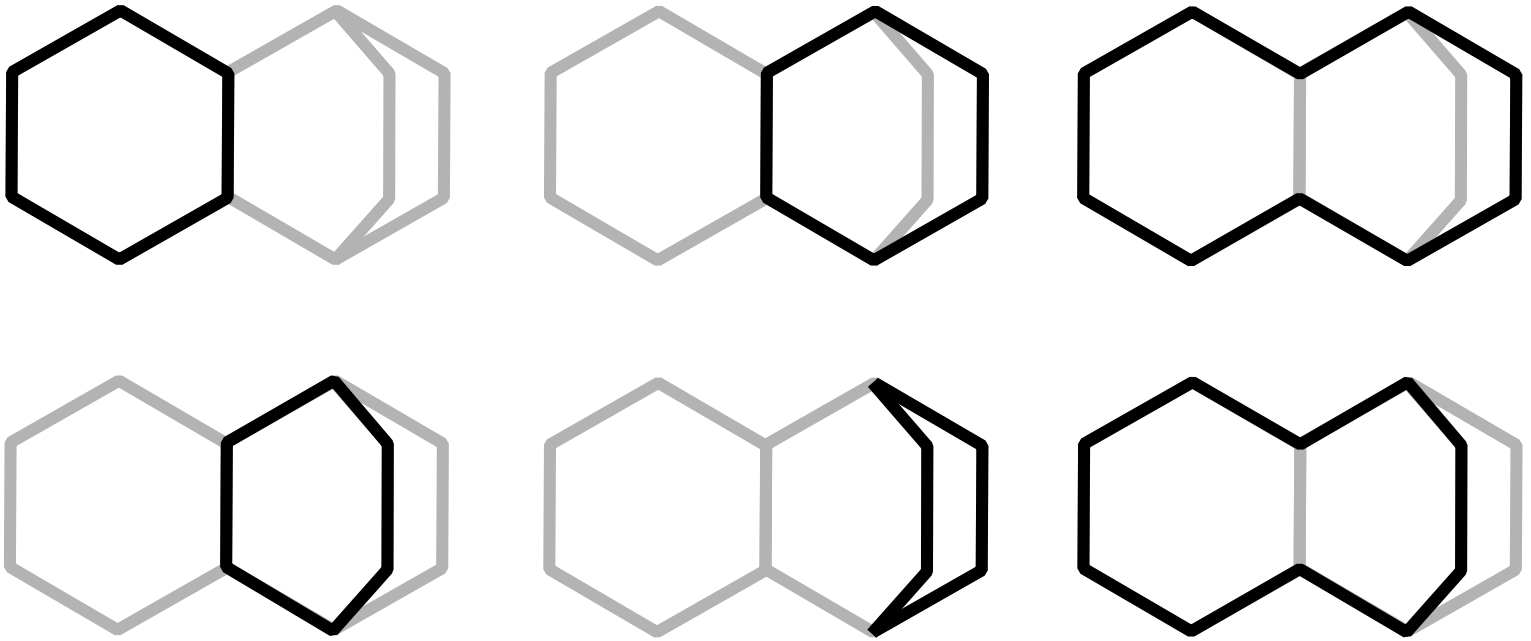
- Cycles can be added to form new cycles
- Symmetric difference („xor“) of their edges



- Cycle base: set of cycles, such that all cycles can be constructed by addition

Smallest Set of Smallest Rings – SSSR[2]

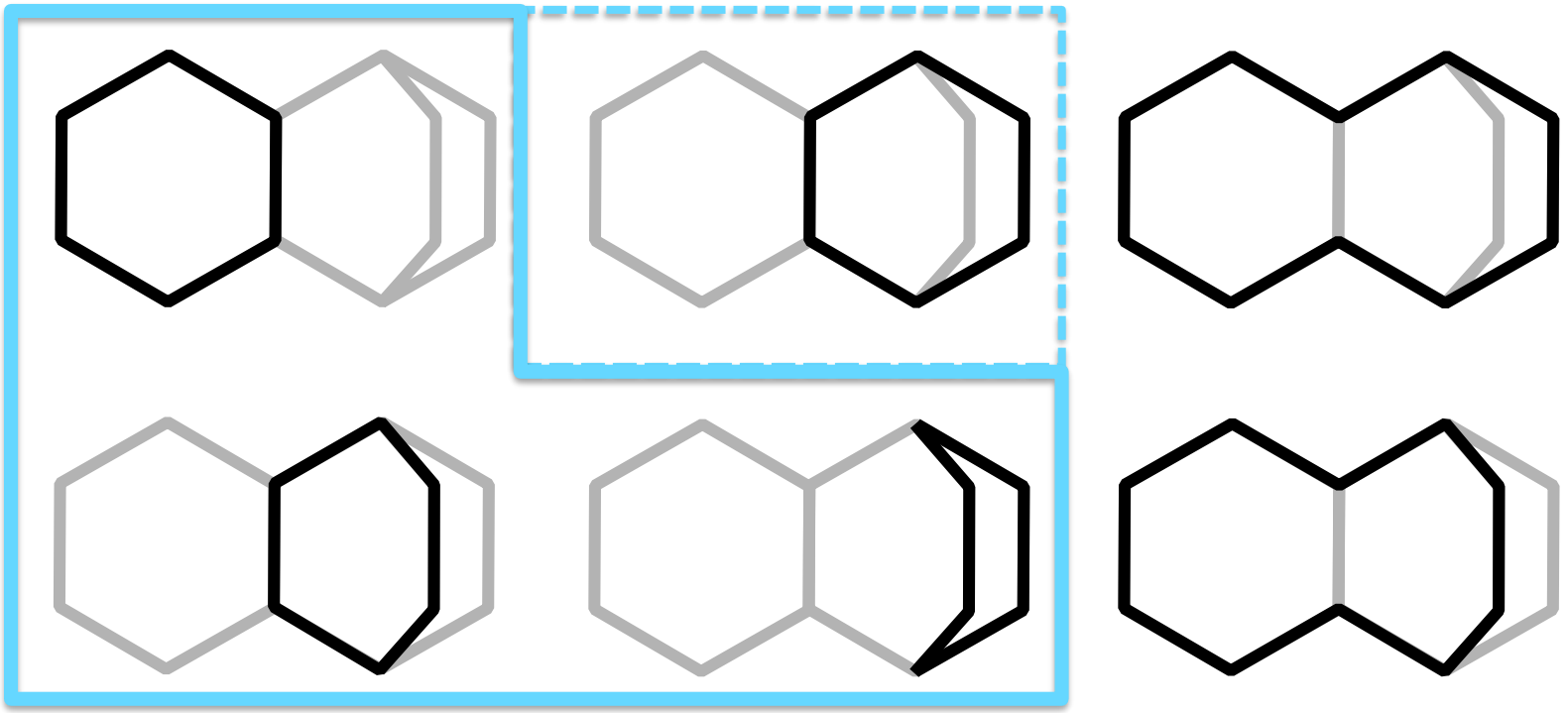
- An SSSR is **efficient to calculate**, mostly **chemically meaningful**, but it isn‘t always **unique**



[2] Zamora, A. Chem. Inf. Comput. Sci. (1976), 16, 40-43

Smallest Set of Smallest Rings – SSSR[2]

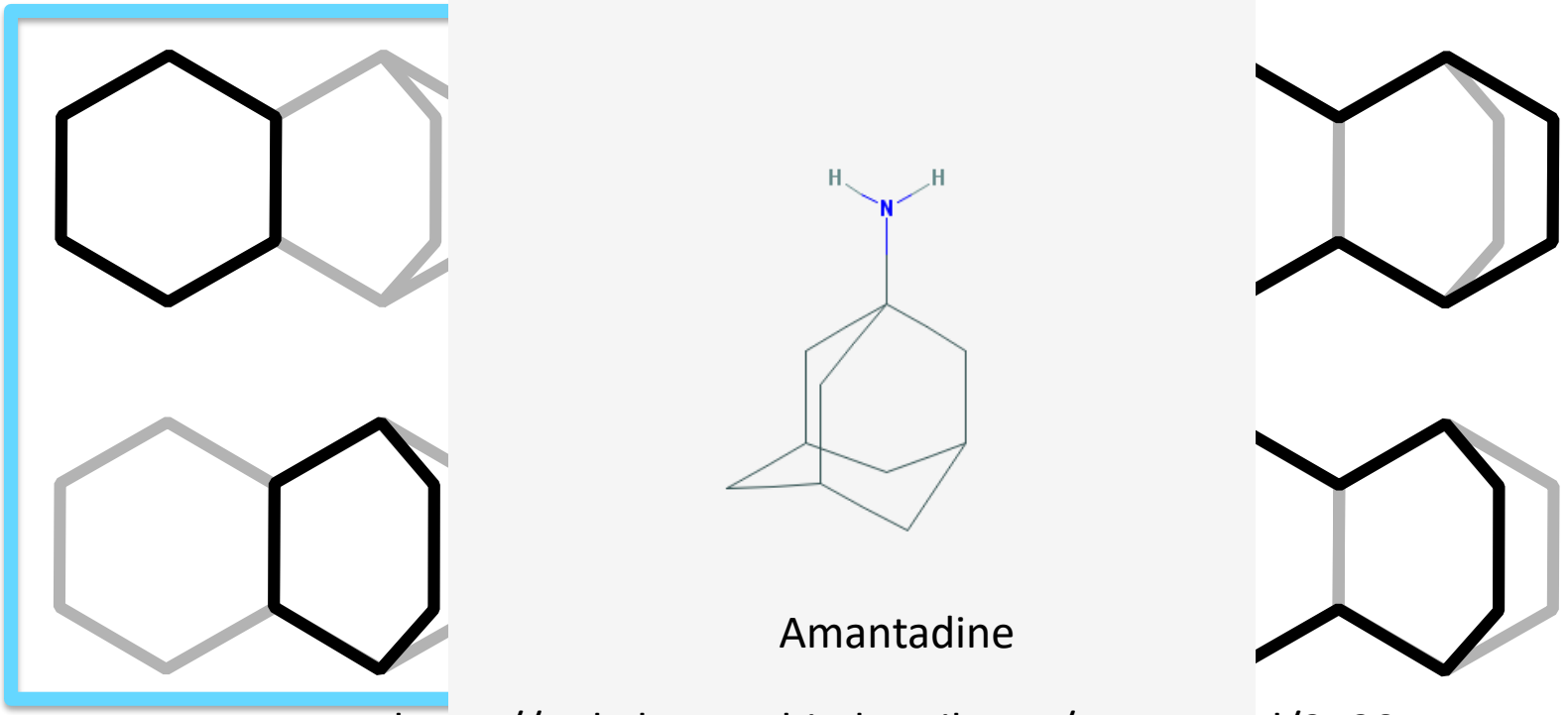
- An SSSR is **efficient to calculate**, mostly **chemically meaningful**, but it isn‘t always **unique**



[2] Zamora, A. Chem. Inf. Comput. Sci. (1976), 16, 40-43

Smallest Set of Smallest Rings – SSSR[2]

- An SSSR is **efficient to calculate**, mostly **chemically meaningful**, but it isn‘t always **unique**



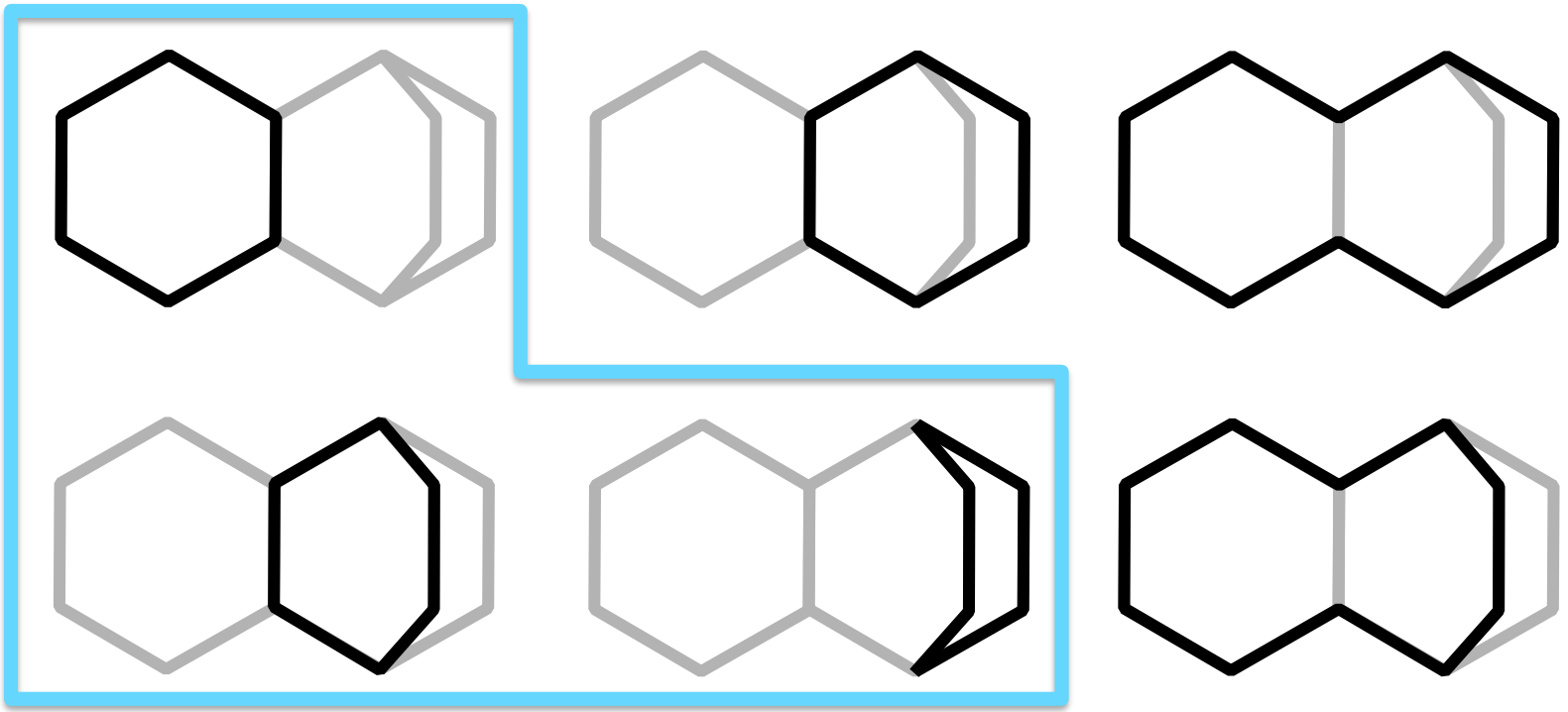
Amantadine

https://pubchem.ncbi.nlm.nih.gov/compound/2130

[2] Zamora, A. Chem. Inf. Comput. Sci. (1976), 16, 40-43

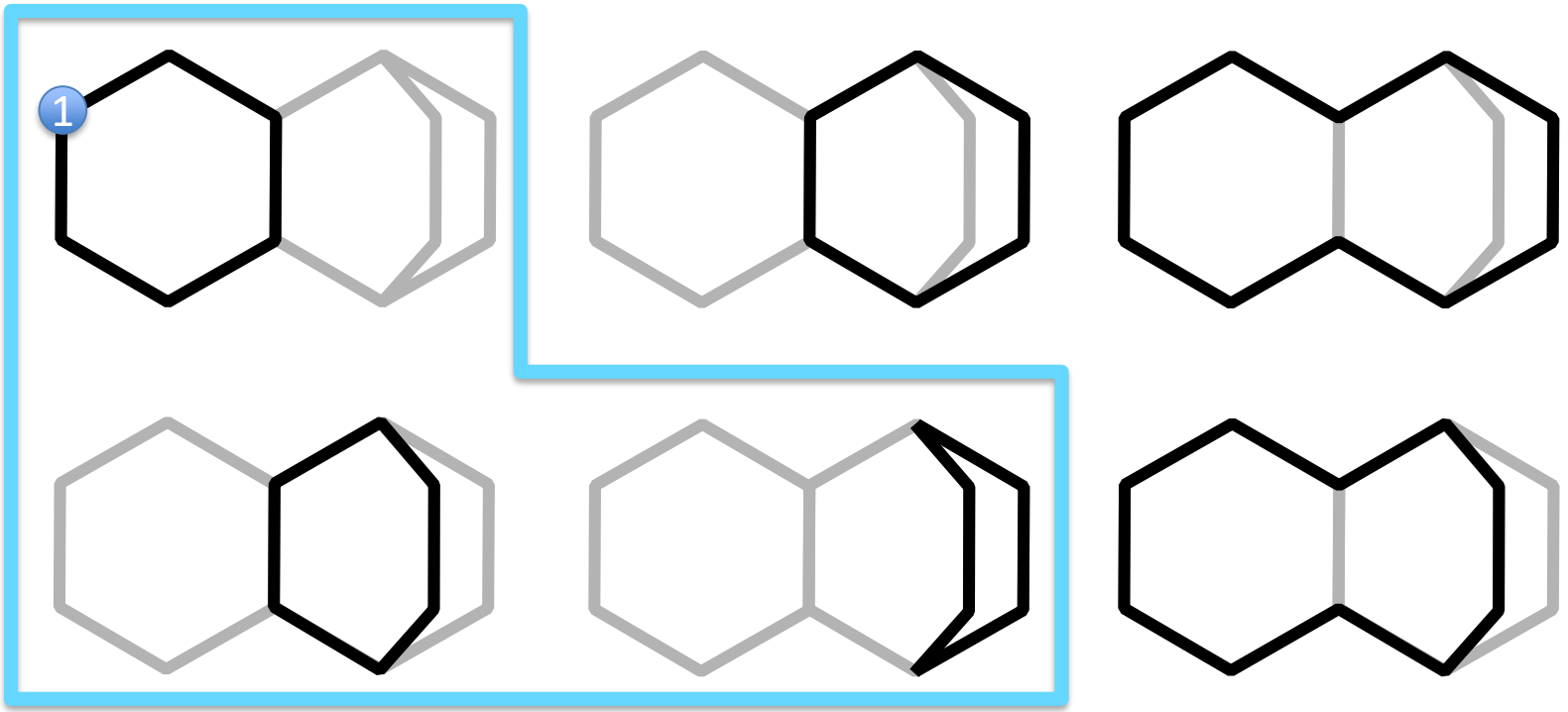
Smallest Set of Smallest Rings - SSSR

- Which atoms does the SMARTS expression [#6R2] match?



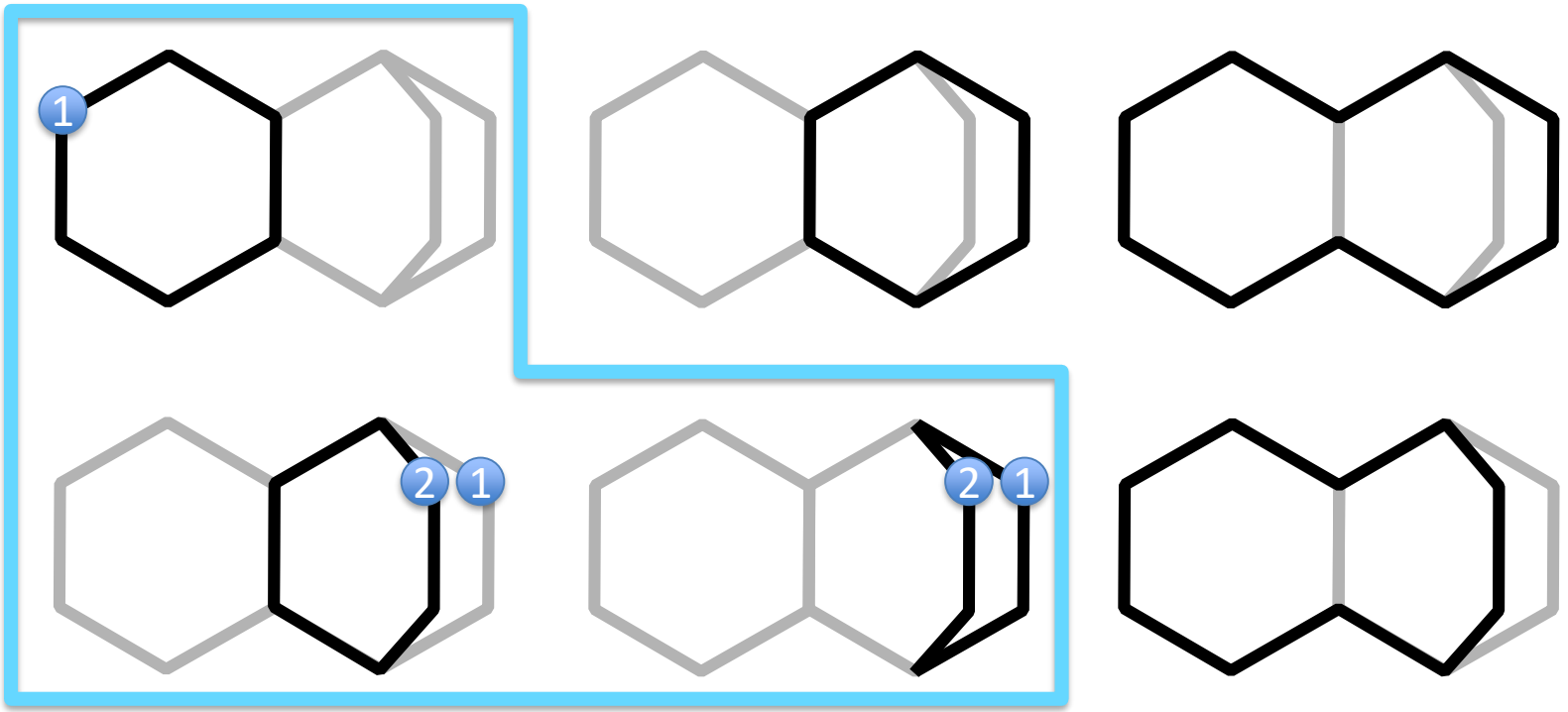
Smallest Set of Smallest Rings - SSSR

- Which atoms does the SMARTS expression [#6R2] match?



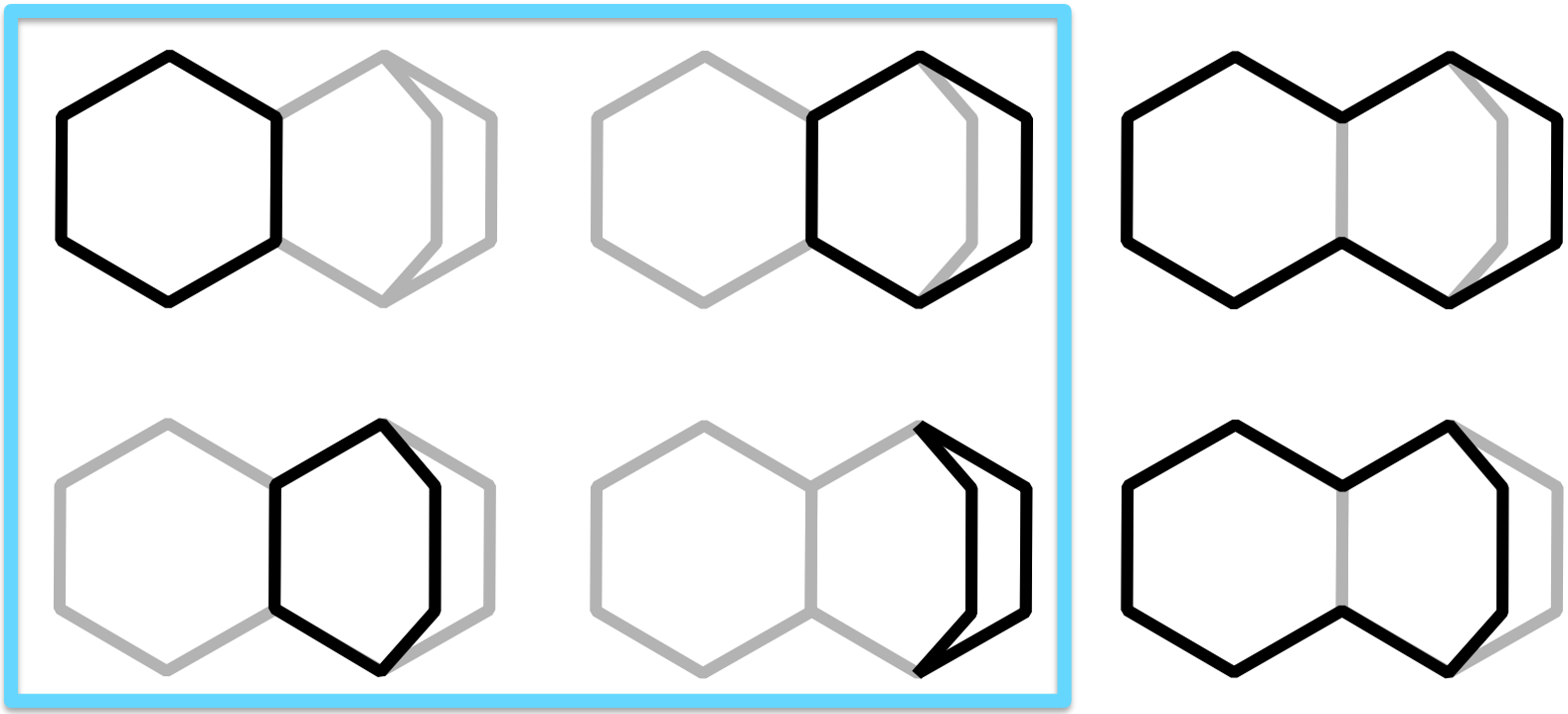
Smallest Set of Smallest Rings - SSSR

- Which atoms does the SMARTS expression [#6R2] match?



Relevant Cycles[3]

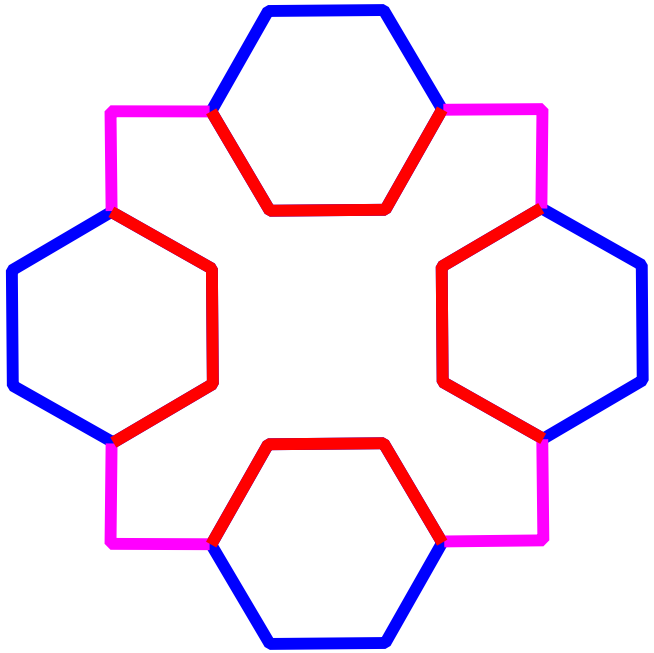
- An RC is a cycle that cannot be constructed from smaller cycles



[3] Vismara, P. Electron. J. Comb. (1997), 4, 1-15

Relevant Cycles

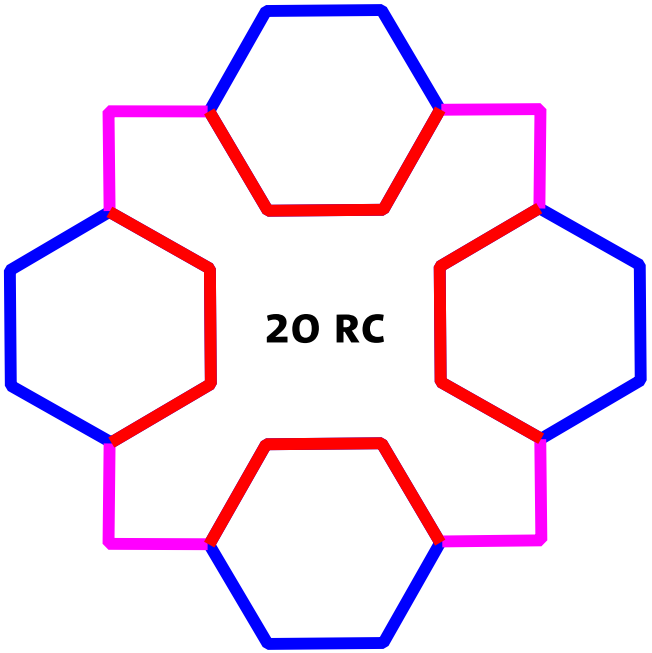
- The RCs are **unique**, but sometimes neither **chemically meaningful** nor **efficient to calculate.**



Combine red and blue subpaths

Relevant Cycles

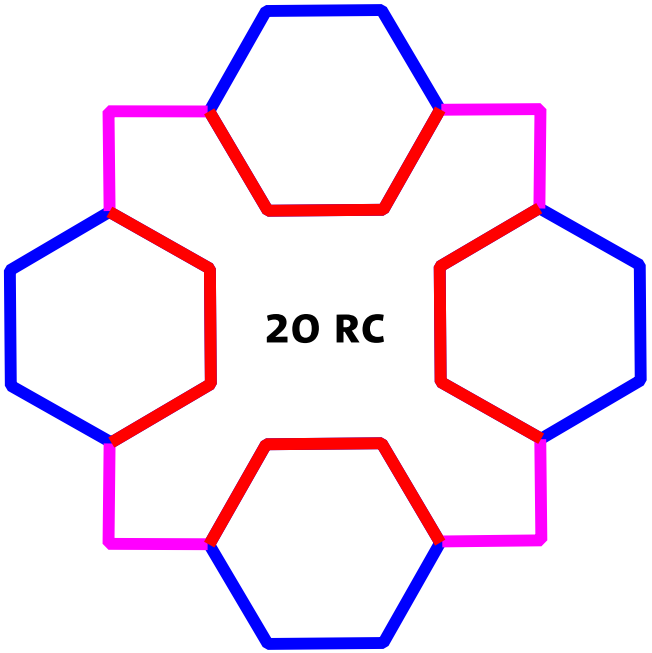
- The RCs are **unique**, but sometimes neither **chemically meaningful** nor **efficient to calculate.**

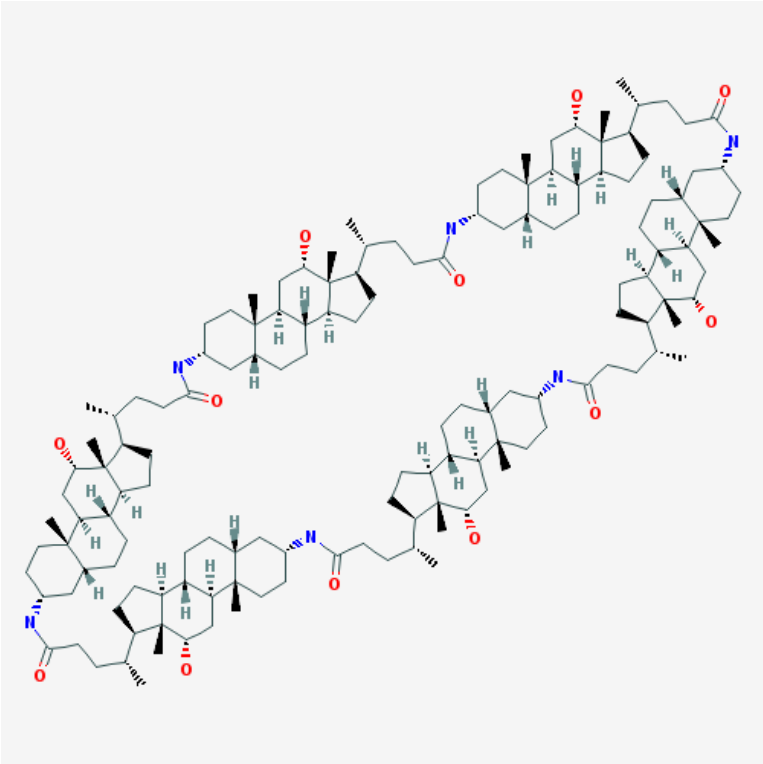


Combine red and blue subpaths

Relevant Cycles

- The RCs are **unique**, but sometimes neither **chemically meaningful** nor **efficient to calculate.**



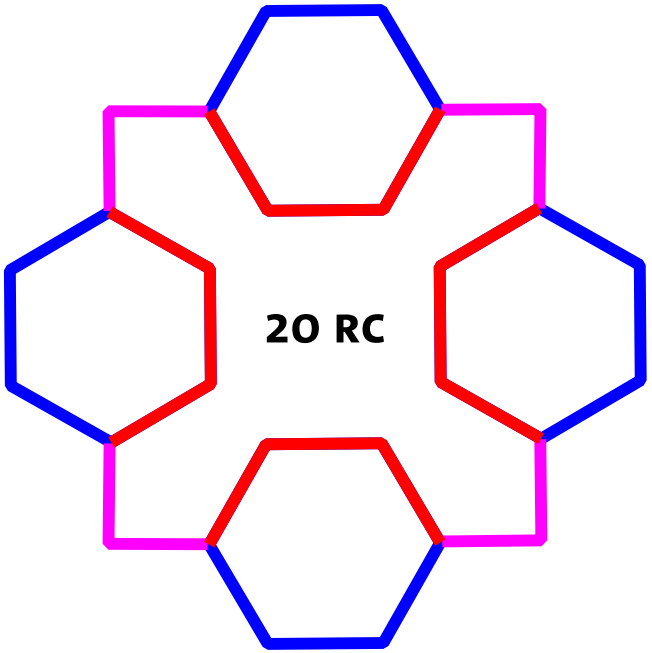


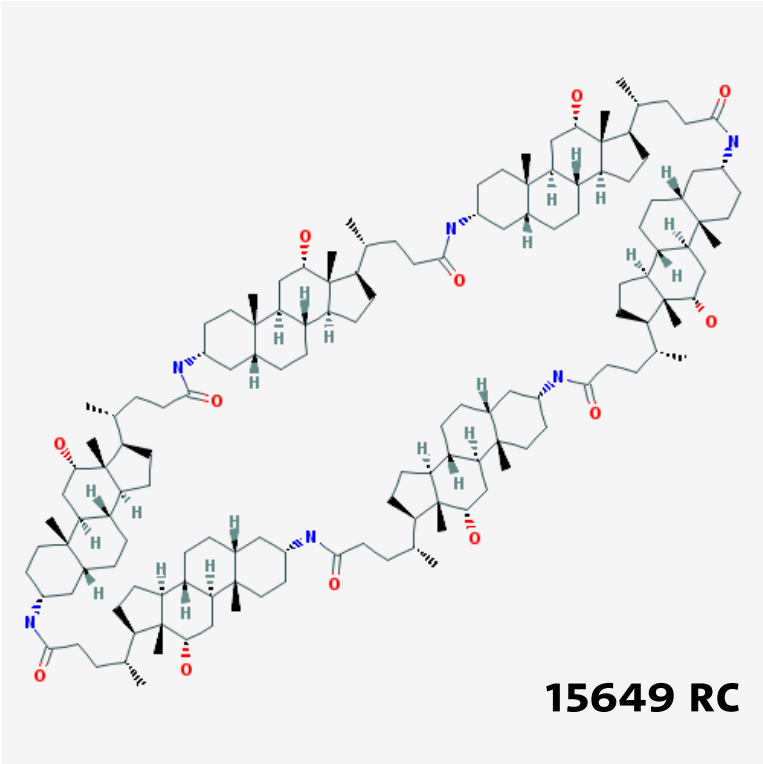
Combine red and blue subpaths

https://pubchem.ncbi.nlm.nih.gov/substance/16783777

Relevant Cycles

- The RCs are **unique**, but sometimes neither **chemically meaningful** nor **efficient to calculate.**



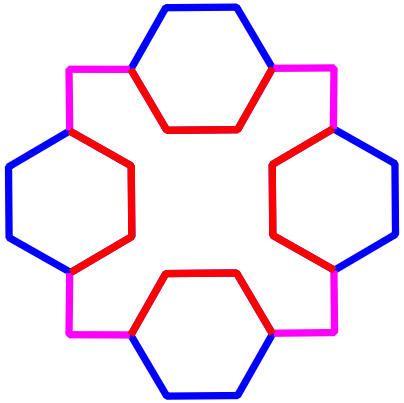


Combine red and blue subpaths

https://pubchem.ncbi.nlm.nih.gov/substance/16783777

Unique Ring Families[4]

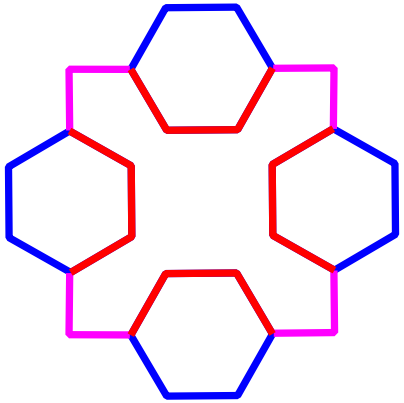
- Combine rings in families
- URF-relation

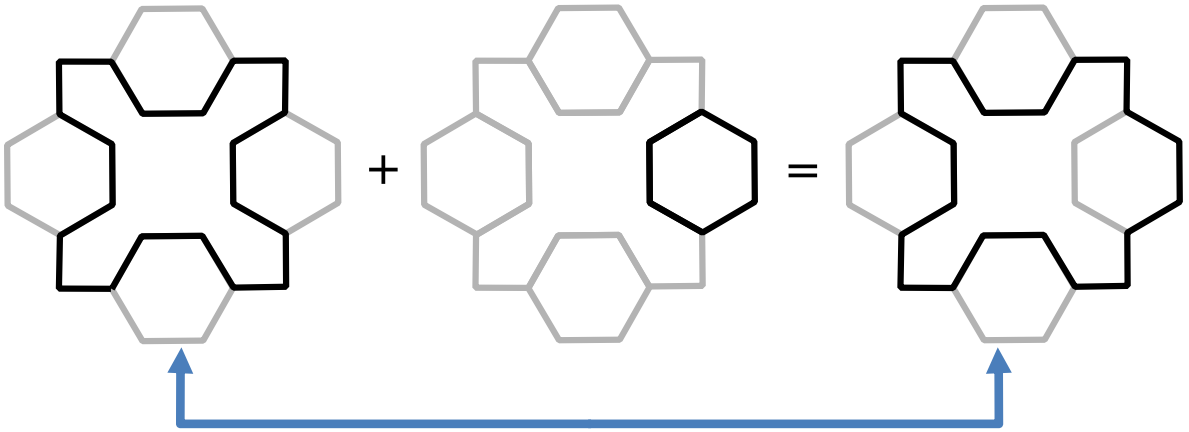


[4] Kolodzik, A.; et al. J. Chem. Inf. Model. (2012), 52, 2013-2021

Unique Ring Families[4]

- Combine rings in families
- URF-relation

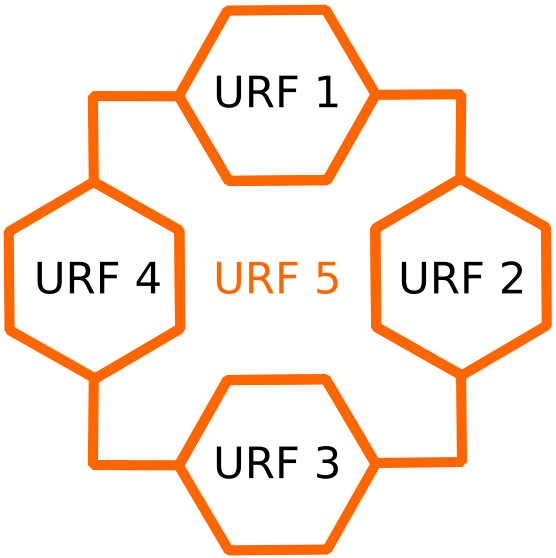




1. Both cycles have the same size: $|C_1| = |C_2|$
2. Both cycles share at least one edge
3. C_1 and C_2 depend linearly on each other and smaller cycles

[4] Kolodzik, A.; et al. J. Chem. Inf. Model. (2012), 52, 2013-2021

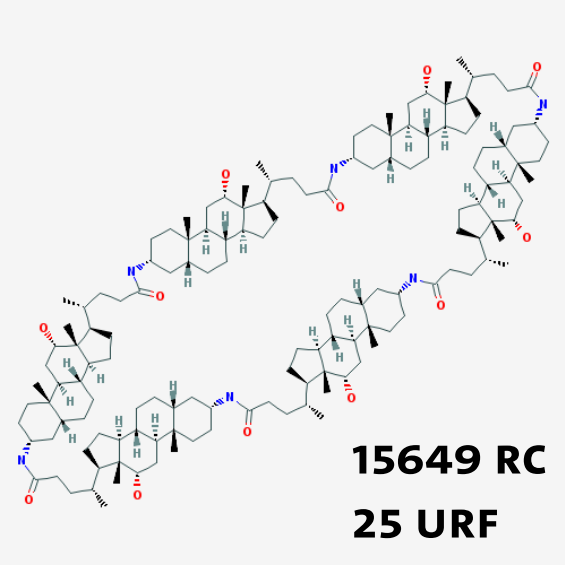
Unique Ring Families



- All 16 different 20-membered rings are URF-related!
- 5 URFs in total
- URFs can be represented by their cycles, but also by their edges

Unique Ring Families

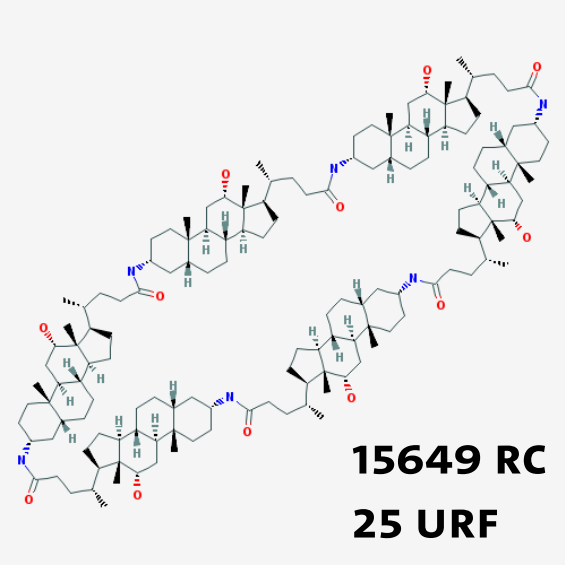
- URFs are
  - **Unique**
  - **Chemically meaningful**
  - **Efficient to calculate**



Unique Ring Families

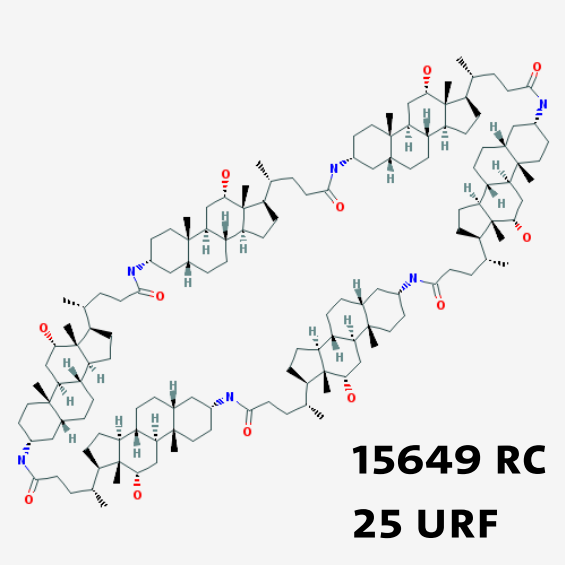
- URFs are
  - **Unique**
  - **Chemically meaningful**
  - **Efficient to calculate**

- **BUT**



Unique Ring Families

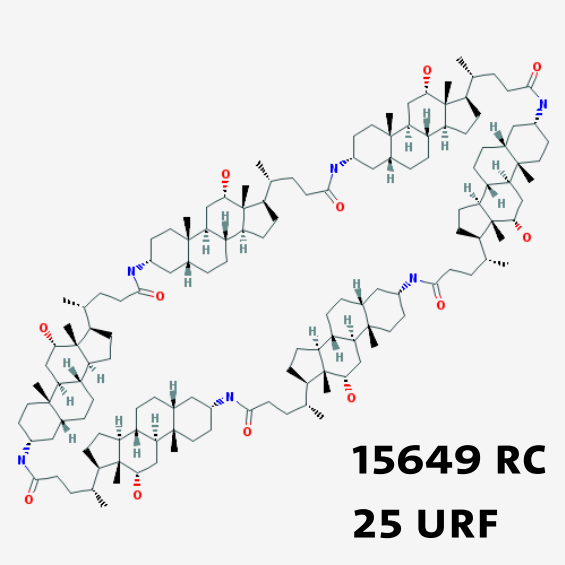
- URFs are
  - **Unique**
  - **Chemically meaningful**
  - **Efficient to calculate**



- **BUT**
  - Complicated implementation

Unique Ring Families

- URFs are
  - **Unique**
  - **Chemically meaningful**
  - **Efficient to calculate**



- **BUT**
  - Complicated implementation
  - Until now only part of NAOMI[5], used for example for 2D depiction in Mona[6]

[5] Urbaczek, S.; et al J. Chem. Inf. Model. (2011), 51, 3199-3207

[6] Hilbig, M.; et al. J. Chem. Inf. (2013), 5

Unique Ring Families – How to calculate[4]

- Step 1: 2-connected components (Tarjan 1985)[7]
- Step 2: Modified RC detection algorithm (Vismara 1997)[3,4]
  - Detection of the cycle prototypes
  - Calculation of relevant cycle prototypes, linear independence of smaller cycles (gaussian elimination)

[3] Vismara, P. Electron. J. Comb. (1997), 4,1-15

[4] Kolodzik, A.; et al. J. Chem. Inf. Model. (2012), 52, 2013-2021

[7] Tarjan, R; et al. SIAM J. Comput. (1985), 14, 862-874

Unique Ring Families – How to calculate[4]

- Step 1: 2-connected components (Tarjan 1985)[7]
- Step 2: Modified RC detection algorithm (Vismara 1997)[3,4]
  - Detection of the cycle prototypes
  - Calculation of relevant cycle prototypes, linear independence of smaller cycles (gaussian elimination)
  - **Dependence on equal sized cycles (URF conditions 1 and 3)**
- **Step 3: Calculate URF-relation**
  - **Detection of shared edges between RCFs (URF condition 2)**
  - **Transitive closure**

(Kolodzik 2012[4])

[3] Vismara, P. Electron. J. Comb. (1997), 4,1-15

[4] Kolodzik, A.; et al. J. Chem. Inf. Model. (2012), 52, 2013-2021

[7] Tarjan, R; et al. SIAM J. Comput. (1985), 14, 862-874

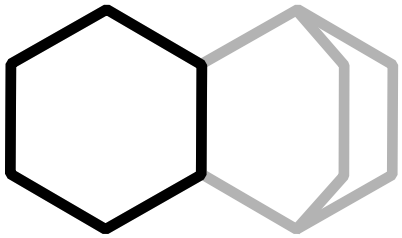
RingDecomposerLib

- Open-source implementation of the URFs and related perception concepts (RCs, SSSR)
- BSD New license

RingDecomposerLib

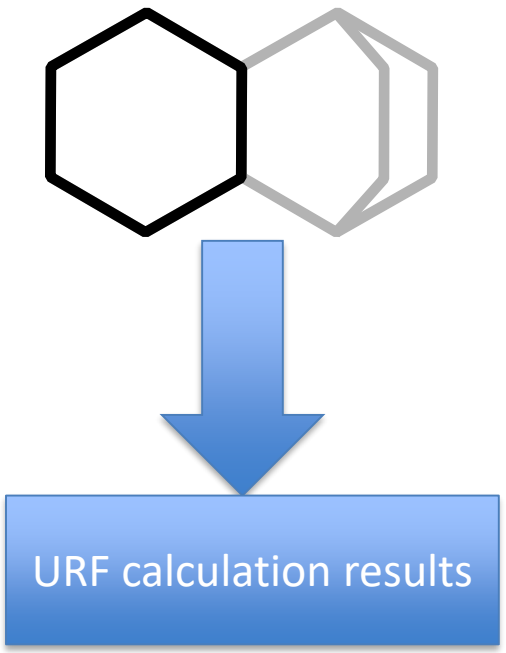
- Open-source implementation of the URFs and related perception concepts (RCs, SSSR)
- BSD New license
- Self-sufficient ANSI C library
  - Portable (tested on Linux, macOS, Windows)
  - Thoroughly validated
  - Well-documented
  - Easy-to-use
  - Fast

RingDecomposerLib



Input:
Bonds of the molecule

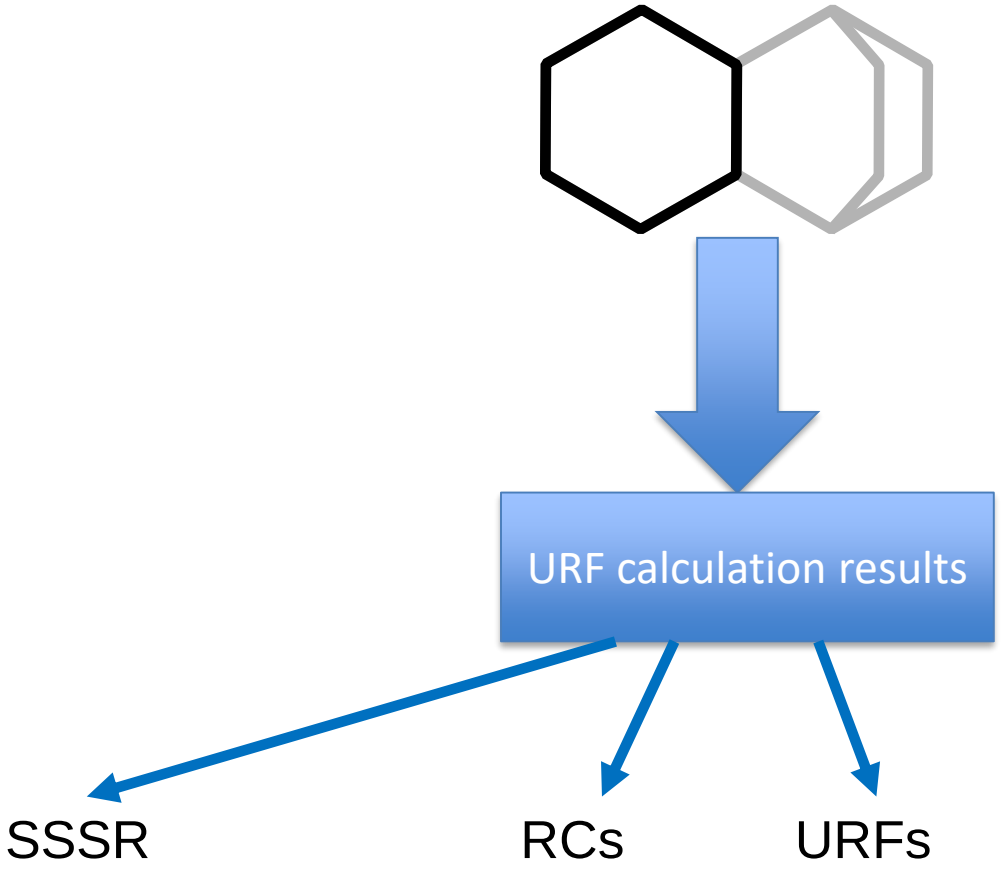
RingDecomposerLib



Input:
Bonds of the molecule

URF calculation

RingDecomposerLib



Input:
Bonds of the molecule

URF calculation

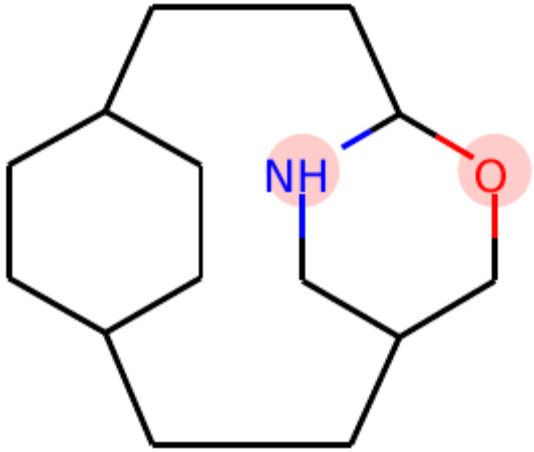
Output

RingDecomposerLib – The Python wrapper py_rdl

- A Python wrapper **py_rdl** is part of the package
- Similar to the C interface, but only requires the bonds and atoms to be hashable
- Very easy to use with RDKit

py_rdl with RDKit – Preparations

In [22]:
```
mol = rdkit.Chem.MolFromMolFile('cyclophane_3.mol')
nitrogen = [a.GetIdx() for a in mol.GetAtoms() if a.GetAtomicNum() == 7][0]
oxygen   = [a.GetIdx() for a in mol.GetAtoms() if a.GetAtomicNum() == 8][0]
```



py_rdl with RDKit – Calculation

- Calculation step (Vismara‘s algorithm, URFs)

In [24]:
```
import py_rdl
rdl_for_mol = py_rdl.Calculator.get_calculated_result(
    mol.GetBonds(),                              # the edges of the graph
    get_node_1=rdkit.Chem.Bond.GetBeginAtom,     # first node of an edge
    get_node_2=rdkit.Chem.Bond.GetEndAtom,       # second node of an edge
    get_node_id=rdkit.Chem.Atom.GetIdx,          # identify a node (atom)
    get_edge_id=rdkit.Chem.Bond.GetIdx)          # identify an endge (bond)
```

py_rdl with RDKit – Calculation

- Calculation step (Vismara‘s algorithm, URFs)

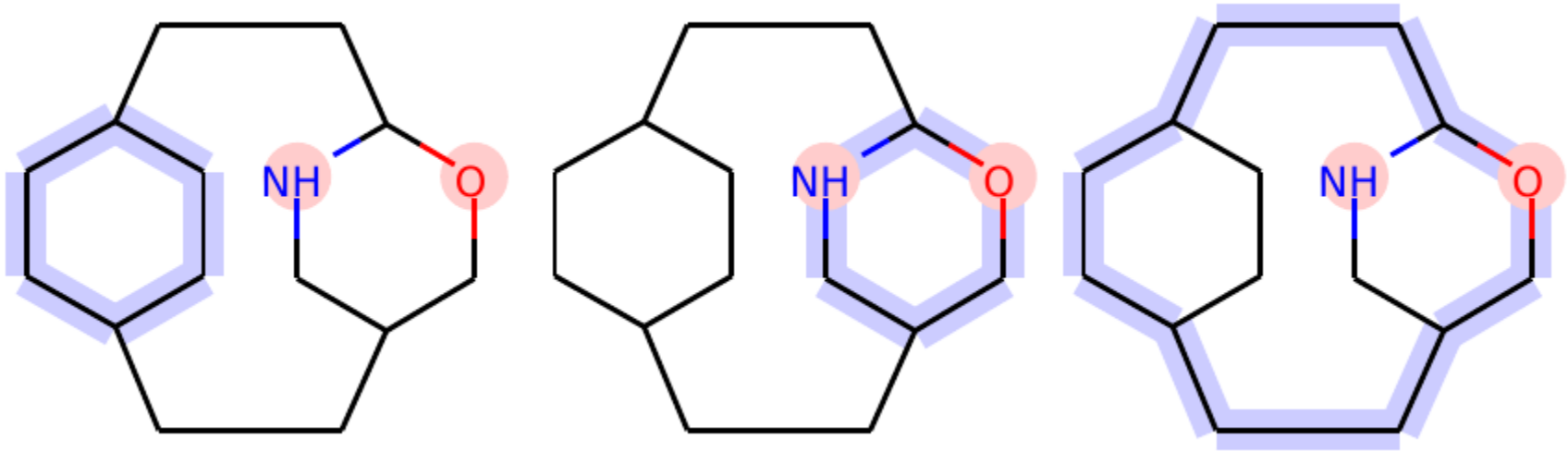
In [24]:
```
import py_rdl
rdl_for_mol = py_rdl.Calculator.get_calculated_result(
    mol.GetBonds(),                          # the edges of the graph
    get_node_1=rdkit.Chem.Bond.GetBeginAtom,    # first node of an edge
    get_node_2=rdkit.Chem.Bond.GetEndAtom,      # second node of an edge
    get_node_id=rdkit.Chem.Atom.GetIdx,         # identify a node (atom)
    get_edge_id=rdkit.Chem.Bond.GetIdx)         # identify an endge (bond)
```

Your molecule goes in here

py_rdl with RDKit – SSSR

In [26]:
```
sssr = rdl_for_mol.get_sssr()

rings_for_N = [ring for ring in sssr if nitrogen in ring.nodes]
print("# SSSR for N: {}".format(len(rings_for_N)))

rings_for_O = [ring for ring in sssr if oxygen in ring.nodes]
print("# SSSR for O: {}".format(len(rings_for_O)))
```

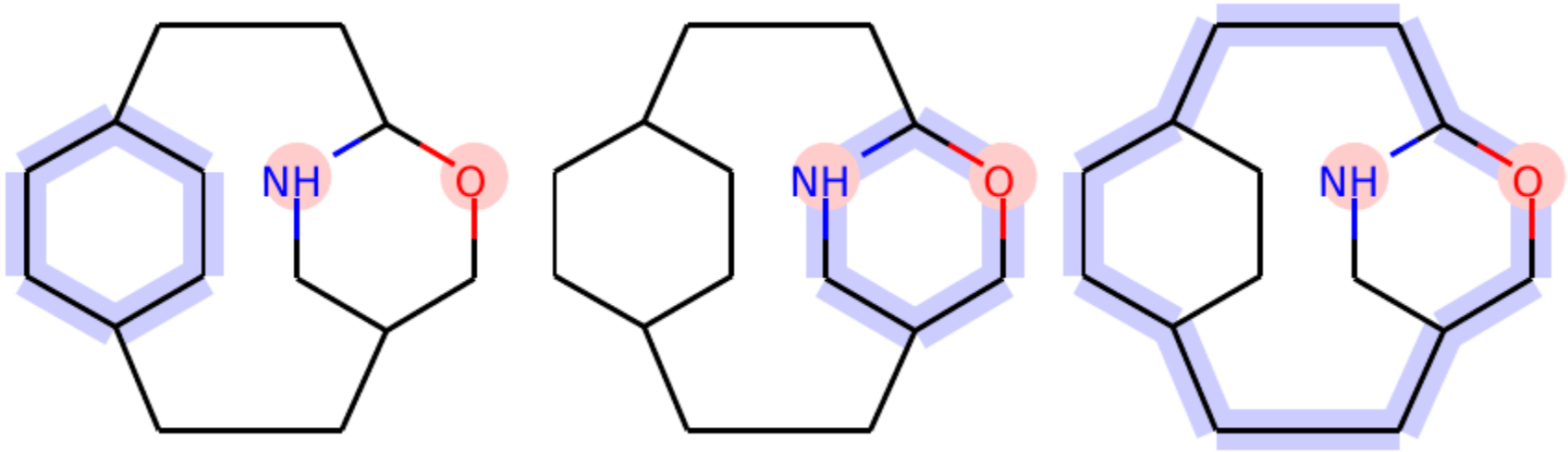
```
# SSSR for N: 1
# SSSR for O: 2
```



py_rdl with RDKit – SSSR

In [26]:
```
sssr = rdl_for_mol.get_sssr()

rings_for_N = [ring for ring in sssr if nitrogen in ring.nodes]
print("# SSSR for N: {}".format(len(rings_for_N)))

rings_for_O = [ring for ring in sssr if oxygen in ring.nodes]
print("# SSSR for O: {}".format(len(rings_for_O)))
```

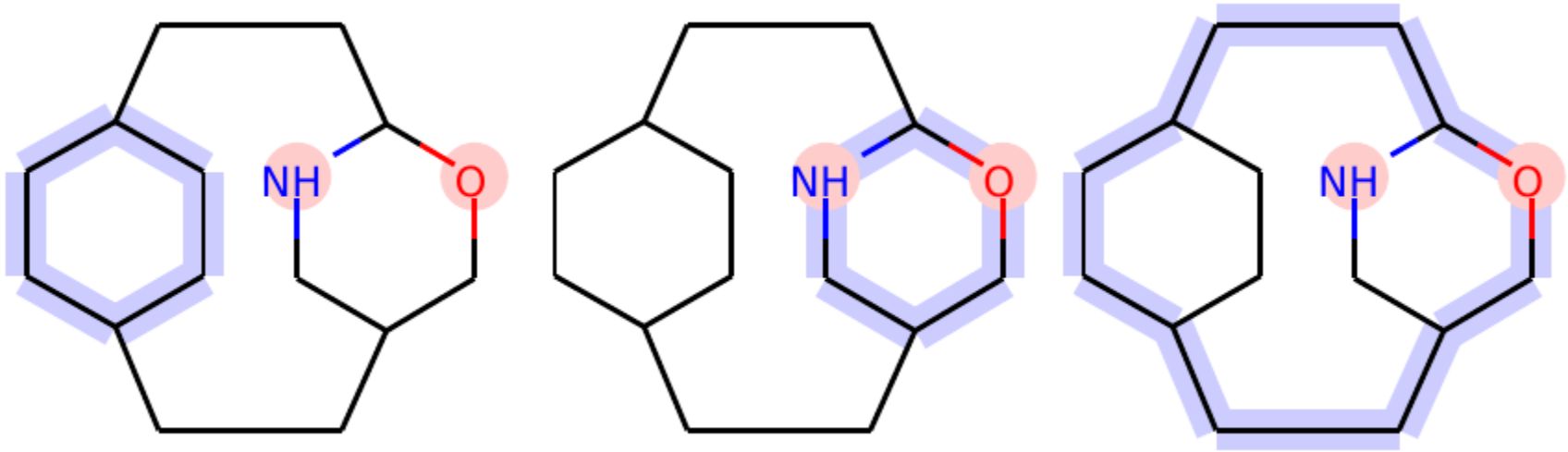
```
# SSSR for N: 1
# SSSR for O: 2
```



py_rdl with RDKit – SSSR

In [26]:
```
sssr = rdl_for_mol.get_sssr()

rings_for_N = [ring for ring in sssr if nitrogen in ring.nodes]
print("# SSSR for N: {}".format(len(rings_for_N)))

rings_for_O = [ring for ring in sssr if oxygen in ring.nodes]
print("# SSSR for O: {}".format(len(rings_for_O)))
```

```
# SSSR for N: 1
# SSSR for O: 2
```

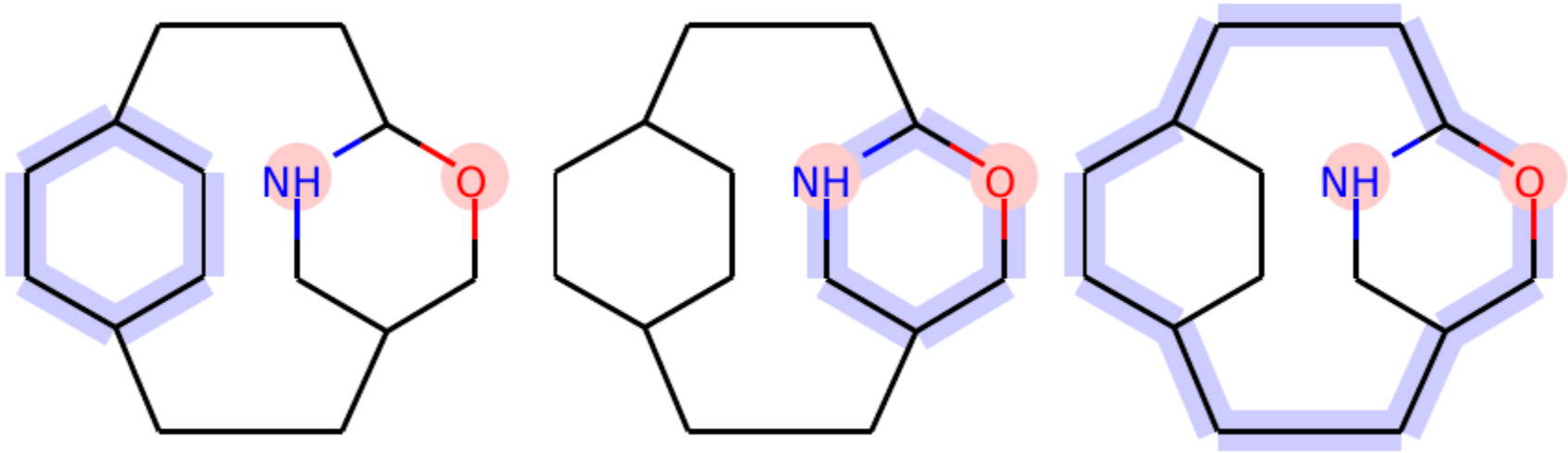


py_rdl with RDKit – SSSR

In [26]:
```
sssr = rdl_for_mol.get_sssr()

rings_for_N = [ring for ring in sssr if nitrogen in ring.nodes]
print("# SSSR for N: {}".format(len(rings_for_N)))

rings_for_O = [ring for ring in sssr if oxygen in ring.nodes]
print("# SSSR for O: {}".format(len(rings_for_O)))
```

```
# SSSR for N: 1
# SSSR for O: 2
```

inconsistent



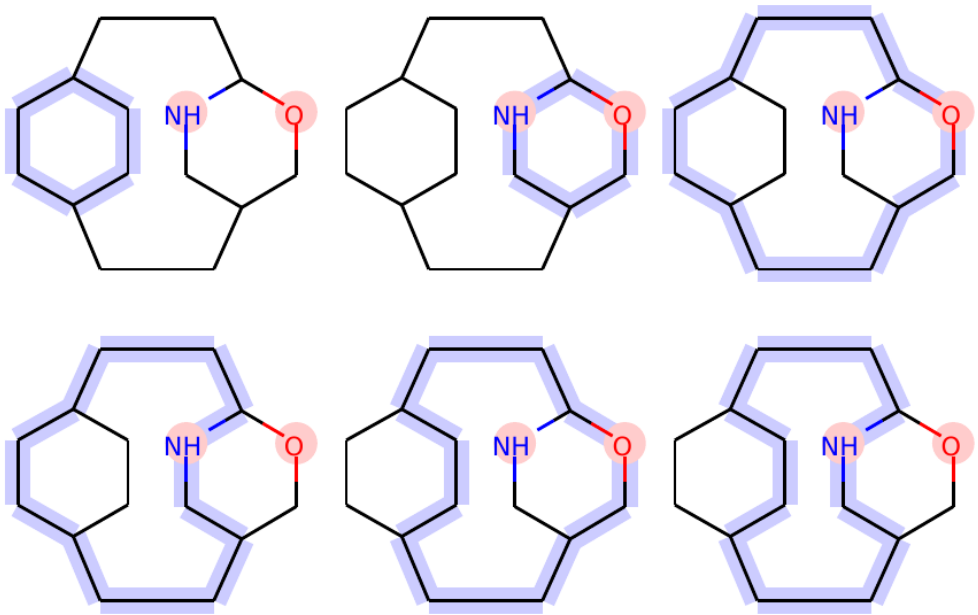
py_rdl with RDKit – RCs

In [28]:
```
nof_rc = rdl_for_mol.get_nof_relevant_cycles()
print('number of RCs: {:.0f}'.format(nof_rc))
```

```
number of RCs: 6
```

In [29]:
```
rcs = rdl_for_mol.get_relevant_cycles()
print(rcs[0], rcs[0].edges)
```

```
(Cycle [WEIGHT 6, URF 0, RCF 0], set([1, 2, 5, 6, 8, 17]))
```



py_rdl with RDKit – RCs

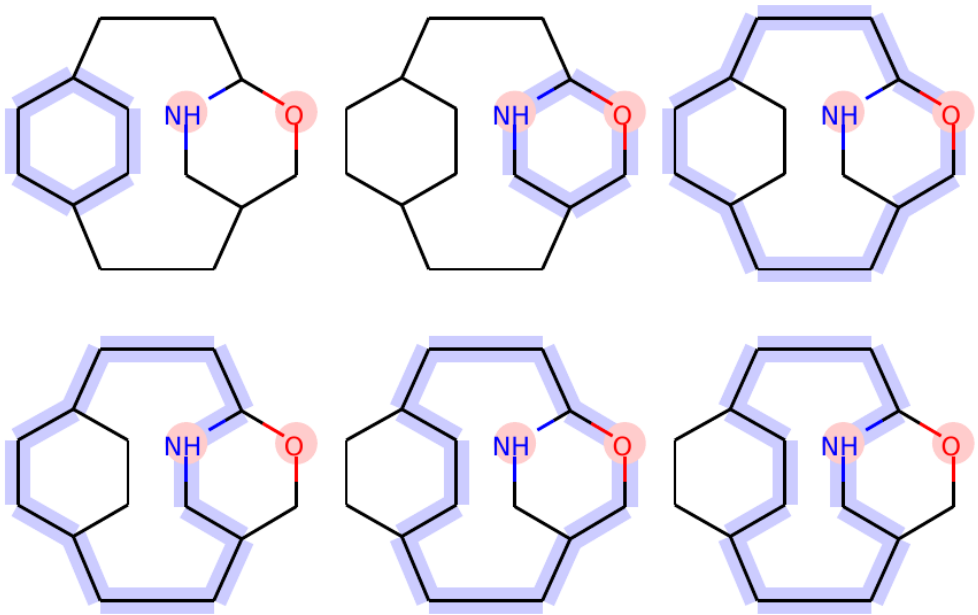
In [28]:
```
nof_rc = rdl_for_mol.get_nof_relevant_cycles()
print('number of RCs: {:.0f}'.format(nof_rc))
```

fast

```
number of RCs: 6
```

In [29]:
```
rcs = rdl_for_mol.get_relevant_cycles()
print(rcs[0], rcs[0].edges)
```

```
(Cycle [WEIGHT 6, URF 0, RCF 0], set([1, 2, 5, 6, 8, 17]))
```



py_rdl with RDKit – RCs

In [28]:
```
nof_rc = rdl_for_mol.get_nof_relevant_cycles()
print('number of RCs: {:.0f}'.format(nof_rc))
```

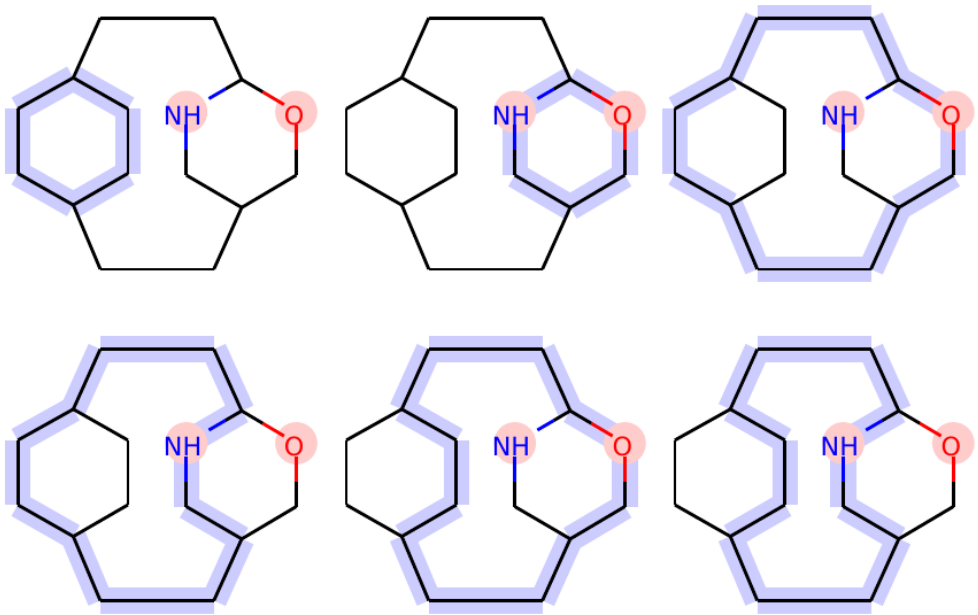
fast

```
number of RCs: 6
```

In [29]:
```
rcs = rdl_for_mol.get_relevant_cycles()
print(rcs[0], rcs[0].edges)
```

potentially slow

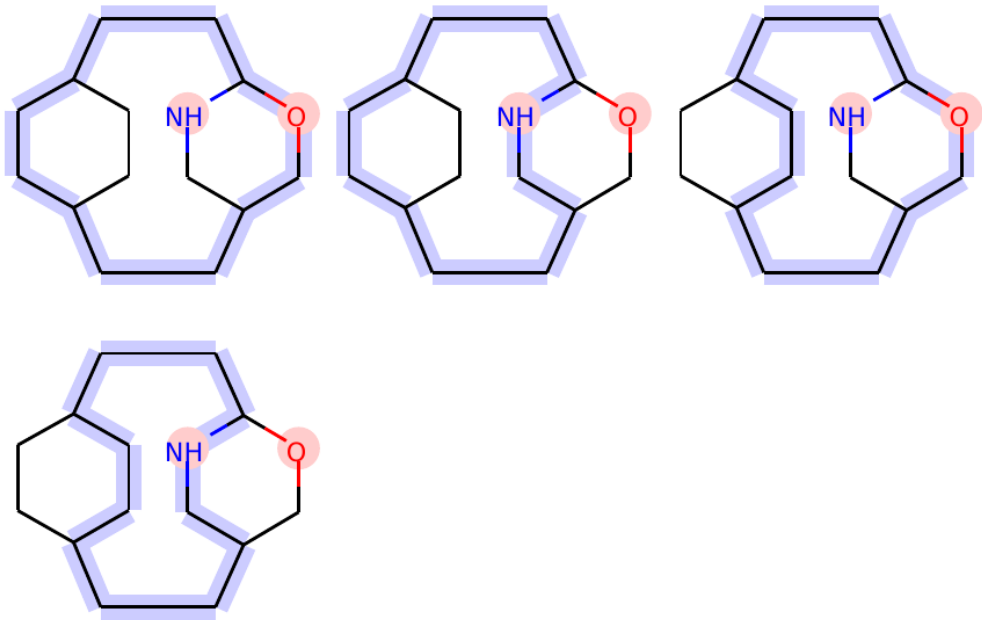
```
(Cycle [WEIGHT 6, URF 0, RCF 0], set([1, 2, 5, 6, 8, 17]))
```



py_rdl with RDKit – URFs

In [31]:
```
nof_urfs = rdl_for_mol.get_nof_urf()
print('number of URFs: {}'.format(nof_urfs))

urf_12 = [urf for urf in rdl_for_mol if urf.weight == 12][0]
rcs_for_urf = rdl_for_mol.get_relevant_cycles_for_urf(urf_12)

for rc in rcs_for_urf:
    print(rc)
```

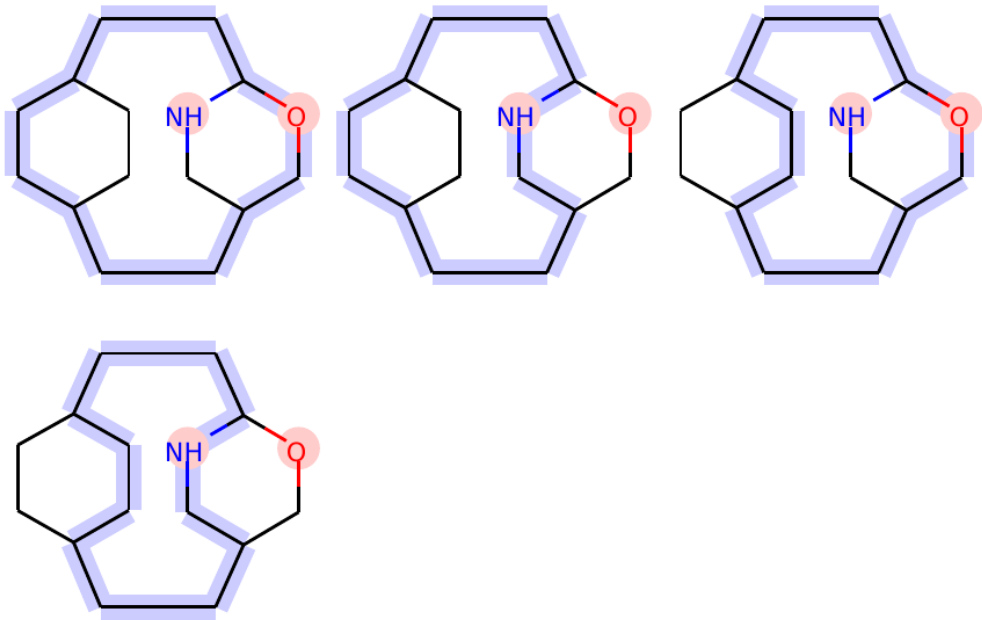
```
number of URFs: 3
Cycle [WEIGHT 12, URF 2, RCF 2]
Cycle [WEIGHT 12, URF 2, RCF 2]
Cycle [WEIGHT 12, URF 2, RCF 3]
Cycle [WEIGHT 12, URF 2, RCF 3]
```



py_rdl with RDKit – URFs

In [31]:
```
nof_urfs = rdl_for_mol.get_nof_urf()
print('number of URFs: {}'.format(nof_urfs))

urf_12 = [urf for urf in rdl_for_mol if urf.weight == 12][0]
rcs_for_urf = rdl_for_mol.get_relevant_cycles_for_urf(urf_12)

for rc in rcs_for_urf:
    print(rc)
```

```
number of URFs: 3
Cycle [WEIGHT 12, URF 2, RCF 2]
Cycle [WEIGHT 12, URF 2, RCF 2]
Cycle [WEIGHT 12, URF 2, RCF 3]
Cycle [WEIGHT 12, URF 2, RCF 3]
```

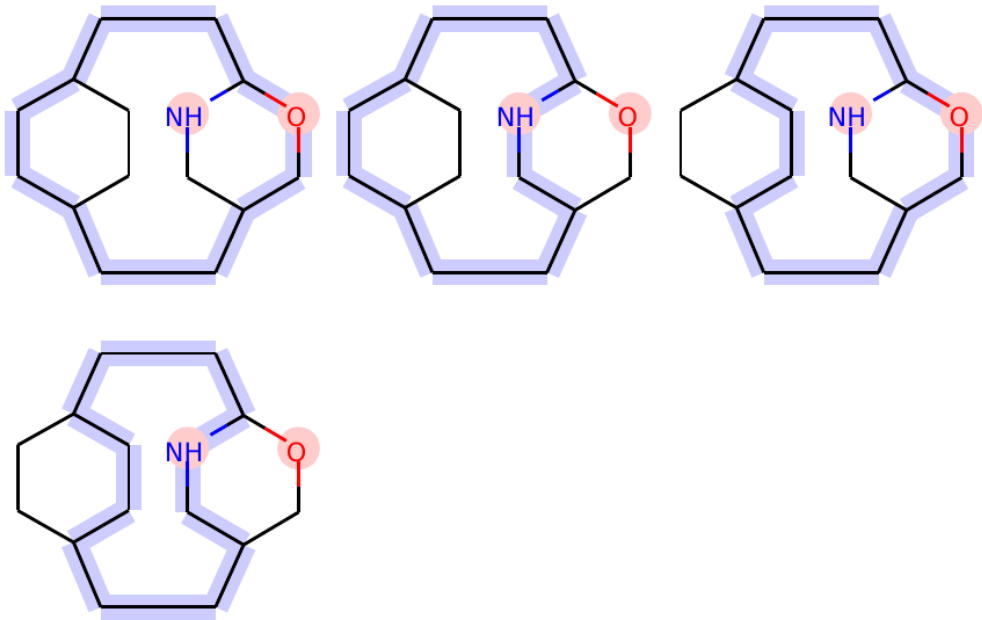


py_rdl with RDKit – URFs

In [31]:
```
nof_urfs = rdl_for_mol.get_nof_urf()
print('number of URFs: {}'.format(nof_urfs))

urf_12 = [urf for urf in rdl_for_mol if urf.weight == 12][0]
rcs_for_urf = rdl_for_mol.get_relevant_cycles_for_urf(urf_12)

for rc in rcs_for_urf:
    print(rc)
```

potentially slow

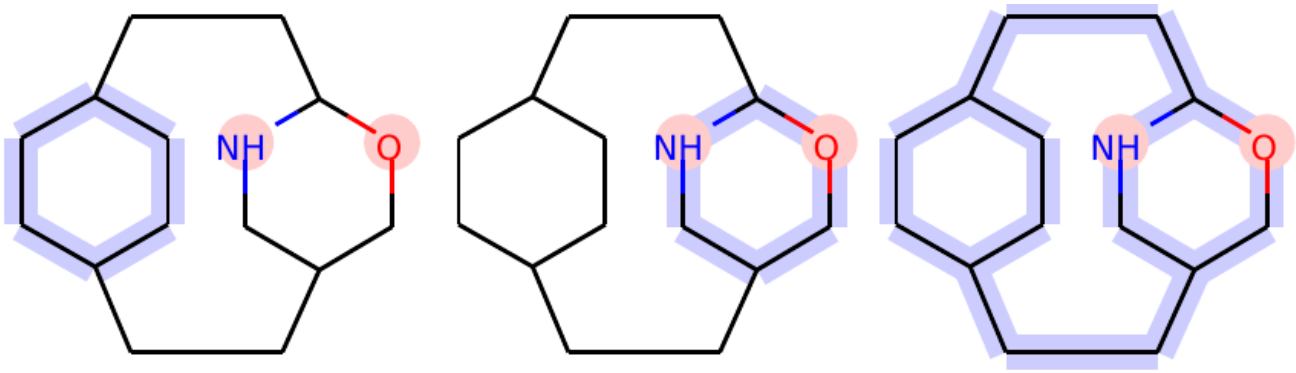
```
number of URFs: 3
Cycle [WEIGHT 12, URF 2, RCF 2]
Cycle [WEIGHT 12, URF 2, RCF 2]
Cycle [WEIGHT 12, URF 2, RCF 3]
Cycle [WEIGHT 12, URF 2, RCF 3]
```



py_rdl with RDKit – URFs and SMARTS

In [33]:
```
for urf in rdl_for_mol:
    print(urf, urf.edges)
```

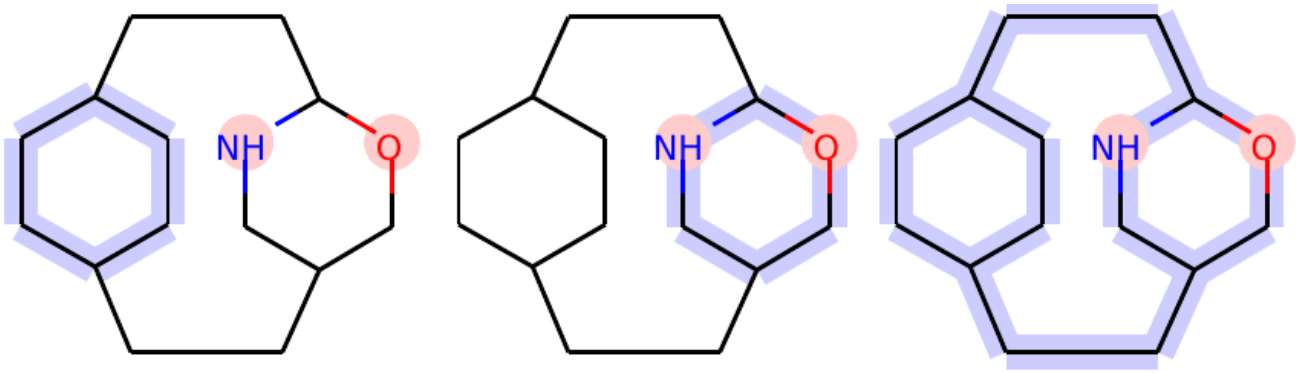
```
(URF 0, set([1, 2, 5, 6, 8, 17]))
(URF 1, set([11, 12, 13, 14, 15, 16]))
(URF 2, set([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]))
```



py_rdl with RDKit – URFs and SMARTS

In [33]:
```
for urf in rdl_for_mol:
    print(urf, urf.edges)
```

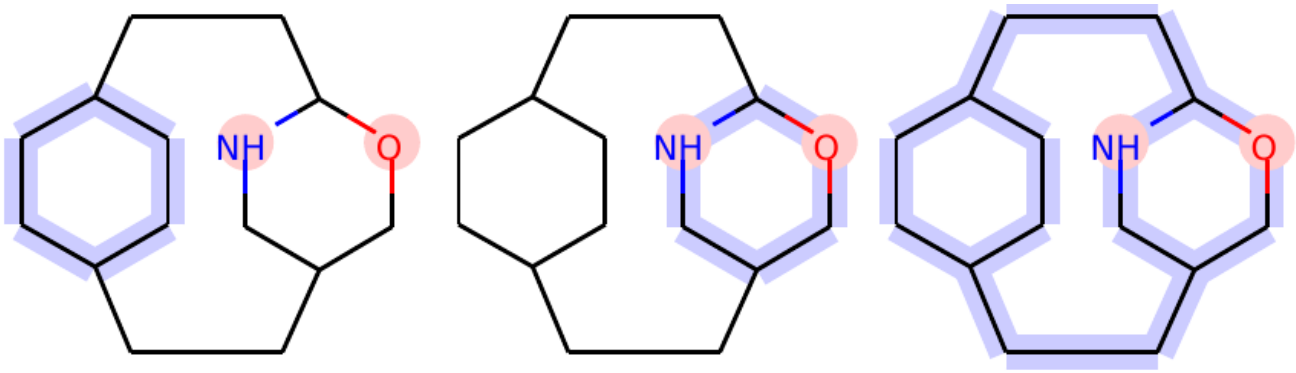
```
(URF 0, set([1, 2, 5, 6, 8, 17]))
(URF 1, set([11, 12, 13, 14, 15, 16]))
(URF 2, set([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]))
```



py_rdl with RDKit – URFs and SMARTS

In [33]:
```
for urf in rdl_for_mol:
    print(urf, urf.edges)
```

```
(URF 0, set([1, 2, 5, 6, 8, 17]))
(URF 1, set([11, 12, 13, 14, 15, 16]))
(URF 2, set([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]))
```



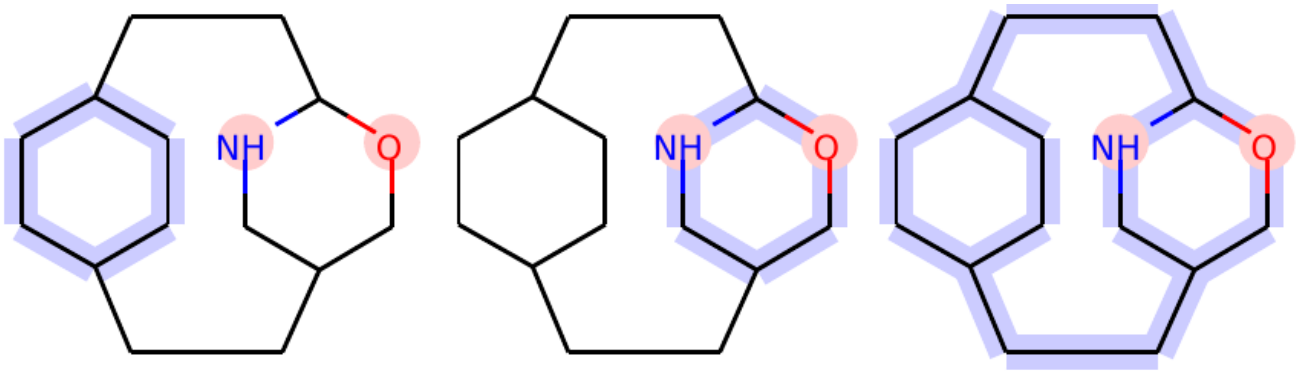
In [35]:
```
urfs_for_N = [urf for urf in rdl_for_mol if nitrogen in urf.nodes]
print("# URF for N: {}".format(len(urfs_for_N)))

urfs_for_O = [urf for urf in rdl_for_mol if oxygen in urf.nodes]
print("# URF for O: {}".format(len(urfs_for_O)))
```

```
# URF for N: 2
# URF for O: 2
```

py_rdl with RDKit – URFs and SMARTS

In [33]:
```
for urf in rdl_for_mol:
    print(urf, urf.edges)
```

```
(URF 0, set([1, 2, 5, 6, 8, 17]))
(URF 1, set([11, 12, 13, 14, 15, 16]))
(URF 2, set([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]))
```



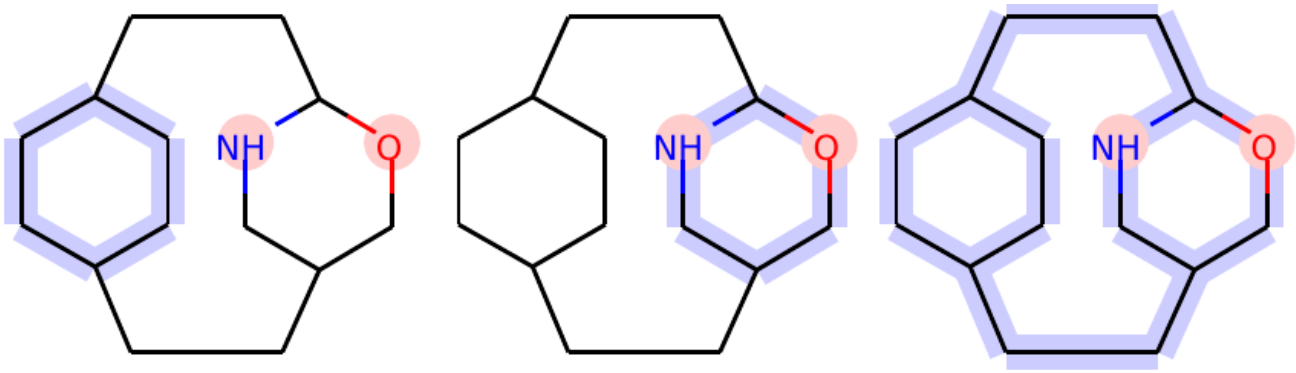
In [35]:
```
urfs_for_N = [urf for urf in rdl_for_mol if nitrogen in urf.nodes]
print("# URF for N: {}".format(len(urfs_for_N)))

urfs_for_O = [urf for urf in rdl_for_mol if oxygen in urf.nodes]
print("# URF for O: {}".format(len(urfs_for_O)))
```

```
# URF for N: 2
# URF for O: 2
```

py_rdl with RDKit – URFs and SMARTS

In [33]:
```
for urf in rdl_for_mol:
    print(urf, urf.edges)
```

```
(URF 0, set([1, 2, 5, 6, 8, 17]))
(URF 1, set([11, 12, 13, 14, 15, 16]))
(URF 2, set([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]))
```

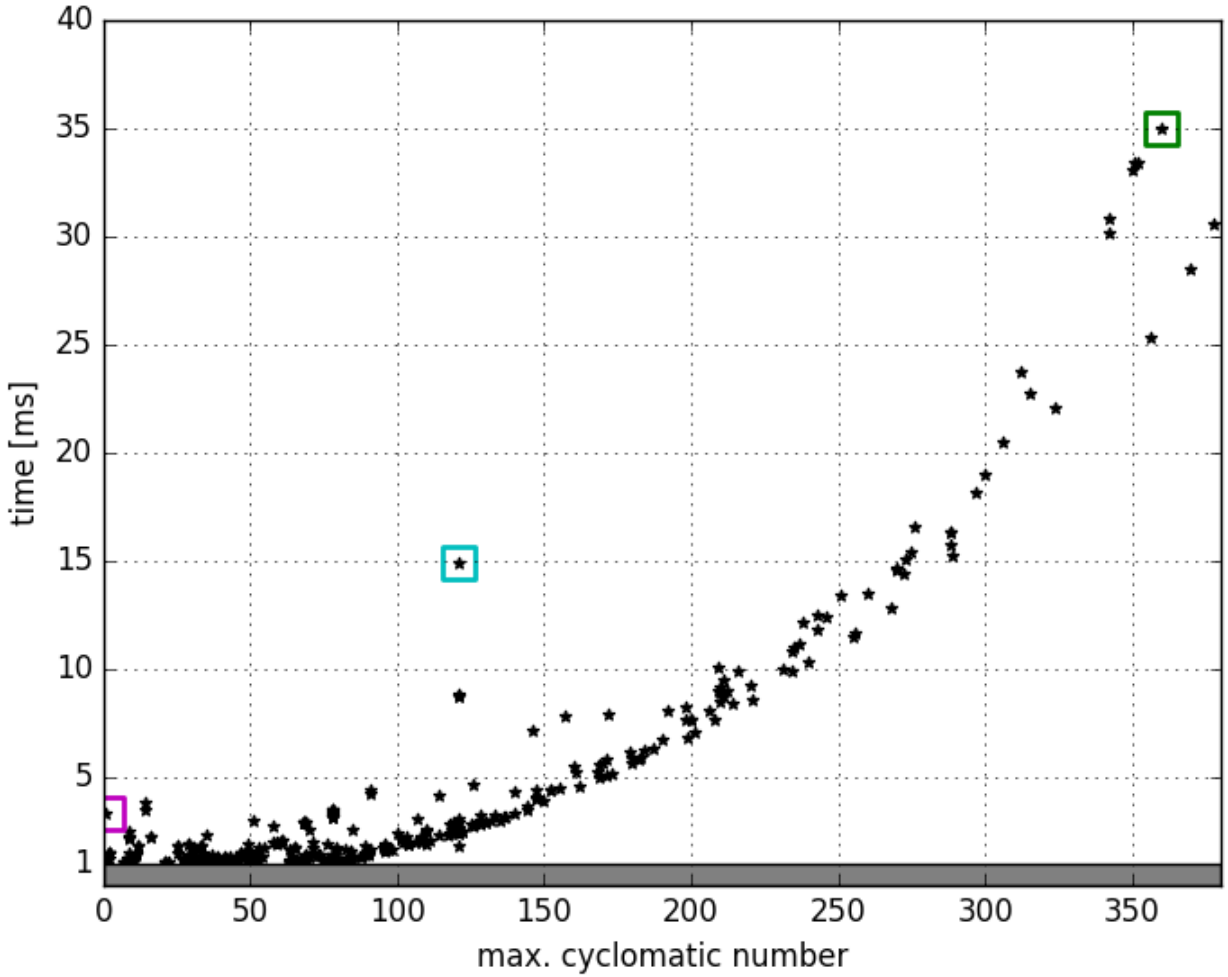


In [35]:
```
urfs_for_N = [urf for urf in rdl_for_mol if nitrogen in urf.nodes]
print("# URF for N: {}".format(len(urfs_for_N)))

urfs_for_O = [urf for urf in rdl_for_mol if oxygen in urf.nodes]
print("# URF for O: {}".format(len(urfs_for_O)))
```

```
# URF for N: 2
# URF for O: 2
```

consistent

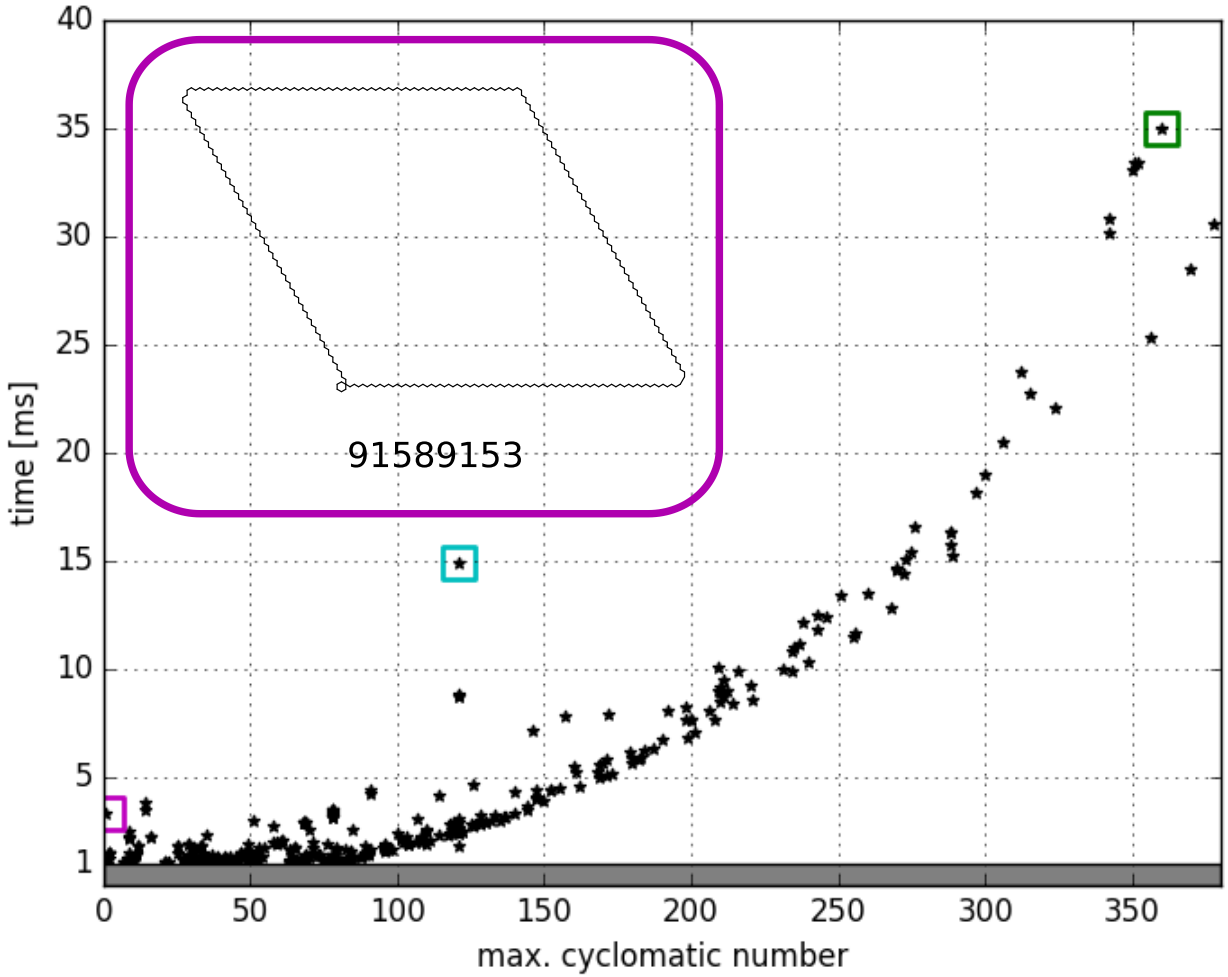
RingDecomposerLib – Runtime



- Complete PubChem compound database[8] (~ 92 million compounds)
- Reading with RDKit 2016.03.1 (C++)
- Only 404 molecules over 1 ms
- Max. 35 ms

[8] Kim, S; et al. Nucleic Acids Res. (2016) 44, D1202-D1213

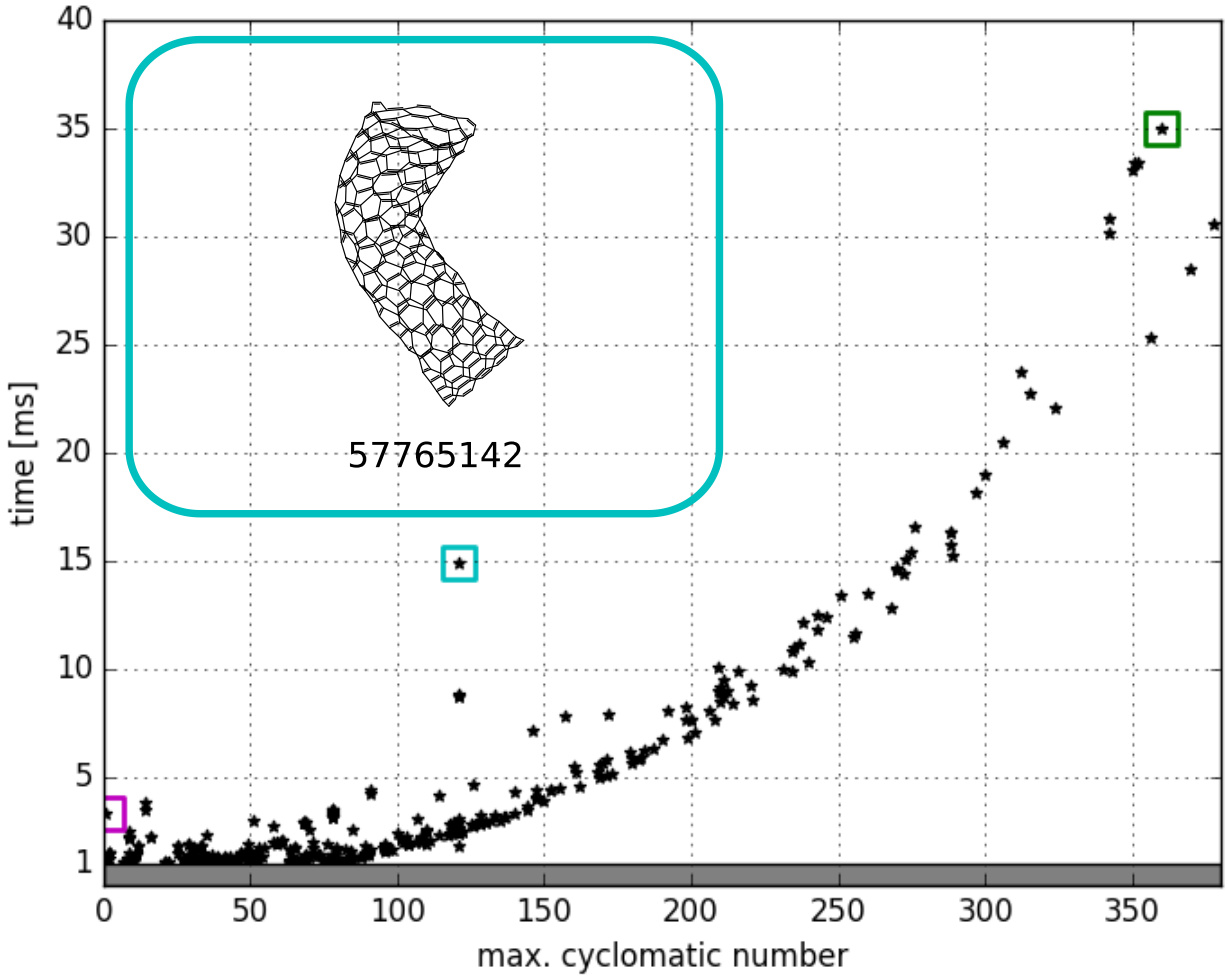
RingDecomposerLib – Runtime



- Complete PubChem compound database[8] (~ 92 million compounds)
- Reading with RDKit 2016.03.1 (C++)
- Only 404 molecules over 1 ms
- Max. 35 ms

[8] Kim, S; et al. Nucleic Acids Res. (2016) 44, D1202-D1213

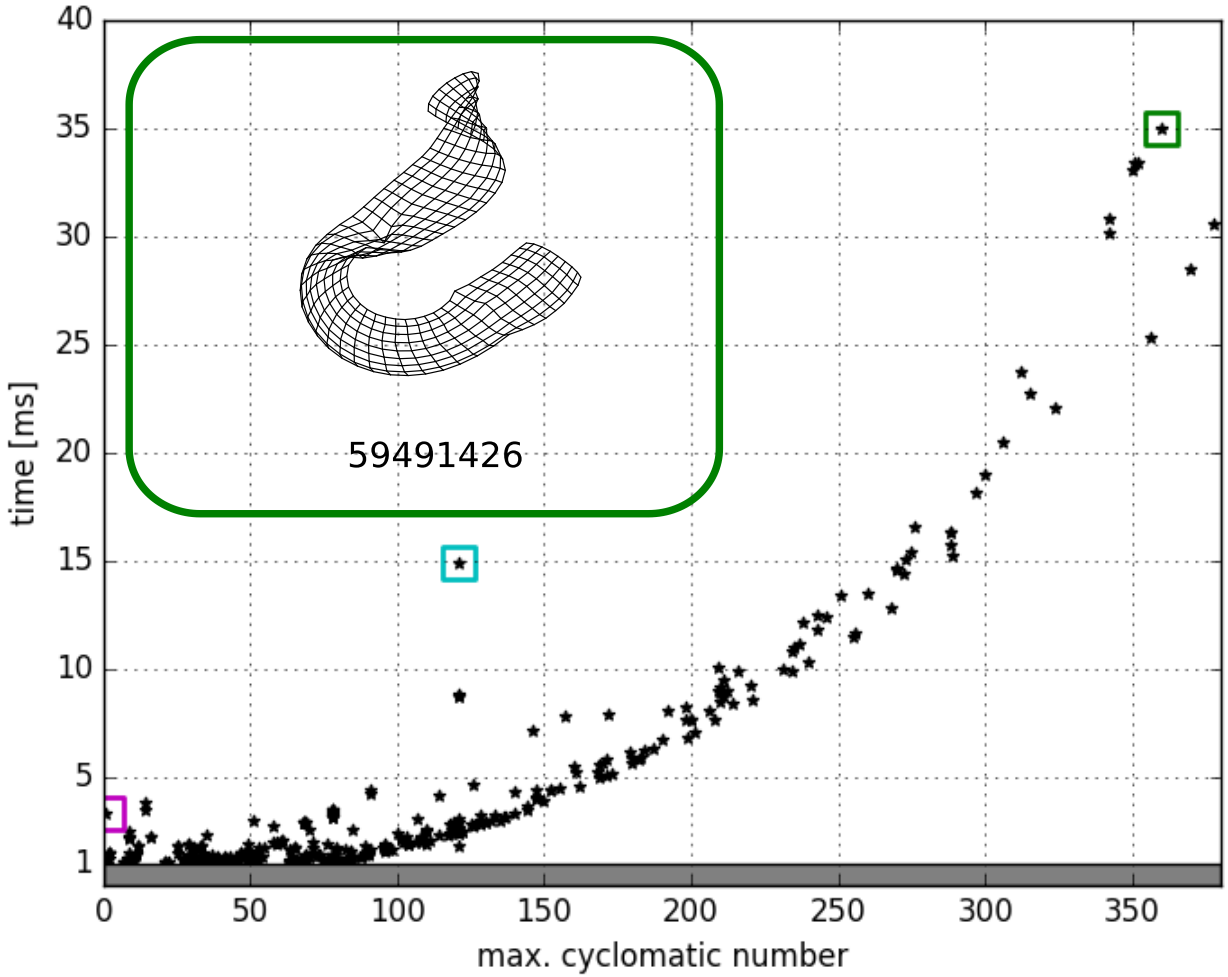
RingDecomposerLib – Runtime



- Complete PubChem compound database[8] (~ 92 million compounds)
- Reading with RDKit 2016.03.1 (C++)
- Only 404 molecules over 1 ms
- Max. 35 ms

[8] Kim, S; et al. Nucleic Acids Res. (2016) 44, D1202-D1213

RingDecomposerLib – Runtime

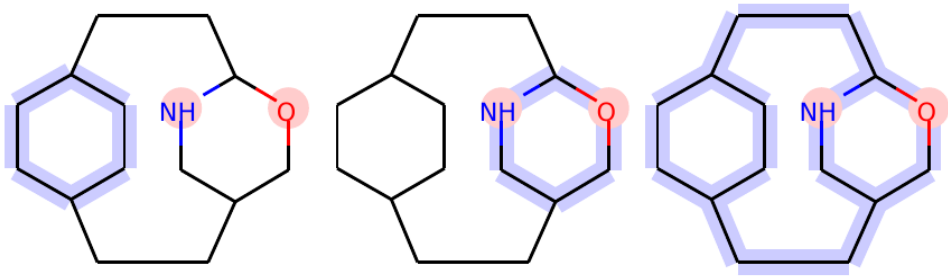


- Complete PubChem compound database[8] (~ 92 million compounds)
- Reading with RDKit 2016.03.1 (C++)
- Only 404 molecules over 1 ms
- Max. 35 ms

[8] Kim, S; et al. Nucleic Acids Res. (2016) 44, D1202-D1213

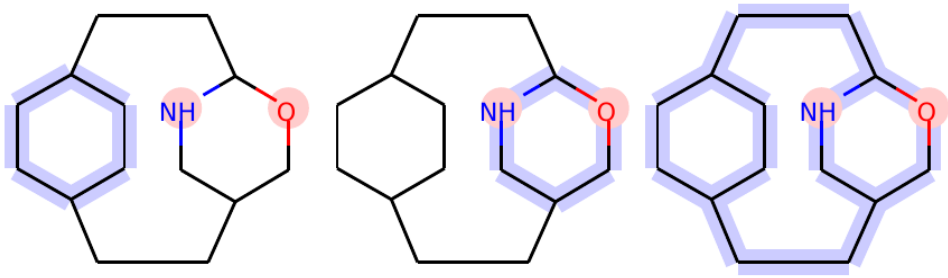
Conclusion

- URFs are **unique**, **chemically meaningful** and **efficient to calculate**



Conclusion

- URFs are **unique**, **chemically meaningful** and **efficient to calculate**



- **RingDecomposerLib** makes URFs available for almost all cheminformatics libraries
  - And it‘s thoroughly validated, fast and easy-to-use

Acknowledgements

THANK YOUR FOR YOUR ATTENTION

- Matthias Rarey
- Thomas Otto
- and the whole AMD group

- Niek Andresen

Availability

- Currently in preparation
  - Flachsenberg, F., Andresen, N.; Rarey, M. RingDecomposerLib: An Open-Source Implementation of Unique Ring Families and Other Cycle Bases.
- RingDecomposerLib
  - www.github.com/rareylab/RingDecomposerLib
- Our Tool Collection: http://molecular-design.net
- Modeling Support Server: http://proteins.plus