RDKIT GEMS

ROGER SAYLE, PHD
NEXTMOVE SOFTWARE LTD
CAMBRIDGE, UK

INTRODUCTION

- This presentation is intended as a short talktorial describing real world code examples, recent features and commonly used RDKit idioms.
- Although many reflect personal/NextMove Software experiences developing with the C++ APIs, most (though not all) observations apply equally to the Java and Python APIs.

LOOPING OVER MOLECULE ATOMS

- Poor

```
for (RWMol::AtomIterator atomIt=mol->beginAtoms();
    atomIt != mol->endAtoms(); atomIt++) …
```

- Better

```
for (RWMol::AtomIterator atomIt=mol->beginAtoms();
    atomIt != mol->endAtoms(); ++atomIt) …
```

- The C++ post increment operator returns the original value prior to the increment, requiring a copy to made.

LOOPING OVER ATOM'S BONDS

- Documented idiom:

```
... molPtr is a const ROMol * ...
... atomPtr is a const Atom * ...
ROMol::OEDGE_ITER beg,end;
boost::tie(beg,end) = molPtr->getAtomBonds(atomPtr);
while(beg!=end){
  const Bond * bond=(*molPtr)[*beg].get();
  ... do something with the Bond ...
  ++beg;
}
```

LOOPING OVER ATOM'S BONDS

- Proposed replacement:

```
... molPtr is a const ROMol * ...
... atomPtr is a const Atom * ...
ROMol::OBOND_ITER_PAIR bondIt;
bondIt = molPtr->getAtomBonds(atomPtr);
while(bondIt.first != bondIt.second){
  const Bond * bond=(*molPtr)[*bondIt.first].get();
  ... do something with the Bond ...
  ++bondIt;
}
```

ACCEPTING STRING ARGUMENTS

- Good practice to accept both C++ constant literals and STL std::strings.
- Poor

```
void foo(const char *ptr) { foo(std::string(ptr)); }
void foo(const std::string str) { … }
```

- Better

```
void foo(const std::string &str) { foo(str.c_str()); }
void foo(const char *ptr) { … }
```

ISOELECTRONIC VALENCES

- Supporting PF_6^- in RDKit currently requires adding 7 to the neutral valences of P. Instead, P^{-1} should be isoelectronic with S.
- Poor

  valence(atomicNo,charge) ~= valence(atomicNo,0)-charge
- Better

  valence(atomicNo,charge) ~= valence(atomicNo-charge,0)

REPRESENTING REACTIONS

- Two possible representations
  - A new object containing vectors of reactant and product (and agent?) molecules.
  - Annotation of existing molecule with reaction roles annotated on each atom.
- Both are representations are equivalent and one may be converted to the other and vice versa.

COMMON ANCESTRY/BASE CLASS

- A significant advantage of avoiding new object types is the common requirement in cheminformatics of reading reactions, molecules, queries, peptides, transforms, and pharmacophores for the same file format.
- OpenBabel and GGA's Indigo require you to know the contents of a file/record (mol vs. rxn) before reading it!?

PROPOSED RDKIT INTERFACE

- Atom::setProp(“rxnRole”,(int)role)
  - RXN_ROLE_NONE 0
  - RXN_ROLE_REACTANT 1
  - RXN_ROLE_AGENT 2
  - RXN_ROLE_PRODUCT 3
- Atom::setProp(“rxnGroup”,(int)group)
- Atom::setProp(“molAtomMapNumber”,idx)
- RWMol::setProp(“isRxn”,isrxn?1:0)

AND WHERE THIS GETS USED...

- Code/GraphMol/FileParser/MolFileWriter.cpp:
- Function GetMolFileAtomLine contains
- Currently unused local variables
  - rxnComponentType
  - rxnComponentNumber

- This would allow support for CPSS-style reactions [i.e. reactions in SD files]

INVERTED PROTEIN HIERARCHY

- Optimize data structures for common ops.
- Poor

```
for (modelIt mi=mol->getModels(); mi; ++mi)
  for (chainIt ci=(*mi)->getChains(); ci; ++ci)
    for (resIt ri=(*ci)->getResidues(); ri; ++ri)
      for (atomIt ai=(*ri)->getAtoms(); ai; ++ai)
        … /* Do something! */
```

- Better

```
for (atomIt ai=mol->getAtoms(); ai; ++ai)
  … /* Do something! */
```

RDKIT PDB FILE WRITER FEATURES

- By default
  - Atoms are written as HETAM records
  - Bonds are written as CONECT records
  - Bond orders encoded by popular extension
  - All atoms are uniquely named
  - Molecule title written as COMPND record
  - Formal charges are preserved.
  - Default residue is “UNL” (not “LIG”).

EXAMPLE PDB FORMAT OUTPUT

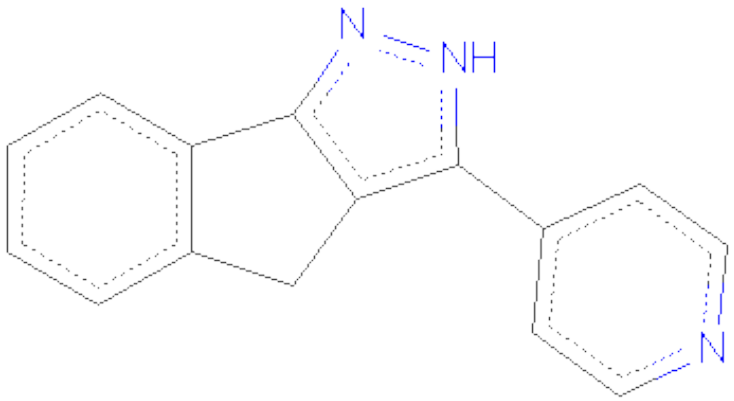
```
COMPND    benzene
HETATM    1  C1  UNL     1       0.000   0.000   0.000  1.00  0.00           C
HETATM    2  C2  UNL     1       0.000   0.000   0.000  1.00  0.00           C
HETATM    3  C3  UNL     1       0.000   0.000   0.000  1.00  0.00           C
HETATM    4  C4  UNL     1       0.000   0.000   0.000  1.00  0.00           C
HETATM    5  C5  UNL     1       0.000   0.000   0.000  1.00  0.00           C
HETATM    6  C6  UNL     1       0.000   0.000   0.000  1.00  0.00           C
CONECT    1    2    2    6
CONECT    2    3
CONECT    3    4    4
CONECT    4    5
CONECT    5    6    6
END
```

UNIQUE ATOM NAMES

- Counts unique per element type
- C1..C99,CA0..CA9,CB0..CZ9,CAA...CZZ
- This allows for up to 1036 indices.

BEWARE OPENBABEL'S LIG RESIDUE

- PDB residue LIG is reserved for 3-pyridin-4-yl-2,4-dihydro-indeno[1,2-c]pyrazole.



- Use of residue UNL (for UNknown Ligand) is much preferred.

PDB FILE READERS

- Not all connectivity is explicit
- Two options for determining bonds
  - Template methods
  - Proximity methods
- Two options for determining bond orders
  - Template methods
  - 3D geometry methods

- See Sayle, “PDB Cruft to Content”, MUG 2001

TEMPLATE-BASED BOND ORDERS

- Need only specify the double bonds
- Most (11/20) amino acids only need C=O.
- Arg: C=O,CZ=NH2
- Asn/Asp: C=O,CG=OD1
- Gln/Glu: C=O,CD=OE1
- His: C=O,CG=CD2,ND1=CE1
- Phe/Tyr: C=O,CG=CD1,CD2=CE2,CE1=CZ
- Trp: C=O,CG=CD1,CD2=CE2,CE3=CZ3,CZ2=CH2

SWITCHING ON SHORT STRINGS

```
// These are macros to allow their use in C++ constants
#define BCNAM(A,B,C)    (((A)<<16) | ((B)<<8) | (C))
#define BCATM(A,B,C,D)  (((A)<<24) | ((B)<<16) | ((C)<<8) | (D))

  switch (rescode) {
  case BCNAM('A','L','A'):
  case BCNAM('C','Y','S'):
  case BCNAM('G','L','Y'):
  case BCNAM('I','L','E'):
  case BCNAM('L','E','U'):
...
    if (atm1==BCATM(' ','C',' ',' ') && atm2==BCATM(' ','O',' ',' '))
      return true;
    break;
```

PROXIMITY BONDING

- The ideal distance between two bonded atoms is the sum of their covalent radii.
- The ideal distance between two non-bonded atoms is the sum of their van der Waal's radii.
- Assume bonding between all atoms closer than the sum of Rcov plus a delta (0.45Å).
- Impose minimum distance threshold (0.4Å)

COMPARE THE SQUARE

- A rate limiting step in many (poorly written) proximity calculations is the calculation of square roots.
- Poor

```
if(sqrt(dx*dx+dy*dy+dz*dz) < radius) ...
```

- Better

```
if(dx*dx + dy*dy + dz*dz < radius*radius) …
```

c.f. http://gcc.gnu.org/ml/gcc-patches/2003-03/msg01683.html

FAIL EARLY, FAIL FAST (LIKE PHARMA)

- Poor

```
return dx*dx + dy*dy + dz*dz <= radius2
```

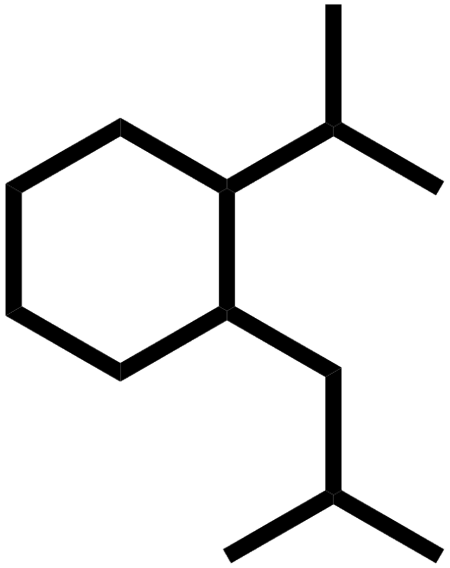
- Better

```
dist2 = dx*dx
if (dist > radius2)
  return false
dist2 += dy*dy
if (dist > radius2)
  return false
dist2 += dz*dz
return dist2 <= radius2
```

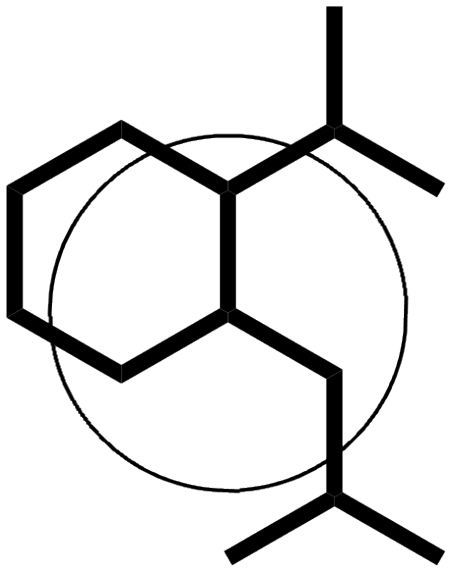
PROXIMITY ALGORITHMS

- Brute Force [FreON] N^2
- Z (Ordinate) Sort [OpenBabel] $N\log N$
- Binary Space Partition [CDK] $N\log N..N^2$
- Fixed Dimension Grid [RasMol] $\sim N$
- Fixed Spacing Grid [Coot] $\sim N$
- Grid Hashing [RDKit/OEChem] $\sim N$

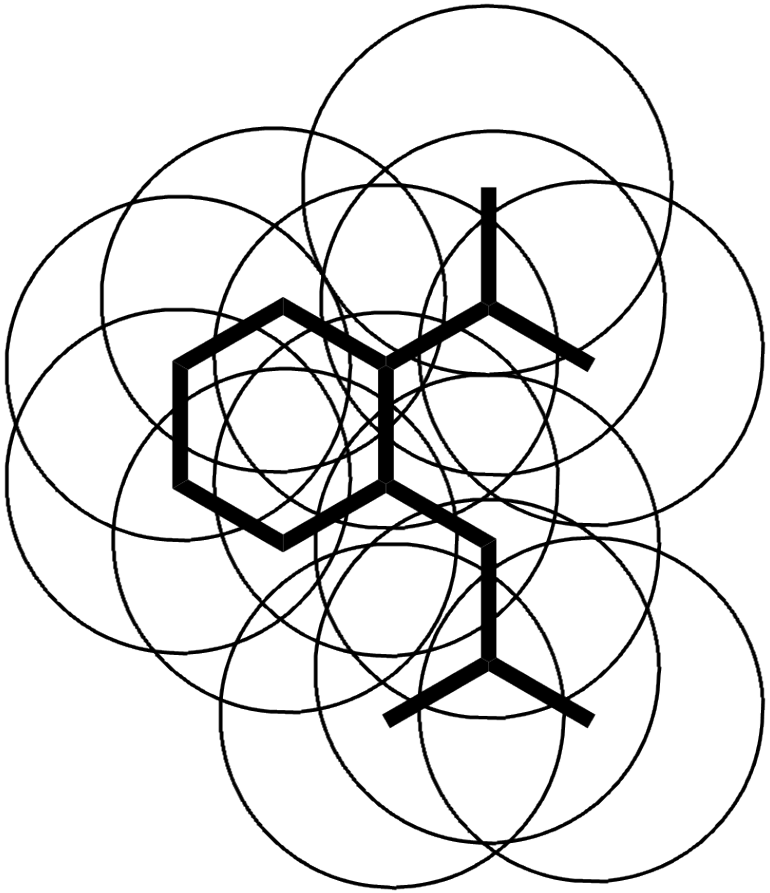
PROXIMITY COMPARISON



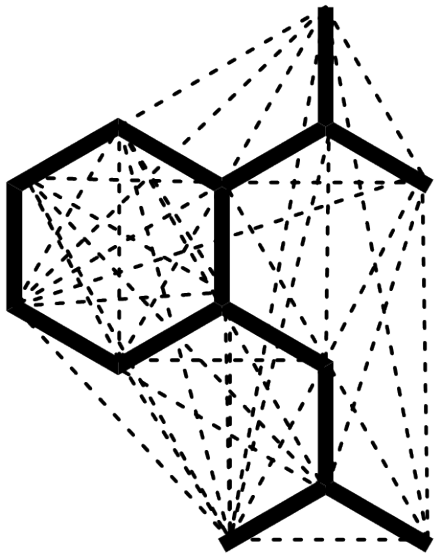
PROXIMITY COMPARISON



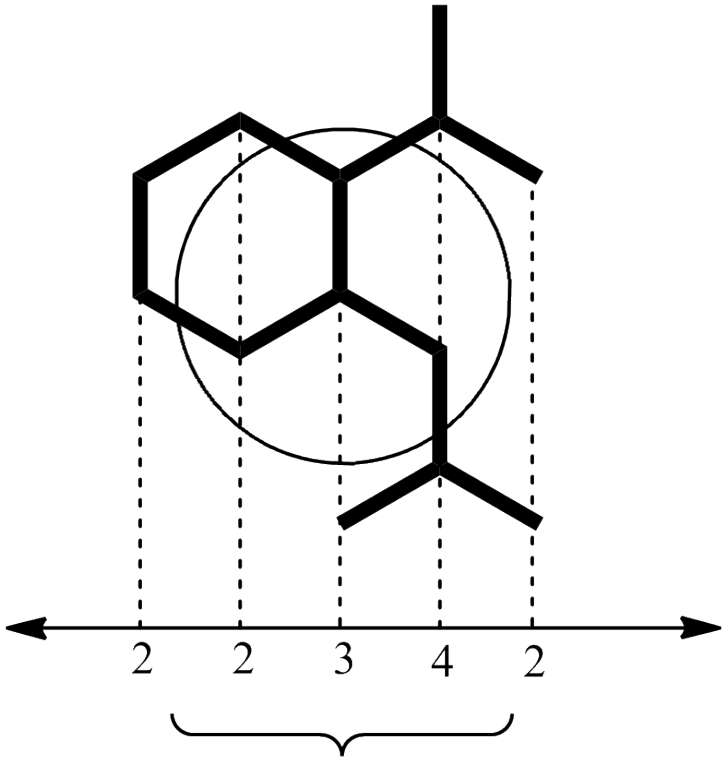
PROXIMITY COMPARISON



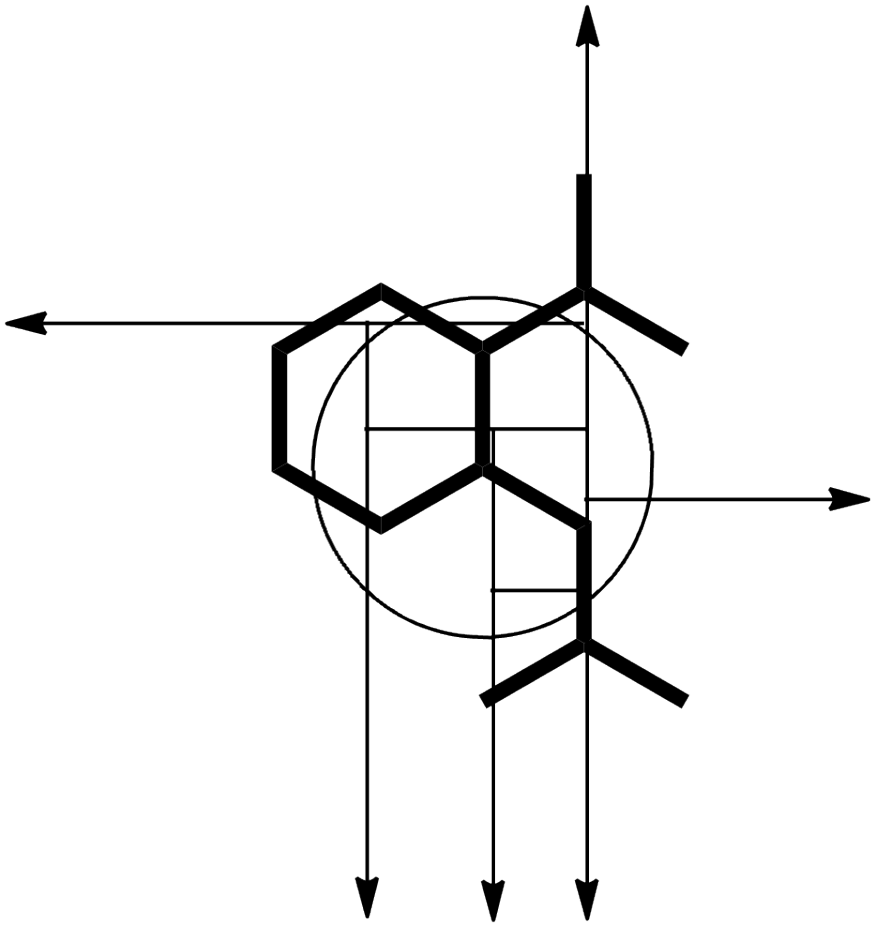
BRUTE FORCE



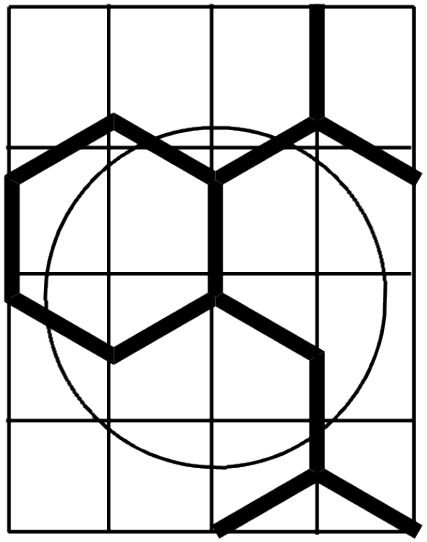
Z (ORDINATE) SORT



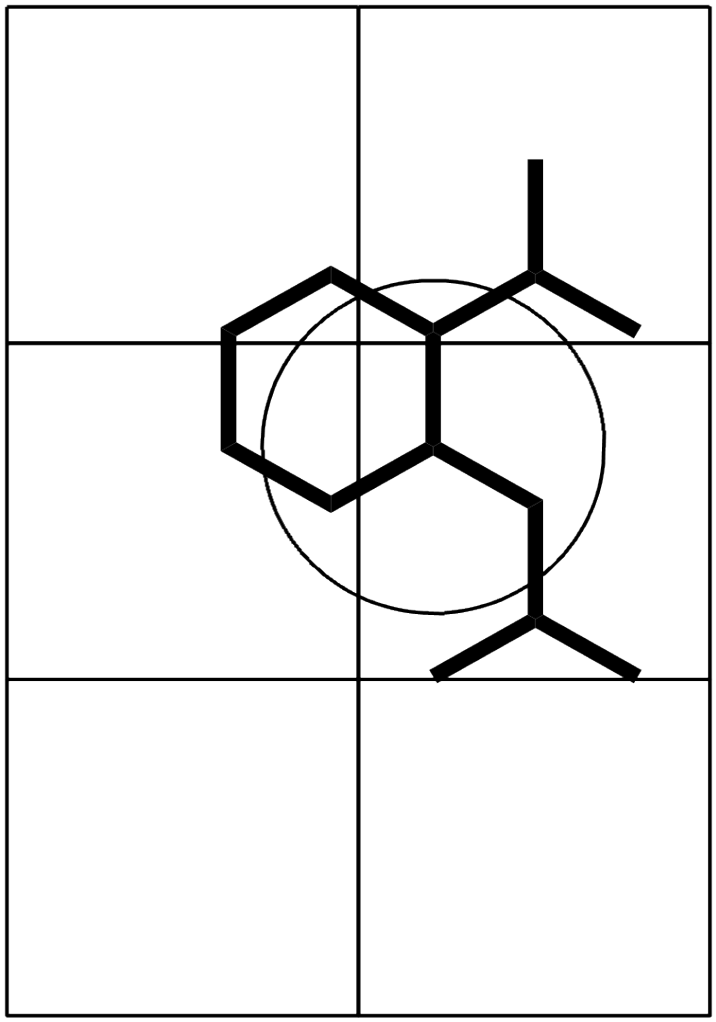
BINARY SPACE PARTITION (BSP)



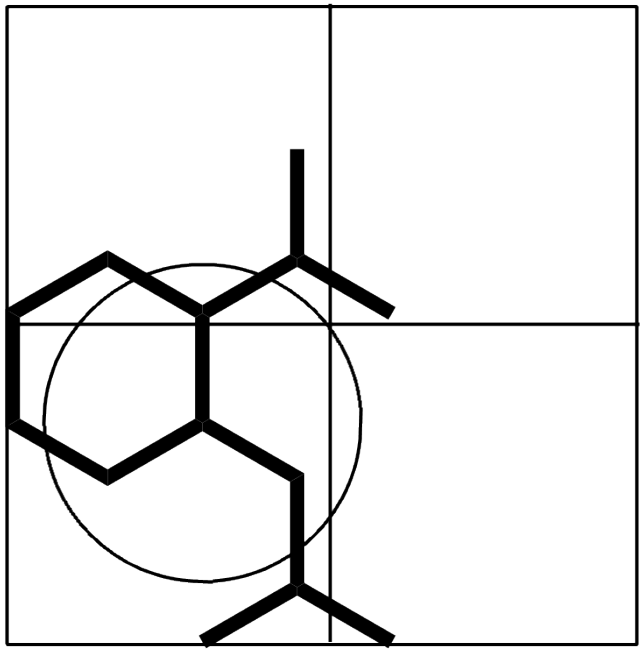
FIXED DIMENSION GRID



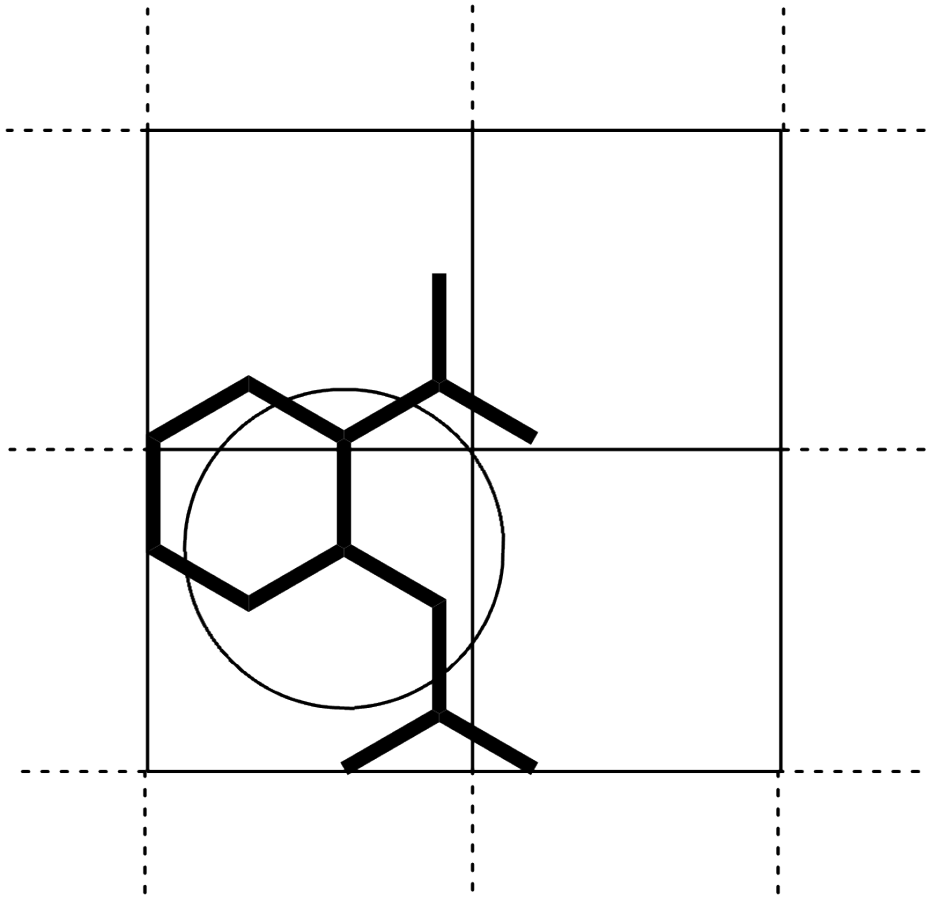
FIXED SPACING GRID



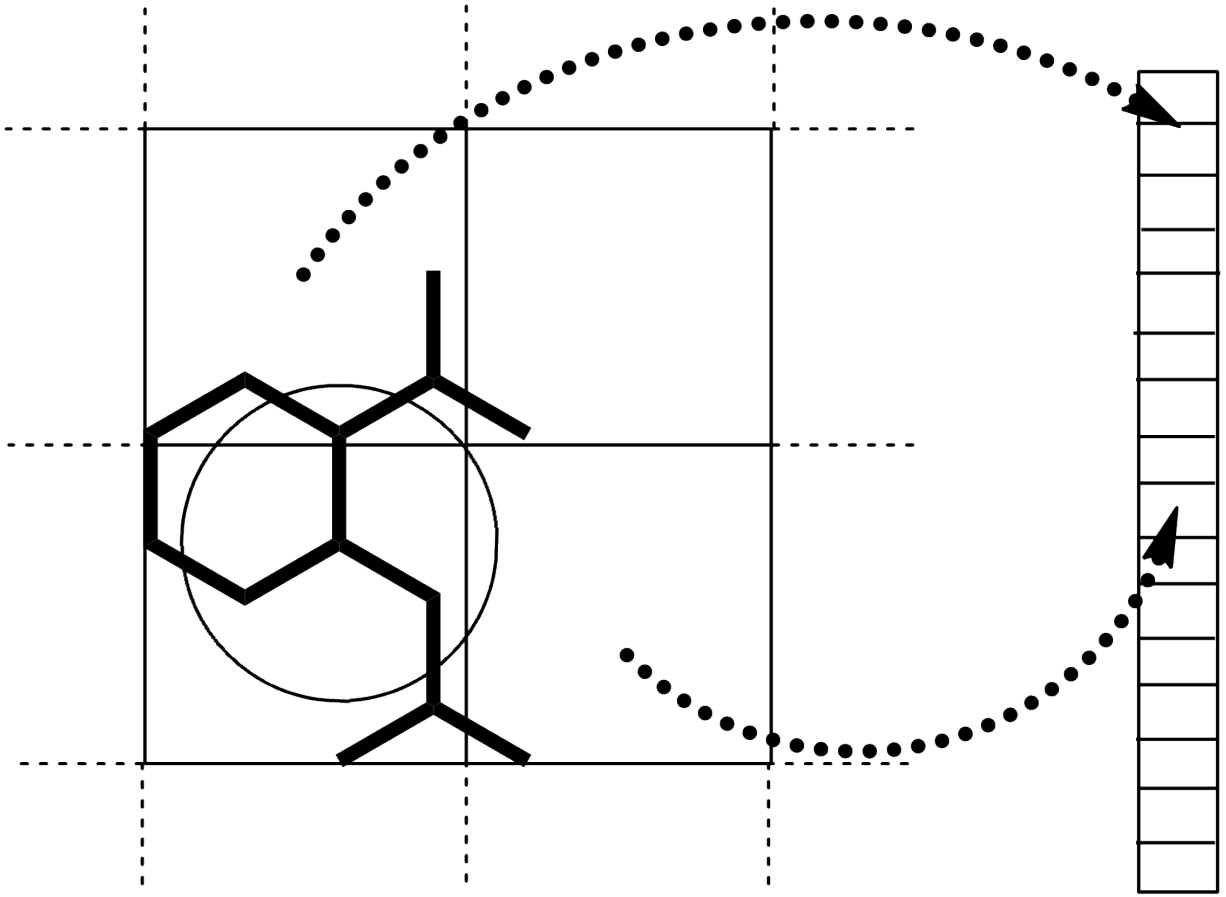
FINITE FIXED SPACING GRID



INFINITE FIXED SPACING GRID



GRID HASHING



FUTURE WORK/WISH LIST

- Atom reordering in RDKit.
- C++ compressed (gzip) file support.
- Non-forward PDB supplier.
- In-place removeHs/addHs.