Thermochemistry Estimation: A Code Perspective

RMG Study Group

Max Liu

September 29, 2016

Questions to be answered

- How does RMG get thermo data for a species?
- What options are there to control this?
- Where can I find the code that does…?

Documentation and API

- http://reactionmechanismgenerator.github.io/RMG-Py/
- Documentation is written manually
  - To change, modify appropriate document in documentation/source folder
  - Pull request to official master
  - Follow instructions to update documentation site, or ask someone to do so: https://github.com/ReactionMechanismGenerator/RMG-Py/wiki/Updating-Documentation
- API is generated automatically from docstrings

```
class RMG(util.Subject):
    """
    A representation of a Reaction Mechanism Generator (RMG) job. The
    attributes are:
    ...
    """
```

Database loading

Specify databases in input file

```
database(
    thermoLibraries = ['primaryThermoLibrary'],
    reactionLibraries = [],
    seedMechanisms = [],
    kineticsDepositories = ['training'],
    kineticsFamilies = 'default',
    kineticsEstimator = 'rate rules',
)
```

`rmgpy.rmg.loadDatabase` creates `rmgpy.data.rmg.RMGDatabase()` object

`RMGDatabase().load`

- Create `rmgpy.data.thermo.ThermoDatabase()` object
  - Create `rmgpy.data.thermo.ThermoLibrary()` object for each library specified
  - Create `rmgpy.data.thermo.ThermoGroups()` object for each groups file

Global RMGDatabase

- RMG stores the database as a module level variable in `rmgpy.data.rmg`
- Upon first instantiation of `RMGDatabase()`, the object is created and assigned to the global variable `database`
- Creating additional `RMGDatabase()` instances will raise a warning:

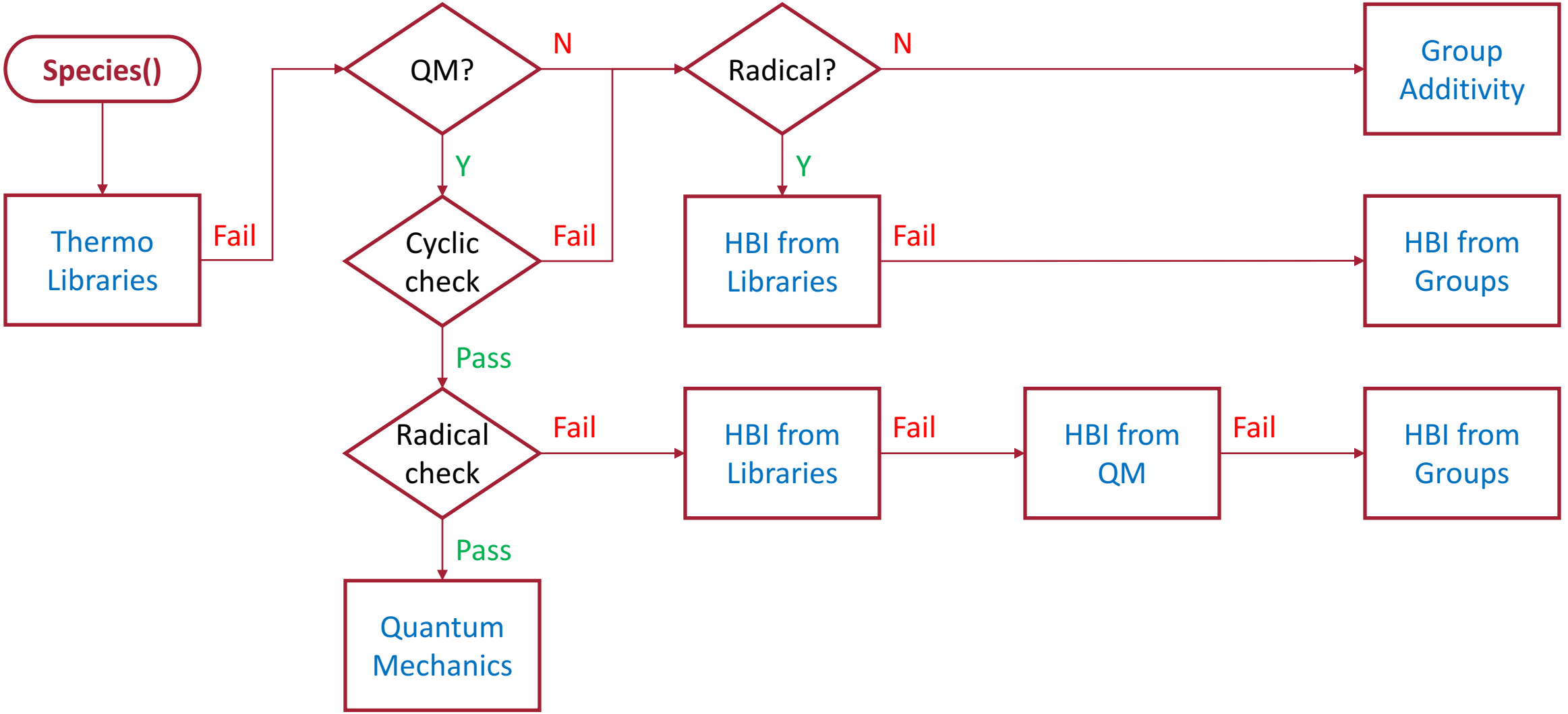
  Should only make one instance of RMGDatabase because it's stored as a module-level variable!

  Unexpected behaviour may result!
- What will happen?
  - `rmgpy.data.rmg.getDB` retrieves databases – queries global `database`
  - Only data loaded into the first `RMGDatabase()` instance will be retrieved
- Not important for normal operation
- Be careful with custom scripts and IPython notebooks

Species creation

- `rmgpy.rmg.model.CoreEdgeReactionModel.makeNewSpecies`
- Creates a new species from `Molecule()` object or `Species()` object (loading from restart)
- Generates resonance isomers: `Species().generateResonanceIsomers()`
- Submit for parallel processing, if enabled: `rmgpy.thermo.thermoengine.submit()`
- Generate resonance isomers again??
- `generateThermoData()` >> `ThermoDatabase().getThermoData()`

Thermo logic



1) Thermo libraries

- `ThermoDatabase().getThermoDataFromLibraries()`
- If solvation is on:
  - Loop through library list and identify liquid phase libraries
  - Loop through liquid phase libraries:
    - `ThermoDatabase().getThermoDataFromLibrary()`
  - Remove liquid phase libraries from list
- Loop through library list
  - `ThermoDatabase().getThermoDataFromLibrary()`
- `getThermoDataFromLibrary()` performs isomorphism checks against every entry in a library and returns the first match with data
- Only 1 result is returned for each species

2) Quantum mechanics

- Activated via input file

```
quantumMechanics(
    software='mopac',
    method='pm3',
    fileStore='QMfiles',
    scratchDirectory = None,
    onlyCyclics = True,
    maxRadicalNumber = 0,
)
```

- Currently supported software/method combinations:
  - `'gaussian': 'pm3', 'pm6'`
  - `'mopac': 'pm3', 'pm6', 'pm7'`

2) Quantum mechanics

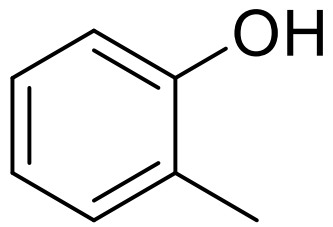
- `rmgpy.qm.main.QMCalculator().getThermoData()`
  - Setup appropriate object for method chosen
  - Run that class's `generateThermoData()` method
- `rmgpy.qm.molecule.QMMolecule().generateThermoData()`
  - Check if QM data already exists (default path QMfiles/method)
  - Call `generateQMData()`, which performs actual setup and calculation
    - Located in `rmgpy.qm.mopac.MopacMol()` or `rmgpy.qm.mopac.GaussianMol()`
    - Check if QM output file already exists in the scratch directory (default None)
    - Returns None if any atoms are N5s, N5d, N5dd, N5t, N5b
  - If successful, tries to determine point group (using SYMMETRY)
  - If successful, saves thermo data file (QMfiles/method) and returns thermo

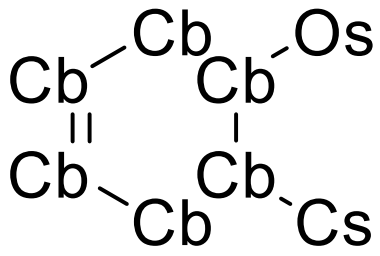
3) Group additivity

- `ThermoDatabase().computeGroupAdditivityThermo()`
- Loop through every non-hydrogen atom in molecule
  - Descend ‘group’ tree to find most specific node with data
  - If not cyclic: search ‘gauche’ tree
  - Search ‘int15’ tree
  - Search ‘other’ tree
- If cyclic:
  - Get list of monorings and polyrings
  - For each monoring, search in ‘ring’ tree
  - For each polyring, try to search in ‘polycyclic’ tree
    - If unsuccessful, apply polycyclic heuristic (bicyclic decomposition)
- Apply symmetry correction to entropy

Group corrections

Benson type groups

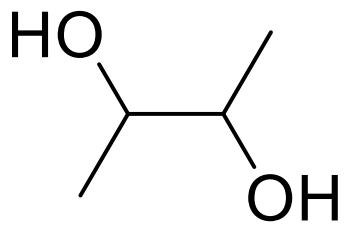


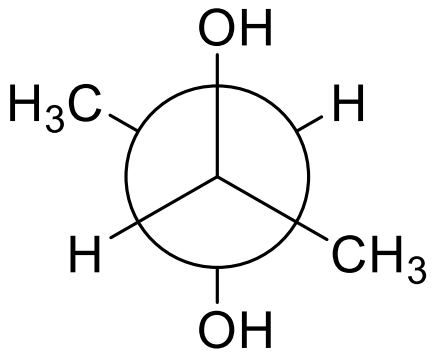



```
4×group(Cb-H)
1×group(Cb-Os)
1×group(Cb-Cs)
1×group(Cs-CbHHH)
1×group(Os-CbH)
```

Non-nearest neighbor corrections

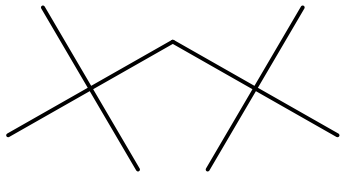
Gauche





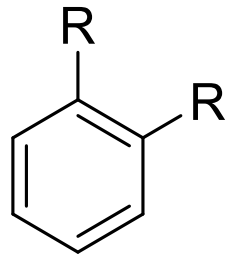

```
2×gauche(Cs(Cs(CsRR)CsRR))
2×gauche(Cs(Cs(CsRR)RRR))
2×gauche(Os(Cs(CsCsR)R))
```

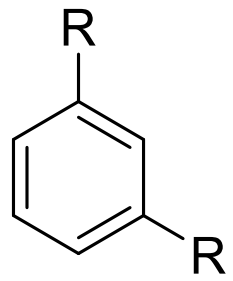
1,5-interaction

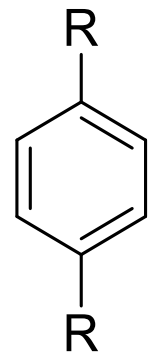


```
1×int15(Cs(Cs(CsCsCs)Cs(CsCsCs)RR))
```

In development: aromatic substituents



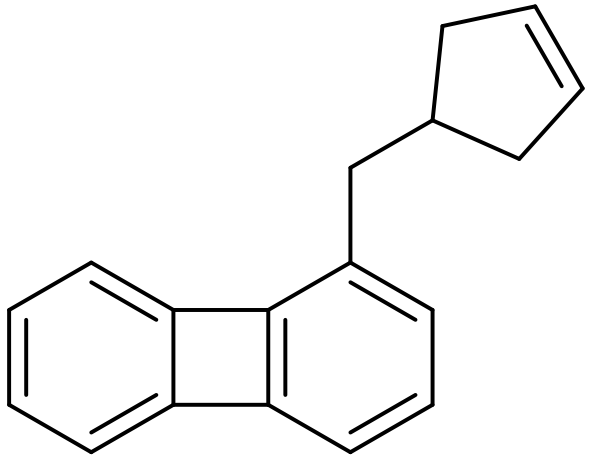




Ring corrections

Smart polycyclic thermo (by Kehang)

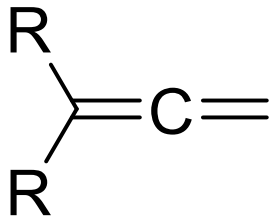
`ThermoDatabase().__addPolycyclicCorrectionThermoData()`



```
1×ring(Cyclopentene)
2×polycyclic(s2_4_6_ben)
-1×polycyclic(Cyclobutane)
```

Other corrections

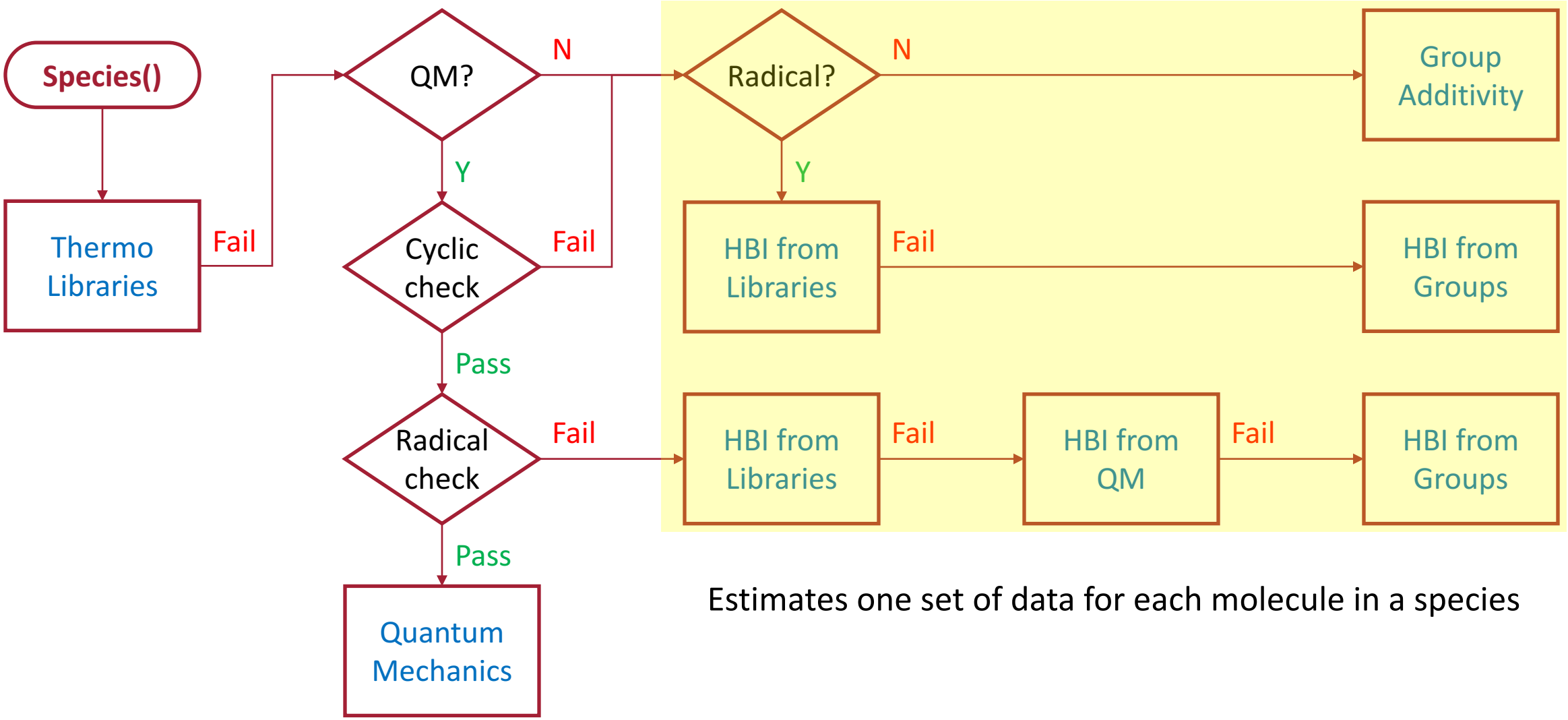
Only ketene corrections right now



4) Hydrogen bond increment

- `ThermoDatabase().estimateRadicalThermoViaHBI()`
- First calculate thermo of saturated molecule using specified method
- Apply symmetry correction based on radical structure
- For each saturated atom
  - Remove hydrogen atom(s)
  - Use the radical atom to descend the ‘radical’ tree and apply correction
  - Restore hydrogen atom(s)
  - Subtract enthalpy of added hydrogen atom(s) – value is hard coded

Thermo logic



Estimates one set of data for each molecule in a species

Choosing thermo values

- QM on, cyclic check satisfied, radical check failed:
  - Assign priorities:
    - HBI from libraries = 1
    - HBI from QM = 2
    - HBI from groups = 3
  - Sort by priority followed by lowest H298
- QM off, radical species, using HBI from libraries: Sort by lowest H298
- Remaining cases (group additivity or HBI from groups)
  - `ThermoDatabase().prioritizeThermo()`
  - Non-cyclic: sort by lowest H298
  - Cyclic:
    - Sum rank of every ring or polycyclic group (entries with no rank are assigned rank 3)
    - Sort by rank followed by lowest H298

ThermoData is finally returned…

- `rmgpy.data.thermo.findCp0andCpInf()`
- `rmgpy.thermo.thermoengine.processThermoData()`
- Convert to Wilhoit format
- Add solvation correction
- Add conformer to species if none exists
- Save E0 to conformer
  - Wilhoit format extrapolates enthalpy to 0 K (coded as 0.001 K)
- Convert to specified thermo format (default is NASA)
- Compute error due to conversion if final format is different from original
  - However, we don’t do anything with this information right now!

How to get better thermo?

- Use an existing library
- Use on-the-fly quantum calculations
- If no suitable library exists, make a library from literature
- If no literature exists, make a library from quantum calculations
  - Can be done using results from on-the-fly quantum calculations
- Add or update group values