CAOS Documentation

Release 0.1.0

Dan Obermiller

November 09, 2015

Contents

CAOS is a useful tool for many organic chemists, but is often a hard one to use in practice. This library will seek to provide an easy method of predicting reactions.

CHAPTER 1

Documentation

Is available at readthedocs.org.

CHAPTER 2

Examples

Are not available at this time - much of it hasn't been implemented to the point where an example would be helpful.

Currently the registration of reaction mechanisms has been implemented, as well as performing reactions. No work on loading molecules, representing those molecules, or analying them has been completed.

CHAPTER 3

Todos:

- [X] Add CI
- [X] Add reaction registration and dispatch
- [ ] Add loading molecules
- [ ] Add molecule inspection
- [ ] Add common requirements functions
- [ ] ???

CAOS is still in early stages of development. Information will be added as it becomes available.

3.1 Motivation

This is a project for my Fall 2015 DSLs class. It is loosely based off of a previous project however with the intent of being more modular, extensible, and language-like. While I can't say much about how it should look, I can say that I'd like to eventually provide this sort of interface to users.

```
import my_reaction_mechanisms
from CAOS import react, load_molecule_from_file

reactant1 = load_molecule_from_file("filename.cml")
reactant2 = load_molecule_from_file("filename.smiles", type="SMILES")
products = react([reactant1, reactant2], conditions={})

products.show()
```

I'd also like to allow users to register new reaction mechanisms and new molecular data structures in order to meet their own needs

```
@register_reaction_mechanism(name, requirements, molecule_type)
def diels_alder_reaction(products, conditions=None):
    ...

@register_molecule_type()
class DielsAlderStructure(object):

  @classmethod
  def from_default(cls, molecule):
      ...
```

As these details become more firmly defined, this file will become more useful.

3.1.1 CAOS package

Submodules

CAOS.dispatch module

Handles registration and dispatch of reactions and molecule types.

Provides two decorators that are aliases for classes:

> decorator alias -> ClassName register_reaction_mechanism -> ReactionDispatcher register_molecule_type -> MoleculeTypeDispatcher

This allows the reaction system to determine which type of reaction and what representation of molecules should be used, all occuring dynamically, at runtime.

Attributes

react: function Function that attempts to react molecules under given conditions

register_reaction_mechanism: function Registers a reaction mechanism with the dispatch system.

reaction_is_registered: function Checks whether or not a reaction has been registered.

class `CAOS.dispatch.`**`ReactionDispatcher`**(*mechanism_name*, *requirements*)

Class that dispatches on reaction types.

`function`

The function to be called when using this reaction.

Returns callable

The function that has been registered for this reaction.

Notes

Should not be called directly - let the *react* function handle that.

`name`

The name assigned to the mechanism.

Returns string

The name the mechanism has been registered as.

Raises **ExistingReactionError**

The name must be unique - if an existing mechanism shares this name it will cause an error.

`namespace`

Shortcut to this mechanism's part of the namespace.

Returns dict

Contains the requirements that must be met to dispatch this function, as well as the function itself.

Notes

Assumes that the name of this mechanism is already known. If you unset the name, this will behave strangely or error.

requirements
The requirements of this reaction mechanism.

Returns dict

Mapping from requirement name to some callable that can be used to determine if the parameters meet the requirement.

Raises **InvalidReactionError**

If any of the requirements aren't callable then an error is raised.

CAOS.dispatch.**register_reaction_mechanism**
alias of *ReactionDispatcher*

CAOS.chem_logging module

Singleton logging for the language.

When verbose mode is enabled, logged messages are written to stdout or stderr, depending on the type of message. Otherwise they are ignored.

class CAOS.chem_logging.**DummyLogger**
Bases: object

Fake logger I'm going to use for now.

Will return something valid in all cases.

CAOS.chem_logging.**logger**
Fake logger I'm going to use for now.

Will return something valid in all cases.

CAOS.util module

Utility functions that aren't core functionality.

CAOS.util.**raises**(*exception_types*, *function*, *args=None*, *kwargs=None*)
Return whether or not the given function raises the error.

Parameters **exception_types: tuple, Exception**

Tuple of the types of the exceptions (or a single type of exception) that should be caught.

function: callable

The function to be called

args: collection, optional

List of positional arguments to be used

kwargs: mapping, optional

Dictionary of keyword arguments to be used

Examples

It should return *False* when given a valid value

```
>>> raises(ValueError, int, ["3"])
False
```

It should return *True* when given an invalid value that results in the expected error

```
>>> raises(ValueError, int, ["hello"])
True
```

It should raise an error if it gets an unexpected error

```
>>> raises(UnboundLocalError, int, ["hello"])
Traceback (most recent call last):
    ...
ValueError: invalid literal for int() with base 10: 'hello'
```

Subpackages

CAOS.exceptions package

Submodules

CAOS.exceptions.dispatch_errors module Errors that occur while dispatching the mechanism or type.

exception CAOS.exceptions.dispatch_errors.**DispatchException**
Bases: exceptions.Exception

Generic error raised when some problem occurs during dispatch.

exception CAOS.exceptions.dispatch_errors.**ExistingReactionError**
Bases: *CAOS.exceptions.dispatch_errors.DispatchException*

A mechanism with this name has already been registered.

exception CAOS.exceptions.dispatch_errors.**InvalidReactionError**
Bases: *CAOS.exceptions.dispatch_errors.DispatchException*

The reaction being registered is invalid in some way.

CAOS.exceptions.reaction_errors module Reaction errors; i.e. those that occur during a reaction.

exception CAOS.exceptions.reaction_errors.**FailedReactionError**
Bases: exceptions.Exception

Indicates that a reaction failed to occur.

Module contents Exceptions used by the library.

Module contents

CAOS module.

3.1.2 CAOS.exceptions package

Submodules

CAOS.exceptions.dispatch_errors module

Errors that occur while dispatching the mechanism or type.

exception CAOS.exceptions.dispatch_errors.**DispatchException**
Bases: exceptions.Exception

Generic error raised when some problem occurs during dispatch.

exception CAOS.exceptions.dispatch_errors.**ExistingReactionError**
Bases: *CAOS.exceptions.dispatch_errors.DispatchException*

A mechanism with this name has already been registered.

exception CAOS.exceptions.dispatch_errors.**InvalidReactionError**
Bases: *CAOS.exceptions.dispatch_errors.DispatchException*

The reaction being registered is invalid in some way.

CAOS.exceptions.reaction_errors module

Reaction errors; i.e. those that occur during a reaction.

exception CAOS.exceptions.reaction_errors.**FailedReactionError**
Bases: exceptions.Exception

Indicates that a reaction failed to occur.

Module contents

Exceptions used by the library.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

C

C

D

E

F

I

L

N

R