Refactoring Rosetta for Chemistry

Meiler Lab

Center for Structural Biology and Institute of Chemical Biology
Departments of Chemistry, Pharmacology, and Biomedical Informatics

Outline

A. Small molecule based challenges we'd like to address with Rosetta

B. Technological advances needed to tackle these challenges

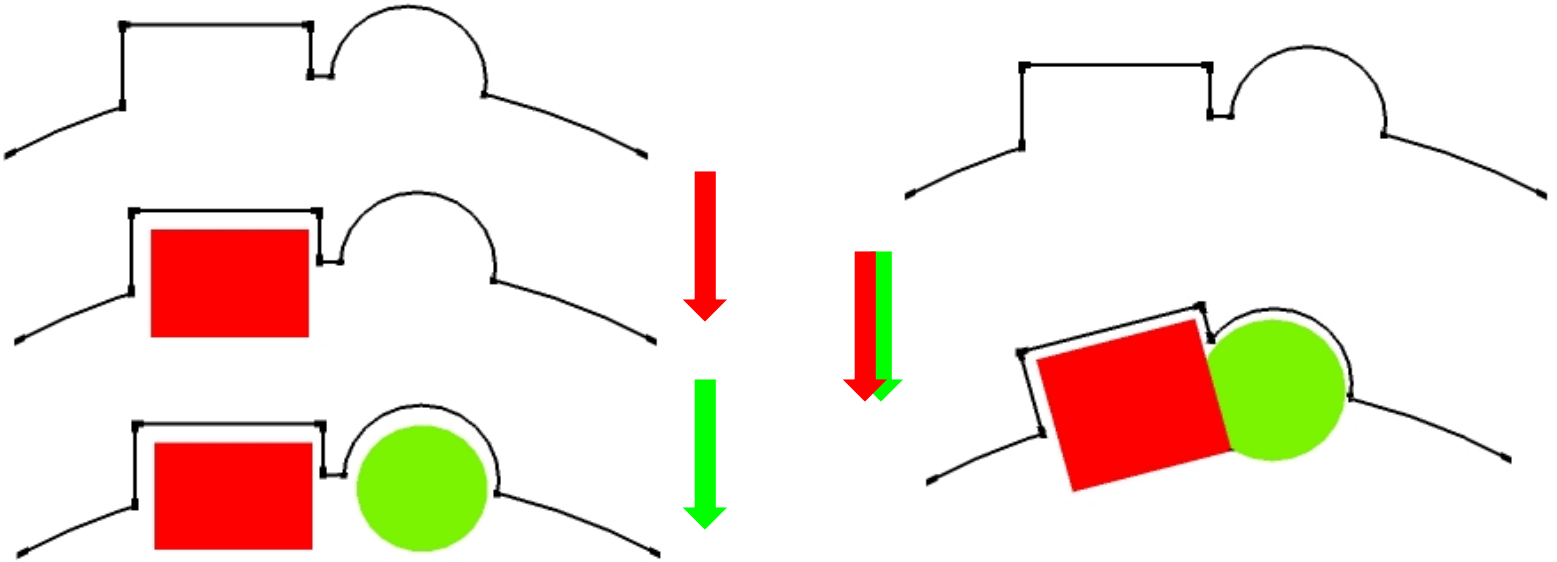
C. Steps to refactoring Rosetta for Chemistry

A) Small molecule challenges for Rosetta

1. **Docking multiple ligands, co-factors, waters, metals**
2. Fragment-based docking and design
3. Merging Ligand based and structure based drug discovery
4. Chemistry inspired atom typing to enable more general scoring functions

Multi-ligand docking captures synergistic effects

- Enzymes bind multiple ligands, cofactors, ions, metals
- Only Simultaneous docking can capture synergy

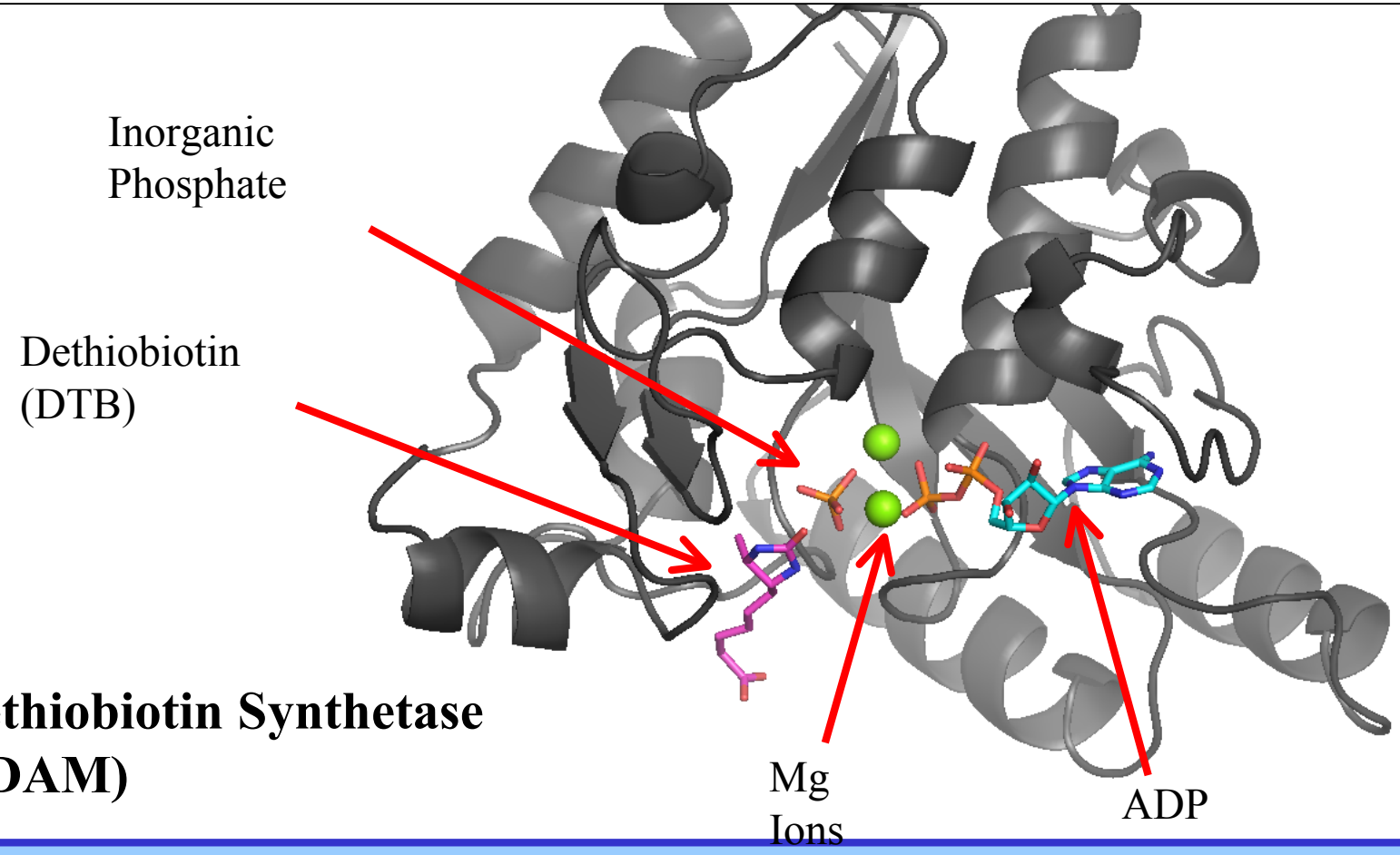


Serial Docking

Simultaneous Docking

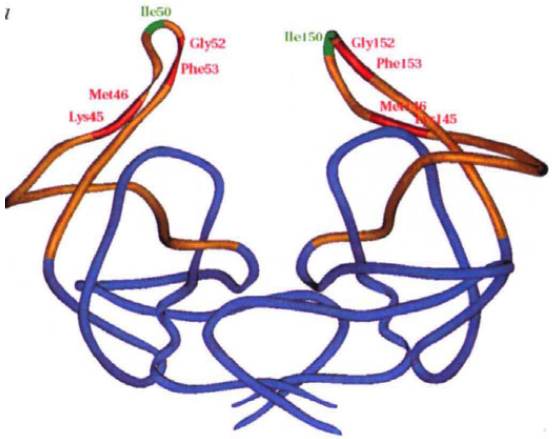
Protein/ligand interactions are complex

ATP + 7,8-diaminononanoate + CO_2 ← → ADP + phosphate + dethiobiotin

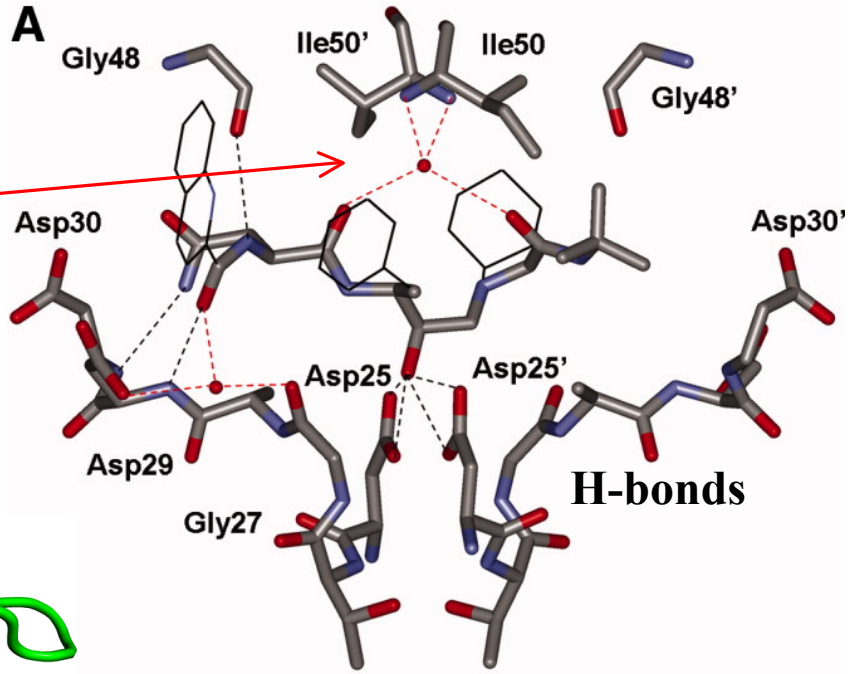


Dethiobiotin Synthetase (1DAM)

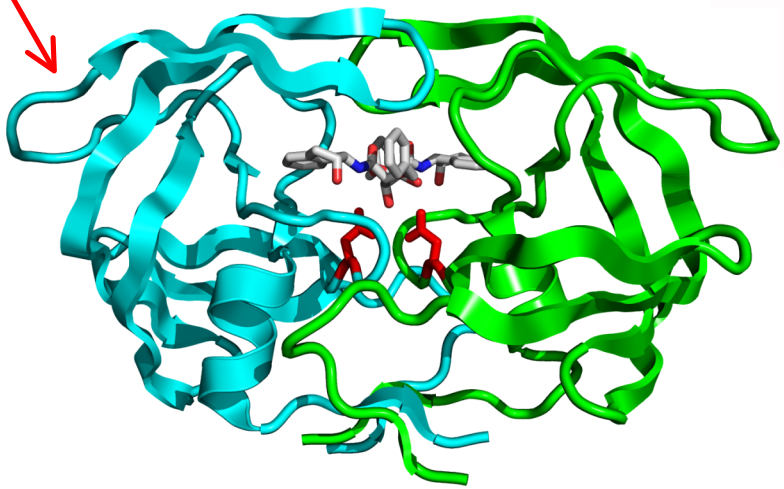
HIV-1 Protease/Inhibitor binding mediated by a key water molecule



H_2O mediates flap/inhibitor interaction



Flap closure upon inhibitor binding



Protein-centric waters improve HIV-1 protease placement and ranking

9 to 1 improvement in RMSD

Ligand/Protein	1HXW	1KZK	1LZQ	1OHR	1SDT	1T7J	2NMW	2O4S	5HVP
1HXW-Ritona	-0.10	-0.95	-0.88	0.31	-0.07	-0.19	0.30	-1.29	-0.21
1KZK-AG1776	0.63	-0.12	-0.55	-0.60	-0.15	-0.85	-0.16	-0.44	-0.33
1KZK-KNI272	0.46	-8.30	-0.18	-0.58	0.14	-0.72	-0.43	-1.06	-0.16
1KZK-KNI764	0.68	0.65	-0.26	-0.54	-0.69	-0.50	0.07	-0.43	0.05
1LZQ-Ethyle	0.03	-2.77	-0.03	-0.02	-0.19	-1.55	-0.58	0.20	0.17
1OHR-Nelfin	-0.60	0.79	-0.20	-0.42	0.34	-1.15	-1.32	-0.06	0.04
1SDT-Indina	-0.16	-0.97	-0.19	-0.23	-0.25	-0.64	-0.54	-0.39	-0.3
1T7J-Ampren	-0.02	-0.10	0.99	0.47	-0.12	-0.43	0.04	-1.07	0.26
2NMW-Saquin	-0.34	0.85	0.34	0.91	0.56	-0.20	0.28	1.07	-0.06
2O4S-Lopina	-0.19	-0.97	-1.09	-0.18	-0.34	-0.09	-0.50	-0.35	0.16
5HVP-Acetyl	-0.04	-0.29	0.75	-0.04	0.05	-0.22	0.35	0.48	-0.06

Protein-centric waters improve HIV-1 protease ranking

12 to 6 improvement in Rank

Ligand/Protein	1HXW	1KZK	1LZQ	1OHR	1SDT	1T7J	2NMW	2O4S	5HVP
1HXW-Ritona	✓	×→✓	×→✓	✓	✓	✓	✓→×	×→✓	✓
1KZK-AG1776	✓	×	✓	✓	✓	✓	✓	✓	✓
1KZK-KNI272	✓	×	✓	✓	✓	✓	✓	×→✓	✓
1KZK-KNI764	✓	✓→×	✓	✓	✓	✓	✓	✓	✓
1LZQ-Ethyle	✓	×→✓	✓	✓	✓	×→✓	×	✓	✓
1OHR-Nelfin	✓	✓→×	✓	✓	✓	×→✓	×→✓	✓	✓
1SDT-Indina	✓	×→✓	✓	✓	✓	✓	×	×→✓	✓
1T7J-Ampren	✓	×	✓	✓	✓	✓	✓	×→✓	✓
2NMW-Saquin	✓	×	×	✓→×	✓→×	✓	×	✓→×	✓
2O4S-Lopina	✓	×	×	✓	✓	✓	×→✓	×	✓
5HVP-Acetyl	✓	×	×	✓	✓	×	×	×	✓

Ligand-centric waters improve CSAR docking RMSDs and ranks

- Waters with 2 protein and 2 ligand contacts = “Tight”
  - Tight waters (~1.1 H_2O per interface)
- Waters with 1 protein and ligand contacts = “Loose”
  - Loose waters (~3.3 H_2O per interface)

Which waters	Study	n	RMSD change > 1		Rank	
			Better	Worse	Better	Worse
Tight	Add water	194	17	13	19	14
	Dock water	194	16	18	25	9
Loose	Add water	299	38	16	51	14
	Dock water	299	35	20	40	20

A) Small molecule challenges for Rosetta

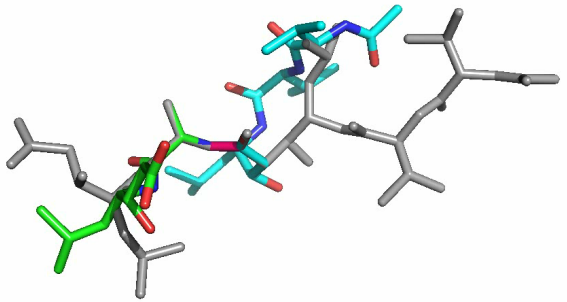
1. Docking multiple ligands, co-factors, waters, metals
2. **Fragment-based docking and design**
3. Merging Ligand based and structure based drug discovery
4. Chemistry inspired atom typing to enable more general scoring functions

Fragment based docking allows for greater ligand flexibility

- Ligands with more than ~7 rotatable bonds fail in docking (David, 2006) because there are too many conformations

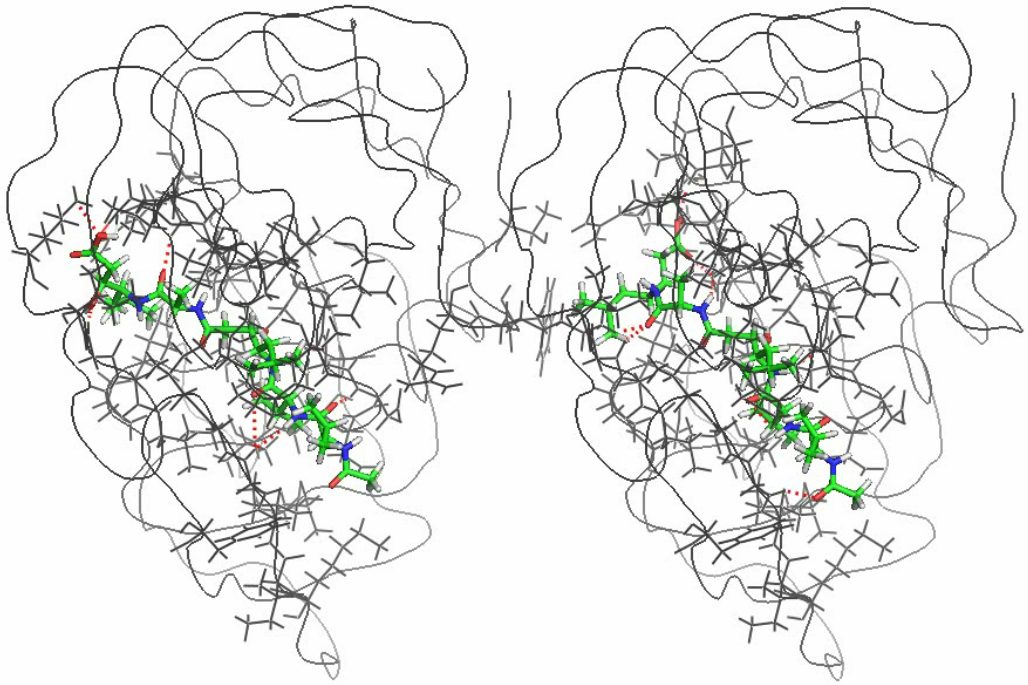
- Solution:

1. Break ligand into fragments
2. Generate fragment rotamer libraries
3. Dock fragments one at a time.



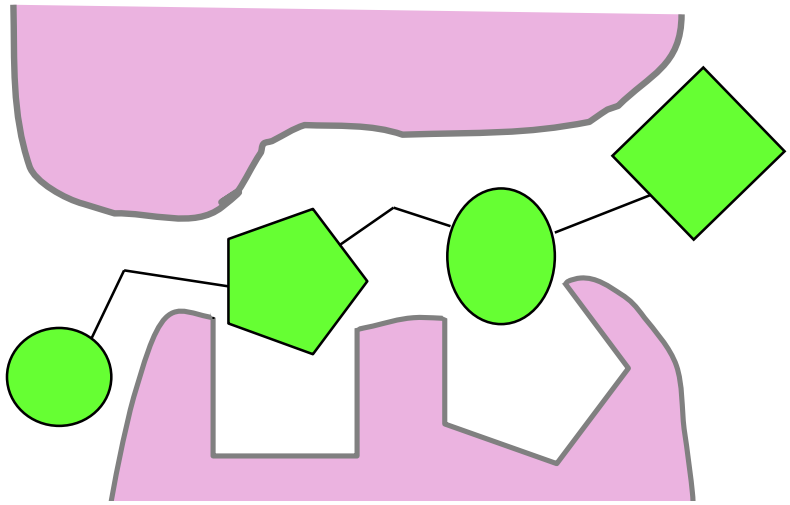
- To demonstrate, HIV-1 PI Acetylpepstatin was split into two fragments, MOE was used to make conformers.

Docking with fragment rotamers increases flexibility

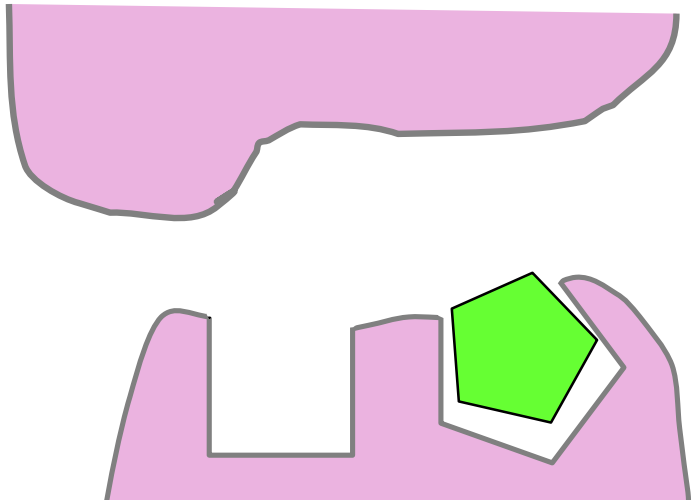


Fragment based screening can greatly expand sampling space

Traditional Screening



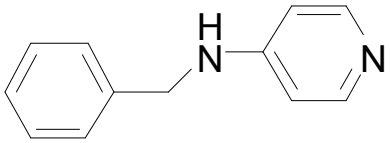
Fragment based screening

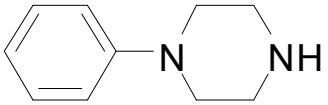


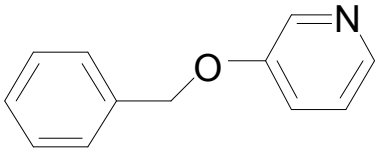
Congreve, M. et al. *Drug Discov.Today* **2003**,8, 876-877

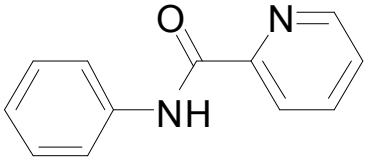
Drug Like Compounds are Built from a Finite Library of Common Fragments

Ring system from drug

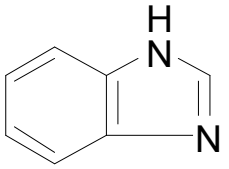


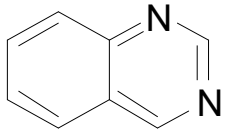


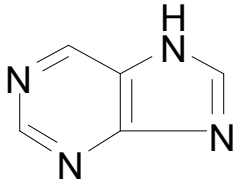


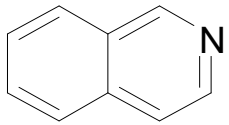


Heterocyclic system

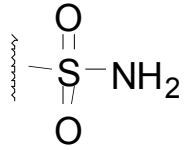


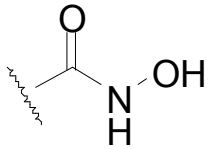


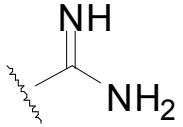


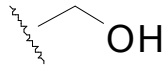


Side chains









Hartshorn, M.J., Murray, C.W.et.al. *J. Med. Chem*. **2005**, 48, 403-413

Ligand design can be directed using chemical property filters

- For instance, Lipinski's rules…

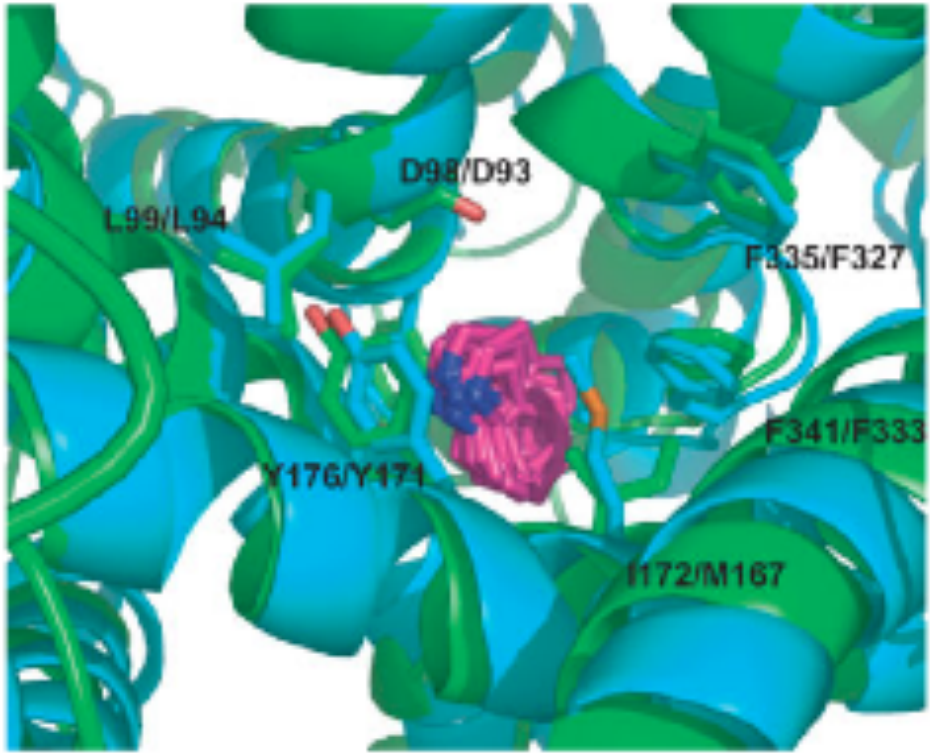
Rule	Filter Name
5 or less H-bond donors	HBondDonorFilter
10 or less H-bond acceptors	HBondAcceptorFilter
Molecular Mass < 500 Daltons	MolecularMassFilter
Log $P_{octonol/water}$ <5	(Future work)

A) Small molecule challenges for Rosetta

1. Docking multiple ligands, co-factors, waters, metals
2. Fragment-based docking and design
3. **Merging Ligand based and structure based drug discovery**
4. Chemistry inspired atom typing to enable more general scoring functions

Integrate Structure and Ligand Based Screening using Structure Activity Relationship (SAR) Data

- ~83,000 structures in PDB
- ~33,000,000 compounds, 600,000 BioAssay studies in PubChem
- Rosetta Comparative modeling can be improved by SAR data
- Kaufmann et al (2009), Structural determinants of species-selective substrate recognition in human and*Drosophila* serotonin transporters revealed through computational docking studies



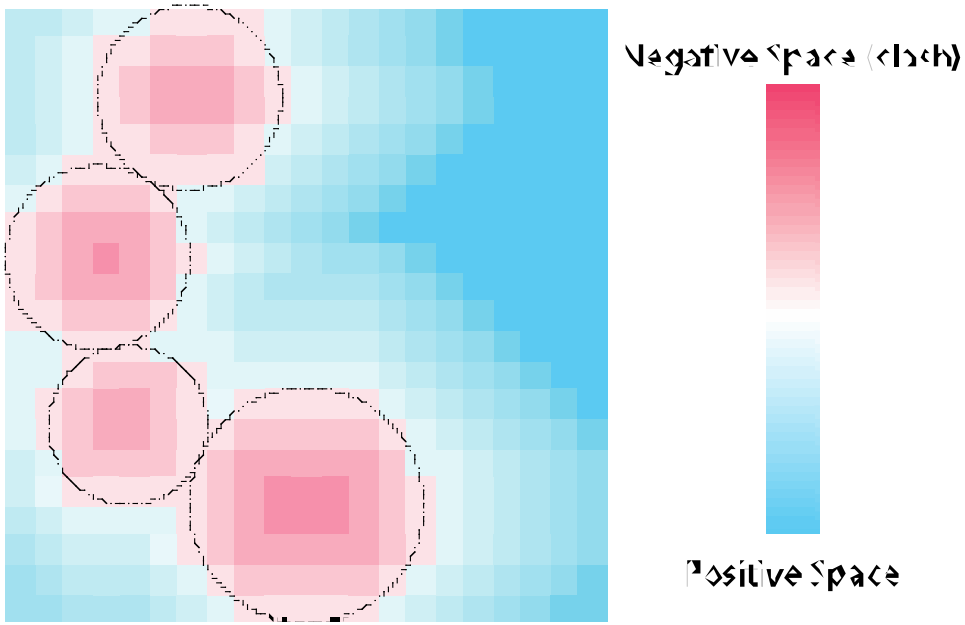
Tryptamine derivatives bind Seratonin Transporters in a *conserved* manner

dSERT model: cyan

hSERT comparative model: green

Precomputed Grids combine ligand and structure based screening

- Low resolution XYZ property grids replace Pose binding site
- Ligand score is the weighted sum of grid space values
- Properties: H-Bonding, Electrostatics, Shape Complementarity
- Ligand Based QSAR data can be used to weight Grid Based Score Function properties



Rosetta: An Integrated Pipeline for High Throughput Screening?

Existing Features

- Loop modeling
- de novo folding
- Protein-protein docking
- Protein-ligand docking

Required Features

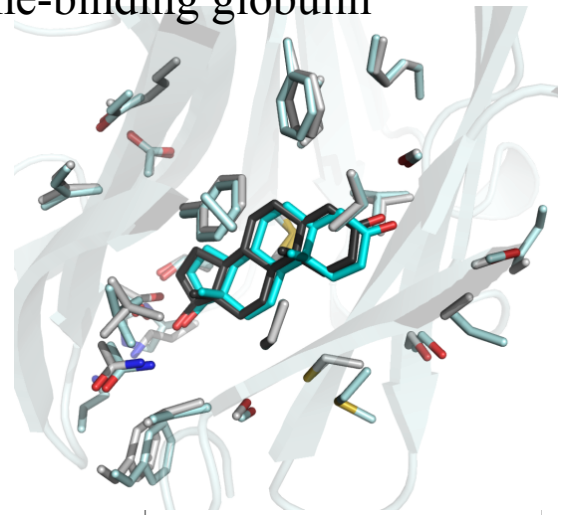
- High Throughput Ligand Docking
- QSAR data integration
- Infrastructure for handling large data sets

A) Small molecule challenges for Rosetta

1. Docking multiple ligands, co-factors, waters, metals
2. Fragment-based docking and design
3. Merging Ligand based and structure based drug discovery
4. **Chemistry inspired atom typing to enable more general scoring functions**

Rosetta score function fails when cation-pi interactions are present

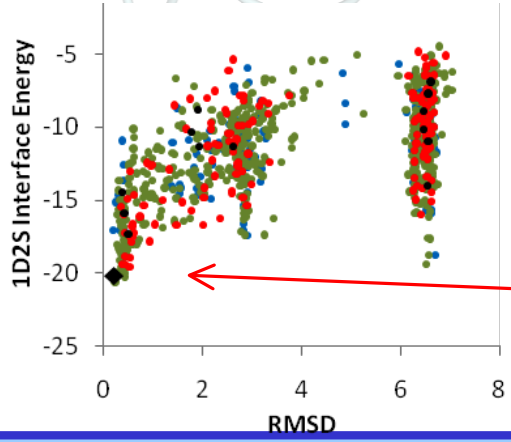
Dihydrotestosterone docked in sex hormone-binding globulin



Gray: native
Teal: model

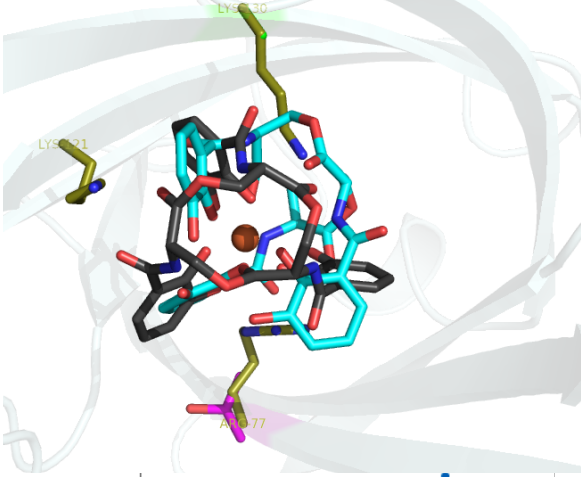
Left: 100% sequence recovery.

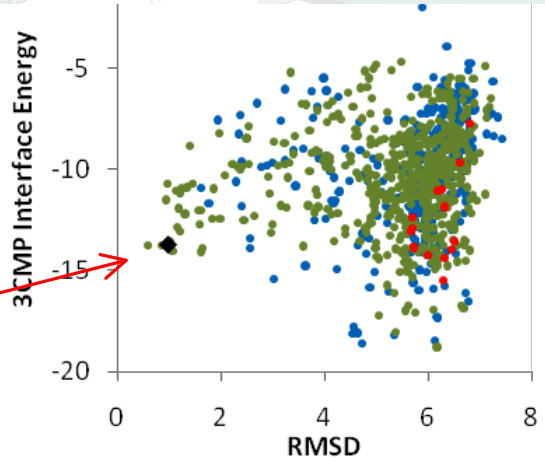
Right: native residues necessary for cation-pi are shown. Two are designed out to Thr (magenta) and Ala (lime).



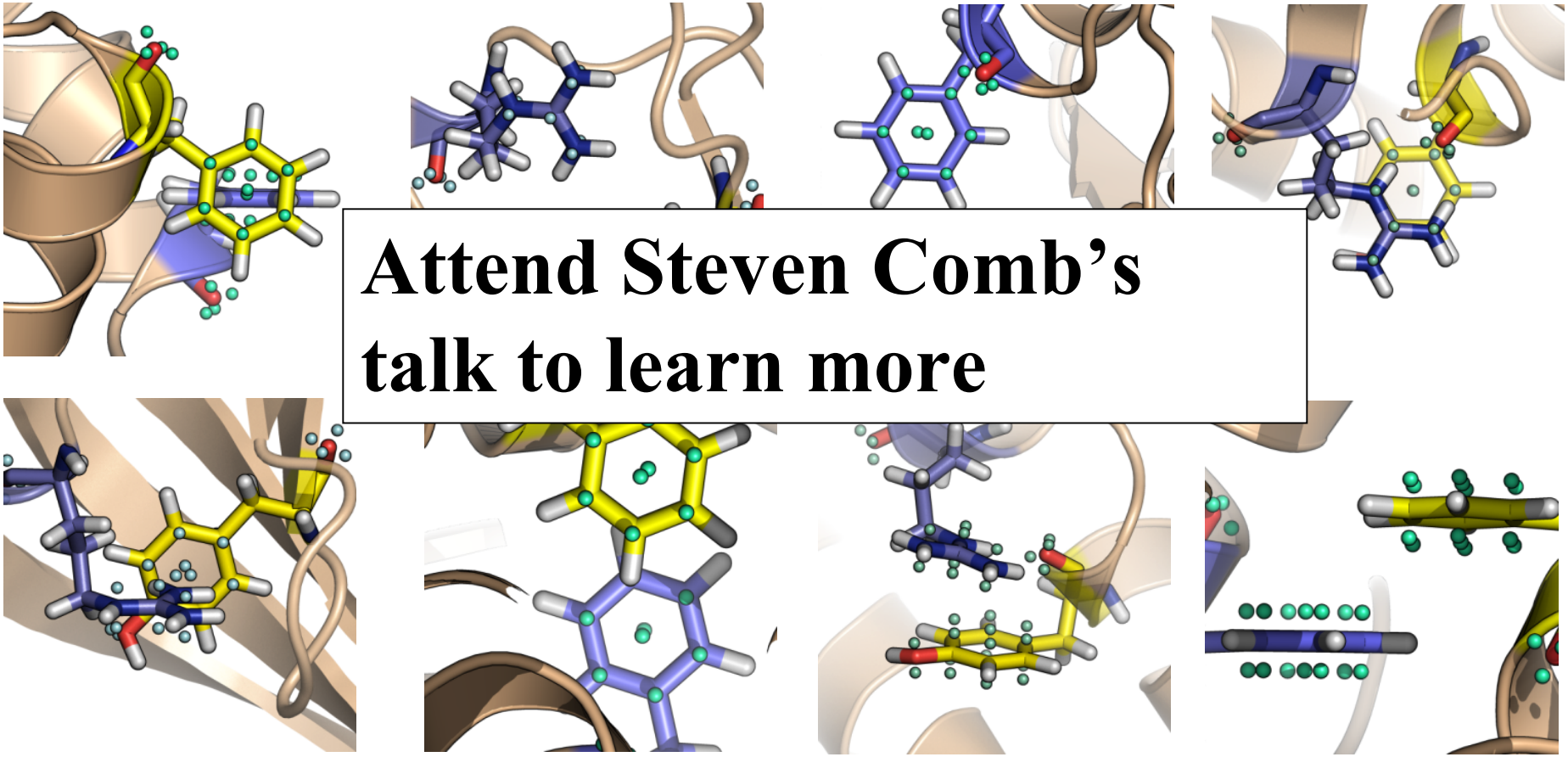
Native

Ferric enterobactin docked in siderocalin





Rosetta Orbitals capture pi-stacking interactions



Attend Steven Comb's talk to learn more

B) Technological Advances needed

1. **Management of large and diverse sets of molecules**
2. Dynamic modification of residues
3. Chemistry-inspired atom typing based on orbital assignments

Managing Large Chemical Libraries in Rosetta is Not Currently Feasible

- Currently
  - Each Residue requires an associated params file
  - Each Residue must have a unique name
  - Each Residue must be loaded into memory prior to PDB/silent file reading
  - For each protein residue and each patch file, a new ResidueType is loaded into memory at run time
- Limitations
  - A large ligand library cannot be processed as a single job
  - Large numbers of patches and non-canonical amino acids cannot be handled
  - Memory Usage increases linearly with the number of loaded Ligands and combinatorily with (non-canonical amino acids)*(patches)

Integrated SQL Database Support Allows for Large Dataset Handling

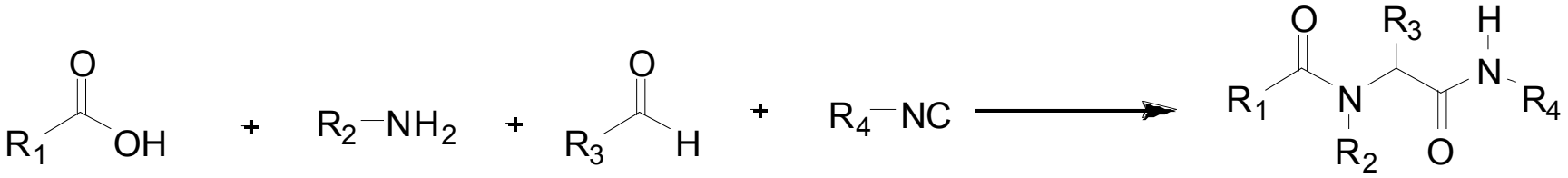
- Cppdb
  - C++ based interface for connecting to a variety of Database Back ends
  - MySQL
  - PostGRE
- Database Pose IO
  - Selectively output based on percentile (output only top 10%) of models)
  - Full support for JD2, including MPI distribution

B) Technological Advances needed

1. Management of large and diverse sets of molecules
2. **Dynamic modification of residues**
3. Chemistry-inspired atom typing based on orbital assignments

Dynamic modification of residues for ligand docking and design

- Add and remove bonds
- Change bond type, bond length
- Change atom types to match new geometry
- Covalently bonding fragments into 1 residue



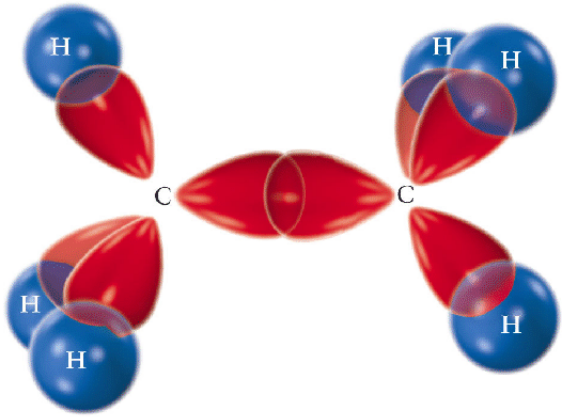
Walters, W.P., Stahl, M.T., Murcko, M.A. *Drug Discovery Today* **1998,** 3,160-178

B) Technological Advances needed

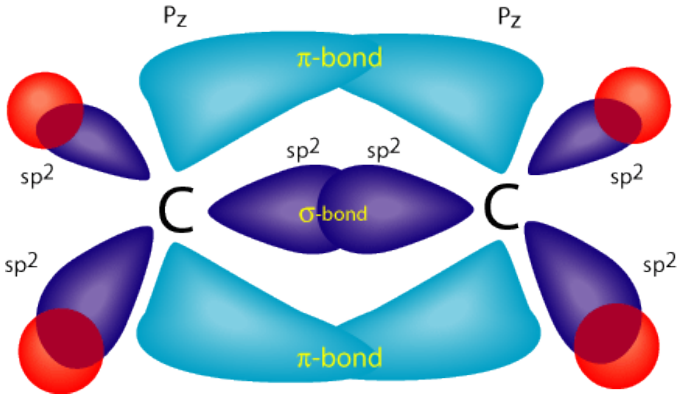
1. Management of large and diverse sets of molecules
2. Dynamic modification of residues
3. **Chemistry-inspired atom typing based on orbital assignments**

Orbital based atom types are unambiguous

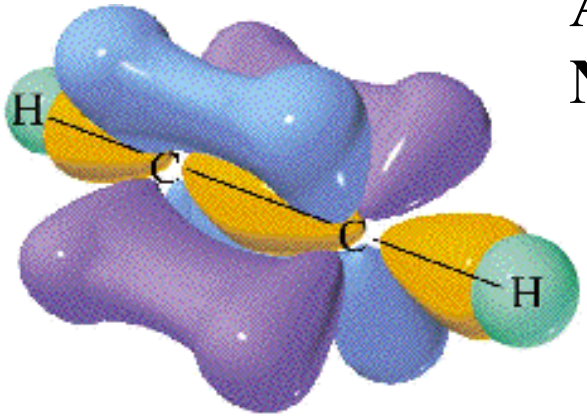
Ethane: **C_TeTeTeTe**



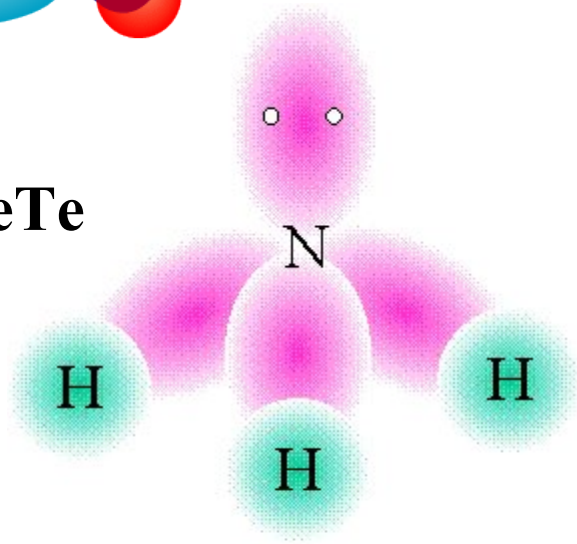
Ethene: **C_TrTrTrPi**



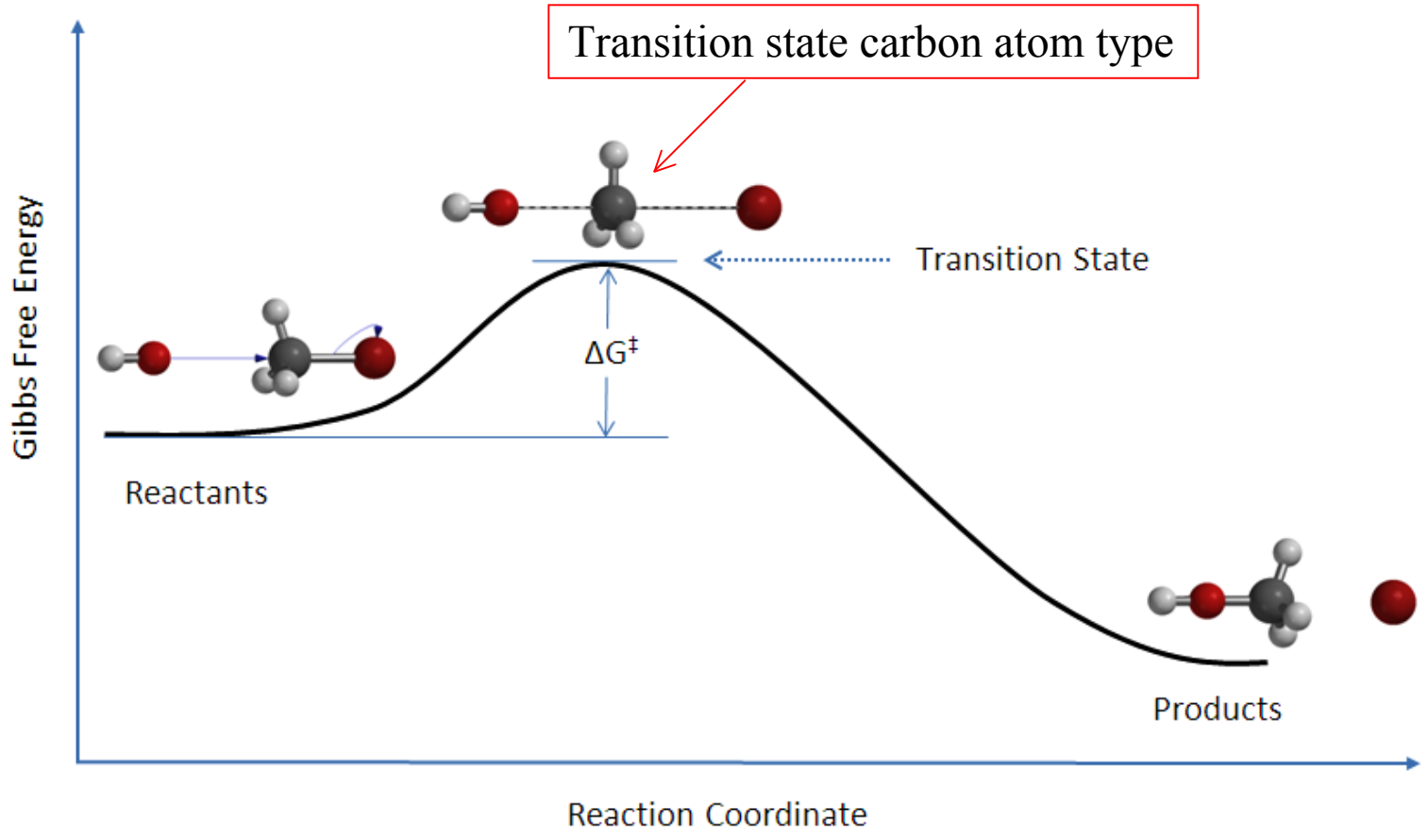
Ethyne:
C_DiDiPiPi



Ammonia:
N_Te2TeTeTe



Orbitals allow Rosetta to model transition states



Reaction: $HO^- + CH_3Br \rightarrow [HO{-}{-}{-}CH_3{-}{-}{-}Br]^{\ddagger} \rightarrow CH_3OH + Br^-$

C) Refactoring Rosetta for Chemistry

1. Residue and ResidueType Refactor
2. Chemical Manager Refactor

ResidueType Refactor: Current adding of 1 Hydrogen

1. Make a new ResidueType from a clone of the original
2. Give the ResidueType a Unique Name (e.g. LG1 -> LG2)
3. Create a Hydrogen atom with a Unique name.
4. Add the atom to the new ResidueType
5. Add a Bond in the new ResidueType to this new Atom
6. Add Icoor data: bond angle, torsion angle, bond length
7. Add new ResidueType to Chemical Manager's ResidueTypeSet
8. Create a Residue of this new ResidueType
9. Select 3 pairs of matching atoms from the old and new residues
10. Replace pose Residue with this new Residue, aligning the new Residue based on the 3 pairs of matching atoms

```
103 void
104 AddHydrogen::apply( core::pose::Pose & pose )
105 {
106     core::conformation::Residue const & res_to_fix= pose.residue(residue_index_);
107     core::chemical::ResidueConnection const & res_connection= res_to_fix.residue_connection(connection
108     core::chemical::AtomICoor const & new_i_coor= res_connection.icoor();
109
110     core::chemical::ResidueTypeOP type_to_fix= res_to_fix.type().clone();
111     type_to_fix->name( generate_unique_name() );
112     core::Size res_conn_atom_index= type_to_fix->residue_connect_atom_index(connection_id_);
113
114
115
116
117
118
119
120
121
122
123     std::string name1= res_to_fix.atom_name(stub_atom1);
124     std::string name2= res_to_fix.atom_name(stub_atom2);
125     std::string name3= res_to_fix.atom_name(stub_atom3);
126
127     core::chemical::SetICoor set_i_coor(
128             "HH",/// name this in the style of the other Hs (H1,H2,H3, etc)
129             new_i_coor.phi(),
130             new_i_coor.theta(),
131             new_i_coor.d(),
132             name1,
133             name2,
134             name3
135     );
136     set_i_coor.apply(*type_to_fix);
137
138     type_to_fix->finalize();
139     {
140         core::chemical::ChemicalManager *cm= core::chemical::ChemicalManager::get_instance();
```

Current approach for adding hydrogens

```
    type_to_fix->finalize();
    {
        core::chemical::ChemicalManager *cm= core::chemical::ChemicalManager::get_instance();
        core::chemical::ResidueTypeSet & rsd_set= cm->nonconst_residue_type_set( core::chemical::FA_STA
        rsd_set.add_residue_type(type_to_fix);
    }
    utility::vector1< std::pair< std::string, std::string > > atom_pairs;
    atom_pairs.push_back(std::pair<std::string, std::string>(name1,name1) );
    atom_pairs.push_back(std::pair<std::string, std::string>(name2,name2) );
    atom_pairs.push_back(std::pair<std::string, std::string>(name3,name3) );

    core::conformation::Residue new_res(*type_to_fix, true);
    //type_to_fix_is_good
    pose.replace_residue(residue_index_, new_res, atom_pairs);
}

std::string generate_unique_name(std::string /*input_name*/){
    core::chemical::ChemicalManager *cm= core::chemical::ChemicalManager::get_instance();
    core::chemical::ResidueTypeSet & rsd_set= cm->nonconst_residue_type_set( core::chemical::FA_STANDAR

    std::string new_name;

    do{
        new_name.clear();
        char a= numeric::random::random_range(65, 90); // ascii range for upper case letters
        char b= numeric::random::random_range(65, 90); // ascii range for upper case letters
        char c= numeric::random::random_range(65, 90); // ascii range for upper case letters


        new_name.append(1,a);
        new_name.append(1,b);
        new_name.append(1,c);

    } while( rsd_set.has_name(new_name));

    return new_name;

}
```

Re

1. A
   A
2. E
   A
3. C

Residue.hh | *ResidueType.hh

```
1842    ResidueTypeSetCAP residue_type_set_;
1843
1844
1845    Size natoms_;
1846    Size nheavyatoms_;
1847    Size n_hbond_acceptors_;
1848    Size n_hbond_donors_;
1849    Size n_orbitals_;
1850    Size n_backbone_heavyatoms_;
1851    Size first_sidechain_hydrogen_;
1852    Size ndihe_;
1853    utility::vector1< std::string > atom_name_;
1854    utility::vector1< std::string > mm_atom_name_;
1855    utility::vector1<std::string > orbital_name_;
1856    utility::vector1< Size        > atom_type_index_;
1857    utility::vector1< Size        > mm_atom_type_index_;
1858    utility::vector1<core::Size   > orbital_type_index_;
1859    utility::vector1< Real        > atomic_charge_;
1860    utility::vector1< utility::vector1<core::Size > > orbital_bonded_neighbor_;
1861    utility::vector1< AtomIndices > bonded_neighbor_;
1862    utility::vector1<utility::vector1<BondName> > bonded_neighbor_type_;
1863    utility::vector1< AtomIndices > cut_bond_neighbor_;
1864    utility::vector1< Size        > atom_base_;
1865    utility::vector1< Size        > abase2_; // acceptors only
1866    utility::vector1< Size        > attached_H_begin_;
1867    utility::vector1< Size        > attached_H_end_;
1868    utility::vector1< AtomICoor   > icoor_;
1869    utility::vector1< orbitals::ICoorOrbitalData> orbital_icoor_id_;
1870    utility::vector1< orbitals::ICoorOrbitalData> new_orbital_icoor_id_;
1871    utility::vector1< Vector > xyz_;
1872    utility::vector1< Vector      > orbital_xyz_;
1873    utility::vector1< Size          > parents_;
1874    utility::vector1< dihedral_atom_set > dihedral_atom_sets_;
1875    utility::vector1< utility::vector1< Size > > dihedrals_for_atom_;
1876    utility::vector1< bondangle_atom_set > bondangle_atom_sets_;
1877    utility::vector1< utility::vector1< Size > > bondangles_for_atom_;
1878    utility::vector1< Size > last_controlling_chi_;
1879    utility::vector1< AtomIndices > atoms_last_controlled_by_chi_;
1880    AtomIndices atoms_with_orb_index_;
1881    AtomIndices Haro_index_;
1882    AtomIndices Hpol_index_;
1883    AtomIndices accpt_pos_;
1884    AtomIndices Hpos_polar_;
1885    AtomIndices Hpos_apolar_;
```

Refactor ResidueType: Step 2

** Atom Ordering Leads to Static Residues **

1. Currently atoms are ordered: N, C, CA, O, <side chain heavy atoms>, <hydrogen atoms>
2. Atom ordering limits ligand design
3. Remove atom ordering while preserving ResidueType interface.
4. Multiple data-structures holding pointers to Atoms: Map of Atom name to AtomOP, List of Hydrogen AtomOPs, List of Heavy AtomOPs, etc.

Refactor Step 3

- Modify Residue Interface, removing backward compatibility

This:

```
pose.residue(3).heavy_atom_is_an_acceptor(atom_id)
```

Becomes:

```
pose.residue(3).atom(atom_id).is_acceptor()
```

- Biopython like selection?
  - model[‘A’][37][‘ca’]

Refactoring the ChemicalManager Will Reduce Memory Requirements

- "Lazy Loading" of Residues:
  - Load residue definitions when needed, unload afterwards
  - Requires coordination between multiple process threads
- "Lazy Loading" of Patches
  - Most patches are never used
  - Identify required patches during input parsing
- Unique Identification of Residues
  - Residues are uniquely identified by name.
  - Meaningful 3-letter code output limits to 46656 ligands