# **OWLAPY**

# Release 1.1.1

# **Ontolearn Team**

Jul 19, 2024

# **Contents:**

1	About owlapy	2
	1.1 What is owlapy?	
	1.2 What does owlapy have to offer?	
	1.3 How to install?	3
2	Basic Usage	3
	2.1 Atomic Classes	4
	2.2 Object Property	4
	2.3 Complex class expressions	
	2.4 Convert to SPARQL, DL or Manchester syntax	5
3	Ontologies	6
	3.1 Loading an Ontology	6
	3.2 Modifying an Ontology	6
	3.3 Save an Ontology	
	3.4 Worlds	8
4	Reasoners	9
	4.1 Usage of the Reasoner	10
	4.2 Class Reasoning	10
	4.3 Object Properties and Data Properties Reasoning	11
	4.4 Find Instances	11
5	Reasoning Details	12
	5.1 Sync Reasoner	12
	5.2 Isolated World	12
	5.3 Capabilities	13
	5.4 Concrete Example	14
6	Owlapi Adaptor	17
	6.1 Initialization	17
	6.2 Notes	17
	6.3 Examples	17
7	owlapy	18
	7.1 Subpackages	18
	7.2 Submodules	52

Index		147
Python 1	Module Index	146
7.5	Package Contents	145
7.4	Functions	145
7.3	Attributes	144

OWLAPY<sup>1</sup>: Representation of OWL objects in python.

# 1 About owlapy

Version: owlapy 1.1.1

GitHub repository: https://github.com/dice-group/owlapy

**Publisher and maintainer:** DICE<sup>2</sup> - data science research group of Paderborn University<sup>3</sup>.

Contact: onto-learn@lists.uni-paderborn.de

License: MIT License

# 1.1 What is owlapy?

Owlapy is an open-source software library in python that is used to represent entities in OWL 2 Web Ontology Language.

We identified the gap of having a library that will serve as a base structure for representing OWL entities and for manipulating OWL Ontologies in python, and like that, owlapy was created. Owlapy is loosely based on its java-counterpart, *owlapi*. Owlapy is currently utilized by powerful libraries such as Ontolearn<sup>4</sup> and OntoSample<sup>5</sup>.

Owlapy is the perfect choice for machine learning projects that are built in python and focus on knowledge graphs and class expression learnings.

# 1.2 What does owlapy have to offer?

- Create, manipulate and save Ontologies.
- Retrieving information from the signature of the ontology.
- · Reasoning over ontology.
- Represent every notation in OWL 2 Structural Specification and Functional-Style Syntax<sup>6</sup> including:
  - Entities, Literals, and Anonymous Individuals

<sup>&</sup>lt;sup>1</sup> https://github.com/dice-group/owlapy

<sup>&</sup>lt;sup>2</sup> https://dice-research.org/

<sup>&</sup>lt;sup>3</sup> https://www.uni-paderborn.de/en/university

<sup>&</sup>lt;sup>4</sup> https://github.com/dice-group/Ontolearn

<sup>&</sup>lt;sup>5</sup> https://github.com/alkidbaci/OntoSample

<sup>6</sup> https://www.w3.org/TR/owl2-syntax/

- Property Expressions
- Data Ranges
- Class Expressions
- Axioms
- Annotations
- · Construct complex class expressions.
- · Provide interfaces for OWL Ontology, Ontology manager and Reasoner.
- · Convert owl expression to SPARQL queries.
- Render owl expression to Description Logics or Manchester syntax.
- Parse Description Logics or Manchester expression to owl expression.

## 1.3 How to install?

Installation from source:

```
git clone https://github.com/dice-group/owlapy conda create -n temp_owlapy python=3.10.13 --no-default-packages && conda activate... otemp_owlapy && pip3 install -e .
```

#### or using PyPI:

```
pip3 install owlapy
```

# 2 Basic Usage

The main usage for owlapy is to use it for class expression construction. Class expression learning algorithms require such basic structure to work upon. Let's walk through an example of constructing some class expressions.

In this example we will be using the *family* ontology, a simple ontology with namespace: http://example.com/family#. Here is a hierarchical diagram that shows the classes and their relationship:

```
Thing

|
person

/ |
male female
```

It contains only one object property which is hasChild and in total there are six persons (individuals), of which four are males and two are females.

# 2.1 Atomic Classes

To represent the classes male, female, and person we can simply use the class OWLClass<sup>7</sup>:

```
from owlapy.class_expression import OWLClass
from owlapy.iri import IRI

namespace = "http://example.com/family#"

male = OWLClass(IRI(namespace, "male"))
female = OWLClass(IRI(namespace, "female"))
person = OWLClass(IRI(namespace, "person"))
```

Notice that we created an IRI object for every class. IRI<sup>8</sup> is used to represent an *IRI*. Every named entity requires an IRI, whereas Anonymous entities does not. However, in owlapy you can create an *OWLClass* by passing the *IRI* directly as a string, like so:

```
male = OWLClass("http://example.com/family#male")
```

# 2.2 Object Property

To represent the object property has Child we can use the class OWLObjectProperty9:

```
from owlapy.owl_property import OWLObjectProperty
hasChild = OWLObjectProperty("http://example.com/family#hasChild")
```

**Tip:** In owlapy the naming of the classes is made in accordance with the notations from OWL 2 specification but with the word "OWL" in the beginning. Example: "OWLObjectProperty" represents the notation "ObjectProperty".

# 2.3 Complex class expressions

Now that we have these atomic entities, we can construct more complex class expressions. Let's say we want to represent all individuals which are male and have at least 1 child.

We already have the concept of male. We need to find the appropriate class for the second part: "have at least 1 child". In OWL 2 specification that would be ObjectMinCardinality<sup>10</sup>. In owlapy, as we said, we simply add the word "OWL" upfront to find the correct class:

```
from owlapy.class_expression import OWLObjectMinCardinality
has_at_least_one_child = OWLObjectMinCardinality(
    cardinality = 1,
    property = hasChild,
    filler = person
)
```

 $<sup>^7\</sup> https://dice-group.github.io/owlapy/autoapi/owlapy/class\_expression/owl\_class/index.html\#owlapy.class\_expression.owl\_class.OWLClass$ 

<sup>&</sup>lt;sup>8</sup> https://dice-group.github.io/owlapy/autoapi/owlapy/iri/index.html#owlapy.iri.IRI

 $<sup>^9 \</sup> https://dice-group.github.io/owlapy/autoapi/owlapy/owl\_property/index.html\#owlapy.owl\_property.OWLObjectProperty$ 

<sup>10</sup> https://www.w3.org/TR/owl2-syntax/#Minimum\_Cardinality

As you can see, to create an object of class OWLObjectMinCardinality<sup>11</sup> is as easy as that. You specify the cardinality which in this case is 1, the object property where we apply this cardinality restriction and the filler class in case you want to restrict the domain of the class expression. In this case we used person.

Now let's merge both class expressions together using OWLObjectIntersectionOf<sup>12</sup>:

```
from owlapy.class_expression import OWLObjectIntersectionOf

ce = OWLObjectIntersectionOf([male, has_at_least_one_child])
```

# 2.4 Convert to SPARQL, DL or Manchester syntax

Owlapy is not just a library to represent OWL entities, you can also use it to convert owl expressions into other formats:

To parse a DL or Manchester expression to owl expression you can use the following convenient methods:

In these examples we showed a fraction of **owlapy**. You can explore the *api documentation* to learn more about all classes in owlapy and check more examples in the examples <sup>13</sup> directory.

 $<sup>^{11}\</sup> https://dice-group.github.io/owlapy/autoapi/owlapy/class\_expression/restriction/index.html\#owlapy.class\_expression.restriction.\\ OWLObjectMinCardinality$ 

<sup>12</sup> https://dice-group.github.io/owlapy/autoapi/owlapy/class\_expression/nary\_boolean\_expression/index.html#owlapy.class\_expression.nary\_boolean\_expression.OWLObjectIntersectionOf

<sup>13</sup> https://github.com/dice-group/owlapy/tree/develop/examples

# 3 Ontologies

To get started with Structured Machine Learning, the first thing required is an Ontology<sup>14</sup> with Named Individuals<sup>15</sup>. In this guide we show the basics of working with ontologies in Owlapy. We will use the *father* ontology for the following examples.

# 3.1 Loading an Ontology

To load an ontology as well as to manage it, you will need an *OWLOntologyManager* An ontology can be loaded using the following Python code:

```
from owlapy.iri import IRI
from owlapy.owl_ontology_manager import OntologyManager

manager = OntologyManager()
onto = manager.load_ontology(IRI.create("file://KGs/Family/father.owl"))
```

First, we import the IRI class and a suitable OWLOntologyManager. To load a file from our computer, we have to reference it with an *IRI*. Secondly, we need the Ontology Manager. Owlapy contains one such manager: The *Ontology-Manager*.

Now, we can already inspect the contents of the ontology. For example, to list all individuals:

```
for ind in onto.individuals_in_signature():
    print(ind)
```

You can get the object properties in the signature:

```
onto.object_properties_in_signature()
```

For more methods, see the abstract class OWLOntology or the concrete implementation Ontology.

# 3.2 Modifying an Ontology

Axioms in ontology serve as the basis for defining the vocabulary of a domain and for making statements about the relationships between individuals and concepts in that domain. They provide a formal and precise way to represent knowledge and allow for automated reasoning and inference. Axioms can be **added**, **modified**, or **removed** from an ontology, allowing the ontology to evolve and adapt as new knowledge is gained.

In owlapy we also have different axioms represented by different classes. You can check all the axioms classes *here*. Some frequently used axioms are:

- OWLDeclarationAxiom
- OWLObjectPropertyAssertionAxiom
- OWLDataPropertyAssertionAxiom
- OWLClassAssertionAxiom
- OWLSubClassOfAxiom
- OWLEquivalentClassesAxiom

<sup>14</sup> https://www.w3.org/TR/owl2-overview/

<sup>15</sup> https://www.w3.org/TR/owl-syntax/#Named\_Individuals

#### Add a new Class

Let's suppose you want to add a new class in our example ontology KGs/Family/father.owl It can be done as follows:

```
from owlapy.class_expression import OWLClass
from owlapy.owl_axiom import OWLDeclarationAxiom

iri = IRI('http://example.com/father#', 'child')
child_class = OWLClass(iri)
child_class_declaration_axiom = OWLDeclarationAxiom(child_class)

manager.add_axiom(onto, child_class_declaration_axiom)
```

In this example, we added the class 'child' to the *father.owl* ontology. Firstly we create an instance of *OWLClass* to represent the concept of 'child' by using an *IRI*. On the other side, an instance of IRI is created by passing two arguments which are the namespace of the ontology and the remainder 'child'. To declare this new class we need an axiom of type OWLDeclarationAxiom. We simply pass the child\_class to create an instance of this axiom. The final step is to add this axiom to the ontology using the *OWLOntologyManager*. We use the add\_axiom method of the manager to add into the ontology onto the axiom child\_class\_declaration\_axiom.

# Add a new Object Property / Data Property

The idea is the same as adding a new class. Instead of OWLClass, for object properties, you can use the class *OWLObjectProperty* and for data properties you can use the class *OWLDataProperty*.

```
from owlapy.owl_property import OWLObjectProperty, OWLDataProperty

# adding the object property 'hasParent'
hasParent_op = OWLObjectProperty(IRI('http://example.com/father#', 'hasParent'))
hasParent_op_declaration_axiom = OWLDeclarationAxiom(hasParent_op)
manager.add_axiom(onto, hasParent_op_declaration_axiom)

# adding the data property 'hasAge'
hasAge_dp = OWLDataProperty(IRI('http://example.com/father#', 'hasAge'))
hasAge_dp_declaration_axiom = OWLDeclarationAxiom(hasAge_dp)
manager.add_axiom(onto, hasAge_dp_declaration_axiom)
```

See the *owlapy* for more OWL entities that you can add as a declaration axiom.

#### Add an Assertion Axiom

To assign a class to a specific individual use the following code:

```
from owlapy.owl_axiom import OWLClassAssertionAxiom
individuals = list(onto.individuals_in_signature())
heinz = individuals[1] # get the 2nd individual in the list which is 'heinz'
class_assertion_axiom = OWLClassAssertionAxiom(heinz, child_class)
manager.add_axiom(onto, class_assertion_axiom)
```

We have used the previous method individuals\_in\_signature () to get all the individuals and converted them to a list, so we can access them by using indexes. In this example, we want to assert a class axiom for the individual heinz.

We have used the class OWLClassAssertionAxiom where the first argument is the 'individual' heinz and the second argument is the 'class\_expression'. As the class expression, we used the previously defined class child\_Class. Finally, add the axiom by using add\_axiom method of the *OWLOntologyManager*.

Let's show one more example using a OWLDataPropertyAssertionAxiom to assign the age of 17 to heinz.

```
from owlapy.owl_literal import OWLLiteral
from owlapy.owl_axiom import OWLDataPropertyAssertionAxiom

literal_17 = OWLLiteral(17)
dp_assertion_axiom = OWLDataPropertyAssertionAxiom(heinz, hasAge_dp, literal_17)

manager.add_axiom(onto, dp_assertion_axiom)
```

OWLLiteral is a class that represents the literal values in Owlapy. We have stored the integer literal value of '18' in the variable literal\_17. Then we construct the OWLDataPropertyAssertionAxiom by passing as the first argument, the individual heinz, as the second argument the data property hasAge\_dp, and the third argument the literal\_17. Finally, add it to the ontology by using add\_axiom method.

Check the *owlapy* to see all the OWL assertion axioms that you can use.

#### Remove an Axiom

To remove an axiom you can use the remove\_axiom method of the ontology manager as follows:

```
manager.remove_axiom(onto,dp_assertion_axiom)
```

The first argument is the ontology you want to remove the axiom from and the second argument is the axiom you want to remove.

# 3.3 Save an Ontology

If you modified an ontology, you may want to save it as a new file. To do this you can use the <code>save\_ontology</code> method of the <code>OWLOntologyManager</code>. It requires two arguments, the first is the ontology you want to save and The second is the IRI of the new ontology.

```
manager.save_ontology(onto, IRI.create('file:/' + 'test' + '.owl'))
```

The above line of code will save the ontology onto in the file *test.owl* which will be created in the same directory as the file you are running this code.

# 3.4 Worlds

Owlready2 stores every triple in a 'World' object, and it can handle several Worlds in parallel. Owlready2 uses an optimized quadstore to store the world. Each world object is stored in a separate quadstore and by default the quadstore is stored in memory, but it can also be stored in an SQLite3 file. The method <code>save\_world()</code> of the ontology manager does the latter. When an <code>OWLOntologyManager</code> object is created, a new world is also created as an attribute of the manager. By calling the method <code>load\_ontology(iri)</code> the ontology is loaded to this world.

It possible to create several isolated "worlds", sometimes called "universe of speech". This makes it possible in particular to load the same ontology several times, independently, that is to say, without the modifications made on one copy affecting the other copy. Sometimes the need to *isolate an ontology* arise. What that means is that you can have multiple reference of the same ontology in different worlds.

It is important that an ontology is associated with a reasoner which is used to inferring knowledge from the ontology, i.e. to perform ontology reasoning. In the next guide we will see how to use a reasoner in Owlapy.

### 4 Reasoners

To validate facts about statements in the ontology, the help of a reasoner component is required.

For this guide we will also consider the 'father' ontology that we slightly described here:

```
from owlapy.owl_ontology_manager import OntologyManager

manager = OntologyManager()
onto = manager.load_ontology(IRI.create("KGs/Family/father.owl"))
```

In our Owlapy library, we provide several **reasoners** to choose from. Currently, there are the following reasoners available:

#### OntologyReasoner

Or differently Structural Reasoner, is the base reasoner in Owlapy. The functionalities of this reasoner are limited. It does not provide full reasoning in *ALCH*. Furthermore, it has no support for instances of complex class expressions, which is covered by the other reasoners (SyncReasoner and FIC). We recommend to use the other reasoners for any heavy reasoning tasks.

#### **Initialization:**

```
from owlapy.owl_reasoner import OntologyReasoner
structural_reasoner = OntologyReasoner(onto)
```

The structural reasoner requires an ontology (*OWLOntology*). The second argument is isolate argument which isolates the world (therefore the ontology) where the reasoner is performing the reasoning. More on that on Reasoning Details.

#### • SyncReasoner

Can perform full reasoning in *ALCH* due to the use of HermiT/Pellet and provides support for complex class expression instances (when using the method instances). SyncReasoner is more useful when your main goal is reasoning over the ontology.

#### **Initialization:**

```
from owlapy.owl_reasoner import SyncReasoner, BaseReasoner

sync_reasoner = SyncReasoner(onto, BaseReasoner.HERMIT, infer_property_values = True)
```

Sync Reasoner requires an ontology and a base reasoner of type <code>BaseReasoner</code> which is just an enumeration with two possible values: <code>BaseReasoner.HERMIT</code> and <code>BaseReasoner.PELLET</code>. You can set the <code>infer\_property\_values</code> argument to <code>True</code> if you want the reasoner to infer property values. <code>infer\_data\_property\_values</code> is an additional argument when the base reasoner is set to <code>BaseReasoner.PELLET</code>. The argument <code>isolated</code> is inherited from the base class

#### • FastInstanceCheckerReasoner (FIC)

FIC also provides support for complex class expression but the rest of the methods are the same as in the base reasoner. It has a cache storing system that allows for faster execution of some reasoning functionalities. Due to this feature, FIC is more appropriate to be used in concept learning.

#### **Initialization:**

Besides the ontology, FIC requires a base reasoner to delegate any reasoning tasks not covered by it. This base reasoner can be any other reasoner in Owlapy. property\_cache specifies whether to cache property values. This requires more memory, but it speeds up the reasoning processes. If negation\_default argument is set to True the missing facts in the ontology means false. The argument sub\_properties is another boolean argument to specify whether you want to take sub properties in consideration for instances () method.

# 4.1 Usage of the Reasoner

All the reasoners available in the Owlapy library inherit from the class: *OWLReasonerEx*. This class provides some extra convenient methods compared to its base class *OWLReasoner*, which is an abstract class. Further on, in this guide, we use *SyncReasoner*. to show the capabilities of a reasoner in Owlapy.

To give examples we consider the *father* dataset. If you are not already familiar with this small dataset, you can find an overview of it *here*.

# 4.2 Class Reasoning

Using an *OWLOntology* you can list all the classes in the signature, but a reasoner can give you more than that. You can get the subclasses, superclasses or the equivalent classes of a class in the ontology:

```
from owlapy.class_expression import OWLClass
from owlapy.iri import IRI

namespace = "http://example.com/father#"
male = OWLClass(IRI(namespace, "male"))

male_super_classes = sync_reasoner.super_classes(male)
male_sub_classes = sync_reasoner.sub_classes(male)
male_equivalent_classes = sync_reasoner.equivalent_classes(male)
```

We define the *male* class by creating an *OWLClass* object. The methods <code>super\_classes</code> and <code>sub\_classes</code> have 2 more boolean arguments: <code>direct</code> and <code>only\_named</code>. If <code>direct=True</code> then only the direct classes in the hierarchy will be returned, else it will return every class in the hierarchy depending on the method(sub\_classes or super\_classes). By default, its value is *False*. The next argument <code>only\_named</code> specifies whether you want to show only named classes or complex classes as well. By default, its value is *True* which means that it will return only the named classes.

**NOTE**: The extra arguments direct and only\_named are also used in other methods that reason upon the class, object property, or data property hierarchy.

You can get all the types of a certain individual using types method:

```
anna = list(onto.individuals_in_signature()).pop()
anna_types = sync_reasoner.types(anna)
```

We retrieve *anna* as the first individual on the list of individuals of the 'Father' ontology. The type method only returns named classes.

# 4.3 Object Properties and Data Properties Reasoning

Owlapy reasoners offers some convenient methods for working with object properties and data properties. Below we show some of them, but you can always check all the methods in the *SyncReasoner* class documentation.

You can get all the object properties that an individual has by using the following method:

```
anna = individuals[0]
object_properties = sync_reasoner.ind_object_properties(anna)
```

In this example, object\_properties contains all the object properties that *anna* has, which in our case would only be *hasChild*. Now we can get the individuals of this object property for *anna*.

```
for op in object_properties:
   object_properties_values = sync_reasoner.object_property_values(anna, op)
   for individual in object_properties_values:
        print(individual)
```

In this example we iterated over the <code>object\_properties</code>, assuming that there are more than 1, and we use the reasoner to get the values for each object property op of the individual anna. The values are individuals which we store in the variable <code>object\_properties\_values</code> and are printed in the end. The method <code>object\_property\_values</code> requires as the first argument, an <code>OWLNamedIndividual</code> that is the subject of the object property values and the second argument an <code>OWLObjectProperty</code> whose values are to be retrieved for the specified individual.

**NOTE:** You can as well get all the data properties of an individual in the same way by using ind\_data\_properties instead of ind\_object\_properties and data\_property\_values instead of object\_property\_values. Keep in mind that data\_property\_values returns literal values (type of *OWLLiteral*).

In the same way as with classes, you can also get the sub object properties or equivalent object properties.

```
from owlapy.owl_property import OWLObjectProperty
hasChild = OWLObjectProperty(IRI(namespace, "hasChild"))
equivalent_to_hasChild = sync_reasoner.equivalent_object_properties(hasChild)
hasChild_sub_properties = sync_reasoner.sub_object_properties(hasChild)
```

In case you want to get the domains and ranges of an object property use the following:

```
hasChild_domains = sync_reasoner.object_property_domains(hasChild)
hasChild_ranges = sync_reasoner.object_property_ranges(hasChild)
```

**NOTE:** Again, you can do the same for data properties but instead of the word 'object' in the method name you should use 'data'.

#### 4.4 Find Instances

The method instances is a very convenient method. It takes only 1 argument that is basically a class expression and returns all the individuals belonging to that class expression. In Owlapy we have implemented a Python class for each type of class expression. The argument is of type *OWLClassExpression*.

Let us now show a simple example by finding the instances of the class *male* and printing them:

```
male_individuals = sync_reasoner.instances(male)
for ind in male_individuals:
    print(ind)
```

In this guide we covered the main functionalities of the reasoners in Owlapy. More details are provided in the next guide.

# 5 Reasoning Details

In the previous guide we explained how to *use reasoners* in Owlapy. Here we cover a detailed explanation of the Owlapy reasoners, particularly *SyncReasoner*. Before we continue to talk about its *capabilities* we have to explain briefly the term *sync\_reasoner*.

# 5.1 Sync Reasoner

sync\_reasoner is a definition used in owlready2 to run HermiT<sup>16</sup> or Pellet<sup>17</sup> and automatically apply the facts deduced to the quadstore. In simple terms, by running HermiT or Pellet, one can infer more knowledge from the ontology (the specification are not mentioned here). We make use of this functionality in Owlapy, and it is represented by SyncReasoner. We explained the concept of "Worlds" in Working with Ontologies. Having that in mind you need to know that sync\_reasoner is applied to the World object. After this particular reasoner is instantiated, because the facts are applied to the quadstore, changes made in the ontology by using the ontology manager will not be reflected to the ontology. The reasoner will use the state of the ontology at the moment it is instantiated.

There are 2 boolean parameters for sync\_reasoner that you can specify when creating an instance of *SyncReasoner*. The first one infer\_property\_values tells HermiT or Pellet whether to infer (or not) property values. The same idea but for data properties is specified by the parameter infer\_data\_property\_values which is only relevant to Pellet.

Note: HermiT and Pellet are Java programs, so you will need to install a Java virtual machine to use them. If you don't have Java, you may install it from www.java.com (for Windows and macOS) or from the packages of your Linux distribution (the packages are often named "jre" or "jdk" for Java Runtime Environment and Java Development Kit).

#### 5.2 Isolated World

In Working with Ontologies we mentioned that we can have multiple reference of in different worlds, which we can use to isolate an ontology to a specific World. For simplicity the terms "isolated world" and "isolated ontology" can be used interchangeably in this guide. The isolation comes in handy when we use multiple reasoners in the same script. If we create an instance of *SyncReasoner* it will apply sync\_reasoner in the world object of the ontology and this will affect also the other reasoner/s which is/are using the same world. To overcome this issue you can set the argument isolate=True when initializing a reasoner. *FastInstanceCheckerReasoner* (FIC) does not have this argument because it uses a base reasoner to delegate most of its methods. Therefore, if the base reasoner has isolate=True then FIC will also operate in the isolated world of it's base reasoner.

<sup>16</sup> http://www.hermit-reasoner.com/

<sup>17</sup> https://github.com/stardog-union/pellet

#### Modifying an isolated ontology

When a reasoner is operating in an isolated ontology, every axiom added to the original ontology before or after the initialization, will not be reflected to the isolated ontology. To update the isolated ontology and add or remove any axiom, you can use update\_isolated\_ontology (axioms\_to\_add, axioms\_to\_remove). This method accepts a list of axioms for every argument (i.e. the axioms that you want to add and the axioms that you want to remove).

# 5.3 Capabilities

*SyncReasoner* provides full reasoning in *ALCH*. We have adapted and build upon owlready2<sup>18</sup> reasoner to provide our own implementation in python. Below we give more details about each functionality of our reasoner:

### Sub and Super Classes

You can retrieve sub (super) classes of a given class expression. Depending on your preferences you can retrieve the whole chain of sub (super) classes or only the direct sub (super) classes (direct argument). It is also possible to get anonymous classes in addition to named classes (only\_named argument). Class equivalence entails subsumption of classes to each other.

### Equivalent Classes

You are able to get the equivalent classes of a given class expression. It can be decided whether only named classes should be returned or anonymous classes as well. If two classes are subclasses of each other they are considered equivalent.

# Disjoint Classes

Every class that is explicitly defined as disjoint with another class will be returned. In addition, every subclass and equivalent class of the disjoint classes will be returned. If a target class does not have explicitly-defined disjoint classes the search is transferred to the superclasses of that target class.

#### Equivalent Properties

You are able to get equivalent properties of a given object or data property. If two properties are sub-properties of each other, they are considered equivalent.

#### Sub and Super Properties

Our reasoner has support also for sub and super properties of a given property. You can set the direct argument like in sub (super) classes. Properties equivalence entails subsumption of properties to each other.

<sup>18</sup> https://owlready2.readthedocs.io/en/latest/

#### Disjoint Properties

Similarly to disjoint classes, you can get the disjoint properties of a property. Same rules apply.

#### Property values

Given an individual(instance) and an object property you can get all the object values. Similarly, given an individual and a data property you can get all the literal values. You can set whether you want only the direct values or all of them.

#### Property domain and range

Easily retrieval available for domain and range for object properties and domain for data properties.

#### Instances

This functionality enables you to get instances for a given named(atomic) class or complex class expression. For the moment direct instances of complex class expressions is not possible.

#### Types

This functionality enables you to get the types of a given instance. It returns only named(atomic) classes. You can set the direct attribute.

#### Same and Different Individuals

Given an individual you can get the individuals that are explicitly defined as same or different to that individual.

# 5.4 Concrete Example

You can find the associated code<sup>19</sup> for the following examples inside examples/example\_reasoner (note that the naming of the classes/relations/individuals may change from the table below). We constructed an ontology for testing purposes. On the table we show for each **method** of the reasoner *SyncReasoner* the results depending on a given **TBox** and **Abox**. The level of complexity of the TBox-es is low compared to real world scenarios, but it's just to show the capabilities of the reasoner.

**Note:** not every method of the reasoner is used in this example. You can check all the methods at the *API documentation*.

Method	TBox	ABox	Returns(T = Thing)
Equivalent_classes(A)	$A \equiv B$	-	[B]
Equivalent_classes(B)	$A \equiv B$	-	[A]
Instances(A)	$A \equiv B$	A(a),B(b)	[a,b]
Instances(B)	$A \equiv B$	A(a),B(b)	[a,b]
Types(a)	$A \equiv B$	A(a),B(b)	[T, A,B]
Types(b)	$A \equiv B$	A(a),B(b)	[T, A,B]

continues on next page

<sup>&</sup>lt;sup>19</sup> https://github.com/dice-group/owlapy/blob/develop/examples/ontology\_reasoning.py

Table 1 - continued from previous page

Method	TBox	ABox	Returns(T = Thing)
Sub_classes(A)	A≡B	-	[B]
Sub_classes(B)	$A \equiv B$	=	[A]
Super_classes(A)	$A \equiv B$	=	[B,T]
Super_classes(B)	$A \equiv B$	-	[A,T]
Equivalent_object_properties(r1)	$r1 \equiv r2$	-	[r2]
Equivalent_object_properties(r2)	$r1 \equiv r2$	-	[r1]
sub_object_properties(r1)	r1 ≡ r2	-	[r2]
sub_object_properties(r2)	r1 ≡ r2	=	[r1]
object_property_values(a, r1, direct=False)	r1 ≡ r2	r1(a,b) r2(a,c)	[c]
object_property_values(a, r2, direct=False)	r1 ≡ r2	r1(a,b) r2(a,c)	[c]
Sub_classes(B)	$A \sqsubseteq B$	-	[A]
Super_classes(A)	$A \sqsubseteq B$	-	[T, B]
Types(a)	$A \sqsubseteq B$	A(a),B(b)	[A,B,T]
Types(b)	$A \sqsubseteq B$	A(a),B(b)	[B,T]
Instances(A)	$A \sqsubseteq B$	A(a),B(b)	[a]
Instances(B)	$A \sqsubseteq B$	A(a),B(b)	[a,b]
sub_object_properties(r1)	r2 ⊑ r1	-	[r2]
object_property_values(a, r2)	r2 ⊑ r1	r2(a,b)	[b]
object_property_values(a, r1, direct=False)	r2 ⊑ r1	r2(a,b)	[b]
Sub_classes(r1.T)	r2 ⊑ r1	-	[r2.T]
Super_classes(D, only_named=False)	D ⊑ ∃r.E	-	[T, ∃r.E]
Sub_classes(∃r.E)	D ⊑ ∃r.E	-	[D]
Instances(D)	D ⊑ ∃r.E	D(d) r(i,e) E(e)	[d]
Instances(∃r.E)	D ⊑ ∃r.E	D(d) r(i,e) E(e)	[i, d]
types(d)	D ⊑ ∃r.E	D(d) r(i,e) E(e)	[D,T]
types(i)	D ⊑ ∃r.E	D(d) r(i,e) E(e)	[T]
object_property_values(i, r)	D ⊑ ∃r.E	r(i,e) E(e)	[e]
Sub_classes(D, only_named=False)	∃r.E ⊑ D	-	[ ∃r.E]
Super_classes(∃r.E)	∃r.E ⊑ D	=	[D, T]
Instances(D)	∃r.E ⊑ D	D(d) r(i,e) E(e)	[i, d]
Instances(∃r.E)	∃r.E ⊑ D	D(d) r(i,e) E(e)	[i]
types(d)	∃r.E ⊑ D	D(d) r(i,e) E(e)	[D, T]
types(i)	∃r.E ⊑ D	D(d) r(i,e) E(e)	[D, T]
object_property_values(i, r)	∃r.E ⊑ D	r(i,e) E(e)	[e]
Sub_classes(A)	$A \sqsubseteq B, B \sqsubseteq A$	-	[A,B]
Sub_classes(B)	$A \sqsubseteq B, B \sqsubseteq A$	-	[A,B]
Super_classes(A)	$A \sqsubseteq B, B \sqsubseteq A$	=	[T, B]
Super_classes(B)	$A \sqsubseteq B, B \sqsubseteq A$	-	[T, A]
Types(a)	$A \sqsubseteq B, B \sqsubseteq A$	A(a),B(b)	[A,B,T]
Types(b)	$A \sqsubseteq B, B \sqsubseteq A$	A(a),B(b)	[A,B,T]
Instances(A)	$A \sqsubseteq B, B \sqsubseteq A$	A(a),B(b)	[a,b]
Instances(B)	$A \sqsubseteq B, B \sqsubseteq A$	A(a),B(b)	[a,b]
Equivalent_classes(A,only_named=False)	$A \sqsubseteq B, B \sqsubseteq A$	-	[B]
Equivalent_classes(B,only_named=False)	$A \sqsubseteq B, B \sqsubseteq A$	-	[A]
sub_object_properties(r1)	r2⊑ r1, r1⊑ r2	-	[r2,r1]
sub_object_properties(r2)	r2⊑ r1, r1⊑ r2	-	[r1,r2]
Equivalent_object_properties(r1)	r2⊑ r1, r1⊑ r2	-	[r2]
Equivalent_object_properties(r2)	r2⊑ r1, r1⊑ r2	-	[r1]
object_property_values(a, r1, direct=False)	$r2 \sqsubseteq r1, r1 \sqsubseteq r2$ $r2 \sqsubseteq r1, r1 \sqsubseteq r2$	r1(a,b) r2(a,c)	[b,c]
object_property_values(a, r2, direct=False)	$r2 \sqsubseteq r1, r1 \sqsubseteq r2$ $r2 \sqsubseteq r1, r1 \sqsubseteq r2$	r1(a,b) r2(a,c)	[b,c]

continues on next page

Table 1 - continued from previous page

Method	TBox	ABox	Returns(T = Thing)
$Sub\_classes(J \sqcap K)$	I⊑J⊓K	-	[I]
Super_classes(I, only_named=False)	I⊑J⊓K	-	$[J \sqcap K, J, K, T]$
Instances( $J \sqcap K$ )	I⊑J⊓K	I(c)	[c]
types(c)	I⊑J⊓K	I(c)	[J, K, I, T]
Super_classes( $J \sqcap K$ )	J⊓K⊑I	-	[I, T]
Sub_classes(I, only_named=False)	J⊓K⊑I	-	[J ⊓ K]
Instances(I)	J⊓K⊑I	J(s),K(s)	[s]
Instances( $J \sqcap K$ )	J⊓K⊑I	J(s),K(s)	[s]
types(s)	J⊓K⊑I	J(s),K(s)	[J, K, I, T]
Sub_classes( $\exists r.E \sqcap B$ )	D ⊑ ∃r.E ⊓ B	-	[D]
Super_classes(D, only_named=False)	D <u>⊆</u> ∃r.E ⊓ B	-	$[T, \exists r.E \sqcap B, B]$
Instances( $\exists r.E \sqcap B$ )	D <u>⊆</u> ∃r.E ⊓ B	D(d) r(b,f) E(f) B(b)	[d,b]
Sub_classes(H, only_named= False)	$F \equiv \exists r.G, F \sqsubseteq H$	-	[F, ∃r.G]
Super_classes(F)	$F \equiv \exists r.G, F \sqsubseteq H$	-	[H,∃r.G,T]
Super_classes(\(\frac{\partial}{3}\)r.G)	$F \equiv \exists r.G, F \sqsubseteq H$	-	[F,H,T]
Equivalent_classes(F, only_named=False)	$F \equiv \exists r.G, F \sqsubseteq H$	-	[∃r.G]
Equivalent_classes(∃r.G)	$F \equiv \exists r.G, F \sqsubseteq H$	-	[F]
Instances( $\exists r.G$ )	$F \equiv \exists r.G, F \sqsubseteq H$	r(i,g) G(g)	[i]
Instances(F)	$F \equiv \exists r.G, F \sqsubseteq H$	r(i,g) G(g)	[i]
Instances(H)	$F \equiv \exists r.G, F \sqsubseteq H$	r(i,g) G(g)	[i]
types(i)	$F \equiv \exists r.G, F \sqsubseteq H$	r(i,g) G(g)	[H,F,T]
Sub_classes(C, only_named=False)	$A \sqcap B \equiv R, R \sqsubseteq C$	-	$[R, A \sqcap B]$
Super_classes(A $\sqcap$ B)	$A \sqcap B \equiv R, R \sqsubseteq C$	-	[R, C,A,B,T]
Equivalent_classes(R,only_named=False)	$A \sqcap B \equiv R, R \sqsubseteq C$	-	[A \cap B]
Equivalent_classes(A $\sqcap$ B)	$A \sqcap B \equiv R, R \sqsubseteq C$	-	[R]
Instances(A $\sqcap$ B)	$A \sqcap B \equiv R, R \sqsubseteq C$	R(e) A(a) B(a)	[e,a]
Instances(R)	$A \sqcap B \equiv R, R \sqsubseteq C$	R(e) A(a) B(a)	[a, e]
Instances(C)	$A \sqcap B \equiv R, R \sqsubseteq C$	R(e) A(a) B(a)	[a, e]
Types(a)	$A \sqcap B \equiv R, R \sqsubseteq C$	R(e) A(a) B(a)	[A,B,R,C,T]
Types(e)	$A \sqcap B \equiv R, R \sqsubseteq C$	R(e) A(a) B(a)	[A,B,R,C,T]
Sub_classes(D, only_named=False)	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	-	$[E, \exists r.P \sqcap C]$
Super_classes( $\exists r.P \sqcap C$ )	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	-	[E, D, T]
Equivalent_classes( $\exists r.P \sqcap C$ )	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	-	[E]
Equivalent_classes(E,only_named=False)	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	-	[∃r.P ⊓ C]
Instances( $\exists r.P \sqcap C$ )	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	r(x,y) C(x) P(y)	[x]
Instances(E)	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	r(x,y) C(x) P(y)	[x]
Instances(D)	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	r(x,y) C(x) P(y)	[X]
Types(x)	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	r(x,y) C(x) P(y)	[C]
disjoint_classes(A)	$A \sqcup B$	-	[B]
disjoint_classes(B)	A⊔B	-	[A]
disjoint_classes(A)	$A \sqcup B$ , $B \equiv C$	-	[B, C]
disjoint_classes(B)	$A \sqcup B, B \equiv C$	-	[A]
disjoint_classes(C)	$A \sqcup B, B \equiv C$	-	[A]
object_property_domains(r)	Domain(r) = A	_	[A,T]
object_property_domains(r)	$Domain(r) = AA \equiv B$	-	[A,T]
object_property_domains(r)	Domain(r1) = Ar2 $\sqsubseteq$ r1	-	[A,T]
cojett_propertj_domains(12)	2 3mm(11) - 1112 = 11		[- <del>- '</del> , <del>-</del> ]

# 6 Owlapi Adaptor

As mentioned earlier, owlapy is loosely based in owlapi<sup>20</sup>, a library for ontology modification in java.

We have created *OWLAPIAdaptor*, an adaptor class that facilitates the conversion of owl class expressions from owlapy to owlapi and vice-versa. This adaptor is still considered experimental, and it's in the initial phase of development.

We are able to use owlapi via Jpype<sup>21</sup>, a python module that provides access to Java via python. To start executing Java code via jpype, one needs to start the java virtual machine (JVM). This is automatically done when initializing a OWLAPIAdaptor object.

#### 6.1 Initialization

To use the adaptor you have to start the JVM via jpype, which is done automatically when you create an *OWLAPIAdaptor* object. After you are finished you can stop the JVM by either using <code>jpype.shutdownJVM()</code> or the static method from the adaptor <code>stopJVM()</code>. This will free the resources used by JPype and the java packages.

```
from owlapy.owlapi_adaptor import OWLAPIAdaptor

adaptor = OWLAPIAdaptor("KGs/Family/father.owl")
# Use the adaptor
print(f"Is the ontology consistent? {adaptor.has_consistent_ontology()}")

# Stop the JVM
adaptor.stopJVM()
```

In the above code snipped, we created an adaptor for the father ontology by passing the local path of that ontology. Then we print whether the ontology is consistent or not.

#### 6.2 Notes

An important note is that when initialising the adaptor you are basically starting a JVM in the background, and therefore you are able to import and use java classes as you would do in python. That means that you can play around with owlapi code in python as long as your JVM is started. Isn't that awesome!

OWLAPIAdaptor uses HermiT reasoner by default. You can choose between: "HermiT", "Pellet", "JFact" and "Openllet".

owlapi version: 5.1.9

# 6.3 Examples

You can check a usage example in the examples<sup>22</sup> folder.

Test cases<sup>23</sup> for the adaptor can also serve as an example, so you can check that out as well.

<sup>&</sup>lt;sup>20</sup> https://github.com/owlcs/owlapi

<sup>&</sup>lt;sup>21</sup> https://jpype.readthedocs.io/en/latest/

<sup>&</sup>lt;sup>22</sup> https://github.com/dice-group/owlapy/tree/develop/examples

<sup>&</sup>lt;sup>23</sup> https://github.com/dice-group/owlapy/tree/develop/tests

# 7 owlapy

# 7.1 Subpackages

#### owlapy.class\_expression

OWL Class Expressions https://www.w3.org/TR/owl2-syntax/#Class\_Expressions ClassExpression :=

owl\_class.py: Class nary\_boolean\_expression.py: ObjectIntersectionOf, ObjectUnionOf class\_expression.py: ObjectComplementOf

restriction.py: ObjectOneOf, ObjectSomeValuesFrom, ObjectAllValuesFrom, ObjectHas-Value,ObjectHasSelf, ObjectMinCardinality, ObjectMaxCardinality, ObjectExactCardinality, Data-SomeValuesFrom, DataAllValuesFrom, DataHasValue, DataMinCardinality, DataMaxCardinality, DataExactCardinality

#### **Submodules**

### owlapy.class\_expression.class\_expression

**OWL Base Classes Expressions** 

#### **Classes**

OWLClassExpression	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
OWLAnonymousClassExpression	A Class Expression which is not a named Class.
OWLBooleanClassExpression	Represent an anonymous boolean class expression.
OWLObjectComplementOf	Represents an ObjectComplementOf class expression in the OWL 2 Specification.

## **Module Contents**

 $\textbf{class} \ \, \textbf{owlapy.class\_expression.class\_expression.} \\ \textbf{OWLClassExpression}$ 

Bases: owlapy.owl\_data\_ranges.OWLPropertyRange

OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the structural specification of OWL 2, class expressions are represented by ClassExpression. (https://www.w3.org/TR/owl2-syntax/#Class\_Expressions)

 $\verb"abstract is_owl_thing"() \to bool$ 

Determines if this expression is the built in class owl: Thing. This method does not determine if the class is equivalent to owl: Thing.

#### **Returns**

Thing.

#### Return type

True if this expression is owl

# $\textbf{abstract is\_owl\_nothing()} \rightarrow bool$

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

### abstract get\_object\_complement\_of() → OWLObjectComplementOf

Gets the object complement of this class expression.

#### Returns

A class expression that is the complement of this class expression.

```
abstract get_nnf() → OWLClassExpression
```

Gets the negation normal form of the complement of this expression.

#### **Returns**

A expression that represents the NNF of the complement of this expression.

 $\textbf{class} \ \, \textbf{owlapy.class\_expression.Class\_expression.OWLAnonymousClassExpression}$ 

Bases: OWLClassExpression

A Class Expression which is not a named Class.

```
is\_owl\_nothing() \rightarrow bool
```

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

```
is\_owl\_thing() \rightarrow bool
```

Determines if this expression is the built in class owl: Thing. This method does not determine if the class is equivalent to owl: Thing.

#### Returns

Thing.

#### Return type

True if this expression is owl

```
get_object_complement_of() → OWLObjectComplementOf
```

Gets the object complement of this class expression.

#### Returns

A class expression that is the complement of this class expression.

```
get nnf() \rightarrow OWLClassExpression
```

Gets the negation normal form of the complement of this expression.

#### Returns

A expression that represents the NNF of the complement of this expression.

class owlapy.class\_expression.class\_expression.OWLBooleanClassExpression

Bases: OWLAnonymousClassExpression

Represent an anonymous boolean class expression.

Bases: OWLBooleanClassExpression, owlapy.meta\_classes.

Represents an ObjectComplementOf class expression in the OWL 2 Specification.

```
__slots__ = '_operand'

type_index: Final = 3003

get_operand() \( \to \) OWLClassExpression
```

#### Returns

The wrapped expression.

 $operands() \rightarrow Iterable[OWLClassExpression]$ 

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

#### **Returns**

The operands.

```
__repr__()
Return repr(self).
__eq__(other)
Return self==value.
__hash__()
Return hash(self).
```

### owlapy.class expression.nary boolean expression

OWL nary boolean expressions

#### **Classes**

OWLNaryBooleanClassExpression	OWLNaryBooleanClassExpression.
OWLObjectUnionOf	A union class expression ObjectUnionOf( CE1 CEn ) contains all individuals that are instances
OWLObjectIntersectionOf	An intersection class expression ObjectIntersectionOf( CE1 CEn ) contains all individuals that are instances

#### **Module Contents**

**class** owlapy.class\_expression.nary\_boolean\_expression.

# ${\tt OWLNaryBooleanClassExpression} \ ($

 $operands: \ Iterable[owlapy.class\_expression.class\_expression.OWLClassExpression])$ 

 $\begin{tabular}{ll} \textbf{Bases:} & owlapy.class\_expression.class\_expression.OWLBooleanClassExpression, owlapy.meta\_classes.HasOperands[owlapy.class\_expression.class\_expression.OWLClassExpression] \end{tabular}$ 

OWLNaryBooleanClassExpression.

```
Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.
              Returns
                  The operands.
     __repr__()
          Return repr(self).
     __eq__(other)
          Return self==value.
     __hash__()
          Return hash(self).
class owlapy.class_expression.nary_boolean_expression.OWLObjectUnionOf(
           operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
     Bases: OWLNaryBooleanClassExpression
     A union class expression ObjectUnionOf( CE1 ... CEn ) contains all individuals that are instances of at least one
     class expression CEi for 1 \le i \le n. (https://www.w3.org/TR/owl2-syntax/#Union_of_Class_Expressions)
     __slots__ = '_operands'
     type_index: Final = 3002
class owlapy.class_expression.nary_boolean_expression.
           OWLObjectIntersectionOf(
           operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
     Bases: OWLNaryBooleanClassExpression
     An intersection class expression ObjectIntersectionOf( CE1 ... CEn ) contains all individuals that are instances of
     all class expressions CEi for 1 \le i \le n. (https://www.w3.org/TR/owl2-syntax/#Intersection_of_Class_Expressions)
     __slots__ = '_operands'
     type_index: Final = 3001
owlapy.class_expression.owl_class
OWL Class
Classes
```

**operands** () → Iterable[owlapy.class\_expression.class\_expression.OWLClassExpression]

sets of individuals.

OWLC1ass

An OWL 2 named Class. Classes can be understood as

#### **Module Contents**

```
class owlapy.class_expression.owl_class.OWLClass(iri: owlapy.iri.IRI | str)
     Bases: owlapy.class_expression.class_expression.OWLClassExpression, owlapy.
     owl_object.OWLEntity
     An OWL 2 named Class. Classes can be understood as sets of individuals. (https://www.w3.org/TR/owl2-syntax/
     #Classes)
     __slots__ = ('_iri', '_is_nothing', '_is_thing')
     type_index: Final = 1001
     property iri: owlapy.iri.IRI
          Gets the IRI of this object.
              Returns
                  The IRI of this object.
     property str
     Gets the string representation of this object
              Returns
                  The IRI as string
     property reminder: str
          The reminder of the IRI
     is\_owl\_thing() \rightarrow bool
          Determines if this expression is the built in class owl: Thing. This method does not determine if the class is
          equivalent to owl:Thing.
              Returns
                  Thing.
              Return type
                  True if this expression is owl
     is\_owl\_nothing() \rightarrow bool
          Determines if this expression is the built in class owl: Nothing. This method does not determine if the class
          is equivalent to owl:Nothing.
     get_object_complement_of()
                  → owlapy.class_expression.class_expression.OWLObjectComplementOf
          Gets the object complement of this class expression.
```

# Returns

A class expression that is the complement of this class expression.

```
get_nnf() \rightarrow OWLClass
```

Gets the negation normal form of the complement of this expression.

#### Returns

A expression that represents the NNF of the complement of this expression.

# owlapy.class\_expression.restriction

OWL Restrictions

# Attributes

Literals

# Classes

OWLRestriction	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
OWLHasValueRestriction	Represent a HasValue restriction in the OWL 2
OWLObjectRestriction	Represents an Object Property Restriction in the OWL 2 specification.
OWLQuantifiedRestriction	Represents a quantified restriction.
OWLCardinalityRestriction	Base interface for owl min and max cardinality restriction.
OWLQuantifiedObjectRestriction	Represents a quantified object restriction.
OWLObjectCardinalityRestriction	Represents Object Property Cardinality Restrictions in the OWL 2 specification.
OWLObjectMinCardinality	A minimum cardinality expression ObjectMinCardinality( n OPE CE ) consists of a nonnegative integer n, an object
OWLObjectMaxCardinality	A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an object
OWLObjectExactCardinality	An exact cardinality expression ObjectExactCardinality( n OPE CE) consists of a nonnegative integer n, an object
OWLObjectSomeValuesFrom	An existential class expression ObjectSomeValuesFrom(OPE CE) consists of an object property expression OPE and
OWLObjectAllValuesFrom	A universal class expression ObjectAllValuesFrom( OPE CE ) consists of an object property expression OPE and a
OWLObjectHasSelf	A self-restriction ObjectHasSelf( OPE ) consists of an object property expression OPE,
OWLObjectHasValue	A has-value class expression ObjectHasValue( OPE a ) consists of an object property expression OPE and an
OWLObjectOneOf	An enumeration of individuals ObjectOneOf( a1 an ) contains exactly the individuals ai with $1 \le i \le n$ .
OWLDataRestriction	Represents a Data Property Restriction.
OWLQuantifiedDataRestriction	Represents a quantified data restriction.
OWLDataCardinalityRestriction	Represents Data Property Cardinality Restrictions.
OWLDataMinCardinality	A minimum cardinality expression DataMinCardinality(n DPE DR) consists of a nonnegative integer n, a data
OWLDataMaxCardinality	A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an object
OWLDataExactCardinality	An exact cardinality expression ObjectExactCardinality( n OPE CE) consists of a nonnegative integer n, an
OWLDataSomeValuesFrom	An existential class expression DataSomeValuesFrom(DPE1 DPEn DR) consists of n data property expressions
OWLDataAllValuesFrom	A universal class expression DataAllValuesFrom( DPE1 DPEn DR ) consists of n data property expressions DPEi,
OWLDataHasValue	A has-value class expression DataHasValue(DPE lt) consists of a data property expression DPE and a literal lt,
OWLDataOneOf	An enumeration of literals DataOneOf( lt1 ltn ) contains exactly the explicitly specified literals lti with
OWLDatatypeRestriction	A datatype restriction DatatypeRestriction( DT F1 lt1 Fn ltn ) consists of a unary datatype DT and n pairs
OWLFacetRestriction	A facet restriction is used to restrict a particular datatype.

#### **Module Contents**

owlapy.class\_expression.restriction.Literals class owlapy.class\_expression.restriction.OWLRestriction Bases: owlapy.class\_expression.class\_expression.OWLAnonymousClassExpression Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification. \_\_slots\_\_ = () **abstract get property**() → *owlapy.owl property.OWLPropertyExpression* Returns Property being restricted. is data restriction()  $\rightarrow$  bool Determines if this is a data restriction. Returns True if this is a data restriction.  $is\_object\_restriction() \rightarrow bool$ Determines if this is an object restriction. Returns True if this is an object restriction.  ${\tt class}$  owlapy.class\_expression.restriction.OWLHasValueRestriction( $value: \_T$ ) Bases: Generic[\_T], OWLRestriction, owlapy.meta\_classes.HasFiller[\_T] Represent a HasValue restriction in the OWL 2 **Parameters**  $_{\mathbf{T}}$  – The value type. \_\_slots\_\_ = ()  $\underline{\phantom{a}}$ eq $\underline{\phantom{a}}$  (other) Return self==value. \_\_hash\_\_() Return hash(self).  $\texttt{get\_filler}\,() \, \to \, \_T$ Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a constant (data value). For quantified restriction this will be a class expression or a data range. Returns the value class owlapy.class\_expression.restriction.OWLObjectRestriction Bases: OWLRestriction Represents an Object Property Restriction in the OWL 2 specification. \_\_slots\_\_ = ()

```
is\_object\_restriction() \rightarrow bool
```

Determines if this is an object restriction.

#### Returns

True if this is an object restriction.

**abstract get\_property**() → *owlapy.owl\_property.OWLObjectPropertyExpression* 

#### Returns

Property being restricted.

```
class owlapy.class_expression.restriction.OWLQuantifiedRestriction
```

Bases: Generic[\_T], OWLRestriction, owlapy.meta\_classes.HasFiller[\_T]

Represents a quantified restriction.

#### **Parameters**

**\_T** – value type

\_\_slots\_\_ = ()

Bases: Generic[\_F], OWLQuantifiedRestriction[\_F], owlapy.meta\_classes.
HasCardinality

Base interface for owl min and max cardinality restriction.

#### **Parameters**

**\_F** – Type of filler.

$$\texttt{get\_cardinality}\,(\,)\,\to int$$

Gets the cardinality of a restriction.

#### Returns

The cardinality. A non-negative integer.

$$\texttt{get\_filler}() \rightarrow \_F$$

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a class expression or a data range.

#### **Returns**

the value

class owlapy.class\_expression.restriction.OWLQuantifiedObjectRestriction(
 filler: owlapy.class expression.class expression.OWLClassExpression)

Bases: OWLQuantifiedRestriction[owlapy.class\_expression.class\_expression. OWLClassExpression], OWLObjectRestriction

Represents a quantified object restriction.

$$\verb"get_filler"() \rightarrow owlapy.class\_expression.class\_expression.OWLClassExpression$$

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a constant (data value). For quantified restriction this will be a class expression or a data range.

```
Returns
```

the value

```
class owlapy.class expression.restriction.OWLObjectCardinalityRestriction(
           cardinality: int, property: owlapy.owl_property.OWLObjectPropertyExpression,
           filler: owlapy.class_expression.class_expression.OWLClassExpression)
               OWLCardinalityRestriction[owlapy.class_expression.class_expression.
     OWLClassExpression], OWLQuantifiedObjectRestriction
     Represents Object Property Cardinality Restrictions in the OWL 2 specification.
     __slots__ = ()
     get property() → owlapy.owl property.OWLObjectPropertyExpression
              Returns
                  Property being restricted.
     __repr__()
          Return repr(self).
     __eq__(other)
          Return self==value.
     __hash__()
          Return hash(self).
class owlapy.class expression.restriction.OWLObjectMinCardinality(
           cardinality: int, property: owlapy.owl property.OWLObjectPropertyExpression,
           filler: owlapy.class_expression.class_expression.OWLClassExpression)
     Bases: OWLObjectCardinalityRestriction
     A minimum cardinality expression ObjectMinCardinality (n OPE CE) consists of a nonnegative integer n, an object
     property expression OPE, and a class expression CE, and it contains all those individuals that are connected by
     OPE to at least n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Minimum_
     Cardinality)
     __slots__ = ('_cardinality', '_filler', '_property')
     type index: Final = 3008
class owlapy.class_expression.restriction.OWLObjectMaxCardinality(
           cardinality: int, property: owlapy.owl_property.OWLObjectPropertyExpression,
           filler: owlapy.class expression.class expression.OWLClassExpression)
     Bases: OWLObjectCardinalityRestriction
     A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an
     object property expression OPE, and a class expression CE, and it contains all those individuals that are connected
     by OPE
          to at most n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/
          #Maximum_Cardinality)
     __slots__ = ('_cardinality', '_filler', '_property')
     type_index: Final = 3010
```

```
class owlapy.class expression.restriction.OWLObjectExactCardinality(
            cardinality: int, property: owlapy.owl_property.OWLObjectPropertyExpression,
           filler: owlapy.class expression.class expression.OWLClassExpression)
     Bases: OWLObjectCardinalityRestriction
     An exact cardinality expression ObjectExactCardinality( n OPE CE ) consists of a nonnegative integer n,
          property expression OPE, and a class expression CE, and it contains all those individuals that are connected
          by to exactly n different individuals that are instances of CE.
     (https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality)
     __slots__ = ('_cardinality', '_filler', '_property')
     type_index: Final = 3009
     as_intersection_of_min_max()
                  → owlapy.class_expression.nary_boolean_expression.OWLObjectIntersectionOf
          Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.
              Returns
                  The semantically equivalent but structurally simpler form (= 1 R C) = >= 1 R C and <= 1 R C.
class owlapy.class_expression.restriction.OWLObjectSomeValuesFrom(
           property: owlapy.owl property.OWLObjectPropertyExpression,
           filler: owlapy.class expression.class expression.OWLClassExpression)
     Bases: OWLQuantifiedObjectRestriction
     An existential class expression ObjectSomeValuesFrom( OPE CE ) consists of an object property expression OPE
     and a class expression CE, and it contains all those individuals that are connected by OPE to an individual that is
     an instance of CE.
     __slots__ = ('_property', '_filler')
     type_index: Final = 3005
      __repr__()
          Return repr(self).
     __eq_ (other)
          Return self==value.
     __hash__()
          Return hash(self).
     get_property() → owlapy.owl_property.OWLObjectPropertyExpression
              Returns
                  Property being restricted.
class owlapy.class expression.restriction.OWLObjectAllValuesFrom(
           property: owlapy.owl_property.OWLObjectPropertyExpression,
           filler: owlapy.class expression.class expression.OWLClassExpression)
     Bases: OWLQuantifiedObjectRestriction
     A universal class expression ObjectAllValuesFrom( OPE CE ) consists of an object property expression OPE and
```

a class expression CE, and it contains all those individuals that are connected by OPE only to individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Universal\_Quantification)

```
__slots__ = ('_property', '_filler')
```

```
type_index: Final = 3006
     __repr__()
          Return repr(self).
     __eq_ (other)
          Return self==value.
     __hash__()
          Return hash(self).
     get_property() → owlapy.owl_property.OWLObjectPropertyExpression
              Returns
                  Property being restricted.
class owlapy.class_expression.restriction.OWLObjectHasSelf(
           property: owlapy.owl_property.OWLObjectPropertyExpression)
     Bases: OWLObjectRestriction
     A self-restriction ObjectHasSelf( OPE ) consists of an object property expression OPE, and it contains all those
     individuals that are connected by OPE to themselves. (https://www.w3.org/TR/owl2-syntax/#Self-Restriction)
     __slots__ = '_property'
     type_index: Final = 3011
     get_property() → owlapy.owl_property.OWLObjectPropertyExpression
              Returns
                  Property being restricted.
     __eq_ (other)
          Return self==value.
     __hash__()
          Return hash(self).
     __repr__()
          Return repr(self).
class owlapy.class_expression.restriction.OWLObjectHasValue(
           property: owlapy.owl_property.OWLObjectPropertyExpression,
           individual: owlapy.owl_individual.OWLIndividual)
     Bases: OWLHasValueRestriction[owlapy.owl_individual.OWLIndividual], OWLObjec-
     tRestriction
     A has-value class expression ObjectHasValue( OPE a ) consists of an object property expression OPE and an
     individual a, and it contains all those individuals that are connected by OPE to a. Each such class expression
     can be seen as a syntactic shortcut for the class expression ObjectSomeValuesFrom( OPE ObjectOneOf( a ) ).
     (https://www.w3.org/TR/owl2-syntax/#Individual_Value_Restriction)
     __slots__ = ('_property', '_v')
     type_index: Final = 3007
     get_property() → owlapy.owl_property.OWLObjectPropertyExpression
              Returns
                  Property being restricted.
```

```
A convenience method that obtains this restriction as an existential restriction with a nominal filler.
               Returns
                   The existential equivalent of this value restriction. simp(HasValue(p a)) = some(p \{a\}).
     __repr__()
          Return repr(self).
class owlapy.class_expression.restriction.OWLObjectOneOf(
            values: owlapy.owl_individual.OWLIndividual | Iterable[owlapy.owl_individual.OWLIndividual])
     Bases: owlapy.class expression.class expression.OWLAnonymousClassExpression,
     owlapy.meta_classes.HasOperands[owlapy.owl_individual.OWLIndividual]
     An enumeration of individuals ObjectOneOf( a1 ... an ) contains exactly the individuals ai with 1 \le i \le n. (https:
     //www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals)
      __slots__ = '_values'
     type_index: Final = 3004
     individuals() \rightarrow Iterable[owlapy.owl\_individual.OWLIndividual]
          Gets the individuals that are in the oneOf. These individuals represent the exact instances (extension) of this
          class expression.
               Returns
                   The individuals that are the values of this {@code ObjectOneOf} class expression.
     operands () → Iterable[owlapy.owl individual.OWLIndividual]
          Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.
               Returns
                   The operands.
     as\_object\_union\_of() \rightarrow owlapy.class\_expression.class\_expression.OWLClassExpression
          Simplifies this enumeration to a union of singleton nominals.
               Returns
                   This enumeration in a more standard DL form. simp({a}) = {a} simp({a0, ..., {an}}) =
                   unionOf(\{a0\}, \ldots, \{an\})
      __hash___()
          Return hash(self).
      __eq__(other)
          Return self==value.
      __repr__()
          Return repr(self).
class owlapy.class_expression.restriction.OWLDataRestriction
     Bases: OWLRestriction
     Represents a Data Property Restriction.
     __slots__ = ()
```

as some values from ()  $\rightarrow$  owlapy.class\_expression.class\_expression.OWLClassExpression

```
\textbf{is\_data\_restriction} \, (\,) \, \to bool \,
```

Determines if this is a data restriction.

#### Returns

True if this is a data restriction.

```
class owlapy.class_expression.restriction.OWLQuantifiedDataRestriction(
     filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: OWLQuantifiedRestriction[owlapy.owl\_data\_ranges.OWLDataRange], OWL-DataRestriction

Represents a quantified data restriction.

```
__slots__ = ()
```

```
get_filler() → owlapy.owl_data_ranges.OWLDataRange
```

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a class expression or a data range.

#### **Returns**

the value

Bases: OWLCardinalityRestriction[owlapy.owl\_data\_ranges.OWLDataRange], OWLQuantifiedDataRestriction, OWLDataRestriction

Represents Data Property Cardinality Restrictions.

```
__slots__ = ()
```

**get property**() → *owlapy.owl property.OWLDataPropertyExpression* 

### Returns

Property being restricted.

```
__repr__()
```

Return repr(self).

\_\_eq\_\_(other)

Return self==value.

\_\_hash\_\_()

Return hash(self).

*filler*: owlapy.owl\_data\_ranges.OWLDataRange)

Bases: OWLDataCardinalityRestriction

A minimum cardinality expression DataMinCardinality( n DPE DR ) consists of a nonnegative integer n, a data property expression DPE, and a unary data range DR, and it contains all those individuals that are connected by DPE to at least n different literals in DR. (https://www.w3.org/TR/owl2-syntax/#Minimum\_Cardinality)

```
__slots__ = ('_cardinality', '_filler', '_property')
type_index: Final = 3015
```

```
class owlapy.class expression.restriction.OWLDataMaxCardinality (cardinality: int.
            property: owlapy.owl_property.OWLDataPropertyExpression,
            filler: owlapy.owl data ranges.OWLDataRange)
     Bases: OWLDataCardinalityRestriction
     A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an
     object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by
     OPE to at most n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Maximum
     Cardinality)
     __slots__ = ('_cardinality', '_filler', '_property')
     type index: Final = 3017
class owlapy.class_expression.restriction.OWLDataExactCardinality(
            cardinality: int, property: owlapy.owl_property.OWLDataPropertyExpression,
            filler: owlapy.owl_data_ranges.OWLDataRange)
     Bases: OWLDataCardinalityRestriction
     An exact cardinality expression ObjectExactCardinality (n OPE CE) consists of a nonnegative integer n, an object
     property expression OPE, and a class expression CE, and it contains all those individuals that are connected
          by OPE to exactly n different individuals that are instances of CE (https://www.w3.org/TR/owl2-syntax/
          #Exact_Cardinality)
     __slots__ = ('_cardinality', '_filler', '_property')
     type_index: Final = 3016
     as_intersection_of_min_max()
                  → owlapy.class_expression.nary_boolean_expression.OWLObjectIntersectionOf
          Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.
              Returns
                  The semantically equivalent but structurally simpler form (= 1 R D) = >= 1 R D and <= 1 R D.
class owlapy.class_expression.restriction.OWLDataSomeValuesFrom(
            property: owlapy.owl property.OWLDataPropertyExpression,
            filler: owlapy.owl data ranges.OWLDataRange)
     Bases: OWLQuantifiedDataRestriction
     An existential class expression DataSomeValuesFrom( DPE1 ... DPEn DR ) consists of n data property expres-
     sions DPEi, 1 \le i \le n, and a data range DR whose arity must be n. Such a class expression contains all those
     individuals that are connected by DPEi to literals lti, 1 \le i \le n, such that the tuple (lt1, ..., ltn) is in DR. A class
     expression of the form DataSomeValuesFrom( DPE DR ) can be seen as a syntactic shortcut for the class expression
     DataMinCardinality( 1 DPE DR ). (https://www.w3.org/TR/owl2-syntax/#Existential_Quantification_2)
     __slots__ = '_property'
     type_index: Final = 3012
     __repr__()
          Return repr(self).
     __eq_ (other)
          Return self==value.
     __hash__()
```

Return hash(self).

```
get_property() → owlapy.owl_property.OWLDataPropertyExpression
              Returns
                  Property being restricted.
class owlapy.class expression.restriction.OWLDataAllValuesFrom(
            property: owlapy.owl_property.OWLDataPropertyExpression,
            filler: owlapy.owl_data_ranges.OWLDataRange)
     Bases: OWLQuantifiedDataRestriction
     A universal class expression DataAllValuesFrom( DPE1 ... DPEn DR ) consists of n data property expressions
     DPEi, 1 \le i \le n, and a data range DR whose arity must be n. Such a class expression contains all those individuals
     that
          are connected by DPEi only to literals lti, 1 \le i \le n, such that each tuple (lt1, ..., ltn) is in DR.
          A class
              expression of the form DataAllValuesFrom( DPE DR ) can be seen as a syntactic shortcut for the
              class expression DataMaxCardinality( 0 DPE DataComplementOf( DR ) ). (https://www.w3.org/
              TR/owl2-syntax/#Universal Quantification 2)
     __slots__ = '_property'
     type_index: Final = 3013
     __repr__()
          Return repr(self).
     __eq_ (other)
          Return self==value.
     __hash__()
          Return hash(self).
     get_property() → owlapy.owl_property.OWLDataPropertyExpression
              Returns
                  Property being restricted.
class owlapy.class expression.restriction.OWLDataHasValue(
            property: owlapy.owl_property.OWLDataPropertyExpression,
            value: owlapy.owl literal.OWLLiteral)
     Bases: OWLHasValueRestriction[owlapy.owl_literal.OWLLiteral], OWLDataRestric-
     tion
     A has-value class expression DataHasValue(DPE lt) consists of a data property expression DPE and a literal lt,
     and it contains all those individuals that are connected by DPE to lt. Each such class expression can be seen as a
     syntactic shortcut for the class expression DataSomeValuesFrom( DPE DataOneOf( lt ) ). (https://www.w3.org/
     TR/owl2-syntax/#Literal Value Restriction)
     __slots__ = '_property'
     type_index: Final = 3014
```

\_\_repr\_\_()

 $\underline{\phantom{a}}$ eq $\underline{\phantom{a}}$  (other)

Return repr(self).

Return self==value.

```
hash ()
           Return hash(self).
     as_some_values_from() → owlapy.class_expression.class_expression.OWLClassExpression
           A convenience method that obtains this restriction as an existential restriction with a nominal filler.
                   The existential equivalent of this value restriction. simp(HasValue(p a)) = some(p \{a\}).
     get property() → owlapy.owl property.OWLDataPropertyExpression
               Returns
                   Property being restricted.
class owlapy.class_expression.restriction.OWLDataOneOf(
            values: owlapy.owl_literal.OWLLiteral | Iterable[owlapy.owl_literal.OWLLiteral])
                        owlapy.owl data ranges.OWLDataRange,
                                                                                  owlapy.meta classes.
     HasOperands[owlapy.owl_literal.OWLLiteral]
     An enumeration of literals DataOneOf(lt1 ... ltn) contains exactly the explicitly specified literals lti with 1 \le i \le
     n. The resulting data range has arity one. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Literals)
     type_index: Final = 4003
     values() \rightarrow Iterable[owlapy.owl\_literal.OWLLiteral]
           Gets the values that are in the oneOf.
               Returns
                   The values of this {@code DataOneOf} class expression.
     operands() \rightarrow Iterable[owlapy.owl\_literal.OWLLiteral]
           Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.
               Returns
                   The operands.
       _hash__()
           Return hash(self).
      \underline{\phantom{a}}eq\underline{\phantom{a}} (other)
           Return self==value.
     __repr__()
           Return repr(self).
class owlapy.class_expression.restriction.OWLDatatypeRestriction(
            type_: owlapy.owl_datatype.OWLDatatype,
            facet_restrictions: OWLFacetRestriction \ Iterable[OWLFacetRestriction])
     Bases: owlapy.owl data ranges.OWLDataRange
     A datatype restriction DatatypeRestriction (DT F1 lt1 ... Fn ltn ) consists of a unary datatype DT and n pairs (
     Fi, lti). The resulting data range is unary and is obtained by restricting the value space of DT according to the
     semantics of all (Fi, vi) (multiple pairs are interpreted conjunctively), where vi are the data values of the literals
     lti. (https://www.w3.org/TR/owl2-syntax/#Datatype_Restrictions)
     __slots__ = ('_type', '_facet_restrictions')
     type index: Final = 4006
     get_datatype() → owlapy.owl_datatype.OWLDatatype
```

```
\texttt{get\_facet\_restrictions} () \rightarrow Sequence[OWLFacetRestriction]
     __eq_ (other)
           Return self==value.
     __hash__()
           Return hash(self).
     __repr__()
           Return repr(self).
class owlapy.class_expression.restriction.OWLFacetRestriction(
            facet: owlapy.vocab.OWLFacet, literal: Literals)
     Bases: owlapy.owl_object.OWLObject
     A facet restriction is used to restrict a particular datatype.
     __slots__ = ('_facet', '_literal')
     type_index: Final = 4007
     \texttt{get\_facet}() \rightarrow owlapy.vocab.OWLFacet
     \texttt{get\_facet\_value} \ () \ \rightarrow owlapy.owl\_literal.OWLLiteral
     __eq_ (other)
          Return self==value.
      __hash__()
           Return hash(self).
     __repr__()
           Return repr(self).
```

#### **Attributes**

OWLThing
OWLNothing

# **Classes**

OWLClassExpression	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
OWLAnonymousClassExpression	A Class Expression which is not a named Class.
OWLBooleanClassExpression	Represent an anonymous boolean class expression.
OWLObjectComplementOf	Represents an ObjectComplementOf class expression in the OWL 2 Specification.

continues on next page

Table 2 - continued from previous page

An OWL 2 named Class. Classes can be understood as sets of individuals.  WLNaryBooleanClassExpression  WLObjectUnionOf  A union class expression ObjectUnionOf(CE1 CEn) contains all individuals that are instances  WLObjectIntersectionOf  A union class expression ObjectUnicnOf(CE1 CEn) contains all individuals that are instances  An intersection class expression ObjectIntersectionOf(CE1 CEn) contains all individuals that are instances  An intersection class expression ObjectIntersectionOf(CE1 CEn) contains all individuals that are instances  WLRestriction  WLRestriction  WLQuantifiedRestriction  WLQuantifiedRestriction  WLQuantifiedObjectRestriction  WLQuantifiedObjectRestriction  WLQuantifiedObjectRestriction  WLDataRestriction  WLDataRestriction  WLDataRestriction  WLDataRestriction  WLDataRestriction  WLDataRestriction  WLDataRestriction  WLDataRestriction  WLDataRestriction  WLObjectLassEelf  WLDataOneOf  A self-restriction ObjectProperty Cardinality Restrictions in the OWL 2 specification.  WLDataOneOf  A self-restriction ObjectProperty Cardinality Restrictions in the OWL 2 specification.  WLDataOneOf  A self-restriction ObjectProperty Cardinality Restrictions in the OWL 2 specification of literals DataOneOf( lt1 ltn ) contains a search the explicitly specified literals lti with Represents a quantified data restriction.  WLDataOneOf  WLDataOneOf  WLDataOneOf  WLDataOneOf  An enumeration of literals DataOneOf( lt1 ltn ) contains a caacty the explicitly specified literals lti with Represents a quantified data restriction.  WLDataOneOf  An existential class expression ObjectMaxCardinality Restrictions on the explicitly specified literals lti with Represents a quantified data restriction.  WLDataOneOf  A universal class expression ObjectMaxCardinality consists of an object property expression OPE and a Navalue class expression ObjectMaxCardinality on OPE CE ) consists of a nonnegative integer n, an object on object property expression OPE and a nonicet object MaxCardinality on OPE CE ) co	Table 2 – continued from previous page			
OWLObjectIntersectionOf         A union class expression ObjectInionOf( CE1 CEn ) contains all individuals that are instances           OWLObjectIntersectionOf         An intersection class expression ObjectIntersectionOf( CE1 CEn ) contains all individuals that are instances           OWLOBATIFIED RESERTATION         Represents an Object Property Restriction of Data Property Restriction in the OWL 2 specification.           OWLOBATIFIED RESERTATION         Represents a quantified restriction.           OWLDATE RESERTATION         Represents a Object Property Restriction in the OWL 2 specification.           OWLDATE RESERTATION         Represents a Data Property Restriction in the OWL 2 specification.           OWLDATE RESERTATION         Represents a Data Property Restriction in the OWL 2 specification.           OWLOBITITION REPRESENTATION OWLOGATION TO SERVICE AND ASSETT COLOR OWLOGATION TO SERVICE ASSETT COLOR OWLOGATION TO SER	OWLClass	sets of individuals.		
contains all individuals that are instances  OWLObjectIntersectionOf  An intersection class expression ObjectIntersectionOf( CEICEn) contains all individuals that are instances  Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.  OWLOwlobjectRestriction  OWLOwlobjectRestriction  OWLOwlobjectRestriction  OWLOBjectRestriction  OWLOBjectSomeValuesFrom  OWLOBjectSomeValuesFrom  OWLOBjectRestriction  OWLOBjectRes	OWLNaryBooleanClassExpression	OWLNaryBooleanClassExpression.		
CEI CEn ) contains all individuals that are instances Represents an Object Property Restriction on Data Property Restriction on Data Property Restriction in the OWL 2 specification.  WLOuantifiedObjectRestriction Represents a quantified object restriction.  WLOuantifiedObjectRestriction Represents a quantified object restriction.  WLOBJectRestriction Represents an Object Property Restriction in the OWL 2 specification.  WLOBJectRestriction Represents an Object Property Restriction in the OWL 2 specification.  WLOBJectRestriction Represents a Data Property Restriction in the OWL 2 Represents a Data Property Restriction.  WLOBJectCardinalityRestriction Base interface for owl min and max cardinality restriction. Owt. ObjectHasSelf A self-restriction ObjectHasSelf (OPE) consists of an object property expression OPE.  An enumeration of literals DataOneOf(ltl ltn) contains exactly the explicitly specified literals lti with Represents a quantified data restriction.  WLOBJectAcardinalityRestriction Represents a quantified data restriction.  WLOBJectSomeValuesFrom An existential class expression ObjectSomeValuesFrom(OPE CE) consists of an object property expression OPE and an An existential class expression ObjectAllValuesFrom(OPE CE) consists of an object property expression OPE and a A has-value class expression ObjectHasValue A has-value class expression ObjectHasValue (OPE a) consists of an object property expression OPE and a A face trestriction is used to restrict a particular datatype.  WLOBJectMinCardinality An ananimum cardinality expression ObjectMinCardinality of DPE CE) consists of a nonnegative integer n, an object operation object property expression ObjectMaxCardinality of DPE CE) consists of a nonnegative integer n, an object operation object property expression ObjectMaxCardinality of DPE CE) consists of n data property expressions DPE, and an animal class expression DataAllValuesFrom (DPE II) operations of the property expression DPE and a literal II, DPEn DR ) consists of n data property expre	OWLObjectUnionOf			
OWLP estriction         Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.           OWLQ uantified Striction         Represents a quantified destriction.           OWLD bject Restriction         Represents a quantified object restriction in the OWL 2 specification.           OWLD at Agreement Striction         Represents an Object Property Restriction in the OWL 2 specification.           OWLD at Agreement Striction         Represents a Data Property Restriction.           OWL Cardinality Restriction         Base interface for owl min and max cardinality restriction in the OWL 2 specification.           OWL Description Striction Stricti	OWLObjectIntersectionOf			
OWLQuantifiedRestriction         Represents a quantified restriction.           OWLDatifiedObjectRestriction         Represents an Object Property Restriction in the OWL 2 specification.           OWLDataRestriction         Represent a Has Value restriction in the OWL 2 specification.           OWLDataRestriction         Represent a Data Property Restriction.           OWLOardinalityRestriction         Base interface for owl min and max cardinality restrictions in the OWL 2 specification.           OWLObjectCardinalityRestriction         Represents Object Property Cardinality Restrictions in the OWL 2 specification.           OWLDataOneOf         An enumeration of literals DataOneOf( ltl ltn ) contains exactly the explicitly specified literals it with Represents a quantified data restriction.           OWLDataCardinalityRestriction         Represents a quantified data restriction.           OWLObjectSomeValuesFrom         An existential class expression ObjectSomeValuesFrom (OPE CE) consists of an object property expression OPE and and and object property expression OPE and and owload of the property expression OPE and an owload of the property expression OPE and and the property expression OPE and and owload of the property expression OPE and an owload of the property expression OPE and and object property expression OPE and an object property expression OPE and an object property expression OPE and an object property expression	OWLRestriction			
OWLOyantifiedObjectRestriction         Represents an Object Property Restriction in the OWL 2 specification.           OWLhasValueRestriction         Represent a HasValue restriction in the OWL 2 specification.           OWLDataRestriction         Represents a Data Property Restriction.           OWLOdification         Represents a Data Property Restriction.           OWLObjectCardinalityRestriction         Represents Object Property Cardinality Restrictions in the OWL 2 specification.           OWLObjectHasSelf         A self-restriction ObjectHasSelf (OPE ) consists of an object property expression OPE.           OWLDataOneOf         An enumeration of literals DataOneOf( ltl ltn ) contains exactly the explicitly specified literals lit with Represents a quantified data restriction.           OWLOdification         Represents a quantified data restriction.           OWLObjectSomeValuesFrom         An existential class expression ObjectSomeValuesFrom (OPE CE) consists of an object property expression OPE and and a wais an advantage of the consist of an object property expression OPE and and a A has-value class expression ObjectHasValue (OPE a) consists of an object property expression OPE and an A datatype restriction DatatypeRestriction (DF and a pairs object Property expression OPE and an A datatype restriction batatypeRestriction (DF and a pairs object MaxCardinality)           OWLDatacted         A facet restriction is used to restrict a particular datatype.           OWLObjectMinCardinality         A maximum cardinality expression ObjectMaxCardinality (n OPE CE) consists of a nonnegative integer n, an object object Analysis of a	OWLQuantifiedRestriction	•		
OWLObjectRestriction         Represents an Object Property Restriction in the OWL 2           OWLDataRestriction         Represent a Has Value restriction in the OWL 2           OWLDataRestriction         Represents a Data Property Restriction.           OWLOardinalityRestriction         Base interface for owl min and max cardinality restrictions in the OWL 2 specification.           OWLObjectCardinalityRestriction         Represents Object Property Cardinality Restrictions in the OWL 2 specification.           OWLDataOneOf         An enumeration of literals DataOneOf( ltl ltn ) contains exactly the explicitly specified literals lit with           OWLQuantifiedDataRestriction         Represents Data Property Cardinality Restriction.           OWLDataCardinalityRestriction         Represents Data Property Cardinality Restrictions.           OWLObjectSomeValuesFrom         An existential class expression ObjectAllValuesFrom (OPE CE) consists of an object property expression OPE and and an A datatype restriction DatatypeRestriction (OPE CE) and an A datatype restriction DatatypeRestriction (OPE In an an A datatype restriction DatatypeRestriction (OPE In		•		
OWLDataRestriction         Represent a HasValue restriction in the OWL 2           OWLDataRestriction         Represents a Data Property Restriction.           OWLODJECTAINALITYRESTRICTION         Base interface for owl min and max cardinality restriction in the OWL 2 specification.           OWLODJECTAINALITYRESTRICTION DISCULTANCE         Represents Object Property Cardinality Restrictions in the OWL 2 specification.           OWLDataOneOF         An enumeration of literals DataOneOf( lt1 ltn ) contains exactly the explicitly specified literals lti with           OWLOQUANTIFIED ACTION OWLDATE CARDINALITY RESTRICTION         Represents a quantified data restriction.           OWLOBJECTSOMEVALUESFROM         An existential class expression ObjectSomeValuesFrom (OPE CE) consists of an object property expression OPE and a           OWLODJECTAILVALUESFROM         A universal class expression ObjectAllValuesFrom (OPE CE) consists of an object property expression OPE and a           OWLODJECTAILVALUESFROM         A datatype restriction DatatypeRestriction (OPE and a on a datatype restriction and a datatype DT and n pairs (OPE and a on a datatype restriction and a datatype of a unary datatype DT and n pairs (OPE and a on a pair and a datatype and a on a consist of a nonnegative integer (OPE and a on a object (OPE and a on a o		Represents an Object Property Restriction in the OWL 2		
OWLDataRestriction         Represents a Data Property Restriction.           OWLCardinalityRestriction         Base interface for owl min and max cardinality restriction.           OWLObjectCardinalityRestriction         Represents Object Property Cardinality Restrictions in the OWL 2 specification.           OWLDataOneOf         An enumeration of literals DataOneOf( lt1 ltn ) contains exactly the explicitly specified literals lt in with           OWLQuantifiedDataRestriction         Represents a quantified data restriction.           OWLDataCardinalityRestriction         Represents Data Property Cardinality Restrictions.           OWLObjectSomeValuesFrom         An existential class expression ObjectSomeValuesFrom(OPE CD consists of an object property expression OPE and and           OWLObjectAllValuesFrom         A universal class expression ObjectAllValuesFrom(OPE CD consists of an object property expression OPE and and A has-value class expression OPE and and A has-value class expression OPE and and A datatype restriction DatatypeRestriction (DT F1 lt1 Fn ltn) consists of a unary datatype DT and n pairs           OWLFacet         Enumerations for OWL facets.           OWLObjectMinCardinality         A facet restriction is used to restrict a particular datatype.           OWLObjectMaxCardinality         A maximum cardinality expression ObjectMaxCardinality (n OPE CE) consists of a nonnegative integer n, an object           OWLObjectExactCardinality         An exact cardinality expression ObjectMaxCardinality (n OPE CE) consists of n data property expressions           <	OWLHasValueRestriction	•		
OWLCardinalityRestriction         Base interface for owl min and max cardinality restriction.           OWLObjectCardinalityRestriction         Represents Object Property Cardinality Restrictions in the OWL 2 specification.           OWLDataOneOf         A self-restriction ObjectHasSelf( OPE ) consists of an object property expression OPE.           OWLQuantifiedDataRestriction         An enumeration of literals DataOneOf( ltl ltn ) contains exactly the explicitly specified literals lti with           OWLQuantifiedDataRestriction         Represents a quantified data restriction.           OWLDataCardinalityRestriction         Represents Data Property Cardinality Restrictions.           OWLObjectSomeValuesFrom         An existential class expression ObjectSomeValuesFrom( OPE CE) consists of an object property expression OPE and an OWLObjectHasValue           OWLObjectHasValue         A has-value class expression ObjectAllValuesFrom( OPE CE) consists of an object property expression OPE and an A datatype restriction DatatypeRestriction (DT Fl ltl Fn ltn) consists of a unary datatype DT and n pairs           OWLFacet         Enumerations for OWL facets.           OWLFacetRestriction         A facet restriction is used to restrict a particular datatype.           OWLObjectMinCardinality         A minimum cardinality expression ObjectMacCardinality (n OPE CE) consists of a nonnegative integer n, an object           OWLObjectExactCardinality         An exact cardinality expression ObjectMacCardinality (n OPE CE) consists of n data property expressions           OWLDataBalVal				
OWLObjectCardinalityRestriction         Represents Object Property Cardinality Restrictions in the OWL 2 specification.           OWLObjectHasSelf         A self-restriction ObjectHasSelf( OPE ) consists of an object property expression OPE.           OWLDataOneOf         An enumeration of literals DataOneOf (lt1 ltn ) contains exactly the explicitly specified literals lti with Represents a quantified data restriction.           OWLQuantifiedDataRestriction         Represents a quantified data restriction.           OWLDataCardinalityRestriction         Represents Data Property Cardinality Restrictions.           OWLObjectSomeValuesFrom         An existential class expression Object property expression OPE and an existential class expression Object property expression OPE and an A datatype restriction DatatypeRestriction (DT F1 lt1 Fn ltn ) consists of a unary datatype DT and n pairs           OWLDatatypeRestriction         A facet restriction is used to restrict a particular datatype.           OWLObjectMinCardinality         A minimum cardinality expression ObjectMinCardinality (n OPE CE) consists of a nonnegative integer n, an object           OWLObjectMaxCardinality         An exact cardinality expression DijectMaxCardinality (n OPE CE) consists of a nonnegative integer n, an object on OPE CE) consists of n data property expressions           OWLDataSomeValuesFrom         An existential class expression DataAllValuesFrom(DPEI DPEn DR) consists of n data property expressions on DPEi DPEn DR) consists of				
the OWL 2 specification.  A self-restriction ObjectHasSelf (OPE ) consists of an object property expression OPE,  An enumeration of literals DataOneOf (lt1 ltn ) contains exactly the explicitly specified literals lti with Represents a quantified data restriction.  OWLQuantifiedDataRestriction Represents a quantified data restriction.  OWLObjectSomeValuesFrom Represents Data Property Cardinality Restrictions.  OWLObjectSomeValuesFrom An existential class expression ObjectSomeValuesFrom (OPE CE ) consists of an object property expression OPE and a OWLObjectHasValue  OWLObjectHasValue A has-value class expression ObjectHasValue (OPE a) consists of an object property expression OPE and a OWLDatatypeRestriction  OWLFacet Enumerations for OWL facets.  OWLFacet Enumerations for OWL facets.  OWLFacet Enumerations for OWL facets.  OWLObjectMinCardinality A minimum cardinality expression ObjectMinCardinality (n OPE CE) consists of a nonnegative integer n, an object OWLObjectMaxCardinality  OWLObjectMaxCardinality An exact cardinality expression ObjectMaxCardinality (n OPE CE) consists of a nonnegative integer n, an object OWLDataSomeValuesFrom  OWLObjectExactCardinality An exact cardinality expression DipectExactCardinality n OPE CE) consists of a nonnegative integer n, an object OWLDataSomeValuesFrom  An exact cardinality expression DataSomeValuesFrom DPEI DPEn DR ) consists of n data property expressions DPEi DPEn DR ) consists of n data property expressions DPEi DPEn DR ) consists of n data property expressions DPEi DPEn DR ) consists of n data property expressions DPEi DPEn DR ) consists of n data property expressions DPEi DPEn DR ) consists of n data property expressions DPEi DPEn DR ) consists of n data property expressions DPEi DPEn DR ) consists of n data property expressions DPEi DPEn DR ) consists of n data property expressions DPEi DPEn DR ) consists of n data property expressions DPEi DPEn DR ) consists of n data property expressions DPEi DPEn DR ) consists		·		
OWLDbjectHasSelf         A self-restriction ObjectHasSelf (OPE ) consists of an object property expression OPE.           OWLDataOneOf         An enumeration of literals DataOneOf (It1 Itn ) contains exactly the explicitly specified literals Iti with Represents a quantified data restriction.           OWLDuartifiedDataRestriction         Represents Data Property Cardinality Restrictions.           OWLObjectSomeValuesFrom         An existential class expression ObjectSomeValuesFrom (OPE CE) consists of an object property expression OPE and and owlobjectHasValue           OWLObjectAllValuesFrom         A universal class expression ObjectHasValue (OPE a) consists of an object property expression OPE and and an object property expression OPE and an owlobject owlobject in owlobject owlobj				
object property expression OPE, An enumeration of literals DataOneOf( lt1 ltn ) contains exactly the explicitly specified literals lti with  WLQuantifiedDataRestriction WLDataCardinalityRestriction Represents a quantified data restriction.  WLObjectSomeValuesFrom An existential class expression ObjectSomeValuesFrom( OPE CE) consists of an object property expression OPE and  WLObjectAllValuesFrom A universal class expression ObjectAllValuesFrom( OPE CE) consists of an object property expression OPE and a OWLObjectHasValue A has-value class expression ObjectHasValue( OPE a) consists of an object property expression OPE and an OWLDatatypeRestriction A datatype restriction DatatypeRestriction( DT F1 lt1 Fn ltn) consists of a unary datatype DT and n pairs  WLFacet Enumerations for OWL facets.  WLFacetsetriction A facet restriction is used to restrict a particular datatype.  WLObjectMinCardinality A minimum cardinality expression ObjectMinCardinality( n OPE CE) consists of a nonnegative integer n, an object  WLObjectMaxCardinality An exact cardinality expression ObjectMaxCardinality( n OPE CE) consists of a nonnegative integer n, an object  WLDataSomeValuesFrom An existential class expression DataSomeValuesFrom( DPE1 DPEn DR) consists of n data property expressions DPE1 DPEn DR) consists of n data property expressions DPE1 DPEn DR) consists of n data property expressions DPE and a literal lt,	OWLObject HasSelf	*		
An enumeration of literals DataOneOf( lt1 ltn ) contains exactly the explicitly specified literals lti with  WLQuantifiedDataRestriction  Represents a quantified data restriction.  WLObjectSomeValuesFrom  An existential class expression ObjectSomeValuesFrom( OPE CE) consists of an object property expression OPE and  WLObjectAllValuesFrom  A universal class expression ObjectAllValuesFrom(OPE CE) consists of an object property expression OPE and a  WLObjectHasValue  A has-value class expression ObjectHasValue(OPE a) consists of an object property expression OPE and a  WLDatatypeRestriction  A datatype restriction DatatypeRestriction(DT F1 lt1 Fn ltn) consists of a unary datatype DT and n pairs  WLFacet  Enumerations for OWL facets.  WLFacetRestriction  A facet restriction is used to restrict a particular datatype.  WLObjectMinCardinality  A minimum cardinality expression ObjectMinCardinality(n OPE CE) consists of a nonnegative integer n, an object  WLObjectMaxCardinality  An exact cardinality expression ObjectMaxCardinality(n OPE CE) consists of a nonnegative integer n, an object  OWLObjectExactCardinality  An exact cardinality expression DetaExactCardinality(n OPE CE) consists of a nonnegative integer n, an object  OWLDataSomeValuesFrom  An existential class expression DataSomeValuesFrom(DPE1 DPEn DR) consists of n data property expressions  OWLDataAllValuesFrom  A universal class expression DataAllValuesFrom(DPE1 DPEn DR) consists of n data property expressions  DPEi DPEn DR) consists of n data property expressions DPEi DPEn DR and a literal lt,		· · · · · · · · · · · · · · · · · · ·		
tains exactly the explicitly specified literals lti with  OWLQuantifiedDataRestriction Represents a quantified data restriction.  OWLObjectSomeValuesFrom An existential class expression ObjectSomeValuesFrom(OPE CE) consists of an object property expression OPE and  OWLObjectAllValuesFrom A universal class expression ObjectAllValuesFrom(OPE CE) consists of an object property expression OPE and a  OWLObjectHasValue A has-value class expression ObjectHasValue(OPE a) consists of an object property expression OPE and an  OWLDatatypeRestriction A datatype restriction DatatypeRestriction(DT F1 lt1 Fn ltn) consists of a unary datatype DT and n pairs  OWLFacet Enumerations for OWL facets.  OWLFacetRestriction A facet restriction is used to restrict a particular datatype.  OWLObjectMinCardinality A minimum cardinality expression ObjectMaxCardinality(n OPE CE) consists of a nonnegative integer n, an object  OWLObjectMaxCardinality An exact cardinality expression ObjectExactCardinality(n OPE CE) consists of a nonnegative integer n, an object  OWLObjectExactCardinality An exact cardinality expression DetataSomeValuesFrom An existential class expression DataAllValuesFrom(DPE1 DPEn DR) consists of n data property expressions  OWLDataAllValuesFrom A universal class expression DataAllValuesFrom(DPE1 DPEn DR) consists of n data property expressions  OWLDataAllValuesFrom A has-value class expression DataAllValuesFrom(DPE1 DPEn DR) consists of n data property expressions  OWLDataAllValuesFrom A has-value class expression DataAllValuesFrom(DPE1 DPEn DR) consists of n data property expressions	OWI.DataOneOf			
OWLQuantifiedDataRestriction         Represents a quantified data restriction.           OWLDataCardinalityRestriction         Represents Data Property Cardinality Restrictions.           OWLObjectSomeValuesFrom         An existential class expression ObjectSomeValuesFrom(OPE and and OPE CE) consists of an object property expression OPE and a           OWLObjectAllValuesFrom         A universal class expression ObjectHalValue(OPE and an object property expression OPE and a nobject property expression OPE and an object operation of the property expression OPE and an object operation of the property expression operation object operation object operation object operation operat				
OWLDataCardinalityRestriction         Represents Data Property Cardinality Restrictions.           OWLObjectSomeValuesFrom         An existential class expression ObjectSomeValuesFrom(OPE CE) consists of an object property expression OPE and and           OWLObjectAllValuesFrom         A universal class expression ObjectAllValuesFrom(OPE CE) consists of an object property expression OPE and and           OWLObjectHasValue         A has-value class expression ObjectHasValue(OPE a) consists of an object property expression OPE and and and and an object property expression OPE and and and A datatype restriction DatatypeRestriction (DT F1 lt1 Fn ltn ) consists of a unary datatype DT and n pairs           OWLFacet         Enumerations for OWL facets.           OWLFacetRestriction         A facet restriction is used to restrict a particular datatype.           OWLObjectMinCardinality         A minimum cardinality expression ObjectMinCardinality(n OPE CE) consists of a nonnegative integer n, an object           OWLObjectMaxCardinality         A maximum cardinality expression ObjectMaxCardinality(n OPE CE) consists of a nonnegative integer n, an object           OWLObjectExactCardinality         An exact cardinality expression ObjectExactCardinality(n OPE CE) consists of a nonnegative integer n, an object           OWLDataSomeValuesFrom         An existential class expression DataSomeValuesFrom(DPE1 DPEn DR) consists of n data property expressions           OWLDataAllValuesFrom         A universal class expression DataAllValuesFrom(DPE1 DPEn DR) consists of n data property expressions DPEi.	OWLOuantifiedDataRestriction			
An existential class expression ObjectSomeValuesFrom OPE CE ) consists of an object property expression OPE and A universal class expression ObjectAllValuesFrom OPE CE) consists of an object property expression OPE and a OWLObjectHasValue A has-value class expression ObjectHasValue (OPE a) consists of an object property expression OPE and an OWLDatatypeRestriction A datatype restriction DatatypeRestriction (DT F1 lt1 Fn ltn) consists of a unary datatype DT and n pairs OWLFacet Enumerations for OWL facets.  OWLFacet Enumerations for OWL facets.  OWLObjectMinCardinality A minimum cardinality expression ObjectMinCardinality (n OPE CE) consists of a nonnegative integer n, an object owlobjectMaxCardinality (n OPE CE) consists of a nonnegative integer n, an object OWLObjectExactCardinality An exact cardinality expression ObjectExactCardinality (n OPE CE) consists of a nonnegative integer n, an object OWLDataSomeValuesFrom An existential class expression DataSomeValuesFrom OPE1 DPEn DR) consists of n data property expressions  OWLDataAllValuesFrom A universal class expression DataAllValuesFrom (DPE1 DPEn DR) consists of n data property expressions DPEi,  OWLDataHasValue A has-value class expression DataHasValue(DPE lt) consists of a data property expression DPE and a literal lt,		•		
CE) consists of an object property expression OPE and a A has-value class expression ObjectHasValue(OPE a) consists of an object property expression OPE and an OWLDatatypeRestriction  A datatype restriction DatatypeRestriction(DT F1 lt1 Fn ltn) consists of a unary datatype DT and n pairs  OWLFacet  Enumerations for OWL facets.  OWLObjectMinCardinality  A minimum cardinality expression ObjectMinCardinality(n OPE CE) consists of a nonnegative integer n, an object  OWLObjectMaxCardinality  A maximum cardinality expression ObjectMaxCardinality(n OPE CE) consists of a nonnegative integer n, an object  OWLObjectExactCardinality  An exact cardinality expression ObjectExactCardinality(n OPE CE) consists of a nonnegative integer n, an object  OWLObjectExactCardinality  An exact cardinality expression ObjectExactCardinality(n OPE CE) consists of a nonnegative integer n, an object  OWLDataSomeValuesFrom  An exact cardinality expression DataSomeValuesFrom(DPE1 DPEn DR) consists of n data property expressions  OWLDataAllValuesFrom  A universal class expression DataAllValuesFrom(DPE1 DPEn DR) consists of n data property expressions DPEi,  OWLDataHasValue  A has-value class expression DataHasValue(DPE lt) consists of a data property expression DPE and a literal lt,		An existential class expression ObjectSomeValuesFrom(OPE CE) consists of an object property expression OPE		
consists of an object property expression OPE and an OWLDatatypeRestriction A datatype restriction DatatypeRestriction (DT F1 lt1 Fn ltn ) consists of a unary datatype DT and n pairs  OWLFacet Enumerations for OWL facets.  OWLFacetRestriction A facet restriction is used to restrict a particular datatype.  OWLObjectMinCardinality A minimum cardinality expression ObjectMinCardinality (n OPE CE) consists of a nonnegative integer n, an object  OWLObjectMaxCardinality A maximum cardinality expression ObjectMaxCardinality (n OPE CE) consists of a nonnegative integer n, an object  OWLObjectExactCardinality An exact cardinality expression ObjectExactCardinality (n OPE CE) consists of a nonnegative integer n, an object  OWLDataSomeValuesFrom An existential class expression DataSomeValuesFrom (DPE1 DPEn DR) consists of n data property expressions  OWLDataAllValuesFrom A universal class expression DataAllValuesFrom (DPE1 DPEn DR) consists of n data property expressions DPEi,  OWLDataHasValue A has-value class expression DataHasValue (DPE lt) consists of a data property expressions DPE and a literal lt,	OWLObjectAllValuesFrom			
Fn ltn ) consists of a unary datatype DT and n pairs  OWLFacet Enumerations for OWL facets.  OWLObjectMinCardinality A minimum cardinality expression ObjectMinCardinality (n OPE CE) consists of a nonnegative integer n, an object  OWLObjectMaxCardinality A maximum cardinality expression ObjectMaxCardinality (n OPE CE) consists of a nonnegative integer n, an object  OWLObjectExactCardinality An exact cardinality expression ObjectExactCardinality (n OPE CE) consists of a nonnegative integer n, an object  OWLObjectExactCardinality An exact cardinality expression ObjectExactCardinality (n OPE CE) consists of a nonnegative integer n, an object  OWLDataSomeValuesFrom An existential class expression DataSomeValuesFrom (DPE1 DPEn DR) consists of n data property expressions  OWLDataAllValuesFrom A universal class expression DataAllValuesFrom (DPE1 DPEn DR) consists of n data property expressions DPEi,  OWLDataHasValue A has-value class expression DataHasValue (DPE lt) consists of a data property expression DPE and a literal lt,	OWLObjectHasValue	· · · · · · · · · · · · · · · · · · ·		
OWLFacetRestriction       A facet restriction is used to restrict a particular datatype.         OWLObjectMinCardinality       A minimum cardinality expression ObjectMinCardinality (n OPE CE) consists of a nonnegative integer n, an object         OWLObjectMaxCardinality       A maximum cardinality expression ObjectMaxCardinality (n OPE CE) consists of a nonnegative integer n, an object         OWLObjectExactCardinality       An exact cardinality expression ObjectExactCardinality (n OPE CE) consists of a nonnegative integer n, an object         OWLDataSomeValuesFrom       An existential class expression DataSomeValuesFrom(DPE1 DPEn DR) consists of n data property expressions         OWLDataAllValuesFrom       A universal class expression DataAllValuesFrom(DPE1 DPEn DR) consists of n data property expressions DPEi,         OWLDataHasValue       A has-value class expression DataHasValue(DPE lt) consists of a data property expression DPE and a literal lt,	OWLDatatypeRestriction			
OWLObjectMinCardinality  A minimum cardinality expression ObjectMinCardinality (n OPE CE) consists of a nonnegative integer n, an object  OWLObjectMaxCardinality  A maximum cardinality expression ObjectMaxCardinality (n OPE CE) consists of a nonnegative integer n, an object  OWLObjectExactCardinality  An exact cardinality expression ObjectExactCardinality (n OPE CE) consists of a nonnegative integer n, an object  OWLDataSomeValuesFrom  An existential class expression DataSomeValuesFrom( DPE1 DPEn DR) consists of n data property expressions  OWLDataAllValuesFrom  A universal class expression DataAllValuesFrom( DPE1 DPEn DR) consists of n data property expressions  DPEi,  OWLDataHasValue  A has-value class expression DataHasValue( DPE lt) consists of a data property expression DPE and a literal lt,				
ity( n OPE CE ) consists of a nonnegative integer n, an object  OWLObjectMaxCardinality  A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an object  OWLObjectExactCardinality  An exact cardinality expression ObjectExactCardinality( n OPE CE ) consists of a nonnegative integer n, an object  OWLDataSomeValuesFrom  An existential class expression DataSomeValuesFrom( DPE1 DPEn DR ) consists of n data property expressions  OWLDataAllValuesFrom  A universal class expression DataAllValuesFrom( DPE1 DPEn DR ) consists of n data property expressions DPEi,  OWLDataHasValue  A has-value class expression DataHasValue( DPE lt ) consists of a data property expression DPE and a literal lt,	OWLFacetRestriction			
ity( n OPE CE ) consists of a nonnegative integer n, an object  OWLObjectExactCardinality  An exact cardinality expression ObjectExactCardinality( n OPE CE) consists of a nonnegative integer n, an object  OWLDataSomeValuesFrom  An existential class expression DataSomeValuesFrom( DPE1 DPEn DR) consists of n data property expressions  OWLDataAllValuesFrom  A universal class expression DataAllValuesFrom( DPE1 DPEn DR) consists of n data property expressions  DPEi,  OWLDataHasValue  A has-value class expression DataHasValue( DPE lt) consists of a data property expression DPE and a literal lt,	OWLObjectMinCardinality	ity( n OPE CE ) consists of a nonnegative integer n, an		
n OPE CE ) consists of a nonnegative integer n, an object  An existential class expression DataSomeValuesFrom( DPE1 DPEn DR ) consists of n data property expressions  OWLDataAllValuesFrom A universal class expression DataAllValuesFrom( DPE1 DPEn DR ) consists of n data property expressions DPEi,  OWLDataHasValue A has-value class expression DataHasValue( DPE lt ) consists of a data property expression DPE and a literal lt,	OWLObjectMaxCardinality	ity( n OPE CE ) consists of a nonnegative integer n, an		
DPE1 DPEn DR ) consists of n data property expressions  OWLDataAllValuesFrom  A universal class expression DataAllValuesFrom( DPE1	OWLObjectExactCardinality			
DPEn DR ) consists of n data property expressions DPEi,  OWLDataHasValue A has-value class expression DataHasValue(DPE lt) consists of a data property expression DPE and a literal lt,	OWLDataSomeValuesFrom	DPE1 DPEn DR ) consists of n data property expressions		
sists of a data property expression DPE and a literal lt,	OWLDataAllValuesFrom	DPEn DR ) consists of n data property expressions DPEi,		
continues on poyt page	OWLDataHasValue	sists of a data property expression DPE and a literal lt,		

continues on next page

Table 2 - continued from previous page

OWLDataMinCardinality	A minimum cardinality expression DataMinCardinality(n DPE DR) consists of a nonnegative integer n, a data
OWLDataMaxCardinality	A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an object
OWLDataExactCardinality	An exact cardinality expression ObjectExactCardinality( n OPE CE) consists of a nonnegative integer n, an
OWLObjectOneOf	An enumeration of individuals ObjectOneOf( a1 an ) contains exactly the individuals ai with $1 \le i \le n$ .
OWLRDFVocabulary	Enumerations for OWL/RDF vocabulary.

# **Package Contents**

class owlapy.class\_expression.OWLClassExpression

Bases: owlapy.owl\_data\_ranges.OWLPropertyRange

OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the structural specification of OWL 2, class expressions are represented by ClassExpression. (https://www.w3.org/TR/owl2-syntax/#Class\_Expressions)

# $\verb"abstract is_owl_thing"() \to bool$

Determines if this expression is the built in class owl: Thing. This method does not determine if the class is equivalent to owl: Thing.

## Returns

Thing.

## Return type

True if this expression is owl

$$abstract is_owl_nothing() \rightarrow bool$$

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

# $\verb"abstract get_object_complement_of"\ () \ \to \mathit{OWLObjectComplementOf}$

Gets the object complement of this class expression.

## Returns

A class expression that is the complement of this class expression.

# $abstract get_nnf() \rightarrow OWLClassExpression$

Gets the negation normal form of the complement of this expression.

#### **Returns**

A expression that represents the NNF of the complement of this expression.

# class owlapy.class\_expression.OWLAnonymousClassExpression

Bases: OWLClassExpression

A Class Expression which is not a named Class.

```
is\_owl\_nothing() \rightarrow bool
```

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

```
is\_owl\_thing() \rightarrow bool
```

Determines if this expression is the built in class owl: Thing. This method does not determine if the class is equivalent to owl: Thing.

#### Returns

Thing.

# Return type

True if this expression is owl

```
\texttt{get\_object\_complement\_of}() \rightarrow OWLObjectComplementOf
```

Gets the object complement of this class expression.

## Returns

A class expression that is the complement of this class expression.

```
\mathtt{get\_nnf}() \to \mathit{OWLClassExpression}
```

Gets the negation normal form of the complement of this expression.

#### Returns

A expression that represents the NNF of the complement of this expression.

```
class owlapy.class_expression.OWLBooleanClassExpression
```

Bases: OWLAnonymousClassExpression

Represent an anonymous boolean class expression.

```
__slots__ = ()
```

```
class owlapy.class_expression.OWLObjectComplementOf(op: OWLClassExpression)
```

Bases: OWLBooleanClassExpression, owlapy.meta\_classes.

 ${\it HasOperands} [{\it OWLClassExpression}]$ 

Represents an ObjectComplementOf class expression in the OWL 2 Specification.

```
__slots__ = '_operand'
type_index: Final = 3003
get operand() → OWLClassExpression
```

## **Returns**

The wrapped expression.

```
operands() \rightarrow Iterable[OWLClassExpression]
```

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

#### Returns

The operands.

```
__repr__()

Return repr(self).
__eq__(other)

Return self==value.
```

```
__hash__()
```

Return hash(self).

```
class owlapy.class_expression.OWLClass(iri: owlapy.iri.IRI | str)
```

Bases: owlapy.class\_expression.class\_expression.OWLClassExpression, owlapy.owl\_object.OWLEntity

An OWL 2 named Class. Classes can be understood as sets of individuals. (https://www.w3.org/TR/owl2-syntax/#Classes)

```
__slots__ = ('_iri', '_is_nothing', '_is_thing')
```

type\_index: Final = 1001

property iri: owlapy.iri.IRI

Gets the IRI of this object.

#### Returns

The IRI of this object.

# property str

Gets the string representation of this object

#### Returns

The IRI as string

# property reminder: str

The reminder of the IRI

# $is\_owl\_thing() \rightarrow bool$

Determines if this expression is the built in class owl: Thing. This method does not determine if the class is equivalent to owl: Thing.

# Returns

Thing.

#### Return type

True if this expression is owl

# $is\_owl\_nothing() \rightarrow bool$

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

```
get_object_complement_of()
```

→ owlapy.class expression.class expression.OWLObjectComplementOf

Gets the object complement of this class expression.

## Returns

A class expression that is the complement of this class expression.

```
\mathtt{get\_nnf}() \to \mathit{OWLClass}
```

Gets the negation normal form of the complement of this expression.

#### Returns

A expression that represents the NNF of the complement of this expression.

 ${\bf class} \ \, {\bf owlapy.class\_expression.OWLNaryBooleanClassExpression} \, ($ 

 $operands:\ Iterable[owlapy.class\_expression.class\_expression.OWLClassExpression])$ 

```
owlapy.class_expression.class_expression.OWLBooleanClassExpression,
     owlapy.meta classes.HasOperands[owlapy.class expression.class expression.
     OWLClassExpression]
     OWLNaryBooleanClassExpression.
     __slots__ = ()
     operands () → Iterable[owlapy.class expression.class expression.OWLClassExpression]
          Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.
              Returns
                 The operands.
     __repr__()
          Return repr(self).
     __eq__(other)
          Return self==value.
     __hash__()
          Return hash(self).
class owlapy.class expression.OWLObjectUnionOf(
           operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
     Bases: OWLNaryBooleanClassExpression
     A union class expression ObjectUnionOf( CE1 ... CEn ) contains all individuals that are instances of at least one
     class expression CEi for 1 \le i \le n. (https://www.w3.org/TR/owl2-syntax/#Union_of_Class_Expressions)
     __slots__ = '_operands'
     type_index: Final = 3002
class owlapy.class_expression.OWLObjectIntersectionOf(
           operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
     Bases: OWLNaryBooleanClassExpression
     An intersection class expression ObjectIntersectionOf( CE1 ... CEn ) contains all individuals that are instances of
     all class expressions CEi for 1 \le i \le n. (https://www.w3.org/TR/owl2-syntax/#Intersection_of_Class_Expressions)
     __slots__ = '_operands'
     type_index: Final = 3001
class owlapy.class_expression.OWLRestriction
     Bases: owlapy.class_expression.class_expression.OWLAnonymousClassExpression
     Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
     __slots__ = ()
     abstract get_property() → owlapy.owl_property.OWLPropertyExpression
                 Property being restricted.
```

```
is_data_restriction() \rightarrow bool
```

Determines if this is a data restriction.

## Returns

True if this is a data restriction.

# $is\_object\_restriction() \rightarrow bool$

Determines if this is an object restriction.

#### Returns

True if this is an object restriction.

# class owlapy.class\_expression.OWLQuantifiedRestriction

Bases: Generic[\_T], OWLRestriction, owlapy.meta\_classes.HasFiller[\_T]

Represents a quantified restriction.

## **Parameters**

**\_T** – value type

# class owlapy.class expression.OWLQuantifiedObjectRestriction(

filler: owlapy.class\_expression.class\_expression.OWLClassExpression)

Bases:  $OWLQuantifiedRestriction[owlapy.class\_expression.class\_expression. OWLClassExpression], OWLObjectRestriction$ 

Represents a quantified object restriction.

$$\texttt{get\_filler}() \rightarrow owlapy.class\_expression.class\_expression.OWLClassExpression$$

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a class expression or a data range.

# Returns

the value

# class owlapy.class\_expression.OWLObjectRestriction

Bases: OWLRestriction

Represents an Object Property Restriction in the OWL 2 specification.

# $is\_object\_restriction() \rightarrow bool$

Determines if this is an object restriction.

#### Returns

True if this is an object restriction.

**abstract get\_property**() → *owlapy.owl\_property.OWLObjectPropertyExpression* 

## Returns

Property being restricted.

Bases: Generic[\_T], OWLRestriction, owlapy.meta\_classes.HasFiller[\_T]

Represent a HasValue restriction in the OWL 2

## **Parameters**

 $_{\mathbf{T}}$  – The value type.

**\_\_eq\_** (*other*)

Return self==value.

Return hash(self).

$$\mathtt{get\_filler}() \rightarrow \_T$$

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a class expression or a data range.

## Returns

the value

class owlapy.class\_expression.OWLDataRestriction

Bases: OWLRestriction

Represents a Data Property Restriction.

$$is\_data\_restriction() \rightarrow bool$$

Determines if this is a data restriction.

#### Returns

True if this is a data restriction.

**class** owlapy.class\_expression.**OWLCardinalityRestriction**(cardinality: int, filler: \_F)

```
Bases: Generic[\_F], OWLQuantifiedRestriction[\_F], owlapy.meta_classes. HasCardinality
```

Base interface for owl min and max cardinality restriction.

# **Parameters**

**\_F** – Type of filler.

$$\texttt{get\_cardinality}\,(\,)\,\to int$$

Gets the cardinality of a restriction.

#### Returns

The cardinality. A non-negative integer.

$$\texttt{get\_filler}\,(\,) \,\to \_F$$

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a class expression or a data range.

# Returns

the value

```
OWLCardinalityRestriction[owlapy.class_expression.class_expression.
     OWLClassExpression], OWLQuantifiedObjectRestriction
     Represents Object Property Cardinality Restrictions in the OWL 2 specification.
     __slots__ = ()
     get_property() → owlapy.owl_property.OWLObjectPropertyExpression
              Returns
                  Property being restricted.
      __repr__()
          Return repr(self).
      \underline{\phantom{a}}eq\underline{\phantom{a}} (other)
          Return self==value.
      __hash__()
          Return hash(self).
class owlapy.class_expression.OWLObjectHasSelf(
           property: owlapy.owl property.OWLObjectPropertyExpression)
     Bases: OWLObjectRestriction
     A self-restriction ObjectHasSelf( OPE ) consists of an object property expression OPE, and it contains all those
     individuals that are connected by OPE to themselves. (https://www.w3.org/TR/owl2-syntax/#Self-Restriction)
     __slots__ = '_property'
     type_index: Final = 3011
     get property() → owlapy.owl property.OWLObjectPropertyExpression
              Returns
                  Property being restricted.
     ___eq__(other)
          Return self==value.
      __hash__()
          Return hash(self).
     __repr__()
          Return repr(self).
class owlapy.class expression.OWLDataOneOf(
            values: owlapy.owl literal.OWLLiteral | Iterable[owlapy.owl literal.OWLLiteral])
                       owlapy.owl data ranges.OWLDataRange,
     Bases:
                                                                             owlapy.meta classes.
     HasOperands[owlapy.owl literal.OWLLiteral]
     An enumeration of literals DataOneOf(lt1...ltn) contains exactly the explicitly specified literals lti with 1 \le i \le
     n. The resulting data range has arity one. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Literals)
     type_index: Final = 4003
```

Bases:

```
Gets the values that are in the oneOf.
               Returns
                  The values of this {@code DataOneOf} class expression.
     operands() \rightarrow Iterable[owlapy.owl\_literal.OWLLiteral]
          Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.
               Returns
                  The operands.
     __hash__()
          Return hash(self).
     __eq__(other)
          Return self==value.
     __repr__()
          Return repr(self).
class owlapy.class_expression.OWLQuantifiedDataRestriction(
            filler: owlapy.owl_data_ranges.OWLDataRange)
               OWLQuantifiedRestriction[owlapy.owl_data_ranges.OWLDataRange], OWL-
     Bases:
     DataRestriction
     Represents a quantified data restriction.
     __slots__ = ()
     get filler() → owlapy.owl data ranges.OWLDataRange
          Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of
          a data restriction this will be a constant (data value). For quantified restriction this will be a class expression
          or a data range.
              Returns
                  the value
class owlapy.class_expression.OWLDataCardinalityRestriction (cardinality: int,
            property: owlapy.owl_property.OWLDataPropertyExpression,
            filler: owlapy.owl_data_ranges.OWLDataRange)
                       OWLCardinalityRestriction[owlapy.owl_data_ranges.OWLDataRange],
     OWLQuantifiedDataRestriction, OWLDataRestriction
     Represents Data Property Cardinality Restrictions.
     __slots__ = ()
     \texttt{get\_property}() \rightarrow owlapy.owl\_property.OWLDataPropertyExpression
               Returns
                  Property being restricted.
     __repr__()
          Return repr(self).
     ___eq__ (other)
          Return self==value.
```

 $values() \rightarrow Iterable[owlapy.owl\_literal.OWLLiteral]$ 

```
__hash__()
          Return hash(self).
class owlapy.class_expression.OWLObjectSomeValuesFrom(
            property: owlapy.owl property.OWLObjectPropertyExpression,
            filler: owlapy.class_expression.class_expression.OWLClassExpression)
     Bases: OWLQuantifiedObjectRestriction
     An existential class expression ObjectSomeValuesFrom(OPE CE) consists of an object property expression OPE
     and a class expression CE, and it contains all those individuals that are connected by OPE to an individual that is
     an instance of CE.
     __slots__ = ('_property', '_filler')
     type_index: Final = 3005
     __repr__()
          Return repr(self).
     __eq__(other)
          Return self==value.
     __hash__()
          Return hash(self).
     get_property() → owlapy.owl_property.OWLObjectPropertyExpression
              Returns
                  Property being restricted.
class owlapy.class_expression.OWLObjectAllValuesFrom(
            property: owlapy.owl_property.OWLObjectPropertyExpression,
            filler: owlapy.class expression.class expression.OWLClassExpression)
     Bases: OWLQuantifiedObjectRestriction
     A universal class expression ObjectAllValuesFrom( OPE CE ) consists of an object property expression OPE and
     a class expression CE, and it contains all those individuals that are connected by OPE only to individuals that are
     instances of CE. (https://www.w3.org/TR/owl2-syntax/#Universal_Quantification)
     __slots__ = ('_property', '_filler')
     type_index: Final = 3006
     __repr__()
          Return repr(self).
     __eq__(other)
          Return self==value.
     __hash__()
          Return hash(self).
     \texttt{get\_property}() \rightarrow owlapy.owl\_property.OWLObjectPropertyExpression
```

## Returns

Property being restricted.

```
class owlapy.class expression.OWLObjectHasValue(
            property: owlapy.owl_property.OWLObjectPropertyExpression,
            individual: owlapy.owl individual.OWLIndividual)
     Bases: OWLHasValueRestriction[owlapy.owl_individual.OWLIndividual], OWLObjec-
      tRestriction
     A has-value class expression ObjectHasValue( OPE a ) consists of an object property expression OPE and an
     individual a, and it contains all those individuals that are connected by OPE to a. Each such class expression
     can be seen as a syntactic shortcut for the class expression ObjectSomeValuesFrom( OPE ObjectOneOf( a ) ).
     (https://www.w3.org/TR/owl2-syntax/#Individual_Value_Restriction)
      __slots__ = ('_property', '_v')
     type_index: Final = 3007
     get_property() → owlapy.owl_property.OWLObjectPropertyExpression
               Returns
                   Property being restricted.
     as\_some\_values\_from() \rightarrow owlapy.class\_expression.class\_expression.OWLClassExpression
          A convenience method that obtains this restriction as an existential restriction with a nominal filler.
               Returns
                   The existential equivalent of this value restriction. simp(HasValue(p a)) = some(p \{a\}).
     __repr__()
          Return repr(self).
class owlapy.class_expression.OWLDatatypeRestriction(
            type: owlapy.owl datatype.OWLDatatype,
            facet restrictions: OWLFacetRestriction | Iterable[OWLFacetRestriction])
     Bases: owlapy.owl_data_ranges.OWLDataRange
     A datatype restriction DatatypeRestriction (DT F1 lt1 ... Fn ltn ) consists of a unary datatype DT and n pairs (
     Fi, lti). The resulting data range is unary and is obtained by restricting the value space of DT according to the
     semantics of all (Fi, vi) (multiple pairs are interpreted conjunctively), where vi are the data values of the literals
     lti. (https://www.w3.org/TR/owl2-syntax/#Datatype_Restrictions)
     __slots__ = ('_type', '_facet_restrictions')
     type_index: Final = 4006
     get_datatype() → owlapy.owl_datatype.OWLDatatype
     \texttt{get\_facet\_restrictions}() \rightarrow \texttt{Sequence}[\textit{OWLFacetRestriction}]
     __eq_ (other)
          Return self==value.
      __hash___()
          Return hash(self).
       _repr__()
          Return repr(self).
class owlapy.class_expression.OWLFacet (remainder: str, symbolic_form: str,
            operator: Callable[[_X, _X], bool])
     Bases: _Vocabulary, enum. Enum
     Enumerations for OWL facets.
```

```
property symbolic_form
     property operator
     static from_str(name: str) → OWLFacet
     MIN_INCLUSIVE: Final
     MIN_EXCLUSIVE: Final
     MAX_INCLUSIVE: Final
     MAX_EXCLUSIVE: Final
     LENGTH: Final
     MIN_LENGTH: Final
     MAX LENGTH: Final
     PATTERN: Final
     TOTAL_DIGITS: Final
     FRACTION_DIGITS: Final
class owlapy.class_expression.OWLFacetRestriction (facet: owlapy.vocab.OWLFacet,
           literal: Literals)
     Bases: owlapy.owl_object.OWLObject
     A facet restriction is used to restrict a particular datatype.
     __slots__ = ('_facet', '_literal')
     type_index: Final = 4007
     get_facet() → owlapy.vocab.OWLFacet
     get facet value() → owlapy.owl literal.OWLLiteral
     __eq_ (other)
         Return self==value.
     __hash__()
         Return hash(self).
     __repr__()
         Return repr(self).
class owlapy.class_expression.OWLObjectMinCardinality (cardinality: int,
           property: owlapy.owl_property.OWLObjectPropertyExpression,
           filler: owlapy.class_expression.class_expression.OWLClassExpression)
     Bases: OWLObjectCardinalityRestriction
     A minimum cardinality expression ObjectMinCardinality (n OPE CE) consists of a nonnegative integer n, an object
     property expression OPE, and a class expression CE, and it contains all those individuals that are connected by
     OPE to at least n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Minimum_
     __slots__ = ('_cardinality', '_filler', '_property')
```

```
type_index: Final = 3008
```

class owlapy.class\_expression.OWLObjectMaxCardinality(cardinality: int,

 $property: owlapy.owl\_property.OWLObjectPropertyExpression,$ 

filler: owlapy.class\_expression.class\_expression.OWLClassExpression)

Bases: OWLObjectCardinalityRestriction

A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE

to at most n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Maximum\_Cardinality)

```
__slots__ = ('_cardinality', '_filler', '_property')
type_index: Final = 3010
```

class owlapy.class\_expression.OWLObjectExactCardinality (cardinality: int,

property: owlapy.owl\_property.OWLObjectPropertyExpression, filler: owlapy.class\_expression.class\_expression.OWLClassExpression)

Bases: OWLObjectCardinalityRestriction

# An exact cardinality expression ObjectExactCardinality( n OPE CE ) consists of a nonnegative integer n, an object

property expression OPE, and a class expression CE, and it contains all those individuals that are connected by to exactly n different individuals that are instances of CE.

(https://www.w3.org/TR/owl2-syntax/#Exact\_Cardinality)

```
__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3009

as_intersection_of_min_max()
```

 $\rightarrow owlapy.class\_expression.nary\_boolean\_expression.OWLObjectIntersectionOf$ 

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

## Returns

The semantically equivalent but structurally simpler form (= 1 R C) = >= 1 R C and <= 1 R C.

 $\textbf{Bases:} \ \textit{OWLQuantifiedDataRestriction}$ 

An existential class expression DataSomeValuesFrom( DPE1 ... DPEn DR ) consists of n data property expressions DPEi,  $1 \le i \le n$ , and a data range DR whose arity must be n. Such a class expression contains all those individuals that are connected by DPEi to literals lti,  $1 \le i \le n$ , such that the tuple ( lt1 , ..., ltn ) is in DR. A class expression of the form DataSomeValuesFrom( DPE DR ) can be seen as a syntactic shortcut for the class expression DataMinCardinality( 1 DPE DR ). (https://www.w3.org/TR/owl2-syntax/#Existential\_Quantification\_2)

```
__slots__ = '_property'

type_index: Final = 3012

__repr__()

Return repr(self).
```

```
\underline{\phantom{a}}eq\underline{\phantom{a}} (other)
           Return self==value.
     __hash__()
           Return hash(self).
     get property() → owlapy.owl property.OWLDataPropertyExpression
               Returns
                   Property being restricted.
class owlapy.class_expression.OWLDataAllValuesFrom(
            property: owlapy.owl_property.OWLDataPropertyExpression,
            filler: owlapy.owl_data_ranges.OWLDataRange)
     Bases: OWLQuantifiedDataRestriction
     A universal class expression DataAllValuesFrom( DPE1 ... DPEn DR ) consists of n data property expressions
     DPEi, 1 \le i \le n, and a data range DR whose arity must be n. Such a class expression contains all those individuals
     that
           are connected by DPEi only to literals lti, 1 \le i \le n, such that each tuple (lt1, ..., ltn) is in DR.
           A class
               expression of the form DataAllValuesFrom( DPE DR ) can be seen as a syntactic shortcut for the
               class expression DataMaxCardinality( 0 DPE DataComplementOf( DR ) ). (https://www.w3.org/
               TR/owl2-syntax/#Universal Quantification 2)
      __slots__ = '_property'
     type_index: Final = 3013
      __repr__()
           Return repr(self).
       \underline{\phantom{a}}eq\underline{\phantom{a}} (other)
           Return self==value.
     __hash__()
           Return hash(self).
     get_property() → owlapy.owl_property.OWLDataPropertyExpression
               Returns
                   Property being restricted.
class owlapy.class expression.OWLDataHasValue(
            property: owlapy.owl property.OWLDataPropertyExpression,
            value: owlapy.owl_literal.OWLLiteral)
     Bases: OWLHasValueRestriction[owlapy.owl literal.OWLLiteral], OWLDataRestric-
      tion
     A has-value class expression DataHasValue(DPE lt) consists of a data property expression DPE and a literal lt,
     and it contains all those individuals that are connected by DPE to lt. Each such class expression can be seen as a
     syntactic shortcut for the class expression DataSomeValuesFrom( DPE DataOneOf( lt ) ). (https://www.w3.org/
     TR/owl2-syntax/#Literal_Value_Restriction)
     __slots__ = '_property'
     type_index: Final = 3014
```

```
__repr__()
          Return repr(self).
     __eq_ (other)
          Return self==value.
     hash ()
          Return hash(self).
     as some values from () \rightarrow owlapy.class expression.class expression.OWLClassExpression
          A convenience method that obtains this restriction as an existential restriction with a nominal filler.
              Returns
                  The existential equivalent of this value restriction. simp(HasValue(p a)) = some(p \{a\}).
     get_property() → owlapy.owl_property.OWLDataPropertyExpression
              Returns
                  Property being restricted.
class owlapy.class expression.OWLDataMinCardinality(cardinality: int,
            property: owlapy.owl_property.OWLDataPropertyExpression,
            filler: owlapy.owl data ranges.OWLDataRange)
     Bases: OWLDataCardinalityRestriction
     A minimum cardinality expression DataMinCardinality( n DPE DR ) consists of a nonnegative integer n, a data
     property expression DPE, and a unary data range DR, and it contains all those individuals that are connected by
     DPE to at least n different literals in DR. (https://www.w3.org/TR/owl2-syntax/#Minimum Cardinality)
     __slots__ = ('_cardinality', '_filler', '_property')
     type index: Final = 3015
class owlapy.class_expression.OWLDataMaxCardinality (cardinality: int,
            property: owlapy.owl property.OWLDataPropertyExpression,
            filler: owlapy.owl_data_ranges.OWLDataRange)
     Bases: OWLDataCardinalityRestriction
     A maximum cardinality expression ObjectMaxCardinality (n OPE CE) consists of a nonnegative integer n, an
     object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by
     OPE to at most n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Maximum_
     Cardinality)
     __slots__ = ('_cardinality', '_filler', '_property')
     type_index: Final = 3017
class owlapy.class_expression.OWLDataExactCardinality(cardinality: int,
            property: owlapy.owl_property.OWLDataPropertyExpression,
            filler: owlapy.owl data ranges.OWLDataRange)
     Bases: OWLDataCardinalityRestriction
     An exact cardinality expression ObjectExactCardinality( n OPE CE ) consists of a nonnegative integer n, an object
     property expression OPE, and a class expression CE, and it contains all those individuals that are connected
          by OPE to exactly n different individuals that are instances of CE (https://www.w3.org/TR/owl2-syntax/
          #Exact Cardinality)
     __slots__ = ('_cardinality', '_filler', '_property')
```

```
type_index: Final = 3016
     as_intersection_of_min_max()
                   → owlapy.class expression.nary boolean expression.OWLObjectIntersectionOf
           Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.
               Returns
                   The semantically equivalent but structurally simpler form (= 1 R D) = >= 1 R D and <= 1 R D.
class owlapy.class expression.OWLObjectOneOf(
            values: owlapy.owl_individual.OWLIndividual \ Iterable[owlapy.owl_individual.OWLIndividual])
     Bases: owlapy.class expression.class expression.OWLAnonymousClassExpression,
     owlapy.meta_classes.HasOperands[owlapy.owl_individual.OWLIndividual]
     An enumeration of individuals ObjectOneOf( a1 ... an ) contains exactly the individuals ai with 1 \le i \le n. (https:
     //www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals)
     __slots__ = '_values'
     type_index: Final = 3004
     individuals() \rightarrow Iterable[owlapy.owl\_individual.OWLIndividual]
           Gets the individuals that are in the oneOf. These individuals represent the exact instances (extension) of this
           class expression.
               Returns
                   The individuals that are the values of this {@code ObjectOneOf} class expression.
     operands () → Iterable[owlapy.owl_individual.OWLIndividual]
           Gets the operands - e.g., the individuals in a same As axiom, or the classes in an equivalent classes axiom.
               Returns
                   The operands.
     as_object\_union\_of() \rightarrow owlapy.class\_expression.class\_expression.OWLClassExpression
           Simplifies this enumeration to a union of singleton nominals.
               Returns
                   This enumeration in a more standard DL form. simp({a}) = {a} simp({a0, ..., {an}}) =
                   unionOf(\{a0\}, \ldots, \{an\})
        _hash___()
           Return hash(self).
      \underline{\phantom{a}}eq\underline{\phantom{a}} (other)
           Return self==value.
        repr__()
           Return repr(self).
class owlapy.class expression.OWLRDFVocabulary(
            namespace: owlapy.namespaces.Namespaces, remainder: str)
     Bases: _Vocabulary, enum. Enum
     Enumerations for OWL/RDF vocabulary.
     OWL_THING
     OWL_NOTHING
```

```
OWL_CLASS
OWL_NAMED_INDIVIDUAL
OWL_TOP_OBJECT_PROPERTY
OWL_BOTTOM_OBJECT_PROPERTY
OWL_TOP_DATA_PROPERTY
OWL_BOTTOM_DATA_PROPERTY
RDFS_LITERAL
```

owlapy.class\_expression.OWLThing: Final

owlapy.class\_expression.OWLNothing: Final

# owlapy.entities

Entities are the fundamental building blocks of OWL 2 ontologies, and they define the vocabulary — the named terms — of an ontology. In logic, the set of entities is usually said to constitute the signature of an ontology.

Classes, datatypes, object properties, data properties, annotation properties, and named individuals are entities, and they are all uniquely identified by an IR.

# 7.2 Submodules

# owlapy.converter

Format converter.

# **Attributes**

converter

# **Classes**

VariablesMapping	Helper class for owl-to-sparql conversion.
Owl2SparqlConverter	Convert owl (owlapy model class expressions) to SPARQL.

# **Functions**

peek(x)	Peek the last element of an array.
owl_expression_to_sparql(→str)	Convert an OWL Class Expression (https://www.w3.org/TR/owl2-syntax/#Class_Expressions) into a SPARQL
	query

# **Module Contents**

```
owlapy.converter.peek (x)
    Peek the last element of an array.
        Returns
            The last element arr[-1].
class owlapy.converter.VariablesMapping
    Helper class for owl-to-sparql conversion.
    __slots__ = ('class_cnt', 'prop_cnt', 'ind_cnt', 'dict')
    get_variable (e: owlapy.owl_object.OWLEntity) → str
    {\tt new\_individual\_variable}\,(\,)\,\to str
    {\tt new\_property\_variable}\,(\,)\,\to str
    __contains__ (item: owlapy.owl_object.OWLEntity) → bool
    getitem (item: owlapy.owl object.OWLEntity) → str
class owlapy.converter.Owl2SparqlConverter
    Convert owl (owlapy model class expressions) to SPARQL.
      _slots__ = ('ce', 'sparql', 'variables', 'parent', 'parent_var',
    'properties', 'variable_entities', 'cnt',...
    ce: owlapy.class_expression.OWLClassExpression
    sparql: List[str]
    variables: List[str]
    parent: List[owlapy.class_expression.OWLClassExpression]
    parent_var: List[str]
    variable_entities: Set[owlapy.owl_object.OWLEntity]
    properties: Dict[int, List[owlapy.owl_object.OWLEntity]]
    mapping: VariablesMapping
    grouping_vars: Dict[owlapy.class_expression.OWLClassExpression, Set[str]]
    having conditions: Dict[owlapy.class_expression.OWLClassExpression,
    Set[str]]
```

```
cnt: int
     for_all_de_morgan: bool
     named_individuals: bool
     convert (root_variable: str, ce: owlapy.class_expression.OWLClassExpression,
                 for_all_de_morgan: bool = True, named_individuals: bool = False)
          Used to convert owl class expression to SPARQL syntax.
              Parameters
                   • root_variable (str) - Root variable name that will be used in SPARQL query.
                   • ce (OWLClassExpression) – The owl class expression to convert.
                   • named individuals (bool) - If 'True' return only entities that are instances of
                    owl:NamedIndividual.
              Returns
                  The SPARQL query.
              Return type
                  list[str]
     property modal_depth
     abstract render (e)
     stack variable(var)
     stack_parent (parent: owlapy.class_expression.OWLClassExpression)
     property current_variable
     abstract process(ce: owlapy.class_expression.OWLClassExpression)
     forAll (ce: owlapy.class_expression.OWLObjectAllValuesFrom)
     forAllDeMorgan (ce: owlapy.class_expression.OWLObjectAllValuesFrom)
     new\_count\_var() \rightarrow str
     append_triple (subject, predicate, object_)
     append (frag)
     triple (subject, predicate, object_)
     as query (root variable: str, ce: owlapy.class expression.OWLClassExpression,
                 for_all_de_morgan: bool = True, count: bool = False,
                 values: Iterable[owlapy.owl individual.OWLNamedIndividual] | None = None,
                 named\_individuals: bool = False) \rightarrow str
owlapy.converter.converter
owlapy.converter.owl expression to sparql(
            expression: owlapy.class\_expression.OWLClassExpression = None, root\_variable: str = '?x',
            values: Iterable[owlapy.owl individual.OWLNamedIndividual] | None = None,
```

 $for\_all\_de\_morgan: bool = True, named\_individuals: bool = False) \rightarrow str$ 

Convert an OWL Class Expression (https://www.w3.org/TR/owl2-syntax/#Class\_Expressions) into a SPARQL query root variable: the variable that will be projected expression: the class expression to be transformed to a SPARQL query

values: positive or negative examples from a class expression problem. Unclear for\_all\_de\_morgan: if set to True, the SPARQL mapping will use the mapping containing the nested FILTER NOT EXISTS patterns for the universal quantifier  $(\neg(\exists r.\neg C))$ , instead of the counting query named\_individuals: if set to True, the generated SPARQL query will return only entities that are instances of owl:NamedIndividual

# owlapy.iri

**OWL IRI** 

# **Classes**

IRI

An IRI, consisting of a namespace and a remainder.

# **Module Contents**

```
class owlapy.iri.IRI (namespace: str | owlapy.namespaces.Namespaces, remainder: str)
               owlapy.owl_annotation.OWLAnnotationSubject, owlapy.owl_annotation.
     OWLAnnotationValue
     An IRI, consisting of a namespace and a remainder.
     __slots__ = ('_namespace', '_remainder', '__weakref__')
     type index: Final = 0
     static create (namespace: owlapy.namespaces, Namespaces, remainder: str) \rightarrow IRI
     	extbf{static} create (namespace: str, remainder: str) 	o IRI
     static create (string: str) \rightarrow IRI
     __repr__()
          Return repr(self).
     __eq_ (other)
          Return self==value.
     __hash__()
          Return hash(self).
     is_nothing()
          Determines if this IRI is equal to the IRI that owl: Nothing is named with.
```

# Returns

True if this IRI is equal to <a href="http://www.w3.org/2002/07/owl#Nothing">http://www.w3.org/2002/07/owl#Nothing</a> and otherwise False.

# is\_thing()

Determines if this IRI is equal to the IRI that owl: Thing is named with.

## Returns

True if this IRI is equal to <a href="http://www.w3.org/2002/07/owl#Thing">http://www.w3.org/2002/07/owl#Thing</a> and otherwise False.

# $is\_reserved\_vocabulary() \rightarrow bool$

Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/2000/01/rdf-schema#</a> or <a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2002/07/owl#</a>.

## **Returns**

True if the IRI is in the reserved vocabulary, otherwise False.

```
as\_iri() \rightarrow IRI
```

## Returns

if the value is an IRI, return it. Return Mone otherwise.

```
as\_str() \rightarrow str
```

CD: Should be deprecated. :returns: The string that specifies the IRI.

```
property str: str
```

Returns: The string that specifies the IRI.

```
property reminder: str
```

Returns: The string corresponding to the reminder of the IRI.

```
\texttt{get\_short\_form}() \rightarrow str
```

Gets the short form.

# Returns

A string that represents the short form.

```
\texttt{get}_{\texttt{namespace}}() \rightarrow \mathsf{str}
```

## Returns

The namespace as string.

```
\texttt{get\_remainder}() \rightarrow str
```

# Returns

The remainder (coincident with NCName usually) for this IRI.

# owlapy.meta\_classes

Meta classes for OWL objects.

# **Classes**

HasIRI	Simple class to access the IRI.
HasOperands	An interface to objects that have a collection of operands.
HasFiller	An interface to objects that have a filler.
HasCardinality	An interface to objects that have a cardinality.

# **Module Contents**

class owlapy.meta\_classes.HasIRI

Simple class to access the IRI.

\_\_slots\_\_ = ()

property iri: IRI

Abstractmethod

Gets the IRI of this object.

**Returns** 

The IRI of this object.

property str: str

Abstractmethod

Gets the string representation of this object

**Returns** 

The IRI as string

class owlapy.meta\_classes.HasOperands

Bases: Generic[\_T]

An interface to objects that have a collection of operands.

**Parameters** 

**\_T** – Operand type.

\_\_slots\_\_ = ()

 $\textbf{abstract operands} \hspace{0.1cm} \textbf{()} \hspace{0.1cm} \rightarrow Iterable[\_T]$ 

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

Returns

The operands.

class owlapy.meta\_classes.HasFiller

Bases: Generic[\_T]

An interface to objects that have a filler.

**Parameters** 

**\_T** – Filler type.

\_\_slots\_\_ = ()

```
\textbf{abstract get\_filler}\,(\,)\,\to \_T
```

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a class expression or a data range.

## **Returns**

the value

class owlapy.meta\_classes.HasCardinality

An interface to objects that have a cardinality.

```
__slots__ = ()
```

 $\textbf{abstract get\_cardinality()} \rightarrow int$ 

Gets the cardinality of a restriction.

## Returns

The cardinality. A non-negative integer.

# owlapy.namespaces

Namespaces.

# **Attributes**

OWL			
RDFS			
RDF			
XSD			

# Classes

Namespaces	Namespaces provide a simple method for qualifying ele-
	ment and attribute names used in Extensible Markup

# **Module Contents**

```
class owlapy.namespaces.Namespaces(prefix: str, ns: str)
```

Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references

```
__slots__ = ('_prefix', '_ns')
property ns: str
```

```
property prefix: str
    __repr__()
        Return repr(self).
    __hash__()
        Return hash(self).
    __eq__(other)
        Return self==value.

owlapy.namespaces.OWL: Final
owlapy.namespaces.RDFS: Final
owlapy.namespaces.RDF: Final
```

# owlapy.owl\_annotation

**OWL** Annotations

## **Classes**

OWLAnnotationObject	A marker interface for the values (objects) of annotations.
OWLAnnotationSubject	A marker interface for annotation subjects, which can either be IRIs or anonymous individuals
OWLAnnotationValue	A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.

# **Module Contents**

if the value is an anonymous, return it. Return None otherwise.

 ${\bf class} \ {\tt owlapy.owl\_annotation.OWLAnnotationSubject}$ 

Bases: OWLAnnotationObject

A marker interface for annotation subjects, which can either be IRIs or anonymous individuals

class owlapy.owl\_annotation.OWLAnnotationValue

Bases: OWLAnnotationObject

A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.

$$\textbf{is\_literal} \, (\,) \, \to bool$$

# Returns

true if the annotation value is a literal

$$as\_literal() \rightarrow \mathit{OWLLiteral} \mid None$$

# Returns

if the value is a literal, returns it. Return None otherwise

# owlapy.owl\_axiom

**OWL** Axioms

# **Classes**

OWLAxiom	Represents Axioms in the OWL 2 Specification.
OWLLogicalAxiom	A base interface of all axioms that affect the logical mean-
	ing of an ontology. This excludes declaration
OWLPropertyAxiom	The base interface for property axioms.
OWLObjectPropertyAxiom	The base interface for object property axioms.
OWLDataPropertyAxiom	The base interface for data property axioms.
OWLIndividualAxiom	The base interface for individual axioms.
OWLClassAxiom	The base interface for class axioms.
OWLDeclarationAxiom	Represents a Declaration axiom in the OWL 2 Specifica-
	tion. A declaration axiom declares an entity in an ontol-
	ogy.
OWLDatatypeDefinitionAxiom	A datatype definition DatatypeDefinition( DT DR ) de-
	fines a new datatype DT as being semantically
OWLHasKeyAxiom	A key axiom HasKey( CE ( OPE1 OPEm ) ( DPE1
	DPEn ) ) states that each
OWLNaryAxiom	Represents an axiom that contains two or more operands
	that could also be represented with multiple pairwise
OWLNaryClassAxiom	Represents an axiom that contains two or more operands
	that could also be represented with
OWLEquivalentClassesAxiom	An equivalent classes axiom EquivalentClasses( CE1
	CEn ) states that all of the class expressions CEi,
OWLDisjointClassesAxiom	A disjoint classes axiom DisjointClasses( CE1 CEn )
	states that all of the class expressions CEi, $1 \le i \le n$ ,
OWLNaryIndividualAxiom	Represents an axiom that contains two or more operands
	that could also be represented with
OWLDifferentIndividualsAxiom	An individual inequality axiom DifferentIndividuals( a1
	an ) states that all of the individuals ai,

continues on next page

Table 3 - continued from previous page

Table 3 – continued from previous page			
OWLSameIndividualAxiom	An individual equality axiom SameIndividual( a1 an ) states that all of the individuals ai, $1 \le i \le n$ ,		
OWLNaryPropertyAxiom	Represents an axiom that contains two or more operands that could also be represented with		
OWLEquivalentObjectPropertiesAxiom	An equivalent object properties axiom EquivalentObject-Properties( OPE1 OPEn ) states that all of the object		
OWLDisjointObjectPropertiesAxiom	A disjoint object properties axiom DisjointObjectProperties(OPE1 OPEn) states that all of the object		
OWLInverseObjectPropertiesAxiom	An inverse object properties axiom InverseObjectProperties( OPE1 OPE2 ) states that the object property		
OWLEquivalentDataPropertiesAxiom	An equivalent data properties axiom EquivalentDataProperties( DPE1 DPEn ) states that all the data property		
OWLDisjointDataPropertiesAxiom	A disjoint data properties axiom DisjointDataProperties(DPE1 DPEn) states that all of the data property		
OWLSubClassOfAxiom	A subclass axiom SubClassOf( CE1 CE2 ) states that the class expression CE1 is a subclass of the class		
OWLDisjointUnionAxiom	A disjoint union axiom DisjointUnion( C CE1 CEn ) states that a class C is a disjoint union of the class		
OWLClassAssertionAxiom	A class assertion ClassAssertion( CE a ) states that the individual a is an instance of the class expression CE.		
OWLAnnotationProperty	Represents an AnnotationProperty in the OWL 2 specification.		
OWLAnnotation	Annotations are used in the various types of annotation axioms, which bind annotations to their subjects		
OWLAnnotationAxiom	A super interface for annotation axioms.		
OWLAnnotationAssertionAxiom	An annotation assertion AnnotationAssertion( AP as av ) states that the annotation subject as — an IRI or an		
OWLSubAnnotationPropertyOfAxiom	An annotation subproperty axiom SubAnnotationPropertyOf( AP1 AP2 ) states that the annotation property AP1 is		
OWLAnnotationPropertyDomainAxiom	An annotation property domain axiom AnnotationPropertyDomain( AP U ) states that the domain of the annotation		
OWLAnnotationPropertyRangeAxiom	An annotation property range axiom AnnotationPropertyRange( AP U )		
OWLSubPropertyAxiom	Base interface for object and data sub-property axioms.		
OWLSubObjectPropertyOfAxiom	Object subproperty axioms are analogous to subclass axioms, and they come in two forms.		
OWLSubDataPropertyOfAxiom	A data subproperty axiom SubDataPropertyOf( DPE1 DPE2 ) states that the data property expression DPE1 is a		
OWLPropertyAssertionAxiom	Base class for Property Assertion axioms.		
OWLObjectPropertyAssertionAxiom	A positive object property assertion ObjectPropertyAssertion(OPE a1 a2) states that the individual a1 is		
OWLNegativeObjectPropertyAssertionAx- iom	A negative object property assertion NegativeObject-PropertyAssertion( OPE a1 a2 ) states that the individual a1		
OWLDataPropertyAssertionAxiom	A positive data property assertion DataPropertyAssertion( DPE a lt ) states that the individual a is connected		
	continues on next page		

continues on next page

Table 3 - continued from previous page

	inom provious page
OWLNegativeDataPropertyAssertionAxiom	A negative data property assertion NegativeDataPropertyAssertion( DPE a lt ) states that the individual a is not
OWLUnaryPropertyAxiom	Base class for Unary property axiom.
OWLObjectPropertyCharacteristicAxiom	Base interface for functional object property axiom.
OWLFunctionalObjectPropertyAxiom	An object property functionality axiom FunctionalObjectProperty( OPE ) states that
OWLAsymmetricObjectPropertyAxiom	An object property asymmetry axiom AsymmetricObjectProperty( OPE ) states that
OWLInverseFunctionalObjectPropertyAx-iom	An object property inverse functionality axiom Inverse-FunctionalObjectProperty( OPE )
OWLIrreflexiveObjectPropertyAxiom	An object property irreflexivity axiom IrreflexiveObject-Property( OPE ) states that the
OWLReflexiveObjectPropertyAxiom	An object property reflexivity axiom ReflexiveObject-Property( OPE ) states that the
OWLSymmetricObjectPropertyAxiom	An object property symmetry axiom SymmetricObject-Property( OPE ) states that
OWLTransitiveObjectPropertyAxiom	An object property transitivity axiom TransitiveObject-Property( OPE ) states that the
OWLDataPropertyCharacteristicAxiom	Base interface for Functional data property axiom.
OWLFunctionalDataPropertyAxiom	A data property functionality axiom FunctionalDataProperty( DPE ) states that
OWLPropertyDomainAxiom	Base class for Property Domain axioms.
OWLPropertyRangeAxiom	Base class for Property Range axioms.
OWLObjectPropertyDomainAxiom	An object property domain axiom ObjectPropertyDomain(OPE CE) states that the domain of the
OWLDataPropertyDomainAxiom	A data property domain axiom DataPropertyDomain(DPE CE) states that the domain of the
OWLObjectPropertyRangeAxiom	An object property range axiom ObjectPropertyRange(OPE CE) states that the range of the object property
OWLDataPropertyRangeAxiom	A data property range axiom DataPropertyRange( DPE DR ) states that the range of the data property

# **Module Contents**

```
\textbf{class} \  \, \texttt{owlapy.owl\_axiom.OWLAxiom} \, (\textit{annotations: Iterable}[OWLAnnotation] \, | \, \textit{None} = \textit{None})
```

Bases: owlapy.owl\_object.OWLObject

Represents Axioms in the OWL 2 Specification.

An OWL ontology contains a set of axioms. These axioms can be annotation axioms, declaration axioms, imports axioms or logical axioms.

```
__slots__ = '_annotations' annotations() \rightarrow List[OWLAnnotation] | None is_annotated() \rightarrow bool is_logical_axiom() \rightarrow bool is_annotation_axiom() \rightarrow bool
```

```
class owlapy.owl_axiom.OWLLogicalAxiom(
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLAxiom
     A base interface of all axioms that affect the logical meaning of an ontology. This excludes declaration axioms
     (including imports declarations) and annotation axioms.
     __slots__ = ()
     is\_logical\_axiom() \rightarrow bool
class owlapy.owl_axiom.OWLPropertyAxiom(
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLLogicalAxiom
     The base interface for property axioms.
     __slots__ = ()
class owlapy.owl_axiom.OWLObjectPropertyAxiom(
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLPropertyAxiom
     The base interface for object property axioms.
     __slots__ = ()
class owlapy.owl_axiom.OWLDataPropertyAxiom(
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLPropertyAxiom
     The base interface for data property axioms.
     __slots__ = ()
class owlapy.owl_axiom.OWLIndividualAxiom(
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLLogicalAxiom
     The base interface for individual axioms.
     __slots__ = ()
class owlapy.owl_axiom.OWLClassAxiom(annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLLogicalAxiom
     The base interface for class axioms.
     __slots__ = ()
class owlapy.owl_axiom.OWLDeclarationAxiom (entity: owlapy.owl_object.OWLEntity,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLAxiom
     Represents a Declaration axiom in the OWL 2 Specification. A declaration axiom declares an entity in an ontology.
     It doesn't affect the logical meaning of the ontology.
     __slots__ = '_entity'
     get_entity() → owlapy.owl_object.OWLEntity
```

A datatype definition DatatypeDefinition( DT DR ) defines a new datatype DT as being semantically equivalent to the data range DR; the latter must be a unary data range. This axiom allows one to use the defined datatype DT as a synonym for DR — that is, in any expression in the ontology containing such an axiom, DT can be replaced with DR without affecting the meaning of the ontology.

A key axiom HasKey( CE ( OPE1 ... OPEm ) ( DPE1 ... DPEn ) ) states that each (named) instance of the class expression CE is uniquely identified by the object property expressions OPEi and/or the data property expressions DPEj — that is, no two distinct (named) instances of CE can coincide on the values of all object property expressions OPEi and all data property expressions DPEj. In each such axiom in an OWL ontology, m or n (or both) must be larger than zero. A key axiom of the form HasKey( owl:Thing ( OPE ) () ) is similar to the axiom InverseFunctionalObjectProperty( OPE ), the main differences being that the former axiom is applicable only to individuals that are explicitly named in an ontology, while the latter axiom is also applicable to anonymous individuals and individuals whose existence is implied by existential quantification.

```
(https://www.w3.org/TR/owl2-syntax/#Keys)
__slots__ = ('_class_expression', '_property_expressions')
get_class_expression() → owlapy.class_expression.OWLClassExpression
get_property_expressions() → List[owlapy.owl_property.OWLPropertyExpression]
```

OWLPropertyExpression]

```
Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.
               Returns
                   The operands.
     __eq_ (other)
          Return self==value.
     __hash__()
          Return hash(self).
     __repr__()
          Return repr(self).
class owlapy.owl_axiom.OWLNaryAxiom (annotations: Iterable[OWLAnnotation] | None = None)
     Bases: Generic[_C], OWLAxiom
     Represents an axiom that contains two or more operands that could also be represented with multiple pairwise
     axioms.
          Parameters
               C – Class of contained objects.
     __slots__ = ()
     abstract as pairwise axioms() → Iterable[OWLNaryAxiom[ C]]
class owlapy.owl axiom.OWLNaryClassAxiom(
            class_expressions: List[owlapy.class_expression.OWLClassExpression],
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLClassAxiom, OWLNaryAxiom[owlapy.class_expression.OWLClassExpression]
     Represents an axiom that contains two or more operands that could also be represented with multiple pairwise
     axioms.
     __slots__ = '_class_expressions'
     class\_expressions() \rightarrow Iterable[owlapy.class\_expression.OWLClassExpression]
          Gets all of the top level class expressions that appear in this axiom.
               Returns
                   Sorted stream of class expressions that appear in the axiom.
     as\_pairwise\_axioms() \rightarrow Iterable[OWLNaryClassAxiom]
          Gets this axiom as a set of pairwise axioms; if the axiom contains only two operands, the axiom itself is
          returned unchanged, including its annotations.
                   This axiom as a set of pairwise axioms.
     __eq_ (other)
          Return self==value.
      __hash___()
          Return hash(self).
      __repr__()
          Return repr(self).
```

**operands** () → Iterable[owlapy.owl\_property.OWLPropertyExpression]

```
class owlapy.owl axiom.OWLEquivalentClassesAxiom(
            class_expressions: List[owlapy.class_expression.OWLClassExpression],
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLNaryClassAxiom
     An equivalent classes axiom Equivalent Classes (CE1 ... CEn ) states that all of the class expressions CEi, 1 \le i \le i
     n, are semantically equivalent to each other. This axiom allows one to use each CEi as a synonym for each CEi —
     that is, in any expression in the ontology containing such an axiom, CEi can be replaced with CEj without affecting
     the meaning of the ontology.
     (https://www.w3.org/TR/owl2-syntax/#Equivalent_Classes)
     __slots__ = ()
     contains named equivalent class() \rightarrow bool
     contains_owl_nothing() \rightarrow bool
     contains owl thing() \rightarrow bool
     named classes() → Iterable[owlapy.class expression.OWLClass]
class owlapy.owl_axiom.OWLDisjointClassesAxiom(
            class_expressions: List[owlapy.class_expression.OWLClassExpression],
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLNarvClassAxiom
     A disjoint classes axiom DisjointClasses (CE1 ... CEn ) states that all of the class expressions CEi, 1 \le i \le n, are
     pairwise disjoint; that is, no individual can be at the same time an instance of both CEi and CEj for i ≠ j.
     (https://www.w3.org/TR/owl2-syntax/#Disjoint_Classes)
     __slots__ = ()
class owlapy.owl_axiom.OWLNaryIndividualAxiom(
            individuals: List[owlapy.owl_individual.OWLIndividual],
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLIndividualAxiom, OWLNaryAxiom[owlapy.owl_individual.OWLIndividual]
     Represents an axiom that contains two or more operands that could also be represented with multiple pairwise
     individual axioms.
     __slots__ = '_individuals'
     individuals() → Iterable[owlapy.owl_individual.OWLIndividual]
           Get the individuals.
               Returns
                   Generator containing the individuals.
     as_pairwise_axioms() \rightarrow Iterable[OWLNaryIndividualAxiom]
      \underline{\phantom{a}}eq\underline{\phantom{a}} (other)
           Return self==value.
     __hash__()
           Return hash(self).
      __repr__()
           Return repr(self).
```

```
class owlapy.owl axiom.OWLDifferentIndividualsAxiom(
            individuals: List[owlapy.owl individual.OWLIndividual],
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLNaryIndividualAxiom
     An individual inequality axiom DifferentIndividuals (a1 ... an) states that all of the individuals ai, 1 \le i \le n, are
     different from each other; that is, no individuals ai and aj with i \neq j can be derived to be equal. This axiom can
     be used to axiomatize the unique name assumption — the assumption that all different individual names denote
     different individuals. (https://www.w3.org/TR/owl2-syntax/#Individual_Inequality)
     __slots__ = ()
class owlapy.owl_axiom.OWLSameIndividualAxiom(
            individuals: List[owlapy.owl individual.OWLIndividual],
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLNaryIndividualAxiom
     An individual equality axiom SameIndividual (a1 ... an ) states that all of the individuals ai, 1 \le i \le n, are equal
     to each other. This axiom allows one to use each ai as a synonym for each ai — that is, in any expression in the
     ontology containing such an axiom, ai can be replaced with aj without affecting the meaning of the ontology.
     (https://www.w3.org/TR/owl2-syntax/#Individual_Equality)
     __slots__ = ()
class owlapy.owl axiom.OWLNaryPropertyAxiom (properties: List[P],
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: Generic[_P], OWLPropertyAxiom, OWLNaryAxiom[_P]
     Represents an axiom that contains two or more operands that could also be represented with multiple pairwise
     property axioms.
     __slots__ = '_properties'
     properties() \rightarrow Iterable[\_P]
          Get all the properties that appear in the axiom.
              Returns
                  Generator containing the properties.
     as pairwise axioms() \rightarrow Iterable[OWLNaryPropertyAxiom]
      eq (other)
          Return self==value.
       hash__()
          Return hash(self).
      __repr__()
          Return repr(self).
class owlapy.owl_axiom.OWLEquivalentObjectPropertiesAxiom(
            properties: List[owlapy.owl_property.OWLObjectPropertyExpression],
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLNaryPropertyAxiom[owlapy.owl_property.OWLObjectPropertyExpression],
     OWLObjectPropertyAxiom
```

An equivalent object properties axiom EquivalentObjectProperties (OPE1 ... OPEn ) states that all of the object property expressions OPEi,  $1 \le i \le n$ , are semantically equivalent to each other. This axiom allows one to use each

OPEi as a synonym for each OPEi — that is, in any expression in the ontology containing such an axiom, OPEi can be replaced with OPEj without affecting the meaning of the ontology.

```
(https://www.w3.org/TR/owl2-syntax/#Equivalent Object Properties)
```

```
__slots__ = ()
```

class owlapy.owl axiom.OWLDisjointObjectPropertiesAxiom(

properties: List[owlapy.owl\_property.OWLObjectPropertyExpression], *annotations: Iterable[OWLAnnotation] | None = None)* 

Bases: OWLNaryPropertyAxiom[owlapy.owl\_property.OWLObjectPropertyExpression], OWLObjectPropertyAxiom

A disjoint object properties axiom DisjointObjectProperties (OPE1 ... OPEn ) states that all of the object property expressions OPEi,  $1 \le i \le n$ , are pairwise disjoint; that is, no individual x can be connected to an individual y by both OPEi and OPEi for  $i \neq j$ .

(https://www.w3.org/TR/owl2-syntax/#Disjoint Object Properties)

```
__slots__ = ()
```

class owlapy.owl\_axiom.OWLInverseObjectPropertiesAxiom(

first: owlapy.owl\_property.OWLObjectPropertyExpression, second: owlapy.owl\_property.OWLObjectPropertyExpression, *annotations: Iterable[OWLAnnotation] | None = None)* 

Bases: OWLNaryPropertyAxiom[owlapy.owl property.OWLObjectPropertyExpression], OWLObjectPropertyAxiom

An inverse object properties axiom InverseObjectProperties(OPE1 OPE2) states that the object property expression OPE1 is an inverse of the object property expression OPE2. Thus, if an individual x is connected by OPE1 to an individual y, then y is also connected by OPE2 to x, and vice versa.

(https://www.w3.org/TR/owl2-syntax/#Inverse\_Object\_Properties\_2)

```
__slots__ = ('_first', '_second')
get_first_property() → owlapy.owl_property.OWLObjectPropertyExpression
get_second_property() → owlapy.owl_property.OWLObjectPropertyExpression
__repr__()
```

Return repr(self).

class owlapy.owl axiom.OWLEquivalentDataPropertiesAxiom(

properties: List[owlapy.owl\_property.OWLDataPropertyExpression], *annotations: Iterable[OWLAnnotation] | None = None)* 

Bases:

OWLNaryPropertyAxiom[owlapy.owl\_property.OWLDataPropertyExpression], OWLDataPropertyAxiom

An equivalent data properties axiom EquivalentDataProperties( DPE1 ... DPEn ) states that all the data property expressions DPEi,  $1 \le i \le n$ , are semantically equivalent to each other. This axiom allows one to use each DPEi as a synonym for each DPEi — that is, in any expression in the ontology containing such an axiom, DPEi can be replaced with DPEj without affecting the meaning of the ontology.

```
(https://www.w3.org/TR/owl2-syntax/#Equivalent Data Properties)
```

```
__slots__ = ()
```

```
class owlapy.owl axiom.OWLDisjointDataPropertiesAxiom(
            properties: List[owlapy.owl_property.OWLDataPropertyExpression],
            annotations: Iterable[OWLAnnotation] | None = None)
                 OWLNaryPropertyAxiom[owlapy.owl_property.OWLDataPropertyExpression],
     Bases:
      OWLDataPropertyAxiom
     A disjoint data properties axiom DisjointDataProperties( DPE1 ... DPEn ) states that all of the data property
     expressions DPEi, 1 \le i \le n, are pairwise disjoint; that is, no individual x can be connected to a literal y by both
           DPEi and DPEj for i \neq j.
           (https://www.w3.org/TR/owl2-syntax/#Disjoint_Data_Properties)
     __slots__ = ()
class owlapy.owl axiom.OWLSubClassOfAxiom(
            sub class: owlapy.class expression.OWLClassExpression,
            super_class: owlapy.class_expression.OWLClassExpression,
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLClassAxiom
     A subclass axiom SubClassOf( CE1 CE2 ) states that the class expression CE1 is a subclass of the class expression
     CE2. Roughly speaking, this states that CE1 is more specific than CE2. Subclass axioms are a fundamental type
     of axioms in OWL 2 and can be used to construct a class hierarchy. Other kinds of class expression axiom can be
     seen as syntactic shortcuts for one or more subclass axioms.
           (https://www.w3.org/TR/owl2-syntax/#Subclass Axioms)
     __slots__ = ('_sub_class', '_super_class')
     get sub class() → owlapy.class expression.OWLClassExpression
     get super class() → owlapy.class expression.OWLClassExpression
      \underline{\phantom{a}}eq\underline{\phantom{a}} (other)
           Return self==value.
      __hash__()
           Return hash(self).
       _repr__()
           Return repr(self).
class owlapy.owl_axiom.OWLDisjointUnionAxiom(cls_: owlapy.class_expression.OWLClass,
            class expressions: List[owlapy.class expression.OWLClassExpression],
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLClassAxiom
     A disjoint union axiom DisjointUnion (C CE1 ... CEn ) states that a class C is a disjoint union of the class expres-
     sions CEi, 1 \le i \le n, all of which are pairwise disjoint. Such axioms are sometimes referred to as covering axioms,
     as they state that the extensions of all CEi exactly cover the extension of C. Thus, each instance of C is an instance
     of exactly one CEi, and each instance of CEi is an instance of C.
     (https://www.w3.org/TR/owl2-syntax/#Disjoint_Union_of_Class_Expressions)
      __slots__ = ('_cls', '_class_expressions')
     \texttt{get\_owl\_class}() \rightarrow owlapy.class\_expression.OWLClass
     \texttt{get\_class\_expressions}() \rightarrow Iterable[\mathit{owlapy.class\_expression.OWLClassExpression}]
```

```
get_owl_equivalent_classes_axiom() → OWLEquivalentClassesAxiom
     \texttt{get\_owl\_disjoint\_classes\_axiom}() \rightarrow OWLDisjointClassesAxiom
     __eq_ (other)
          Return self==value.
     __hash__()
          Return hash(self).
     __repr__()
          Return repr(self).
class owlapy.owl_axiom.OWLClassAssertionAxiom(
           individual: owlapy.owl_individual.OWLIndividual,
           class_expression: owlapy.class_expression.OWLClassExpression,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLIndividualAxiom
     A class assertion ClassAssertion (CE a) states that the individual a is an instance of the class expression CE.
     (https://www.w3.org/TR/owl2-syntax/#Class_Assertions)
     __slots__ = ('_individual', '_class_expression')
     get_individual() → owlapy.owl_individual.OWLIndividual
     \texttt{get\_class\_expression}() \rightarrow \textit{owlapy.class\_expression.OWLClassExpression}
     __eq_ (other)
          Return self==value.
     __hash__()
          Return hash(self).
     __repr__()
          Return repr(self).
class owlapy.owl_axiom.OWLAnnotationProperty(iri: owlapy.iri.IRI | str)
     Bases: owlapy.owl_property.OWLProperty
     Represents an AnnotationProperty in the OWL 2 specification.
     __slots__ = '_iri'
     property iri: owlapy.iri.IRI
          Gets the IRI of this object.
                  The IRI of this object.
     property str: str
          Gets the string representation of this object
              Returns
                  The IRI as string
```

```
class owlapy.owl_axiom.OWLAnnotation(property: OWLAnnotationProperty,
            value: owlapy.owl annotation.OWLAnnotationValue)
     Bases: owlapy.owl_object.OWLObject
     Annotations are used in the various types of annotation axioms, which bind annotations to their subjects (i.e. axioms
     or declarations).
     __slots__ = ('_property', '_value')
     get_property() → OWLAnnotationProperty
          Gets the property that this annotation acts along.
               Returns
                   The annotation property.
     \texttt{get\_value}() \rightarrow owlapy.owl\_annotation.OWLAnnotationValue
          Gets the annotation value. The type of value will depend upon the type of the annotation e.g. whether the
          annotation is an OWLLiteral, an IRI or an OWLAnonymousIndividual.
               Returns
                   The annotation value.
     __eq_ (other)
          Return self==value.
     __hash__()
          Return hash(self).
      __repr__()
          Return repr(self).
class owlapy.owl_axiom.OWLAnnotationAxiom(
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLAxiom
     A super interface for annotation axioms.
     __slots__ = ()
     is\_annotation\_axiom() \rightarrow bool
class owlapy.owl_axiom.OWLAnnotationAssertionAxiom(
            subject: owlapy.owl annotation.OWLAnnotationSubject, annotation: OWLAnnotation)
     Bases: OWLAnnotationAxiom
     An annotation assertion AnnotationAssertion( AP as av ) states that the annotation subject as — an IRI or an
     anonymous individual — is annotated with the annotation property AP and the annotation value av.
     (https://www.w3.org/TR/owl2-syntax/#Annotation_Assertion)
     __slots__ = ('_subject', '_annotation')
     \texttt{get\_subject} () \rightarrow owlapy.owl_annotation.OWLAnnotationSubject
          Gets the subject of this object.
```

## Returns

The subject.

```
get_property() → OWLAnnotationProperty
          Gets the property.
              Returns
                  The property.
     \texttt{get\_value}() \rightarrow owlapy.owl\_annotation.OWLAnnotationValue
          Gets the annotation value. This is either an IRI, an OWLAnonymousIndividual or an OWLLiteral.
              Returns
                  The annotation value.
     ___eq__(other)
          Return self==value.
      __hash__()
          Return hash(self).
     __repr__()
          Return repr(self).
class owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom(
           sub_property: OWLAnnotationProperty, super_property: OWLAnnotationProperty,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLAnnotationAxiom
     An annotation subproperty axiom SubAnnotationPropertyOf( AP1 AP2 ) states that the annotation property AP1
     is a subproperty of the annotation property AP2.
     (https://www.w3.org/TR/owl2-syntax/#Annotation_Subproperties)
     __slots__ = ('_sub_property', '_super_property')
     get_sub_property() → OWLAnnotationProperty
     get super property() → OWLAnnotationProperty
     __eq_ (other)
          Return self==value.
      hash__()
          Return hash(self).
       _repr__()
          Return repr(self).
class owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom(
           property_: OWLAnnotationProperty, domain: owlapy.iri.IRI,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLAnnotationAxiom
     An annotation property domain axiom AnnotationPropertyDomain(APU) states that the domain of the annotation
     property AP is the IRI U.
          (https://www.w3.org/TR/owl2-syntax/#Annotation_Property_Domain)
     __slots__ = ('_property', '_domain')
     \texttt{get\_property}() \rightarrow OWLAnnotationProperty
```

```
get_domain() → owlapy.iri.IRI
     __eq__(other)
          Return self==value.
     __hash__()
          Return hash(self).
     __repr__()
          Return repr(self).
class owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom(
            property_: OWLAnnotationProperty, range_: owlapy.iri.IRI,
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLAnnotationAxiom
     An annotation property range axiom AnnotationPropertyRange( AP U ) states that the range of the annotation
     property AP is the IRI U.
     (https://www.w3.org/TR/owl2-syntax/#Annotation_Property_Range)
     __slots__ = ('_property', '_range')
     \texttt{get\_property}() \rightarrow OWLAnnotationProperty
     \texttt{get\_range}() \rightarrow owlapy.iri.IRI
     __eq_ (other)
          Return self==value.
       hash__()
          Return hash(self).
     __repr__()
          Return repr(self).
class owlapy.owl_axiom.OWLSubPropertyAxiom(sub_property: _P, super_property: _P,
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: Generic[_P], OWLPropertyAxiom
     Base interface for object and data sub-property axioms.
     __slots__ = ('_sub_property', '_super_property')
     \texttt{get} sub property() \rightarrow P
     \texttt{get\_super\_property}\,(\,)\,\to \_P
     __eq_ (other)
          Return self==value.
     __hash___()
          Return hash(self).
     __repr__()
          Return repr(self).
```

```
class owlapy.owl axiom.OWLSubObjectPropertyOfAxiom(
            sub_property: owlapy.owl_property.OWLObjectPropertyExpression,
            super property: owlapy.owl property.OWLObjectPropertyExpression,
            annotations: Iterable[OWLAnnotation] | None = None)
              OWLSubPropertyAxiom[owlapy.owl property.OWLObjectPropertyExpression],
     Bases:
     OWLObjectPropertyAxiom
     Object subproperty axioms are analogous to subclass axioms, and they come in two forms. The basic form is
     SubObjectPropertyOf(OPE1 OPE2). This axiom states that the object property expression OPE1 is a subproperty
     of the object property expression OPE2 — that is, if an individual x is connected by OPE1 to an individual y, then
     x is also connected by OPE2 to y. The more complex form is SubObjectPropertyOf( ObjectPropertyChain( OPE1
     ... OPEn ) OPE ) but ObjectPropertyChain is not represented in owlapy yet.
     (https://www.w3.org/TR/owl2-syntax/#Object_Subproperties)
     __slots__ = ()
class owlapy.owl_axiom.OWLSubDataPropertyOfAxiom(
            sub_property: owlapy.owl_property.OWLDataPropertyExpression,
            super_property: owlapy.owl_property.OWLDataPropertyExpression,
            annotations: Iterable[OWLAnnotation] | None = None)
                 OWLSubPropertyAxiom[owlapy.owl_property.OWLDataPropertyExpression],
     OWLDataPropertyAxiom
     A data subproperty axiom SubDataPropertyOf( DPE1 DPE2 ) states that the data property expression DPE1 is a
     subproperty of the data property expression DPE2 — that is, if an individual x is connected by DPE1 to a literal y,
          then x is connected by DPE2 to y as well.
          (https://www.w3.org/TR/owl2-syntax/#Data_Subproperties)
     __slots__ = ()
class owlapy.owl axiom.OWLPropertyAssertionAxiom(
            subject: owlapy.owl_individual.OWLIndividual, property_: _P, object_: _C,
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: Generic[ P, C], OWLIndividualAxiom
     Base class for Property Assertion axioms.
     __slots__ = ('_subject', '_property', '_object')
     get_subject() → owlapy.owl_individual.OWLIndividual
     \texttt{get property}() \rightarrow P
     \mathtt{get\_object}() \rightarrow \_C
     \underline{\phantom{a}}eq\underline{\phantom{a}} (other)
          Return self==value.
      __hash__()
          Return hash(self).
     __repr__()
          Return repr(self).
class owlapy.owl_axiom.OWLObjectPropertyAssertionAxiom(
            subject: owlapy.owl individual.OWLIndividual,
            property_: owlapy.owl_property.OWLObjectPropertyExpression,
            object: owlapy.owl individual.OWLIndividual,
            annotations: Iterable[OWLAnnotation] | None = None)
```

```
Bases: OWLPropertyAssertionAxiom[owlapy.owl_property.OWLObjectPropertyExpression, owlapy.owl individual.OWLIndividual]
```

A positive object property assertion ObjectPropertyAssertion(OPE a1 a2) states that the individual a1 is connected by the object property expression OPE to the individual a2.

```
(https://www.w3.org/TR/owl2-syntax/#Positive Object Property Assertions)
```

```
__slots__ = ()
```

```
class owlapy.owl_axiom.OWLNegativeObjectPropertyAssertionAxiom(
```

subject: owlapy.owl individual.OWLIndividual,

property\_: owlapy.owl\_property.OWLObjectPropertyExpression,

object\_: owlapy.owl\_individual.OWLIndividual,

*annotations: Iterable[OWLAnnotation] | None = None)* 

 ${\bf Bases:}~{\it OWLPropertyAssertionAxiom[owlapy.owl\_property.OWLObjectPropertyExpression, owlapy.owl\_individual.OWLIndividual]}$ 

A negative object property assertion NegativeObjectPropertyAssertion( OPE a1 a2 ) states that the individual a1 is not connected by the object property expression OPE to the individual a2.

(https://www.w3.org/TR/owl2-syntax/#Negative\_Object\_Property\_Assertions)

```
__slots__ = ()
```

class owlapy.owl\_axiom.OWLDataPropertyAssertionAxiom(

subject: owlapy.owl\_individual.OWLIndividual,

property\_: owlapy.owl\_property.OWLDataPropertyExpression,

object\_: owlapy.owl\_literal.OWLLiteral, annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLPropertyAssertionAxiom[owlapy.owl\_property.OWLDataPropertyExpression, owlapy.owl\_literal.OWLLiteral]

A positive data property assertion DataPropertyAssertion( DPE a lt ) states that the individual a is connected by the data property expression DPE to the literal lt.

(https://www.w3.org/TR/owl2-syntax/#Positive Data Property Assertions)

```
__slots__ = ()
```

class owlapy.owl axiom.OWLNegativeDataPropertyAssertionAxiom(

subject: owlapy.owl\_individual.OWLIndividual,

property\_: owlapy.owl\_property.OWLDataPropertyExpression,

object\_: owlapy.owl\_literal.OWLLiteral, annotations: Iterable[OWLAnnotation] | None = None)

 ${\bf Bases:}~{\it OWLP roperty Assertion Axiom [owlapy.owl\_property.OWLData Property Expression, owlapy.owl\_literal.OWLLiteral]}$ 

A negative data property assertion NegativeDataPropertyAssertion( DPE a lt ) states that the individual a is not connected by the data property expression DPE to the literal lt.

(https://www.w3.org/TR/owl2-syntax/#Negative\_Data\_Property\_Assertions)

```
__slots__ = ()
```

class owlapy.owl\_axiom.OWLUnaryPropertyAxiom(property\_: \_P,

*annotations: Iterable[OWLAnnotation] | None = None)* 

Bases: Generic[\_P], OWLPropertyAxiom

Base class for Unary property axiom.

```
__slots__ = '_property'
     \texttt{get\_property}\,(\,)\,\to \_P
class owlapy.owl_axiom.OWLObjectPropertyCharacteristicAxiom(
           property_: owlapy.owl_property.OWLObjectPropertyExpression,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLUnaryPropertyAxiom[owlapy.owl_property.OWLObjectPropertyExpression],
     OWLObjectPropertyAxiom
     Base interface for functional object property axiom.
     __slots__ = ()
     ___eq__(other)
          Return self==value.
      hash ()
          Return hash(self).
     __repr__()
          Return repr(self).
class owlapy.owl_axiom.OWLFunctionalObjectPropertyAxiom(
           property: owlapy.owl property.OWLObjectPropertyExpression,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLObjectPropertyCharacteristicAxiom
     An object property functionality axiom FunctionalObjectProperty(OPE) states that the object property expression
     OPE is functional — that is, for each individual x, there can be at most one distinct individual y such that x is
     connected by OPE to y.
     (https://www.w3.org/TR/owl2-syntax/#Functional_Object_Properties)
     __slots__ = ()
class owlapy.owl_axiom.OWLAsymmetricObjectPropertyAxiom(
           property_: owlapy.owl_property.OWLObjectPropertyExpression,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLObjectPropertyCharacteristicAxiom
     An object property asymmetry axiom AsymmetricObjectProperty(OPE) states that the object property expression
     OPE is asymmetric — that is, if an individual x is connected by OPE to an individual y, then y cannot be connected
     by OPE to x.
     (https://www.w3.org/TR/owl2-syntax/#Symmetric_Object_Properties)
     __slots__ = ()
class owlapy.owl_axiom.OWLInverseFunctionalObjectPropertyAxiom(
           property_: owlapy.owl_property.OWLObjectPropertyExpression,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLObjectPropertyCharacteristicAxiom
     An object property inverse functionality axiom InverseFunctionalObjectProperty( OPE ) states that the object
```

An object property inverse functionality axiom InverseFunctionalObjectProperty( OPE ) states that the object property expression OPE is inverse-functional — that is, for each individual x, there can be at most one individual y such that y is connected by OPE with x.

(https://www.w3.org/TR/owl2-syntax/#Inverse-Functional\_Object\_Properties)

```
__slots__ = ()
class owlapy.owl_axiom.OWLIrreflexiveObjectPropertyAxiom(
           property: owlapy.owl property.OWLObjectPropertyExpression,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLObjectPropertyCharacteristicAxiom
     An object property irreflexivity axiom IrreflexiveObjectProperty(OPE) states that the object property expression
     OPE is irreflexive — that is, no individual is connected by OPE to itself.
     (https://www.w3.org/TR/owl2-syntax/#Irreflexive_Object_Properties)
     __slots__ = ()
class owlapy.owl axiom.OWLReflexiveObjectPropertyAxiom(
           property_: owlapy.owl_property.OWLObjectPropertyExpression,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLObjectPropertyCharacteristicAxiom
     An object property reflexivity axiom ReflexiveObjectProperty( OPE ) states that the object property expression
     OPE is reflexive — that is, each individual is connected by OPE to itself. Each such axiom can be seen as a
     syntactic shortcut for the following axiom: SubClassOf( owl:Thing ObjectHasSelf( OPE ) )
     (https://www.w3.org/TR/owl2-syntax/#Reflexive Object Properties)
     __slots__ = ()
class owlapy.owl_axiom.OWLSymmetricObjectPropertyAxiom(
           property: owlapy.owl property.OWLObjectPropertyExpression,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLObjectPropertyCharacteristicAxiom
     An object property symmetry axiom SymmetricObjectProperty (OPE) states that the object property expression
     OPE is symmetric — that is, if an individual x is connected by OPE to an individual y, then y is also connected by
     OPE to x. Each such axiom can be seen as a syntactic shortcut for the following axiom:
          SubObjectPropertyOf( OPE ObjectInverseOf( OPE ) )
          (https://www.w3.org/TR/owl2-syntax/#Symmetric Object Properties)
      __slots__ = ()
class owlapy.owl_axiom.OWLTransitiveObjectPropertyAxiom(
           property: owlapy.owl property.OWLObjectPropertyExpression,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLObjectPropertyCharacteristicAxiom
     An object property transitivity axiom TransitiveObjectProperty( OPE ) states that the object property expres-
     sionOPE is transitive — that is, if an individual x is connected by OPE to an individual y that is connected by OPE
     to an individual z, then x is also connected by OPE to z. Each such axiom can be seen as a syntactic shortcut for
     the following axiom: SubObjectPropertyOf( ObjectPropertyChain( OPE OPE ) OPE )
          (https://www.w3.org/TR/owl2-syntax/#Transitive Object Properties)
     __slots__ = ()
class owlapy.owl_axiom.OWLDataPropertyCharacteristicAxiom(
           property_: owlapy.owl_property.OWLDataPropertyExpression,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLUnaryPropertyAxiom[owlapy.owl property.OWLDataPropertyExpression],
```

OWLDataPropertyAxiom

```
Base interface for Functional data property axiom.
     __slots__ = ()
     __eq_ (other)
           Return self==value.
     __hash__()
          Return hash(self).
     __repr__()
          Return repr(self).
class owlapy.owl_axiom.OWLFunctionalDataPropertyAxiom(
            property_: owlapy.owl_property.OWLDataPropertyExpression,
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLDataPropertyCharacteristicAxiom
     A data property functionality axiom FunctionalDataProperty (DPE) states that the data property expression DPE
     is functional — that is, for each individual x, there can be at most one distinct literal y such that x is connected by
     DPE with y. Each such axiom can be seen as a syntactic shortcut for the following axiom: SubClassOf( owl:Thing
     DataMaxCardinality( 1 DPE ) )
     (https://www.w3.org/TR/owl2-syntax/#Transitive_Object_Properties)
     __slots__ = ()
class owlapy.owl_axiom.OWLPropertyDomainAxiom(property_: _P,
            domain: owlapy.class_expression.OWLClassExpression,
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: Generic[P], OWLUnaryPropertyAxiom[P]
     Base class for Property Domain axioms.
     __slots__ = '_domain'
     get_domain() → owlapy.class_expression.OWLClassExpression
     \underline{\phantom{a}}eq\underline{\phantom{a}} (other)
           Return self==value.
     __hash__()
          Return hash(self).
     __repr__()
          Return repr(self).
class owlapy.owl axiom.OWLPropertyRangeAxiom(property: P, range: R,
            annotations: Iterable[OWLAnnotation] | None = None)
     Bases: Generic[_P, _R], OWLUnaryPropertyAxiom[_P]
     Base class for Property Range axioms.
     __slots__ = '_range'
     \mathtt{get}\mathtt{\_range}\left(\right) \to \mathtt{\_R}
     __eq_ (other)
           Return self==value.
```

```
hash ()
          Return hash(self).
     __repr__()
          Return repr(self).
class owlapy.owl axiom.OWLObjectPropertyDomainAxiom(
           property: owlapy.owl property.OWLObjectPropertyExpression,
           domain: owlapy.class expression.OWLClassExpression,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLPropertyDomainAxiom[owlapy.owl_property.OWLObjectPropertyExpression]
     An object property domain axiom ObjectPropertyDomain( OPE CE) states that the domain of the object property
     expression OPE is the class expression CE — that is, if an individual x is connected by OPE with some other
     individual, then x is an instance of CE. Each such axiom can be seen as a syntactic shortcut for the following
     axiom: SubClassOf(ObjectSomeValuesFrom(OPE owl:Thing) CE)
     (https://www.w3.org/TR/owl2-syntax/#Object Property Domain)
     __slots__ = ()
class owlapy.owl_axiom.OWLDataPropertyDomainAxiom(
           property_: owlapy.owl_property.OWLDataPropertyExpression,
           domain: owlapy.class_expression.OWLClassExpression,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLPropertyDomainAxiom[owlapy.owl property.OWLDataPropertyExpression]
     A data property domain axiom DataPropertyDomain( DPE CE ) states that the domain of the data property ex-
     pression DPE is the class expression CE — that is, if an individual x is connected by DPE with some literal, then
     x is an instance of CE. Each such axiom can be seen as a syntactic shortcut for the following axiom: SubClassOf(
     DataSomeValuesFrom( DPE rdfs:Literal) CE )
     (https://www.w3.org/TR/owl2-syntax/#Data_Property_Domain)
     __slots__ = ()
class owlapy.owl_axiom.OWLObjectPropertyRangeAxiom(
           property_: owlapy.owl_property.OWLObjectPropertyExpression,
           range_: owlapy.class_expression.OWLClassExpression,
           annotations: Iterable[OWLAnnotation] | None = None)
     Bases: OWLPropertyRangeAxiom[owlapy.owl_property.OWLObjectPropertyExpression,
     owlapy.class_expression.OWLClassExpression]
     An object property range axiom ObjectPropertyRange( OPE CE ) states that the range of the object property
     expression OPE is the class expression CE — that is, if some individual is connected by OPE with an individual
     x, then x is an instance of CE. Each such axiom can be seen as a syntactic shortcut for the following axiom:
     SubClassOf( owl:Thing ObjectAllValuesFrom( OPE CE ) )
     (https://www.w3.org/TR/owl2-syntax/#Object_Property_Range)
     __slots__ = ()
class owlapy.owl_axiom.OWLDataPropertyRangeAxiom(
           property: owlapy.owl property.OWLDataPropertyExpression,
           range: owlapy.owl_datatype.OWLDataRange,
           annotations: Iterable[OWLAnnotation] | None = None)
              OWLPropertyRangeAxiom[owlapy.owl_property.OWLDataPropertyExpression,
     owlapy.owl_datatype.OWLDataRange]
```

A data property range axiom DataPropertyRange( DPE DR ) states that the range of the data property expression DPE is the data range DR — that is, if some individual is connected by DPE with a literal x, then x is in DR. The arity of DR must be one. Each such axiom can be seen as a syntactic shortcut for the following axiom: SubClassOf( owl:Thing DataAllValuesFrom( DPE DR ) )

(https://www.w3.org/TR/owl2-syntax/#Data\_Property\_Range)

\_\_slots\_\_ = ()

# owlapy.owl\_data\_ranges

**OWL Data Ranges** 

https://www.w3.org/TR/owl2-syntax/#Data\_Ranges

 $DataRange := Datatype \mid DataIntersectionOf \mid DataUnionOf \mid DataComplementOf \mid DataOneOf \mid DatatypeRestriction$ 

# **Classes**

OWLPropertyRange	OWL Objects that can be the ranges of properties.
OWLDataRange	Represents a DataRange in the OWL 2 Specification.
OWLNaryDataRange	OWLNaryDataRange.
OWLDataIntersectionOf	An intersection data range DataIntersectionOf( DR1
	DRn ) contains all tuples of literals that are contained
OWLDataUnionOf	A union data range DataUnionOf( DR1 DRn ) contains
	all tuples of literals that are contained in the at least
OWLDataComplementOf	A complement data range DataComplementOf( DR )
	contains all tuples of literals that are not contained in the

# **Module Contents**

```
class owlapy.owl_data_ranges.OWLPropertyRange

Bases: owlapy.owl_object.OWLObject

OWL Objects that can be the ranges of properties.

class owlapy.owl_data_ranges.OWLDataRange

Bases: OWLPropertyRange

Represents a DataRange in the OWL 2 Specification.

class owlapy.owl_data_ranges.OWLNaryDataRange (operands: Iterable[OWLDataRange])

Bases: OWLDataRange, owlapy.meta_classes.HasOperands[OWLDataRange]

OWLNaryDataRange.

__slots__ = ()

operands() → Iterable[OWLDataRange]

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.
```

Returns

The operands.

```
__repr__()
          Return repr(self).
     __eq_ (other)
          Return self==value.
     hash ()
          Return hash(self).
class owlapy.owl_data_ranges.OWLDataIntersectionOf(
           operands: Iterable[OWLDataRange])
     Bases: OWLNaryDataRange
     An intersection data range DataIntersectionOf( DR1 ... DRn ) contains all tuples of literals that are contained in
     each data range DRi for 1 \le i \le n. All data ranges DRi must be of the same arity, and the resulting data range is
     of that arity as well.
     (https://www.w3.org/TR/owl2-syntax/#Intersection_of_Data_Ranges)
     __slots__ = '_operands'
     type_index: Final = 4004
class owlapy.owl_data_ranges.OWLDataUnionOf(operands: Iterable[OWLDataRange])
     Bases: OWLNaryDataRange
     A union data range DataUnionOf( DR1 ... DRn ) contains all tuples of literals that are contained in the at least one
     data range DRi for 1 \le i \le n. All data ranges DRi must be of the same arity, and the resulting data range is of that
     arity as well.
     (https://www.w3.org/TR/owl2-syntax/#Union of Data Ranges)
     __slots__ = '_operands'
     type_index: Final = 4005
class owlapy.owl_data_ranges.OWLDataComplementOf(data_range: OWLDataRange)
     Bases: OWLDataRange
     A complement data range DataComplementOf( DR ) contains all tuples of literals that are not contained in the
     data range DR. The resulting data range has the arity equal to the arity of DR.
     (https://www.w3.org/TR/owl2-syntax/#Complement_of_Data_Ranges)
     type_index: Final = 4002
     get_data_range() → OWLDataRange
              Returns
                  The wrapped data range.
     __repr__()
          Return repr(self).
     __eq__(other)
          Return self==value.
     __hash__()
          Return hash(self).
```

# owlapy.owl\_datatype

**OWL** Datatype

# **Classes**

OWLDatatype	Datatypes are entities that refer to sets of data values.
	Thus, datatypes are analogous to classes,

# **Module Contents**

```
class owlapy.owl_datatype.OWLDatatype (iri: owlapy.iri.IRI | owlapy.meta_classes.HasIRI)
    Bases: owlapy.owl_object.OWLEntity, owlapy.owl_data_ranges.OWLDataRange
```

Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes, the main difference being that the former contain data values such as strings and numbers, rather than individuals. Datatypes are a kind of data range, which allows them to be used in restrictions. Each data range is associated with an arity; for datatypes, the arity is always one. The built-in datatype rdfs:Literal denotes any set of data values that contains the union of the value spaces of all datatypes.

(https://www.w3.org/TR/owl2-syntax/#Datatypes)

```
__slots__ = '_iri'

type_index: Final = 4001

property iri: owlapy.iri.IRI

Gets the IRI of this object.

Returns

The IRI of this object.

property str: str

Gets the string representation of this object
```

Returns

#### The

The IRI as string

# owlapy.owl\_hierarchy

Classes representing hierarchy in OWL.

# **Classes**

AbstractHierarchy	Representation of an abstract hierarchy which can be used for classes or properties.
ClassHierarchy	Representation of a class hierarchy.
ObjectPropertyHierarchy	Representation of an objet property hierarchy.
DatatypePropertyHierarchy	Representation of a data property hierarchy.

# **Module Contents**

Representation of an abstract hierarchy which can be used for classes or properties.

#### **Parameters**

- hierarchy\_down A downwards hierarchy given as a mapping of Entities to sub-entities.
- reasoner Alternatively, a reasoner whose root\_ontology is queried for entities.

```
__slots__ = ('_Type', '_ent_set', '_parents_map', '_parents_map_trans',
'_children_map',...
```

 ${\tt classmethod\ get\_top\_entity}\,(\,)\,\to \_S$ 

# Abstractmethod

The most general entity in this hierarchy, which contains all the entities.

```
{\tt classmethod\ get\_bottom\_entity}\,(\,)\,\to \_S
```

#### Abstractmethod

The most specific entity in this hierarchy, which contains none of the entities.

static restrict (hierarchy:  $\_U$ , \*, remove: Iterable[ $\_S$ ] = None, allow: Iterable[ $\_S$ ] = None)  $\rightarrow$   $\_U$  Restrict a given hierarchy to a set of allowed/removed entities.

# **Parameters**

- hierarchy An existing Entity hierarchy to restrict.
- **remove** Set of entities which should be ignored.
- allow Set of entities which should be used.

#### Returns

The restricted hierarchy.

```
restrict_and_copy (*, remove: Iterable[\_S] = None, allow: Iterable[\_S] = None) \rightarrow _U Restrict this hierarchy.
```

See restrict for more info.

```
\texttt{parents} \ (\textit{entity: \_S}, \textit{direct: bool} = \textit{True}) \ \rightarrow \text{Iterable}[\_S]
```

Parents of an entity.

# **Parameters**

- **entity** Entity for which to query parent entities.
- direct False to return transitive parents.

# Returns

Super-entities.

**is\_parent\_of** (a:  $\_S$ , b:  $\_S$ )  $\rightarrow$  bool if A is a parent of B.

# 1 Note

A is always a parent of A.

 $\verb"is_child_of" (a: \_S, b: \_S) \to bool$ 

If A is a child of B.

# 1 Note

A is always a child of A.

**children** (*entity*:  $\_S$ , *direct*: bool = True)  $\rightarrow$  Iterable[ $\_S$ ]

Children of an entity.

# **Parameters**

- **entity** Entity for which to guery child entities.
- direct False to return transitive children.

#### Returns

Sub-entities.

**siblings** (*entity:*  $\_S$ )  $\rightarrow$  Iterable[ $\_S$ ]

 $\textbf{items}\,(\,)\,\rightarrow Iterable[\_S]$ 

 $roots(of: \_S \mid None = None) \rightarrow Iterable[\_S]$ 

**leaves** (of:  $\_S \mid None = None$ )  $\rightarrow$  Iterable[ $\_S$ ]

 $\_$ contains $\_$ ( $item: \_S$ )  $\rightarrow$  bool $\_$ len $\_$ ()

class owlapy.owl\_hierarchy.ClassHierarchy(

hierarchy\_down: Iterable[Tuple[owlapy.class\_expression.OWLClass, Iterable[owlapy.class\_expression.OWLClass]]])

class owlapy.owl\_hierarchy.ClassHierarchy (reasoner: owlapy.owl\_reasoner.OWLReasoner)

Bases: AbstractHierarchy[owlapy.class\_expression.OWLClass]

Representation of a class hierarchy.

# **Parameters**

- hierarchy\_down A downwards hierarchy given as a mapping of Class to sub-classes.
- reasoner Alternatively, a reasoner whose root\_ontology is queried for classes and subclasses.

classmethod get\_top\_entity() → owlapy.class\_expression.OWLClass

The most general entity in this hierarchy, which contains all the entities.

```
classmethod get_bottom_entity() → owlapy.class_expression.OWLClass
          The most specific entity in this hierarchy, which contains none of the entities.
     sub classes (entity: owlapy.class expression.OWLClass, direct: bool = True)
                  → Iterable[owlapy.class expression.OWLClass]
     super_classes (entity: owlapy.class_expression.OWLClass, direct: bool = True)
                  → Iterable[owlapy.class expression.OWLClass]
     is_subclass_of (subclass: owlapy.class_expression.OWLClass,
                 superclass: owlapy.class\_expression.OWLClass) \rightarrow bool
class owlapy.owl_hierarchy.ObjectPropertyHierarchy(
           hierarchy_down: Iterable[Tuple[owlapy.owl_property.OWLObjectProperty, Iterable[owlapy.owl_property.OWLObjectProperty]
class owlapy.owl_hierarchy.ObjectPropertyHierarchy(
           reasoner: owlapy.owl reasoner.OWLReasoner)
     Bases: AbstractHierarchy[owlapy.owl_property.OWLObjectProperty]
     Representation of an objet property hierarchy.
     classmethod get_top_entity() \rightarrow owlapy.owl_property.OWLObjectProperty
          The most general entity in this hierarchy, which contains all the entities.
     classmethod get_bottom_entity() → owlapy.owl_property.OWLObjectProperty
          The most specific entity in this hierarchy, which contains none of the entities.
     sub_object_properties (entity: owlapy.owl_property.OWLObjectProperty, direct: bool = True)
                  → Iterable[owlapy.owl_property.OWLObjectProperty]
     super_object_properties (entity: owlapy.owl_property.OWLObjectProperty, direct: bool = True)
                  → Iterable[owlapy.owl_property.OWLObjectProperty]
     more general roles (role: owlapy.owl property.OWLObjectProperty, direct: bool = True)
                  → Iterable[owlapy.owl_property.OWLObjectProperty]
     more_special_roles (role: owlapy.owl_property.OWLObjectProperty, direct: bool = True)
                  → Iterable[owlapy.owl property.OWLObjectProperty]
     is_sub_property_of (sub_property: owlapy.owl_property.OWLObjectProperty,
                 super\_property: owlapy.owl\_property.OWLObjectProperty) \rightarrow bool
     most\_general\_roles() \rightarrow Iterable[owlapy.owl\_property.OWLObjectProperty]
     most\_special\_roles() \rightarrow Iterable[owlapy.owl\_property.OWLObjectProperty]
class owlapy.owl_hierarchy.DatatypePropertyHierarchy(
           hierarchy_down: Iterable[Tuple[owlapy.owl_property.OWLDataProperty, Iterable[owlapy.owl_property.OWLDataProperty]]
class owlapy.owl_hierarchy.DatatypePropertyHierarchy(
            reasoner: owlapy.owl_reasoner.OWLReasoner)
     Bases: AbstractHierarchy[owlapy.owl_property.OWLDataProperty]
     Representation of a data property hierarchy.
     classmethod get_top_entity() → owlapy.owl_property.OWLDataProperty
```

The most general entity in this hierarchy, which contains all the entities.

# owlapy.owl\_individual

**OWL** Individuals

# **Classes**

OWLIndividual	Represents a named or anonymous individual.
OWLNamedIndividual	Named individuals are identified using an IRI. Since they
	are given an IRI, named individuals are entities.

# **Module Contents**

```
class owlapy.owl_individual.OWLIndividual
    Bases: owlapy.owl_object.OWLObject
    Represents a named or anonymous individual.
    __slots__ = ()

class owlapy.owl_individual.OWLNamedIndividual (iri: owlapy.iri.IRI | str)
    Bases: OWLIndividual, owlapy.owl_object.OWLEntity
    Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities. IRIs from the reserved vocabulary must not be used to identify named individuals in an OWL 2 DL ontology.
    (https://www.w3.org/TR/owl2-syntax/#Named_Individuals)
    __slots__ = '__iri'
    type_index: Final = 1005
```

```
property iri: owlapy.iri.IRI

Gets the IRI of this object.

Returns
The IRI of this object.

property str

Gets the string representation of this object

Returns
The IRI as string
```

# owlapy.owl\_literal

**OWL** Literals

# **Attributes**

Literals
OWLTopObjectProperty
OWLBottomObjectProperty
OWLTopDataProperty
OWLBottomDataProperty
DoubleOWLDatatype
IntegerOWLDatatype
BooleanOWLDatatype
StringOWLDatatype StringOWLDatatype
DateOWLDatatype
DateTimeOWLDatatype
DurationOWLDatatype
TopOWLDatatype
NUMERIC_DATATYPES
TIME_DATATYPES

**OWLLiteral** 

Literals represent data values such as particular strings or integers. They are analogous to typed RDF

# **Module Contents**

```
owlapy.owl_literal.Literals
```

# class owlapy.owl\_literal.OWLLiteral

Bases: owlapy.owl annotation.OWLAnnotationValue

Literals represent data values such as particular strings or integers. They are analogous to typed RDF literals and can also be understood as individuals denoting data values. Each literal consists of a lexical form, which is a string, and a datatype.

(https://www.w3.org/TR/owl2-syntax/#Literals)

type\_index: Final = 4008

$$\texttt{get\_literal}() \rightarrow str$$

Gets the lexical value of this literal. Note that the language tag is not included.

#### Returns

The lexical value of this literal.

# $is\_boolean() \rightarrow bool$

Whether this literal is typed as boolean.

```
parse\_boolean() \rightarrow bool
```

Parses the lexical value of this literal into a bool. The lexical value of this literal should be in the lexical space of the boolean datatype ("http://www.w3.org/2001/XMLSchema#boolean").

# Returns

A bool value that is represented by this literal.

# $is\_double() \rightarrow bool$

Whether this literal is typed as double.

$$parse\_double() \rightarrow float$$

Parses the lexical value of this literal into a double. The lexical value of this literal should be in the lexical space of the double datatype ("http://www.w3.org/2001/XMLSchema#double").

#### Returns

A double value that is represented by this literal.

# $is\_integer() \rightarrow bool$

Whether this literal is typed as integer.

$$\textbf{parse\_integer}\,(\,)\,\rightarrow int$$

Parses the lexical value of this literal into an integer. The lexical value of this literal should be in the lexical space of the integer datatype ("http://www.w3.org/2001/XMLSchema#integer").

#### Returns

An integer value that is represented by this literal.

```
is\_string() \rightarrow bool
```

Whether this literal is typed as string.

```
{\tt parse\_string}\,(\,)\,\to str
```

Parses the lexical value of this literal into a string. The lexical value of this literal should be in the lexical space of the string datatype ("http://www.w3.org/2001/XMLSchema#string").

#### Returns

A string value that is represented by this literal.

```
is\_date() \rightarrow bool
```

Whether this literal is typed as date.

```
parse\_date() \rightarrow datetime.date
```

Parses the lexical value of this literal into a date. The lexical value of this literal should be in the lexical space of the date datatype ("http://www.w3.org/2001/XMLSchema#date").

# Returns

A date value that is represented by this literal.

```
is\_datetime() \rightarrow bool
```

Whether this literal is typed as dateTime.

```
parse datetime () \rightarrow datetime.datetime
```

Parses the lexical value of this literal into a datetime. The lexical value of this literal should be in the lexical space of the dateTime datatype ("http://www.w3.org/2001/XMLSchema#dateTime").

#### Returns

A datetime value that is represented by this literal.

```
\textbf{is\_duration}\,(\,)\,\rightarrow bool
```

Whether this literal is typed as duration.

```
parse\_duration() \rightarrow pandas.Timedelta
```

Parses the lexical value of this literal into a Timedelta. The lexical value of this literal should be in the lexical space of the duration datatype ("http://www.w3.org/2001/XMLSchema#duration").

#### Returns

A Timedelta value that is represented by this literal.

```
is_literal() \rightarrow bool
```

#### Returns

true if the annotation value is a literal

```
as\_literal() \rightarrow OWLLiteral
```

# Returns

if the value is a literal, returns it. Return None otherwise

```
to_python() \rightarrow Literals
```

# **abstract get\_datatype**() → owlapy.owl\_datatype.OWLDatatype

Gets the OWLDatatype which types this literal.

# Returns

The OWLDatatype that types this literal.

owlapy.owl literal.OWLTopObjectProperty: Final

```
owlapy.owl_literal.OWLTopDataProperty: Final
owlapy.owl_literal.OWLBottomDataProperty: Final
owlapy.owl_literal.OWLBottomDataProperty: Final
owlapy.owl_literal.DoubleOWLDatatype: Final
owlapy.owl_literal.IntegerOWLDatatype: Final
owlapy.owl_literal.BooleanOWLDatatype: Final
owlapy.owl_literal.StringOWLDatatype: Final
owlapy.owl_literal.DateOWLDatatype: Final
owlapy.owl_literal.DateTimeOWLDatatype: Final
owlapy.owl_literal.DurationOWLDatatype: Final
owlapy.owl_literal.TopOWLDatatype: Final
owlapy.owl_literal.TopOWLDatatype: Final
owlapy.owl_literal.NUMERIC_DATATYPES:
Final[Set[owlapy.owl_datatype.OWLDatatype]]
```

# owlapy.owl object

**OWL** Base classes

# Classes

OWLObject	Base interface for OWL objects
OWLObjectRenderer	Abstract class with a render method to render an OWL
	Object into a string.
OWLObjectParser	Abstract class with a parse method to parse a string to an
	OWL Object.
OWLNamedObject	Represents a named object for example, class, property,
	ontology etc i.e. anything that has an
OWLEntity	Represents Entities in the OWL 2 Specification.

# **Module Contents**

```
class owlapy.owl_object.OWLObject
Base interface for OWL objects
   __slots__ = ()
abstract __eq__(other)
Return self==value.
```

```
abstract __hash__()
          Return hash(self).
     abstract __repr__()
          Return repr(self).
     is\_anonymous() \rightarrow bool
class owlapy.owl_object.OWLObjectRenderer
     Abstract class with a render method to render an OWL Object into a string.
     \verb"abstract set_short_form_provider" (short_form_provider") \to None
          Configure a short form provider that shortens the OWL objects during rendering.
              Parameters
                  short_form_provider - Short form provider.
     abstract render (o: OWLObject) \rightarrow str
          Render OWL Object to string.
              Parameters
                  o - OWL Object.
              Returns
                  String rendition of OWL object.
class owlapy.owl_object.OWLObjectParser
     Abstract class with a parse method to parse a string to an OWL Object.
     abstract\ parse\_expression\ (expression\_str:\ str)\ 	o OWLObject
          Parse a string to an OWL Object.
              Parameters
                  expression_str (str) - Expression string.
              Returns
                  The OWL Object which is represented by the string.
class owlapy.owl_object.OWLNamedObject
     Bases: OWLObject, owlapy.meta_classes.HasIRI
     Represents a named object for example, class, property, ontology etc. - i.e. anything that has an IRI as its name.
     __slots__ = ()
      eq (other)
          Return self==value.
     ___1t___(other)
          Return self<value.
      hash__()
          Return hash(self).
      __repr__()
          Return repr(self).
class owlapy.owl_object.OWLEntity
     Bases: OWLNamedObject
```

Represents Entities in the OWL 2 Specification.

```
__slots__ = ()  to_string_id() \rightarrow str   is_anonymous() \rightarrow bool
```

# owlapy.owl\_ontology

**OWL** Ontology

# **Attributes**

```
logger

OWLREADY2_FACET_KEYS
```

# **Classes**

OWLOntologyID	An object that identifies an ontology. Since OWL 2, ontologies do not have to have an ontology IRI, or if they
OWLOntology	Represents an OWL 2 Ontology in the OWL 2 specification.
Ontology	Represents an OWL 2 Ontology in the OWL 2 specification.
ToOwlready2	
FromOwlready2	Map owlready2 classes to owlapy model classes.

# **Module Contents**

```
owlapy.owl_ontology.logger
```

An object that identifies an ontology. Since OWL 2, ontologies do not have to have an ontology IRI, or if they have an ontology IRI then they can optionally also have a version IRI. Instances of this OWLOntologyID class bundle identifying information of an ontology together. If an ontology doesn't have an ontology IRI then we say that it is "anonymous".

```
__slots__ = ('_ontology_iri', '_version_iri')

get_ontology_iri() \( \rightarrow owlapy.iri.IRI \) | None

Gets the ontology IRI.
```

# Returns

Ontology IRI. If the ontology is anonymous, it will return None.

```
get_version_iri() → owlapy.iri.IRI | None
```

Gets the version IRI.

#### Returns

Version IRI or None.

```
get_default_document_iri() → owlapy.iri.IRI | None
```

Gets the IRI which is used as a default for the document that contain a representation of an ontology with this ID. This will be the version IRI if there is an ontology IRI and version IRI, else it will be the ontology IRI if there is an ontology IRI but no version IRI, else it will be None if there is no ontology IRI. See Ontology Documents in the OWL 2 Structural Specification.

#### Returns

the IRI that can be used as a default for an ontology document, or None.

```
is_anonymous() → bool

__repr__()
    Return repr(self).

__eq__(other)
    Return self==value.

class owlapy.owl_ontology.OWLOntology
    Bases: owlapy.owl_object.OWLObject
```

Represents an OWL 2 Ontology in the OWL 2 specification.

An OWLOntology consists of a possibly empty set of OWLAxioms and a possibly empty set of OWLAnnotations. An ontology can have an ontology IRI which can be used to identify the ontology. If it has an ontology IRI then it may also have an ontology version IRI. Since OWL 2, an ontology need not have an ontology IRI. (See the OWL 2 Structural Specification).

An ontology cannot be modified directly. Changes must be applied via its OWLOntologyManager.

```
__slots__ = ()
type_index: Final = 1
abstract classes_in_signature() → Iterable[owlapy.class_expression.OWLClass]
Gets the classes in the signature of this object.
```

#### Returns

Classes in the signature of this object.

```
abstract data_properties_in_signature()

→ Iterable[owlapy.owl_property.OWLDataProperty]
```

Get the data properties that are in the signature of this object.

#### Returns

Data properties that are in the signature of this object.

```
abstract object_properties_in_signature()

→ Iterable[owlapy.owl_property.OWLObjectProperty]
```

A convenience method that obtains the object properties that are in the signature of this object.

# Returns

Object properties that are in the signature of this object.

# abstract individuals\_in\_signature()

→ Iterable[owlapy.owl\_individual.OWLNamedIndividual]

A convenience method that obtains the individuals that are in the signature of this object.

#### Returns

Individuals that are in the signature of this object.

# abstract equivalent\_classes\_axioms (c: owlapy.class\_expression.OWLClass)

→ Iterable[owlapy.owl\_axiom.OWLEquivalentClassesAxiom]

Gets all of the equivalent axioms in this ontology that contain the specified class as an operand.

# **Parameters**

**c** – The class for which the EquivalentClasses axioms should be retrieved.

#### Returns

EquivalentClasses axioms contained in this ontology.

```
abstract general_class_axioms() → Iterable[owlapy.owl_axiom.OWLClassAxiom]
```

# Get the general class axioms of this ontology. This includes SubClass axioms with a complex class expression

as the sub class and EquivalentClass axioms and DisjointClass axioms with only complex class expressions.

#### Returns

General class axioms contained in this ontology.

# abstract data\_property\_domain\_axioms (property: owlapy.owl\_property.OWLDataProperty) → Iterable[owlapy.owl\_axiom.OWLDataPropertyDomainAxiom]

Gets the OWLDataPropertyDomainAxiom objects where the property is equal to the specified property.

#### **Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

# Returns

The axioms matching the search.

# $\verb|abstract| \verb|data_property_range_axioms| (property: owlapy.owl_property.OWLDataProperty)| \\$

→ Iterable[owlapy.owl\_axiom.OWLDataPropertyRangeAxiom]

Gets the OWLDataPropertyRangeAxiom objects where the property is equal to the specified property.

#### **Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

# Returns

The axioms matching the search.

# abstract object\_property\_domain\_axioms(

property: owlapy.owl\_property.OWLObjectProperty)

→ Iterable[owlapy.owl\_axiom.OWLObjectPropertyDomainAxiom]

Gets the OWLObjectPropertyDomainAxiom objects where the property is equal to the specified property.

# **Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

#### Returns

The axioms matching the search.

```
abstract object_property_range_axioms(
```

property: owlapy.owl\_property.OWLObjectProperty)

→ Iterable[owlapy.owl\_axiom.OWLObjectPropertyRangeAxiom]

Gets the OWLObjectPropertyRangeAxiom objects where the property is equal to the specified property.

#### **Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

# Returns

The axioms matching the search.

```
\verb|abstract get_owl_ontology_manager()| \to \_M
```

Gets the manager that manages this ontology.

```
abstract get_ontology_id() \rightarrow OWLOntologyID
```

Gets the OWLOntologyID belonging to this object.

#### Returns

The OWLOntologyID.

```
is\_anonymous() \rightarrow bool
```

Check whether this ontology does contain an IRI or not.

Bases: OWLOntology

Represents an OWL 2 Ontology in the OWL 2 specification.

An OWLOntology consists of a possibly empty set of OWLAxioms and a possibly empty set of OWLAnnotations. An ontology can have an ontology IRI which can be used to identify the ontology. If it has an ontology IRI then it may also have an ontology version IRI. Since OWL 2, an ontology need not have an ontology IRI. (See the OWL 2 Structural Specification).

An ontology cannot be modified directly. Changes must be applied via its OWLOntologyManager.

```
__slots__ = ('_manager', '_iri', '_world', '_onto')
```

 $classes_{in\_signature}() \rightarrow Iterable[owlapy.class\_expression.OWLClass]$ 

Gets the classes in the signature of this object.

# Returns

Classes in the signature of this object.

data\_properties\_in\_signature() \rightarrow Iterable[owlapy.owl\_property.OWLDataProperty]

Get the data properties that are in the signature of this object.

# Returns

Data properties that are in the signature of this object.

```
object properties in signature() → Iterable[owlapy.owl property.OWLObjectProperty]
```

A convenience method that obtains the object properties that are in the signature of this object.

# **Returns**

Object properties that are in the signature of this object.

# $\textbf{individuals\_in\_signature} \ () \ \rightarrow Iterable[\textit{owlapy.owl\_individual}.OWLNamedIndividual}]$

A convenience method that obtains the individuals that are in the signature of this object.

#### Returns

Individuals that are in the signature of this object.

# equivalent\_classes\_axioms (c: owlapy.class\_expression.OWLClass)

→ Iterable[owlapy.owl\_axiom.OWLEquivalentClassesAxiom]

Gets all of the equivalent axioms in this ontology that contain the specified class as an operand.

#### **Parameters**

**c** – The class for which the EquivalentClasses axioms should be retrieved.

#### Returns

EquivalentClasses axioms contained in this ontology.

 $general\_class\_axioms() \rightarrow Iterable[owlapy.owl\_axiom.OWLClassAxiom]$ 

# Get the general class axioms of this ontology. This includes SubClass axioms with a complex class expression

as the sub class and EquivalentClass axioms and DisjointClass axioms with only complex class expressions.

#### Returns

General class axioms contained in this ontology.

# $\verb"get_owl_ontology_manager"() \to Ontology Manager"$

Gets the manager that manages this ontology.

# $\texttt{get\_ontology\_id}() \rightarrow OWLOntologyID$

Gets the OWLOntologyID belonging to this object.

# Returns

The OWLOntologyID.

# data\_property\_domain\_axioms (pe: owlapy.owl\_property.OWLDataProperty)

→ Iterable[owlapy.owl\_axiom.OWLDataPropertyDomainAxiom]

Gets the OWLDataPropertyDomainAxiom objects where the property is equal to the specified property.

# **Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

# Returns

The axioms matching the search.

# data\_property\_range\_axioms (pe: owlapy.owl\_property.OWLDataProperty)

→ Iterable[owlapy.owl axiom.OWLDataPropertyRangeAxiom]

Gets the OWLDataPropertyRangeAxiom objects where the property is equal to the specified property.

#### **Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

# Returns

The axioms matching the search.

# object\_property\_domain\_axioms (pe: owlapy.owl\_property.OWLObjectProperty)

→ Iterable[owlapy.owl\_axiom.OWLObjectPropertyDomainAxiom]

Gets the OWLObjectPropertyDomainAxiom objects where the property is equal to the specified property.

# **Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

# **Returns**

The axioms matching the search.

```
object_property_range_axioms (pe: owlapy.owl_property.OWLObjectProperty)
                  → Iterable[owlapy.owl_axiom.OWLObjectPropertyRangeAxiom]
          Gets the OWLObjectPropertyRangeAxiom objects where the property is equal to the specified property.
              Parameters
                  property – The property which is equal to the property of the retrieved axioms.
              Returns
                  The axioms matching the search.
     get_original_iri()
          Get the IRI argument that was used to create this ontology.
     __eq_ (other)
          Return self==value.
     __hash__()
          Return hash(self).
      _repr__()
          Return repr(self).
owlapy.owl_ontology.OWLREADY2_FACET_KEYS
class owlapy.owl_ontology.ToOwlready2(world: owlready2.World)
     __slots__ = '_world'
     abstract map_object(o: owlapy.owl_object.OWLObject)
          Map owlapy object classes.
     abstract map_concept (o: owlapy.class_expression.OWLClassExpression)
                  → owlready2.ClassConstruct | owlready2.ThingClass
          Map owlapy concept classes.
     abstract map_datarange (p: owlapy.owl_data_ranges.OWLDataRange)
                  \rightarrow owlready2.ClassConstruct | type
          Map owlapy data range classes.
class owlapy.owl_ontology.FromOwlready2
     Map owlready2 classes to owlapy model classes.
     __slots__ = ()
     abstract map_concept (c: owlready2.ClassConstruct | owlready2.ThingClass)
                  → owlapy.class_expression.OWLClassExpression
          Map concept classes.
     abstract map_datarange (p: owlready2.ClassConstruct)
                  → owlapy.owl_data_ranges.OWLDataRange
          Map data range classes.
```

# owlapy.owl\_ontology\_manager

# **Classes**

OWLOntologyChange	Represents an ontology change.
OWLOntologyManager	An OWLOntologyManager manages a set of ontologies.
	It is the main point for creating, loading and accessing
OWLImportsDeclaration	Represents an import statement in an ontology.
AddImport	Represents an ontology change where an import statement
	is added to an ontology.
OntologyManager	An OWLOntologyManager manages a set of ontologies.
	It is the main point for creating, loading and accessing

# **Module Contents**

Represents an ontology change.

```
__slots__ = ()
```

get\_ontology() → owlapy.owl\_ontology.OWLOntology

Gets the ontology that the change is/was applied to.

# Returns

The ontology that the change is applicable to.

```
class owlapy.owl_ontology_manager.OWLOntologyManager
```

An OWLOntologyManager manages a set of ontologies. It is the main point for creating, loading and accessing ontologies.

```
abstract create_ontology (iri: owlapy.iri.IRI) → owlapy.owl_ontology.OWLOntology
```

Creates a new (empty) ontology that that has the specified ontology IRI (and no version IRI).

#### **Parameters**

iri – The IRI of the ontology to be created.

# Returns

The newly created ontology, or if an ontology with the specified IRI already exists then this existing ontology will be returned.

```
abstract load_ontology (iri: owlapy.iri.IRI) → owlapy.owl_ontology.OWLOntology
```

Loads an ontology that is assumed to have the specified ontology IRI as its IRI or version IRI. The ontology IRI will be mapped to an ontology document IRI.

# **Parameters**

iri – The IRI that identifies the ontology. It is expected that the ontology will also have this IRI (although the OWL API should tolerate situations where this is not the case).

# Returns

The OWLOntology representation of the ontology that was loaded.

```
abstract apply_change (change: OWLOntologyChange)
```

A convenience method that applies just one change to an ontology. When this method is used through an OWLOntologyManager implementation, the instance used should be the one that the ontology returns through the get\_owl\_ontology\_manager() call.

#### **Parameters**

**change** – The change to be applied.

#### **Raises**

**ChangeApplied. UNSUCCESSFULLY** – if the change was not applied successfully.

A convenience method that adds a single axiom to an ontology.

#### **Parameters**

- ontology The ontology to add the axiom to.
- axiom The axiom to be added.

A convenience method that removes a single axiom from an ontology.

#### **Parameters**

- **ontology** The ontology to remove the axiom from.
- axiom The axiom to be removed.

Saves the specified ontology, using the specified document IRI to determine where/how the ontology should be saved.

# **Parameters**

- ontology The ontology to be saved.
- **document\_iri** The document IRI where the ontology should be saved to.

```
class owlapy.owl ontology manager.OWLImportsDeclaration (import iri: owlapy.iri.IRI)
```

```
Bases: owlapy.meta_classes.HasIRI
```

Represents an import statement in an ontology.

```
__slots__ = '_iri'

property iri: owlapy.iri.IRI

Gets the import IRI.
```

# Returns

The import IRI that points to the ontology to be imported. The imported ontology might have this IRI as its ontology IRI but this is not mandated. For example, an ontology with a non-resolvable ontology IRI can be deployed at a resolvable URL.

# property str: str

Gets the string representation of this object

# Returns

The IRI as string

Bases: OWLOntologyChange

Represents an ontology change where an import statement is added to an ontology.

```
__slots__ = ('_ont', '_declaration')
```

get\_import\_declaration() → OWLImportsDeclaration

Gets the import declaration that the change pertains to.

#### Returns

The import declaration.

```
class owlapy.owl_ontology_manager.OntologyManager(world_store=None)
```

Bases: OWLOntologyManager

An OWLOntologyManager manages a set of ontologies. It is the main point for creating, loading and accessing ontologies.

```
__slots__ = '_world'
```

create\_ontology (iri: owlapy.iri.IRI) → owlapy.owl\_ontology.Ontology

Creates a new (empty) ontology that that has the specified ontology IRI (and no version IRI).

#### **Parameters**

iri - The IRI of the ontology to be created.

#### Returns

The newly created ontology, or if an ontology with the specified IRI already exists then this existing ontology will be returned.

```
load\_ontology (iri: owlapy.iri.IRI) \rightarrow owlapy.owl_ontology.Ontology
```

Loads an ontology that is assumed to have the specified ontology IRI as its IRI or version IRI. The ontology IRI will be mapped to an ontology document IRI.

# **Parameters**

iri – The IRI that identifies the ontology. It is expected that the ontology will also have this IRI (although the OWL API should tolerate situations where this is not the case).

# Returns

The OWLOntology representation of the ontology that was loaded.

```
apply_change (change: OWLOntologyChange)
```

A convenience method that applies just one change to an ontology. When this method is used through an OWLOntologyManager implementation, the instance used should be the one that the ontology returns through the get\_owl\_ontology\_manager() call.

# **Parameters**

**change** – The change to be applied.

# Raises

ChangeApplied.UNSUCCESSFULLY - if the change was not applied successfully.

add\_axiom(ontology: owlapy.owl\_ontology.OWLOntology, axiom: owlapy.owl\_axiom.OWLAxiom)

A convenience method that adds a single axiom to an ontology.

#### **Parameters**

- ontology The ontology to add the axiom to.
- axiom The axiom to be added.

remove axiom (ontology: owlapy.owl\_ontology.OWLOntology, axiom: owlapy.owl\_axiom.OWLAxiom)

A convenience method that removes a single axiom from an ontology.

# **Parameters**

- **ontology** The ontology to remove the axiom from.
- axiom The axiom to be removed.

save\_ontology (ontology: owlapy.owl\_ontology.OWLOntology, document\_iri: owlapy.iri.IRI)

Saves the specified ontology, using the specified document IRI to determine where/how the ontology should be saved.

# **Parameters**

- **ontology** The ontology to be saved.
- **document\_iri** The document IRI where the ontology should be saved to.

save\_world()

Saves the actual state of the quadstore in the SQLite3 file.

# owlapy.owl\_property

**OWL** Properties

# **Classes**

OWLPropertyExpression	Represents a property or possibly the inverse of a property.
OWLObjectPropertyExpression	A high level interface to describe different types of object properties.
OWLDataPropertyExpression	A high level interface to describe different types of data properties.
OWLProperty	A base class for properties that aren't expression i.e. named properties. By definition, properties
OWLObjectProperty	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
OWLObjectInverseOf	Represents the inverse of a property expression (Object-InverseOf). An inverse object property expression
OWLDataProperty	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.

# **Module Contents**

class owlapy.owl\_property.OWLPropertyExpression
 Bases: owlapy.owl\_object.OWLObject
 Represents a property or possibly the inverse of a property.
 \_\_slots\_\_ = ()

```
is_data_property_expression() \rightarrow bool
               Returns
                   True if this is a data property.
     is\_object\_property\_expression() \rightarrow bool
               Returns
                   True if this is an object property.
     is\_owl\_top\_object\_property() \rightarrow bool
           Determines if this is the owl:topObjectProperty.
               Returns
                   topObjectProperty.
               Return type
                   True if this property is the owl
     is\_owl\_top\_data\_property() \rightarrow bool
           Determines if this is the owl:topDataProperty.
               Returns
                   topDataProperty.
               Return type
                   True if this property is the owl
class owlapy.owl_property.OWLObjectPropertyExpression
     Bases: OWLPropertyExpression
     A high level interface to describe different types of object properties.
     __slots__ = ()
     abstract get_inverse_property() → OWLObjectPropertyExpression
           Obtains the property that corresponds to the inverse of this property.
               Returns
                   The inverse of this property. Note that this property will not necessarily be in the simplest form.
     abstract get_named_property() → OWLObjectProperty
           Get the named object property used in this property expression.
               Returns
                   P if this expression is either inv(P) or P.
     is\_object\_property\_expression() \rightarrow bool
               Returns
                   True if this is an object property.
class owlapy.owl_property.OWLDataPropertyExpression
     Bases: OWLPropertyExpression
     A high level interface to describe different types of data properties.
     __slots__ = ()
```

```
is_data_property_expression()
```

#### Returns

True if this is a data property.

```
class owlapy.owl_property.OWLProperty(iri: owlapy.iri.IRI | str)
```

Bases: OWLPropertyExpression, owlapy.owl\_object.OWLEntity

A base class for properties that aren't expression i.e. named properties. By definition, properties are either data properties or object properties.

```
__slots__ = '_iri'
```

# property str: str

Gets the string representation of this object

#### Returns

The IRI as string

property iri: owlapy.iri.IRI

Gets the IRI of this object.

#### Returns

The IRI of this object.

```
class owlapy.owl_property.OWLObjectProperty(iri: owlapy.iri.IRI | str)
```

Bases: OWLObjectPropertyExpression, OWLProperty

Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.

(https://www.w3.org/TR/owl2-syntax/#Object\_Properties)

```
__slots__ = '_iri'
```

type\_index: Final = 1002

```
\verb"get_named_property"() \to OWLObjectProperty"
```

Get the named object property used in this property expression.

# Returns

P if this expression is either inv(P) or P.

```
\texttt{get\_inverse\_property}() \rightarrow OWLObjectInverseOf
```

Obtains the property that corresponds to the inverse of this property.

#### Returns

The inverse of this property. Note that this property will not necessarily be in the simplest form.

```
is\_owl\_top\_object\_property() \rightarrow bool
```

Determines if this is the owl:topObjectProperty.

#### Returns

topObjectProperty.

# Return type

True if this property is the owl

```
class owlapy.owl_property.OWLObjectInverseOf(property: OWLObjectProperty)
```

Bases: OWLObjectPropertyExpression

Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression ObjectInverseOf(P) connects an individual I1 with I2 if and only if the object property P connects I2 with I1. This can

be used to refer to the inverse of a property, without actually naming the property. For example, consider the property hasPart, the inverse property of hasPart (isPartOf) can be referred to using this interface inverseOf(hasPart), which can be used in restrictions e.g. inverseOf(hasPart) some Car refers to the set of things that are part of at least one car.

```
(https://www.w3.org/TR/owl2-syntax/#Inverse_Object_Properties)
```

```
__slots__ = '_inverse_property'

type_index: Final = 1003

get_inverse() \( \rightarrow \text{OWLObjectProperty} \)
```

Gets the property expression that this is the inverse of.

# Returns

The object property expression such that this object property expression is an inverse of it.

```
get_inverse_property() → OWLObjectProperty
```

Obtains the property that corresponds to the inverse of this property.

#### Returns

The inverse of this property. Note that this property will not necessarily be in the simplest form.

```
\texttt{get\_named\_property}() \rightarrow OWLObjectProperty
```

Get the named object property used in this property expression.

#### Returns

P if this expression is either inv(P) or P.

```
__repr__()
    Return repr(self).
__eq__(other)
    Return self==value.
__hash__()
```

Return hash(self).

class owlapy.owl\_property.OWLDataProperty(iri: owlapy.iri.IRI | str)

Bases: OWLDataPropertyExpression, OWLProperty

Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals. In some knowledge representation systems, functional data properties are called attributes.

(https://www.w3.org/TR/owl2-syntax/#Data\_Properties)

```
__slots__ = '_iri'
type_index: Final = 1004
is_owl_top_data_property() -> bool
```

Determines if this is the owl:topDataProperty.

#### **Returns**

topDataProperty.

# Return type

True if this property is the owl

# owlapy.owl reasoner

**OWL** Reasoner

# **Attributes**

logger			

# **Classes**

OWLReasoner	An OWLReasoner reasons over a set of axioms (the set of reasoner axioms) that is based on the imports closure of
BaseReasoner	Enumeration class for base reasoner when calling sync_reasoner.
OWLReasonerEx	Extra convenience methods for OWL Reasoners
OntologyReasoner	Extra convenience methods for OWL Reasoners
FastInstanceCheckerReasoner	Tries to check instances fast (but maybe incomplete).
SyncReasoner	Extra convenience methods for OWL Reasoners

# **Module Contents**

```
owlapy.owl_reasoner.logger
```

class owlapy.owl\_reasoner.OWLReasoner(ontology: owlapy.owl\_ontology.OWLOntology)

An OWLReasoner reasons over a set of axioms (the set of reasoner axioms) that is based on the imports closure of a particular ontology - the "root" ontology.

```
__slots__ = ()
```

**abstract** data\_property\_domains (pe: owlapy.owl\_property.OWLDataProperty, direct: bool = False) → Iterable[owlapy.class\_expression.OWLClassExpression]

Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.

# **Parameters**

- **pe** The property expression whose domains are to be retrieved.
- **direct** Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

# Returns

Let  $N = equivalent\_classes(DataSomeValuesFrom(pe rdfs:Literal))$ . If direct is True: then if N is not empty then the return value is N, else the return value is the result of super\\_classes(DataSomeValuesFrom(pe rdfs:Literal), true). If direct is False: then the result of super\\_classes(DataSomeValuesFrom(pe rdfs:Literal), false) together with N if N is non-empty. (Note, rdfs:Literal is the top datatype).

```
abstract object_property_domains (pe: owlapy.owl_property.OWLObjectProperty, direct: bool = False) → Iterable[owlapy.class expression.OWLClassExpression]
```

Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.

#### **Parameters**

- **pe** The property expression whose domains are to be retrieved.
- **direct** Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

#### Returns

Let  $N = equivalent\_classes(ObjectSomeValuesFrom(pe owl:Thing))$ . If direct is True: then if N is not empty then the return value is N, else the return value is the result of super\\_classes(ObjectSomeValuesFrom(pe owl:Thing), true). If direct is False: then the result of super\\_classes(ObjectSomeValuesFrom(pe owl:Thing), false) together with N if N is non-empty.

```
abstract object_property_ranges (pe: owlapy.owl_property.OWLObjectProperty, direct: bool = False) → Iterable[owlapy.class_expression.OWLClassExpression]
```

Gets the class expressions that are the direct or indirect ranges of this property with respect to the imports closure of the root ontology.

# **Parameters**

- **pe** The property expression whose ranges are to be retrieved.
- **direct** Specifies if the direct ranges should be retrieved (True), or if all ranges should be retrieved (False).

# **Returns**

Let  $N = equivalent\_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe) owl:Thing))$ . If direct is True: then if N is not empty then the return value is N, else the return value is the result of super\\_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe) owl:Thing), true). If direct is False: then the result of super\\_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe) owl:Thing), false) together with N if N is non-empty.

```
abstract equivalent_classes (ce: owlapy.class_expression.OWLClassExpression, only_named: bool = True) \rightarrow Iterable[owlapy.class_expression.OWLClassExpression]
```

Gets the class expressions that are equivalent to the specified class expression with respect to the set of reasoner axioms.

#### **Parameters**

- $\bullet$  ce The class expression whose equivalent classes are to be retrieved.
- **only\_named** Whether to only retrieve named equivalent classes or also complex class expressions.

# Returns

All class expressions C where the root ontology imports closure entails EquivalentClasses(ce C). If ce is not a class name (i.e. it is an anonymous class expression) and there are no such classes C then there will be no result. If ce is unsatisfiable with respect to the set of reasoner axioms then owl:Nothing, i.e. the bottom node, will be returned.

# abstract disjoint\_classes (ce: owlapy.class\_expression.OWLClassExpression, only named: bool = True) → Iterable[owlapy.class expression.OWLClassExpression]

Gets the class expressions that are disjoint with specified class expression with respect to the set of reasoner axioms.

#### **Parameters**

- **ce** The class expression whose disjoint classes are to be retrieved.
- only\_named Whether to only retrieve named disjoint classes or also complex class expressions.

#### Returns

All class expressions D where the set of reasoner axioms entails EquivalentClasses(D Object-ComplementOf(ce)) or StrictSubClassOf(D ObjectComplementOf(ce)).

# abstract different\_individuals(ind: owlapy.owl\_individual.OWLNamedIndividual)

→ Iterable[owlapy.owl\_individual.OWLNamedIndividual]

Gets the individuals that are different from the specified individual with respect to the set of reasoner axioms.

# **Parameters**

ind – The individual whose different individuals are to be retrieved.

#### Returns

All individuals x where the set of reasoner axioms entails DifferentIndividuals(ind x).

# abstract same\_individuals(ind: owlapy.owl\_individual.OWLNamedIndividual)

→ Iterable[owlapy.owl\_individual.OWLNamedIndividual]

Gets the individuals that are the same as the specified individual with respect to the set of reasoner axioms.

# **Parameters**

ind – The individual whose same individuals are to be retrieved.

#### Returns

All individuals x where the root ontology imports closure entails SameIndividual(ind x).

# abstract equivalent\_object\_properties(

op: owlapy.owl\_property.OWLObjectPropertyExpression)

→ Iterable[owlapy.owl\_property.OWLObjectPropertyExpression]

Gets the simplified object properties that are equivalent to the specified object property with respect to the set of reasoner axioms.

# **Parameters**

**op** – The object property whose equivalent object properties are to be retrieved.

#### Returns

All simplified object properties e where the root ontology imports closure entails EquivalentObjectProperties(op e). If op is unsatisfiable with respect to the set of reasoner axioms then owl:bottomDataProperty will be returned.

# abstract equivalent\_data\_properties(dp: owlapy.owl\_property.OWLDataProperty)

→ Iterable[owlapy.owl\_property.OWLDataProperty]

Gets the data properties that are equivalent to the specified data property with respect to the set of reasoner axioms.

# **Parameters**

**dp** – The data property whose equivalent data properties are to be retrieved.

#### Returns

All data properties e where the root ontology imports closure entails EquivalentDataProp-

erties(dp e). If dp is unsatisfiable with respect to the set of reasoner axioms then owl:bottomDataProperty will be returned.

# abstract data\_property\_values (ind: owlapy.owl\_individual.OWLNamedIndividual, pe: owlapy.owl\_property.OWLDataProperty, direct: bool = True) $\rightarrow$ Iterable[owlapy.owl\_literal.OWLLiteral]

Gets the data property values for the specified individual and data property expression.

# **Parameters**

- ind The individual that is the subject of the data property values.
- **pe** The data property expression whose values are to be retrieved for the specified individual.
- **direct** Specifies if the direct values should be retrieved (True), or if all values should be retrieved (False), so that sub properties are taken into account.

#### Returns

A set of OWLLiterals containing literals such that for each literal l in the set, the set of reasoner axioms entails DataPropertyAssertion(pe ind l).

```
\begin{tabular}{ll} \textbf{abstract} & \textbf{object\_property\_values} & (ind: owlapy.owl\_individual.OWLNamedIndividual, \\ pe: owlapy.owl\_property.OWLObjectPropertyExpression, direct: bool = True) \\ & \rightarrow \textbf{Iterable}[owlapy.owl\_individual.OWLNamedIndividual]} \end{tabular}
```

Gets the object property values for the specified individual and object property expression.

#### **Parameters**

- **ind** The individual that is the subject of the object property values.
- **pe** The object property expression whose values are to be retrieved for the specified individual.
- **direct** Specifies if the direct values should be retrieved (True), or if all values should be retrieved (False), so that sub properties are taken into account.

# Returns

The named individuals such that for each individual j, the set of reasoner axioms entails ObjectPropertyAssertion(pe ind j).

```
abstract flush() \rightarrow None
```

Flushes any changes stored in the buffer, which causes the reasoner to take into consideration the changes the current root ontology specified by the changes.

```
abstract instances (ce: owlapy.class_expression.OWLClassExpression, direct: bool = False)

→ Iterable[owlapy.owl_individual.OWLNamedIndividual]
```

Gets the individuals which are instances of the specified class expression.

#### **Parameters**

- **ce** The class expression whose instances are to be retrieved.
- **direct** Specifies if the direct instances should be retrieved (True), or if all instances should be retrieved (False).

#### Returns

If direct is True, each named individual j where the set of reasoner axioms entails DirectClassAssertion(ce, j). If direct is False, each named individual j where the set of reasoner axioms entails ClassAssertion(ce, j). If ce is unsatisfiable with respect to the set of reasoner axioms then nothing returned.

```
abstract sub_classes (ce: owlapy.class_expression.OWLClassExpression, direct: bool = False, only_named: bool = True) \rightarrow Iterable[owlapy.class_expression.OWLClassExpression]
```

Gets the set of named classes that are the strict (potentially direct) subclasses of the specified class expression with respect to the reasoner axioms.

#### **Parameters**

- ce The class expression whose strict (direct) subclasses are to be retrieved.
- **direct** Specifies if the direct subclasses should be retrieved (True) or if the all subclasses (descendant) classes should be retrieved (False).
- only\_named Whether to only retrieve named sub-classes or also complex class expressions.

#### Returns

If direct is True, each class C where reasoner axioms entails DirectSubClassOf(C, ce). If direct is False, each class C where reasoner axioms entails StrictSubClassOf(C, ce). If ce is equivalent to owl:Nothing then nothing will be returned.

# abstract disjoint\_object\_properties(

op: owlapy.owl\_property.OWLObjectPropertyExpression)

→ Iterable[owlapy.owl\_property.OWLObjectPropertyExpression]

Gets the simplified object properties that are disjoint with the specified object property with respect to the set of reasoner axioms.

### **Parameters**

op – The object property whose disjoint object properties are to be retrieved.

### Returns

All simplified object properties e where the root ontology imports closure entails EquivalentObjectProperties(e ObjectPropertyComplementOf(op)) or StrictSubObjectPropertyOf(e ObjectPropertyComplementOf(op)).

```
abstract disjoint_data_properties (dp: owlapy.owl_property.OWLDataProperty)

→ Iterable[owlapy.owl_property.OWLDataProperty]
```

Gets the data properties that are disjoint with the specified data property with respect to the set of reasoner axioms.

### **Parameters**

**dp** – The data property whose disjoint data properties are to be retrieved.

#### Returns

All data properties e where the root ontology imports closure entails EquivalentDataProperties(e DataPropertyComplementOf(dp)) or StrictSubDataPropertyOf(e DataPropertyComplementOf(dp)).

```
abstract sub_data_properties(dp: owlapy.owl_property.OWLDataProperty, direct: bool = False) → Iterable[owlapy.owl_property.OWLDataProperty]
```

Gets the set of named data properties that are the strict (potentially direct) subproperties of the specified data property expression with respect to the imports closure of the root ontology.

# **Parameters**

- dp The data property whose strict (direct) subproperties are to be retrieved.
- **direct** Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

### Returns

If direct is True, each property P where the set of reasoner axioms entails DirectSubDataPropertyOf(P, pe). If direct is False, each property P where the set of reasoner axioms entails StrictSubDataPropertyOf(P, pe). If pe is equivalent to owl:bottomDataProperty then nothing will be returned.

```
abstract super_data_properties (dp: owlapy.owl_property.OWLDataProperty, direct: bool = False) → Iterable[owlapy.owl_property.OWLDataProperty]
```

Gets the stream of data properties that are the strict (potentially direct) super properties of the specified data property with respect to the imports closure of the root ontology.

#### **Parameters**

- **dp** (OWLDataProperty) The data property whose super properties are to be retrieved.
- **direct** (bool) Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

#### Returns

Iterable of super properties.

```
abstract sub_object_properties (op: owlapy.owl_property.OWLObjectPropertyExpression, direct: bool = False) → Iterable[owlapy.owl_property.OWLObjectPropertyExpression]
```

Gets the stream of simplified object property expressions that are the strict (potentially direct) subproperties of the specified object property expression with respect to the imports closure of the root ontology.

### **Parameters**

- op The object property expression whose strict (direct) subproperties are to be retrieved.
- **direct** Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

# Returns

If direct is True, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails DirectSubObjectPropertyOf(P, pe). If direct is False, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails StrictSubObjectPropertyOf(P, pe). If pe is equivalent to owl:bottomObjectProperty then nothing will be returned.

```
abstract super_object_properties (op: owlapy.owl_property.OWLObjectPropertyExpression, direct: bool = False) → Iterable[owlapy.owl_property.OWLObjectPropertyExpression]
```

Gets the stream of object properties that are the strict (potentially direct) super properties of the specified object property with respect to the imports closure of the root ontology.

#### **Parameters**

- **op** (OWLObjectPropertyExpression) The object property expression whose super properties are to be retrieved.
- **direct** (bool) Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

# Returns

Iterable of super properties.

```
\label{eq:abstract} \textbf{types} (\textit{ind: owlapy.owl\_individual.OWLNamedIndividual, direct: bool = False}) \\ \rightarrow \textbf{Iterable}[\textit{owlapy.class\_expression.OWLClass}]
```

Gets the named classes which are (potentially direct) types of the specified named individual.

# **Parameters**

- ind The individual whose types are to be retrieved.
- **direct** Specifies if the direct types should be retrieved (True), or if all types should be retrieved (False).

#### Returns

If direct is True, each named class C where the set of reasoner axioms entails DirectClassAssertion(C, ind). If direct is False, each named class C where the set of reasoner axioms entails ClassAssertion(C, ind).

```
abstract get_root_ontology() → owlapy.owl_ontology.OWLOntology
```

Gets the "root" ontology that is loaded into this reasoner. The reasoner takes into account the axioms in this ontology and its import's closure.

```
abstract is_isolated()
```

Return True if this reasoner is using an isolated ontology.

abstract super\_classes (ce: owlapy.class\_expression.OWLClassExpression, direct: bool = False, only\_named: bool = True)  $\rightarrow$  Iterable[owlapy.class\_expression.OWLClassExpression]

Gets the stream of named classes that are the strict (potentially direct) super classes of the specified class expression with respect to the imports closure of the root ontology.

### **Parameters**

- **ce** The class expression whose strict (direct) super classes are to be retrieved.
- **direct** Specifies if the direct super classes should be retrieved (True) or if the all super classes (ancestors) classes should be retrieved (False).
- only\_named Whether to only retrieve named super classes or also complex class expressions.

#### Returns

If direct is True, each class C where the set of reasoner axioms entails DirectSubClassOf(ce,

- C). If direct is False, each class C where set of reasoner axioms entails StrictSubClassOf(ce,
- C). If ce is equivalent to owl: Thing then nothing will be returned.

```
class owlapy.owl_reasoner.BaseReasoner
```

Bases: enum.Enum

Enumeration class for base reasoner when calling sync\_reasoner.

#### PELLET

Pellet base reasoner.

# HERMIT

HermiT base reasoner.

PELLET

HERMIT

class owlapy.owl\_reasoner.OWLReasonerEx (ontology: owlapy.owl\_ontology.OWLOntology)

Bases: OWLReasoner

Extra convenience methods for OWL Reasoners

```
\label{lem:data_property_anges} \begin{subarray}{l} $\text{data\_property.owl\_property.owl\_property.owl\_property, direct: bool = False)} \\ \to & \text{Iterable}[owlapy.owl\_data\_ranges.owl\_bataRange] \end{subarray}
```

Gets the data ranges that are the direct or indirect ranges of this property with respect to the imports closure of the root ontology.

### **Parameters**

- **pe** The property expression whose ranges are to be retrieved.
- **direct** Specifies if the direct ranges should be retrieved (True), or if all ranges should be retrieved (False).

Returns:

```
all_data_property_values (pe: owlapy.owl_property.OWLDataProperty, direct: bool = True)

→ Iterable[owlapy.owl_literal.OWLLiteral]
```

Gets all values for the given data property expression that appear in the knowledge base.

#### **Parameters**

- pe The data property expression whose values are to be retrieved
- **direct** Specifies if only the direct values of the data property pe should be retrieved (True), or if the values of sub properties of pe should be taken into account (False).

#### Returns

A set of OWLLiterals containing literals such that for each literal l in the set, the set of reasoner axioms entails DataPropertyAssertion(pe ind l) for any ind.

```
\label{lower_lower} \begin{split} & \texttt{ind\_data\_properties} \ (\textit{ind: owlapy.owl\_individual.OWLNamedIndividual, direct: bool = True}) \\ & \rightarrow \\ & \texttt{Iterable}[\textit{owlapy.owl\_property.OWLDataProperty}] \end{split}
```

Gets all data properties for the given individual that appear in the knowledge base.

### **Parameters**

- ind The named individual whose data properties are to be retrieved
- **direct** Specifies if the direct data properties should be retrieved (True), or if all data properties should be retrieved (False), so that sub properties are taken into account.

### **Returns**

All data properties pe where the set of reasoner axioms entails DataPropertyAssertion(pe ind l) for atleast one l.

Gets all object properties for the given individual that appear in the knowledge base.

# **Parameters**

- ind The named individual whose object properties are to be retrieved
- **direct** Specifies if the direct object properties should be retrieved (True), or if all object properties should be retrieved (False), so that sub properties are taken into account.

### Returns

All data properties pe where the set of reasoner axioms entails ObjectPropertyAssertion(pe ind ind2) for atleast one ind2.

Extra convenience methods for OWL Reasoners

```
__slots__ = ('_ontology', '_world')
```

Add or remove axioms to the isolated ontology that the reasoner is using.

#### **Parameters**

- axioms\_to\_add (List[OWLAxiom]) Axioms to add to the isolated ontology.
- axioms\_to\_remove (List[OWLAxiom]) Axioms to remove from the isolated ontology.

data\_property\_domains (pe: owlapy.owl\_property.OWLDataProperty, direct: bool = False)

→ Iterable[owlapy.class\_expression.OWLClassExpression]

Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.

#### **Parameters**

- **pe** The property expression whose domains are to be retrieved.
- **direct** Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

#### Returns

Let  $N = equivalent\_classes(DataSomeValuesFrom(pe rdfs:Literal))$ . If direct is True: then if N is not empty then the return value is N, else the return value is the result of super\\_classes(DataSomeValuesFrom(pe rdfs:Literal), true). If direct is False: then the result of super\\_classes(DataSomeValuesFrom(pe rdfs:Literal), false) together with N if N is non-empty. (Note, rdfs:Literal is the top datatype).

Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.

### **Parameters**

- **pe** The property expression whose domains are to be retrieved.
- **direct** Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

# Returns

Let  $N = equivalent\_classes(ObjectSomeValuesFrom(pe owl:Thing))$ . If direct is True: then if N is not empty then the return value is N, else the return value is the result of super\\_classes(ObjectSomeValuesFrom(pe owl:Thing), true). If direct is False: then the result of super\\_classes(ObjectSomeValuesFrom(pe owl:Thing), false) together with N if N is nonempty.

```
object_property_ranges (pe: owlapy.owl_property.OWLObjectProperty, direct: bool = False)

→ Iterable[owlapy.class_expression.OWLClassExpression]
```

Gets the class expressions that are the direct or indirect ranges of this property with respect to the imports closure of the root ontology.

# **Parameters**

• **pe** – The property expression whose ranges are to be retrieved.

• **direct** – Specifies if the direct ranges should be retrieved (True), or if all ranges should be retrieved (False).

#### Returns

Let N = equivalent\_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe) owl:Thing)). If direct is True: then if N is not empty then the return value is N, else the return value is the result of super\_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe) owl:Thing), true). If direct is False: then the result of super\_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe) owl:Thing), false) together with N if N is non-empty.

equivalent\_classes (ce: owlapy.class\_expression.OWLClassExpression, only\_named: bool = True)

→ Iterable[owlapy.class\_expression.OWLClassExpression]

Gets the class expressions that are equivalent to the specified class expression with respect to the set of reasoner axioms.

### **Parameters**

- **ce** The class expression whose equivalent classes are to be retrieved.
- **only\_named** Whether to only retrieve named equivalent classes or also complex class expressions.

#### Returns

All class expressions C where the root ontology imports closure entails EquivalentClasses(ce C). If ce is not a class name (i.e. it is an anonymous class expression) and there are no such classes C then there will be no result. If ce is unsatisfiable with respect to the set of reasoner axioms then owl:Nothing, i.e. the bottom node, will be returned.

disjoint\_classes (ce: owlapy.class\_expression.OWLClassExpression, only\_named: bool = True)

→ Iterable[owlapy.class expression.OWLClassExpression]

Gets the class expressions that are disjoint with specified class expression with respect to the set of reasoner axioms.

# **Parameters**

- **ce** The class expression whose disjoint classes are to be retrieved.
- only\_named Whether to only retrieve named disjoint classes or also complex class expressions.

#### **Returns**

All class expressions D where the set of reasoner axioms entails EquivalentClasses(D ObjectComplementOf(ce)) or StrictSubClassOf(D ObjectComplementOf(ce)).

different\_individuals (ind: owlapy.owl\_individual.OWLNamedIndividual)

 $\rightarrow$  Iterable[owlapy.owl\_individual.OWLNamedIndividual]

Gets the individuals that are different from the specified individual with respect to the set of reasoner axioms.

#### **Parameters**

ind – The individual whose different individuals are to be retrieved.

# Returns

All individuals x where the set of reasoner axioms entails DifferentIndividuals(ind x).

 $same\_individuals$  (ind: owlapy.owl\_individual.OWLNamedIndividual)

→ Iterable[owlapy.owl\_individual.OWLNamedIndividual]

Gets the individuals that are the same as the specified individual with respect to the set of reasoner axioms.

# **Parameters**

ind – The individual whose same individuals are to be retrieved.

### Returns

All individuals x where the root ontology imports closure entails SameIndividual(ind x).

```
data_property_values (ind: owlapy.owl_individual.OWLNamedIndividual,
```

pe: owlapy.owl\_property.OWLDataProperty, direct: bool = True)

→ Iterable[owlapy.owl\_literal.OWLLiteral]

Gets the data property values for the specified individual and data property expression.

# **Parameters**

- ind The individual that is the subject of the data property values.
- pe The data property expression whose values are to be retrieved for the specified individual.
- **direct** Specifies if the direct values should be retrieved (True), or if all values should be retrieved (False), so that sub properties are taken into account.

# Returns

A set of OWLLiterals containing literals such that for each literal l in the set, the set of reasoner axioms entails DataPropertyAssertion(pe ind l).

all\_data\_property\_values (pe: owlapy.owl\_property.OWLDataProperty, direct: bool = True)

→ Iterable[owlapy.owl\_literal.OWLLiteral]

Gets all values for the given data property expression that appear in the knowledge base.

## **Parameters**

- pe The data property expression whose values are to be retrieved
- **direct** Specifies if only the direct values of the data property pe should be retrieved (True), or if the values of sub properties of pe should be taken into account (False).

#### Returns

A set of OWLLiterals containing literals such that for each literal l in the set, the set of reasoner axioms entails DataPropertyAssertion(pe ind l) for any ind.

object\_property\_values (ind: owlapy.owl\_individual.OWLNamedIndividual,

pe: owlapy.owl\_property.OWLObjectPropertyExpression, direct: bool = False)

→ Iterable[owlapy.owl\_individual.OWLNamedIndividual]

Gets the object property values for the specified individual and object property expression.

# **Parameters**

- ind The individual that is the subject of the object property values.
- pe The object property expression whose values are to be retrieved for the specified individual.
- **direct** Specifies if the direct values should be retrieved (True), or if all values should be retrieved (False), so that sub properties are taken into account.

#### Returns

The named individuals such that for each individual j, the set of reasoner axioms entails ObjectPropertyAssertion(pe ind j).

# $\textbf{flush}\,(\,)\,\to None$

Flushes any changes stored in the buffer, which causes the reasoner to take into consideration the changes the current root ontology specified by the changes.

Gets the individuals which are instances of the specified class expression.

#### **Parameters**

- **ce** The class expression whose instances are to be retrieved.
- **direct** Specifies if the direct instances should be retrieved (True), or if all instances should be retrieved (False).

### Returns

If direct is True, each named individual j where the set of reasoner axioms entails DirectClassAssertion(ce, j). If direct is False, each named individual j where the set of reasoner axioms entails ClassAssertion(ce, j). If ce is unsatisfiable with respect to the set of reasoner axioms then nothing returned.

```
sub\_classes (ce: owlapy.class_expression.OWLClassExpression, direct: bool = False, only_named: bool = True) \rightarrow Iterable[owlapy.class_expression.OWLClassExpression]
```

Gets the set of named classes that are the strict (potentially direct) subclasses of the specified class expression with respect to the reasoner axioms.

### **Parameters**

- ce The class expression whose strict (direct) subclasses are to be retrieved.
- **direct** Specifies if the direct subclasses should be retrieved (True) or if the all subclasses (descendant) classes should be retrieved (False).
- only\_named Whether to only retrieve named sub-classes or also complex class expressions.

#### Returns

If direct is True, each class C where reasoner axioms entails DirectSubClassOf(C, ce). If direct is False, each class C where reasoner axioms entails StrictSubClassOf(C, ce). If ce is equivalent to owl:Nothing then nothing will be returned.

```
super\_classes (ce: owlapy.class_expression.OWLClassExpression, direct: bool = False, only named: bool = True) \rightarrow Iterable[owlapy.class expression.OWLClassExpression]
```

Gets the stream of named classes that are the strict (potentially direct) super classes of the specified class expression with respect to the imports closure of the root ontology.

# **Parameters**

- ce The class expression whose strict (direct) super classes are to be retrieved.
- **direct** Specifies if the direct super classes should be retrieved (True) or if the all super classes (ancestors) classes should be retrieved (False).
- only\_named Whether to only retrieve named super classes or also complex class expressions.

# Returns

If direct is True, each class C where the set of reasoner axioms entails DirectSubClassOf(ce,

- C). If direct is False, each class C where set of reasoner axioms entails StrictSubClassOf(ce,
- C). If ce is equivalent to owl:Thing then nothing will be returned.

```
equivalent_object_properties (op: owlapy.owl_property.OWLObjectPropertyExpression)

→ Iterable[owlapy.owl_property.OWLObjectPropertyExpression]
```

Gets the simplified object properties that are equivalent to the specified object property with respect to the set of reasoner axioms.

### **Parameters**

op – The object property whose equivalent object properties are to be retrieved.

#### Returns

All simplified object properties e where the root ontology imports closure entails EquivalentObjectProperties(op e). If op is unsatisfiable with respect to the set of reasoner axioms then owl:bottomDataProperty will be returned.

# equivalent\_data\_properties (dp: owlapy.owl\_property.OWLDataProperty)

→ Iterable[owlapy.owl\_property.OWLDataProperty]

Gets the data properties that are equivalent to the specified data property with respect to the set of reasoner axioms.

#### **Parameters**

**dp** – The data property whose equivalent data properties are to be retrieved.

#### Returns

All data properties e where the root ontology imports closure entails EquivalentDataProperties(dp e). If dp is unsatisfiable with respect to the set of reasoner axioms then owl:bottomDataProperty will be returned.

# disjoint\_object\_properties (op: owlapy.owl\_property.OWLObjectPropertyExpression)

→ Iterable[owlapy.owl\_property.OWLObjectPropertyExpression]

Gets the simplified object properties that are disjoint with the specified object property with respect to the set of reasoner axioms.

#### **Parameters**

op – The object property whose disjoint object properties are to be retrieved.

#### Returns

All simplified object properties e where the root ontology imports closure entails EquivalentObjectProperties(e ObjectPropertyComplementOf(op)) or StrictSubObjectPropertyOf(e ObjectPropertyComplementOf(op)).

# disjoint\_data\_properties (dp: owlapy.owl\_property.OWLDataProperty)

→ Iterable[owlapy.owl\_property.OWLDataProperty]

Gets the data properties that are disjoint with the specified data property with respect to the set of reasoner axioms.

#### **Parameters**

**dp** – The data property whose disjoint data properties are to be retrieved.

# Returns

All data properties e where the root ontology imports closure entails EquivalentDataProperties(e DataPropertyComplementOf(dp)) or StrictSubDataPropertyOf(e DataPropertyComplementOf(dp)).

# $\verb|super_data_properties| (dp: owlapy.owl_property.OWLDataProperty, direct: bool = False)|$

→ Iterable[owlapy.owl\_property.OWLDataProperty]

Gets the stream of data properties that are the strict (potentially direct) super properties of the specified data property with respect to the imports closure of the root ontology.

### **Parameters**

- **dp** (OWLDataProperty) The data property whose super properties are to be retrieved.
- **direct** (bool) Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

# Returns

Iterable of super properties.

```
\begin{tabular}{ll} \textbf{sub\_data\_properties} (dp: owlapy.owl\_property.OWLDataProperty, direct: bool = False) \\ &\rightarrow \textbf{Iterable}[owlapy.owl\_property.OWLDataProperty] \\ \end{tabular}
```

Gets the set of named data properties that are the strict (potentially direct) subproperties of the specified data property expression with respect to the imports closure of the root ontology.

#### **Parameters**

- **dp** The data property whose strict (direct) subproperties are to be retrieved.
- **direct** Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

#### Returns

If direct is True, each property P where the set of reasoner axioms entails DirectSubDataPropertyOf(P, pe). If direct is False, each property P where the set of reasoner axioms entails StrictSubDataPropertyOf(P, pe). If pe is equivalent to owl:bottomDataProperty then nothing will be returned.

Gets the stream of object properties that are the strict (potentially direct) super properties of the specified object property with respect to the imports closure of the root ontology.

#### **Parameters**

- **op** (OWLObjectPropertyExpression) The object property expression whose super properties are to be retrieved.
- **direct** (bool) Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

### Returns

Iterable of super properties.

Gets the stream of simplified object property expressions that are the strict (potentially direct) subproperties of the specified object property expression with respect to the imports closure of the root ontology.

#### **Parameters**

- op The object property expression whose strict (direct) subproperties are to be retrieved.
- **direct** Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

#### Returns

If direct is True, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails DirectSubObjectPropertyOf(P, pe). If direct is False, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails StrictSubObjectPropertyOf(P, pe). If pe is equivalent to owl:bottomObjectProperty then nothing will be returned.

```
\label{eq:continuity}  \textbf{types} \ (\textit{ind: owlapy.owl\_individual.OWLNamedIndividual, direct: bool = False}) \\ \rightarrow \text{Iterable}[\textit{owlapy.class\_expression.OWLClass}]
```

Gets the named classes which are (potentially direct) types of the specified named individual.

# **Parameters**

• ind – The individual whose types are to be retrieved.

• **direct** – Specifies if the direct types should be retrieved (True), or if all types should be retrieved (False).

#### Returns

If direct is True, each named class C where the set of reasoner axioms entails DirectClassAssertion(C, ind). If direct is False, each named class C where the set of reasoner axioms entails ClassAssertion(C, ind).

```
get_root_ontology() → owlapy.owl_ontology.OWLOntology
```

Gets the "root" ontology that is loaded into this reasoner. The reasoner takes into account the axioms in this ontology and its import's closure.

```
is_isolated()
```

Return True if this reasoner is using an isolated ontology.

Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.

#### **Parameters**

- **pe** The property expression whose domains are to be retrieved.
- **direct** Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

# Returns

Let  $N = equivalent\_classes(DataSomeValuesFrom(pe rdfs:Literal))$ . If direct is True: then if N is not empty then the return value is N, else the return value is the result of super\\_classes(DataSomeValuesFrom(pe rdfs:Literal), true). If direct is False: then the result of super\\_classes(DataSomeValuesFrom(pe rdfs:Literal), false) together with N if N is non-empty. (Note, rdfs:Literal is the top datatype).

Gets the data ranges that are the direct or indirect ranges of this property with respect to the imports closure of the root ontology.

# **Parameters**

- **pe** The property expression whose ranges are to be retrieved.
- **direct** Specifies if the direct ranges should be retrieved (True), or if all ranges should be retrieved (False).

Returns:

Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.

#### **Parameters**

- **pe** The property expression whose domains are to be retrieved.
- **direct** Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

#### Returns

Let  $N = equivalent\_classes(ObjectSomeValuesFrom(pe owl:Thing))$ . If direct is True: then if N is not empty then the return value is N, else the return value is the result of super\\_classes(ObjectSomeValuesFrom(pe owl:Thing), true). If direct is False: then the result of super\\_classes(ObjectSomeValuesFrom(pe owl:Thing), false) together with N if N is non-empty.

object\_property\_ranges (pe: owlapy.owl\_property.OWLObjectProperty, direct: bool = False)

→ Iterable[owlapy.class\_expression.OWLClassExpression]

Gets the class expressions that are the direct or indirect ranges of this property with respect to the imports closure of the root ontology.

### **Parameters**

- **pe** The property expression whose ranges are to be retrieved.
- **direct** Specifies if the direct ranges should be retrieved (True), or if all ranges should be retrieved (False).

# Returns

Let  $N = equivalent\_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe) owl:Thing))$ . If direct is True: then if N is not empty then the return value is N, else the return value is the result of super\\_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe) owl:Thing), true). If direct is False: then the result of super\\_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe) owl:Thing), false) together with N if N is non-empty.

equivalent\_classes (ce: owlapy.class\_expression.OWLClassExpression, only\_named: bool = True)

→ Iterable[owlapy.class\_expression.OWLClassExpression]

Gets the class expressions that are equivalent to the specified class expression with respect to the set of reasoner axioms.

# **Parameters**

- ce The class expression whose equivalent classes are to be retrieved.
- only\_named Whether to only retrieve named equivalent classes or also complex class expressions.

### Returns

All class expressions C where the root ontology imports closure entails EquivalentClasses(ce C). If ce is not a class name (i.e. it is an anonymous class expression) and there are no such classes C then there will be no result. If ce is unsatisfiable with respect to the set of reasoner axioms then owl:Nothing, i.e. the bottom node, will be returned.

disjoint\_classes (ce: owlapy.class\_expression.OWLClassExpression, only\_named: bool = True)

→ Iterable[owlapy.class expression.OWLClassExpression]

Gets the class expressions that are disjoint with specified class expression with respect to the set of reasoner axioms.

#### **Parameters**

- **ce** The class expression whose disjoint classes are to be retrieved.
- only\_named Whether to only retrieve named disjoint classes or also complex class expressions.

### **Returns**

All class expressions D where the set of reasoner axioms entails EquivalentClasses(D Object-ComplementOf(ce)) or StrictSubClassOf(D ObjectComplementOf(ce)).

 ${\tt different\_individuals} \ (\textit{ce: owlapy.owl\_individual.OWLNamedIndividual})$ 

→ Iterable[owlapy.owl\_individual.OWLNamedIndividual]

Gets the individuals that are different from the specified individual with respect to the set of reasoner axioms.

#### **Parameters**

ind – The individual whose different individuals are to be retrieved.

#### Returns

All individuals x where the set of reasoner axioms entails DifferentIndividuals(ind x).

 ${\tt same\_individuals} \ (\textit{ce: owlapy.owl\_individual.OWLNamedIndividual})$ 

→ Iterable[owlapy.owl\_individual.OWLNamedIndividual]

Gets the individuals that are the same as the specified individual with respect to the set of reasoner axioms.

# **Parameters**

ind – The individual whose same individuals are to be retrieved.

#### Returns

All individuals x where the root ontology imports closure entails SameIndividual(ind x).

data\_property\_values (ind: owlapy.owl\_individual.OWLNamedIndividual,

pe: owlapy.owl\_property.OWLDataProperty, direct: bool = True)

→ Iterable[owlapy.owl\_literal.OWLLiteral]

Gets the data property values for the specified individual and data property expression.

### **Parameters**

- ind The individual that is the subject of the data property values.
- **pe** The data property expression whose values are to be retrieved for the specified individual.
- direct Specifies if the direct values should be retrieved (True), or if all values should be retrieved (False), so that sub properties are taken into account.

#### Returns

A set of OWLLiterals containing literals such that for each literal l in the set, the set of reasoner axioms entails DataPropertyAssertion(pe ind l).

```
all_data_property_values (pe: owlapy.owl_property.OWLDataProperty, direct: bool = True)

→ Iterable[owlapy.owl literal.OWLLiteral]
```

Gets all values for the given data property expression that appear in the knowledge base.

#### **Parameters**

- pe The data property expression whose values are to be retrieved
- **direct** Specifies if only the direct values of the data property pe should be retrieved (True), or if the values of sub properties of pe should be taken into account (False).

### Returns

A set of OWLLiterals containing literals such that for each literal l in the set, the set of reasoner axioms entails DataPropertyAssertion(pe ind l) for any ind.

# 

→ Iterable[owlapy.owl\_individual.OWLNamedIndividual]

Gets the object property values for the specified individual and object property expression.

### **Parameters**

- ind The individual that is the subject of the object property values.
- pe The object property expression whose values are to be retrieved for the specified individual.
- **direct** Specifies if the direct values should be retrieved (True), or if all values should be retrieved (False), so that sub properties are taken into account.

#### Returns

The named individuals such that for each individual j, the set of reasoner axioms entails ObjectPropertyAssertion(pe ind j).

### flush() $\rightarrow$ None

Flushes any changes stored in the buffer, which causes the reasoner to take into consideration the changes the current root ontology specified by the changes.

```
instances (ce: owlapy.class_expression.OWLClassExpression, direct: bool = False)
```

→ Iterable[owlapy.owl\_individual.OWLNamedIndividual]

Gets the individuals which are instances of the specified class expression.

# **Parameters**

- **ce** The class expression whose instances are to be retrieved.
- **direct** Specifies if the direct instances should be retrieved (True), or if all instances should be retrieved (False).

### **Returns**

If direct is True, each named individual j where the set of reasoner axioms entails DirectClassAssertion(ce, j). If direct is False, each named individual j where the set of reasoner axioms entails ClassAssertion(ce, j). If ce is unsatisfiable with respect to the set of reasoner axioms then nothing returned.

```
sub_classes (ce: owlapy.class_expression.OWLClassExpression, direct: bool = False, only named: bool = True) \rightarrow Iterable[owlapy.class expression.OWLClassExpression]
```

Gets the set of named classes that are the strict (potentially direct) subclasses of the specified class expression with respect to the reasoner axioms.

# **Parameters**

- **ce** The class expression whose strict (direct) subclasses are to be retrieved.
- direct Specifies if the direct subclasses should be retrieved (True) or if the all subclasses (descendant) classes should be retrieved (False).
- only\_named Whether to only retrieve named sub-classes or also complex class expressions.

#### Returns

If direct is True, each class C where reasoner axioms entails DirectSubClassOf(C, ce). If direct is False, each class C where reasoner axioms entails StrictSubClassOf(C, ce). If ce is equivalent to owl:Nothing then nothing will be returned.

```
super\_classes (ce: owlapy.class_expression.OWLClassExpression, direct: bool = False, only_named: bool = True) \rightarrow Iterable[owlapy.class_expression.OWLClassExpression]
```

Gets the stream of named classes that are the strict (potentially direct) super classes of the specified class expression with respect to the imports closure of the root ontology.

#### **Parameters**

- **ce** The class expression whose strict (direct) super classes are to be retrieved.
- **direct** Specifies if the direct super classes should be retrieved (True) or if the all super classes (ancestors) classes should be retrieved (False).
- only\_named Whether to only retrieve named super classes or also complex class expressions.

# Returns

If direct is True, each class C where the set of reasoner axioms entails DirectSubClassOf(ce,

- C). If direct is False, each class C where set of reasoner axioms entails StrictSubClassOf(ce,
- C). If ce is equivalent to owl: Thing then nothing will be returned.

 ${\tt types} \ (ind: owlapy.owl\_individual.OWLNamedIndividual, \ direct: bool = False)$ 

→ Iterable[owlapy.class\_expression.OWLClass]

Gets the named classes which are (potentially direct) types of the specified named individual.

## **Parameters**

- ind The individual whose types are to be retrieved.
- **direct** Specifies if the direct types should be retrieved (True), or if all types should be retrieved (False).

### Returns

If direct is True, each named class C where the set of reasoner axioms entails DirectClassAssertion(C, ind). If direct is False, each named class C where the set of reasoner axioms entails ClassAssertion(C, ind).

```
equivalent_object_properties (dp: owlapy.owl_property.OWLObjectPropertyExpression)

→ Iterable[owlapy.owl_property.OWLObjectPropertyExpression]
```

Gets the simplified object properties that are equivalent to the specified object property with respect to the set of reasoner axioms.

### **Parameters**

**op** – The object property whose equivalent object properties are to be retrieved.

#### Returns

All simplified object properties e where the root ontology imports closure entails EquivalentObjectProperties(op e). If op is unsatisfiable with respect to the set of reasoner axioms then owl:bottomDataProperty will be returned.

```
equivalent_data_properties (dp: owlapy.owl_property.OWLDataProperty)

→ Iterable[owlapy.owl_property.OWLDataProperty]
```

Gets the data properties that are equivalent to the specified data property with respect to the set of reasoner axioms.

#### **Parameters**

dp – The data property whose equivalent data properties are to be retrieved.

#### Returns

All data properties e where the root ontology imports closure entails EquivalentDataProperties(dp e). If dp is unsatisfiable with respect to the set of reasoner axioms then owl:bottomDataProperty will be returned.

```
disjoint_object_properties (dp: owlapy.owl_property.OWLObjectPropertyExpression)

→ Iterable[owlapy.owl_property.OWLObjectPropertyExpression]
```

Gets the simplified object properties that are disjoint with the specified object property with respect to the set of reasoner axioms.

#### **Parameters**

op – The object property whose disjoint object properties are to be retrieved.

# Returns

All simplified object properties e where the root ontology imports closure entails EquivalentObjectProperties(e ObjectPropertyComplementOf(op)) or StrictSubObjectPropertyOf(e ObjectPropertyComplementOf(op)).

Gets the data properties that are disjoint with the specified data property with respect to the set of reasoner axioms.

# **Parameters**

**dp** – The data property whose disjoint data properties are to be retrieved.

# Returns

All data properties e where the root ontology imports closure entails EquivalentDataProperties(e DataPropertyComplementOf(dp)) or StrictSubDataPropertyOf(e DataPropertyComplementOf(dp)).

Gets the set of named data properties that are the strict (potentially direct) subproperties of the specified data property expression with respect to the imports closure of the root ontology.

### **Parameters**

- **dp** The data property whose strict (direct) subproperties are to be retrieved.
- **direct** Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

### Returns

If direct is True, each property P where the set of reasoner axioms entails DirectSubDataPropertyOf(P, pe). If direct is False, each property P where the set of reasoner axioms entails StrictSubDataPropertyOf(P, pe). If pe is equivalent to owl:bottomDataProperty then nothing will be returned.

Gets the stream of data properties that are the strict (potentially direct) super properties of the specified data property with respect to the imports closure of the root ontology.

# **Parameters**

- **dp** (OWLDataProperty) The data property whose super properties are to be retrieved.
- **direct** (bool) Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

### **Returns**

Iterable of super properties.

Gets the stream of object properties that are the strict (potentially direct) super properties of the specified object property with respect to the imports closure of the root ontology.

#### **Parameters**

- **op** (OWLObjectPropertyExpression) The object property expression whose super properties are to be retrieved.
- **direct** (bool) Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

#### Returns

Iterable of super properties.

Gets the stream of simplified object property expressions that are the strict (potentially direct) subproperties of the specified object property expression with respect to the imports closure of the root ontology.

# **Parameters**

- op The object property expression whose strict (direct) subproperties are to be retrieved.
- **direct** Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

### **Returns**

If direct is True, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails DirectSubObjectPropertyOf(P, pe). If direct is False, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails StrictSubObjectPropertyOf(P, pe). If pe is equivalent to owl:bottomObjectProperty then nothing will be returned.

```
\verb"get_root_ontology"() \to owlapy.owl_ontology". OWLOntology
```

Gets the "root" ontology that is loaded into this reasoner. The reasoner takes into account the axioms in this ontology and its import's closure.

```
__slots__ = ('_cnt', '_conv', '_base_reasoner')
```

Add or remove axioms to the isolated ontology that the reasoner is using.

#### **Parameters**

- axioms\_to\_add (List [OWLAxiom]) Axioms to add to the isolated ontology.
- axioms\_to\_remove (List[OWLAxiom]) Axioms to remove from the isolated ontology.

Gets the individuals which are instances of the specified class expression.

### **Parameters**

- **ce** The class expression whose instances are to be retrieved.
- **direct** Specifies if the direct instances should be retrieved (True), or if all instances should be retrieved (False).

# **Returns**

If direct is True, each named individual j where the set of reasoner axioms entails DirectClassAssertion(ce, j). If direct is False, each named individual j where the set of reasoner axioms entails ClassAssertion(ce, j). If ce is unsatisfiable with respect to the set of reasoner axioms then nothing returned.

```
__del__()
```

# owlapy.owlapi adaptor

### **Classes**

OWLAPIAdaptor

A class to interface with the OWL API using the HermiT reasoner, enabling ontology management,

### **Module Contents**

class owlapy.owlapi\_adaptor.OWLAPIAdaptor(path: str, name\_reasoner: str = 'HermiT')

A class to interface with the OWL API using the HermiT reasoner, enabling ontology management, reasoning, and parsing class expressions in Manchester OWL Syntax.

### path

The file path to the ontology.

**Type** 

str

# name reasoner

The reasoner to be used, default is "HermiT".

**Type** 

str

#### manager

The OWL ontology manager.

### ontology

The loaded OWL ontology.

#### reasoner

Choose from (case-sensitive): ["HermiT", "Pellet", "JFact", "Openllet"]. Default: "HermiT".

#### parser

The Manchester OWL Syntax parser.

# renderer

The Manchester OWL Syntax renderer.

```
stopJVM (*args, **kwargs) \rightarrow None
```

Detaches the thread from Java packages and shuts down the java virtual machine hosted by jpype.

```
generate_inferred_class_assertion_axioms (output='temp.ttl', format: str = None)
```

Generates inferred class assertion axioms for the ontology managed by this instance's reasoner and saves them to a file.

This function uses the OWL API to generate inferred class assertion axioms based on the ontology and reasoner associated with this instance. The inferred axioms are saved to the specified output file in the desired format.

#### Parameters:

### output

[str, optional] The name of the file where the inferred axioms will be saved. Default is "temp.ttl".

#### format

[str, optional] The format in which to save the inferred axioms. Supported formats are: - "ttl" or "turtle" for Turtle format - "rdf/xml" for RDF/XML format - "owl/xml" for OWL/XML format If not specified, the format of the original ontology is used.

# Notes:

- The function supports saving in multiple formats: Turtle, RDF/XML, and OWL/XML.
- The inferred axioms are generated using the reasoner associated with this instance and the OWL API's InferredClassAssertionAxiomGenerator.
- The inferred axioms are added to a new ontology which is then saved in the specified format.

# **Example:**

```
>>> instance.generate_inferred_class_assertion_axioms(output="inferred_axioms.

+ttl", format="ttl")
```

This will save the inferred class assertion axioms to the file "inferred\_axioms.ttl" in Turtle format.

```
convert_to_owlapi (ce: owlapy.class_expression.OWLClassExpression)
```

Converts an OWLAPY class expression to an OWLAPI class expression.

# **Parameters**

**ce** (OWLClassExpression) – The class expression in OWLAPY format to be converted.

#### Returns

The class expression in OWLAPI format.

**convert from owlapi** (ce, namespace:  $str) \rightarrow owlapy.class$  expression.OWLClassExpression

Converts an OWLAPI class expression to an OWLAPY class expression.

### **Parameters**

- ce The class expression in OWLAPI format.
- namespace (str) The ontology's namespace where the class expression belongs.

### **Returns**

The class expression in OWLAPY format.

# Return type

**OWLClassExpression** 

instances (ce: owlapy.class\_expression.OWLClassExpression)

 $\rightarrow$  List[owlapy.owl\_individual.OWLNamedIndividual]

Get the instances for a given class expression using HermiT.

# **Parameters**

**ce** (OWLClassExpression) – The class expression in OWLAPY format.

### **Returns**

A list of individuals classified by the given class expression.

# Return type

list

# ${\tt has\_consistent\_ontology}\,(\,)\,\to bool$

Check if the used ontology is consistent.

# Returns

True if the ontology is consistent, False otherwise.

# Return type

bool

# owlapy.parser

String to OWL parsers.

# **Attributes**

MANCHESTER_GRAMMAR
DL_GRAMMAR
DLparser
ManchesterParser

### **Classes**

ManchesterOWLSyntaxParser	Manchester Syntax parser to parse strings to OWLClass- Expressions.
DLSyntaxParser	Description Logic Syntax parser to parse strings to OWL-ClassExpressions.

# **Functions**

```
dl_to_owl_expression(dl_expression, names-
pace)
manchester_to_owl_expression(manchester_ex
...)
```

# **Module Contents**

The OWL Object which is represented by the string.

```
visit\_union (node, children) \rightarrow owlapy.class_expression.OWLClassExpression
visit\_intersection (node, children) \rightarrow owlapy.class_expression.OWLClassExpression
visit\_primary(node, children) \rightarrow owlapy.class\_expression.OWLClassExpression
{\tt visit\_some\_only\_res}\ (node, children) \rightarrow owlapy.class\_expression.OWLQuantifiedObjectRestriction
visit_cardinality_res (node, children)
             → owlapy.class expression.OWLObjectCardinalityRestriction
visit value res(node, children) \rightarrow owlapy.class expression.OWLObjectHasValue
visit has self (node, children) \rightarrow owlapy.class expression.OWLObjectHasSelf
visit\_object\_property(node, children) \rightarrow owlapy.owl\_property.OWLObjectPropertyExpression
visit\_class\_expression (node, children) \rightarrow owlapy.class\_expression.OWLClassExpression
visit_individual_list (node, children) → owlapy.class_expression.OWLObjectOneOf
visit_data_primary (node, children) → owlapy.owl_data_ranges.OWLDataRange
visit_data_some_only_res (node, children)
             → owlapy.class_expression.OWLQuantifiedDataRestriction
visit_data_cardinality_res (node, children)
             → owlapy.class expression.OWLDataCardinalityRestriction
visit_data_value_res(node, children) \rightarrow owlapy.class_expression.OWLDataHasValue
visit_data_union (node, children) → owlapy.owl_data_ranges.OWLDataRange
visit\_data\_intersection (node, children) \rightarrow owlapy.owl\_data\_ranges.OWLDataRange
visit\_literal\_list (node, children) \rightarrow owlapy.class\_expression.OWLDataOneOf
visit_data_parentheses (node, children) → owlapy.owl_data_ranges.OWLDataRange
visit_datatype_restriction (node, children)
             → owlapy.class expression.OWLDatatypeRestriction
visit_facet_restrictions (node, children)
             → List[owlapy.class_expression.OWLFacetRestriction]
visit\_facet\_restriction (node, children) \rightarrow owlapy.class\_expression.OWLFacetRestriction
visit literal (node, children) → owlapy.owl literal.OWLLiteral
visit typed literal (node, children) → owlapy.owl literal.OWLLiteral
abstract visit string literal language (node, children)
visit_string_literal_no_language (node, children) → owlapy.owl_literal.OWLLiteral
visit\_quoted\_string(node, children) \rightarrow str
visit_float_literal (node, children) → owlapy.owl_literal.OWLLiteral
visit\_decimal\_literal(node, children) \rightarrow owlapy.owl\_literal.OWLLiteral
```

```
visit\_integer\_literal (node, children) \rightarrow owlapy.owl\_literal.OWLLiteral
visit\_boolean\_literal(node, children) \rightarrow owlapy.owl\_literal.OWLLiteral
visit\_datetime\_literal(node, children) \rightarrow owlapy.owl\_literal.OWLLiteral
visit\_duration\_literal (node, children) \rightarrow owlapy.owl_literal.OWLLiteral
visit\_date\_literal(node, children) \rightarrow owlapy.owl\_literal.OWLLiteral
visit_non_negative_integer (node, children) → int
visit_datatype_iri (node, children) → str
visit\_datatype (node, children) \rightarrow owlapy.owl\_datatype.OWLDatatype
visit\_facet(node, children) \rightarrow owlapy.vocab.OWLFacet
visit_class_iri (node, children) → owlapy.class_expression.OWLClass
visit\_individual\_iri (node, children) \rightarrow owlapy.owl_individual.OWLNamedIndividual
visit_object_property_iri (node, children) → owlapy.owl_property.OWLObjectProperty
visit_data_property_iri (node, children) → owlapy.owl_property.OWLDataProperty
visit_iri(node, children) \rightarrow owlapy.iri.IRI
visit\_full\_iri(node, children) \rightarrow owlapy.iri.IRI
abstract visit_abbreviated_iri (node, children)
visit_simple_iri(node, children) → owlapy.iri.IRI
visit\_parentheses (node, children) \rightarrow owlapy.class\_expression.OWLClassExpression
generic_visit (node, children)
     Default visitor method
```

# **Parameters**

- node The node we're visiting
- visited children The results of visiting the children of that node, in a list

I'm not sure there's an implementation of this that makes sense across all (or even most) use cases, so we leave it to subclasses to implement for now.

```
Parse a string to an OWL Object.
         Parameters
             expression str(str) – Expression string.
             The OWL Object which is represented by the string.
visit\_union (node, children) \rightarrow owlapy.class_expression.OWLClassExpression
visit\_intersection (node, children) \rightarrow owlapy.class_expression.OWLClassExpression
visit\_primary(node, children) \rightarrow owlapy.class\_expression.OWLClassExpression
visit\_some\_only\_res(node, children) \rightarrow owlapy.class\_expression.OWLQuantifiedObjectRestriction
visit_cardinality_res (node, children)
             → owlapy.class expression.OWLObjectCardinalityRestriction
visit\_value\_res(node, children) \rightarrow owlapy.class\_expression.OWLObjectHasValue
visit\_has\_self(node, children) \rightarrow owlapy.class\_expression.OWLObjectHasSelf
visit object property (node, children) → owlapy.owl property.OWLObjectPropertyExpression
visit\_class\_expression (node, children) \rightarrow owlapy.class\_expression.OWLClassExpression
visit\_individual\_list (node, children) \rightarrow owlapy.class\_expression.OWLObjectOneOf
visit_data_primary (node, children) → owlapy.owl_data_ranges.OWLDataRange
visit_data_some_only_res (node, children)
             → owlapy.class_expression.OWLQuantifiedDataRestriction
visit_data_cardinality_res (node, children)
             → owlapy.class expression.OWLDataCardinalityRestriction
visit data value res(node, children) \rightarrow owlapy.class expression.OWLDataHasValue
visit\_data\_union (node, children) \rightarrow owlapy.owl_data_ranges.OWLDataRange
visit_{data_intersection} (node, children) \rightarrow owlapy.owl_data_ranges.OWLDataRange
visit\_literal\_list(node, children) \rightarrow owlapy.class\_expression.OWLDataOneOf
visit_data_parentheses (node, children) → owlapy.owl_data_ranges.OWLDataRange
visit_datatype_restriction (node, children)
             → owlapy.class expression.OWLDatatypeRestriction
visit_facet_restrictions (node, children)
             → List[owlapy.class_expression.OWLFacetRestriction]
visit_facet_restriction (node, children) \rightarrow owlapy.class_expression.OWLFacetRestriction
visit_literal (node, children) → owlapy.owl_literal.OWLLiteral
visit typed literal (node, children) → owlapy.owl literal.OWLLiteral
abstract visit_string_literal_language (node, children)
```

 $parse\_expression\ (expression\_str: str) \rightarrow owlapy.class\_expression.OWLClassExpression$ 

```
visit_string_literal_no_language (node, children) → owlapy.owl_literal.OWLLiteral
visit\_quoted\_string(node, children) \rightarrow str
visit_float_literal (node, children) \rightarrow owlapy.owl_literal.OWLLiteral
visit\_decimal\_literal(node, children) \rightarrow owlapy.owl\_literal.OWLLiteral
visit_integer_literal (node, children) → owlapy.owl_literal.OWLLiteral
visit\_boolean\_literal(node, children) \rightarrow owlapy.owl\_literal.OWLLiteral
visit\_datetime\_literal(node, children) \rightarrow owlapy.owl\_literal.OWLLiteral
visit_duration_literal (node, children) → owlapy.owl_literal.OWLLiteral
visit\_date\_literal(node, children) \rightarrow owlapy.owl\_literal.OWLLiteral
visit\_non\_negative\_integer(node, children) \rightarrow int
visit datatype iri(node, children) \rightarrow str
visit_datatype (node, children) → owlapy.owl_datatype. OWLDatatype
visit\_facet(node, children) \rightarrow owlapy.vocab.OWLFacet
visit\_class\_iri(node, children) \rightarrow owlapy.class\_expression.OWLClass
visit\_individual\_iri (node, children) \rightarrow owlapy.owl_individual.OWLNamedIndividual
visit_object_property_iri (node, children) → owlapy.owl_property.OWLObjectProperty
visit data property iri (node, children) → owlapy.owl property.OWLDataProperty
visit_iri(node, children) \rightarrow owlapy.iri.IRI
visit_full_iri(node, children) \rightarrow owlapy.iri.IRI
abstract visit_abbreviated_iri (node, children)
visit simple iri(node, children) → owlapy.iri.IRI
visit\_parentheses(node, children) \rightarrow owlapy.class\_expression.OWLClassExpression
generic_visit (node, children)
     Default visitor method
```

# **Parameters**

- node The node we're visiting
- visited\_children The results of visiting the children of that node, in a list

I'm not sure there's an implementation of this that makes sense across all (or even most) use cases, so we leave it to subclasses to implement for now.

```
owlapy.parser.DLparser
owlapy.parser.ManchesterParser
owlapy.parser.dl_to_owl_expression(dl_expression: str, namespace: str)
owlapy.parser.manchester_to_owl_expression(manchester_expression: str, namespace: str)
```

# owlapy.providers

OWL Datatype restriction constructors.

## **Attributes**

```
Restriction_Literals
```

# **Functions**

```
      owl_datatype_max_exclusive_restriction
      Create a max exclusive restriction.

      owl_datatype_min_exclusive_restriction
      Create a min exclusive restriction.

      owl_datatype_max_inclusive_restriction
      Create a max inclusive restriction.

      owl_datatype_min_inclusive_restriction
      Create a min inclusive restriction.

      owl_datatype_min_max_exclusive_restric
      Create a min-max exclusive restriction.

      owl_datatype_min_max_inclusive_restric
      Create a min-max inclusive restriction.
```

### **Module Contents**

```
owlapy.providers.Restriction_Literals
owlapy.providers.owl datatype max exclusive restriction (max: Restriction Literals)
            → owlapy.class expression.OWLDatatypeRestriction
     Create a max exclusive restriction.
owlapy.providers.owl_datatype_min_exclusive_restriction (min_: Restriction_Literals)
            → owlapy.class expression.OWLDatatypeRestriction
     Create a min exclusive restriction.
owlapy.providers.owl_datatype_max_inclusive_restriction (max_: Restriction_Literals)
            → owlapy.class expression.OWLDatatypeRestriction
     Create a max inclusive restriction.
owlapy.providers.owl_datatype_min_inclusive_restriction (min_: Restriction_Literals)
            → owlapy.class expression.OWLDatatypeRestriction
     Create a min inclusive restriction.
owlapy.providers.owl_datatype_min_max_exclusive_restriction(
           min_: Restriction_Literals, max_: Restriction_Literals)
            → owlapy.class_expression.OWLDatatypeRestriction
     Create a min-max exclusive restriction.
owlapy.providers.owl_datatype_min_max_inclusive_restriction(
           min_: Restriction_Literals, max_: Restriction_Literals)
            → owlapy.class_expression.OWLDatatypeRestriction
     Create a min-max inclusive restriction.
```

# owlapy.render

Renderers for different syntax.

# **Attributes**

```
DLrenderer
```

*ManchesterRenderer* 

### **Classes**

DLSyntaxObjectRenderer	DL Syntax renderer for OWL Objects.
ManchesterOWLSyntaxOWLObjectRenderer	Manchester Syntax renderer for OWL Objects

# **Functions**

```
owl\_expression\_to\_dl(\rightarrow str) owl\_expression\_to\_manchester(\rightarrow str)
```

# **Module Contents**

### Returns

String rendition of OWL object.

```
class owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer(
           short_form_provider: Callable[[owlapy.owl_object.OWLEntity], str] = _simple_short_form_provider,
           no_render_thing=False)
     Bases: owlapy.owl_object.OWLObjectRenderer
     Manchester Syntax renderer for OWL Objects
     __slots__ = ('_sfp', '_no_render_thing')
     set_short_form_provider (short_form_provider: Callable[[owlapy.owl_object.OWLEntity], str])
                 \rightarrow None
          Configure a short form provider that shortens the OWL objects during rendering.
             Parameters
                 short_form_provider - Short form provider.
     render (o: owlapy.owl_object.OWLObject) → str
          Render OWL Object to string.
             Parameters
                 o – OWL Object.
             Returns
                 String rendition of OWL object.
owlapy.render.DLrenderer
owlapy.render.ManchesterRenderer
owlapy.render.owl_expression_to_dl(o: owlapy.owl_object.OWLObject) → str
```

owlapy.render.owl\_expression\_to\_manchester(o: owlapy.owl\_object.OWLObject) 

str

# owlapy.static\_funcs

Static functions for general purposes.

# **Functions**

move(*args)	"Move" an imported class to the current module by setting the classesmodule attribute.
download_external_files(ftp_link)	

# **Module Contents**

```
owlapy.static_funcs.move(*args)
```

"Move" an imported class to the current module by setting the classes \_\_module\_\_ attribute.

This is useful for documentation purposes to hide internal packages in sphinx.

# **Parameters**

args - List of classes to move.

owlapy.static\_funcs.download\_external\_files (ftp\_link: str)

# owlapy.utils

Owlapy utils.

# **Attributes**

measurer

# **Classes**

OWLClassExpressionLengthMetric	Length calculation of OWLClassExpression
EvaluatedDescriptionSet	Abstract base class for generic types.
ConceptOperandSorter	
OperandSetTransform	
HasIndex	Interface for types with an index; this is used to group objects by type when sorting.
OrderedOWLObject	Holder of OWL Objects that can be used for Python sorted.
NNF	This class contains functions to transform a Class Expression into Negation Normal Form.
TopLevelCNF	This class contains functions to transform a class expression into Top-Level Conjunctive Normal Form.
TopLevelDNF	This class contains functions to transform a class expression into Top-Level Disjunctive Normal Form.
LRUCache	Constants shares by all lru cache instances.

# **Functions**

get_expression_length(→ int)	
combine_nary_expressions()	Shortens an OWLClassExpression or OWLDataRange by combining all nested nary expressions of the same type.
iter_count(→int)	Count the number of elements in an iterable.
$as\_index(\rightarrow HasIndex)$	Cast OWL Object to HasIndex.

### **Module Contents**

class owlapy.utils.OWLClassExpressionLengthMetric (\*, class\_length: int, object\_intersection\_length: int, object\_union\_length: int, object\_complement\_length: int, object\_some\_values\_length: int, object\_all\_values\_length: int, object\_has\_value\_length: int, object\_cardinality\_length: int, object\_has\_self\_length: int, object\_one\_of\_length: int, data\_some\_values\_length: int, data\_all\_values\_length: int, data\_has\_value\_length: int, data\_cardinality\_length: int, object\_property\_length: int, object\_inverse\_length: int, data\_property\_length: int, data\_one\_of\_length: int, data\_union\_length: int)

Length calculation of OWLClassExpression

#### **Parameters**

- class\_length Class: "C"
- object\_intersection\_length Intersection:  $A \sqcap B$
- object\_union\_length Union: A \subseteq B
- object\_complement\_length Complement: ¬ C
- object\_some\_values\_length Obj. Some Values: ∃ r.C
- object\_all\_values\_length Obj. All Values: \( \forall \) r.C
- object\_has\_value\_length Obj. Has Value:  $\exists r.\{I\}$
- object\_cardinality\_length Obj. Cardinality restriction: ≤n r.C
- object\_has\_self\_length Obj. Self restriction: 3 r.Self
- object\_one\_of\_length Obj. One of:  $\exists r.\{X,Y,Z\}$
- data\_some\_values\_length Data Some Values: ∃ p.t
- data\_all\_values\_length Data All Values: ∀ p.t
- data\_has\_value\_length Data Has Value: ∃ p.{V}
- data\_cardinality\_length Data Cardinality restriction: ≤n r.t
- object\_property\_length Obj. Property: ∃ r.C
- object\_inverse\_length Inverse property: ∃ r¯.C
- $data\_property\_length$  Data Property:  $\exists p.t$
- datatype length Datatype: ^^datatype
- data\_one\_of\_length Data One of: ∃ p.{U,V,W}

```
• data_complement_length - Data Complement: ¬datatype
             • data_intersection_length - Data Intersection: datatype □ datatype
             • data_union_length – Data Union: datatype ☐ datatype
    __slots__ = ('class_length', 'object_intersection_length',
    'object_union_length', . . .
    class length: int
    object_intersection_length: int
    object_union_length: int
    object_complement_length: int
    object some values length: int
    object_all_values_length: int
    object_has_value_length: int
    object_cardinality_length: int
    object_has_self_length: int
    object_one_of_length: int
    data_some_values_length: int
    data_all_values_length: int
    data_has_value_length: int
    data_cardinality_length: int
    object_property_length: int
    object_inverse_length: int
    data_property_length: int
    datatype_length: int
    data_one_of_length: int
    data_complement_length: int
    data_intersection_length: int
    data_union_length: int
    static get_default() → OWLClassExpressionLengthMetric
    abstract length (o: owlapy.owl_object.OWLObject) → int
owlapy.utils.measurer
owlapy.utils.get_expression_length(ce: owlapy.class_expression.OWLClassExpression) \rightarrow int
```

Abstract base class for generic types.

Return self==value.

A generic type is typically declared by inheriting from this class parameterized with one or more type variables. For example, a generic mapping type might be defined as:

```
class Mapping(Generic[KT, VT]):
    def __getitem__(self, key: KT) -> VT:
        ...
# Etc.
```

```
This class can then be used as follows:
     def lookup_name (mapping: Mapping[KT, VT], key: KT, default: VT) -> VT:
              return mapping[key]
          except KeyError:
              return default
     __slots__ = ('items', '_max_size', '_Ordering')
     items: SortedSet[_N]
     maybe_add (node: _N)
     clean()
     worst()
     best()
     \textbf{best\_quality\_value} \, (\,) \, \to float
     \_iter\_() \rightarrow Iterable[\_N]
class owlapy.utils.ConceptOperandSorter
     \texttt{abstract} \ \texttt{sort} \ (o\text{: } \_O) \ \to \_O
class owlapy.utils.OperandSetTransform
     simplify (o: owlapy.class_expression.OWLClassExpression)
                  \rightarrow owlapy.class_expression.OWLClassExpression
class owlapy.utils.HasIndex
     Bases: Protocol
     Interface for types with an index; this is used to group objects by type when sorting.
     type_index: ClassVar[int]
     __eq_ (other)
```

```
class owlapy.utils.OrderedOWLObject (o: _HasIndex)
```

Holder of OWL Objects that can be used for Python sorted.

The Ordering is dependent on the type\_index of the impl. classes recursively followed by all components of the OWL Object.

```
OWL object.
__slots__ = ('o', '_chain')
o: _HasIndex
__lt__(other)
    Return self<value.
__eq__(other)
    Return self==value.</pre>
```

class owlapy.utils.NNF

This class contains functions to transform a Class Expression into Negation Normal Form.

```
abstract get_class_nnf (ce: owlapy.class_expression.OWLClassExpression, negated: bool = False) \rightarrow owlapy.class expression.OWLClassExpression
```

Convert a Class Expression to Negation Normal Form. Operands will be sorted.

#### **Parameters**

- ce Class Expression.
- **negated** Whether the result should be negated.

# Returns

Class Expression in Negation Normal Form.

```
class owlapy.utils.TopLevelCNF
```

This class contains functions to transform a class expression into Top-Level Conjunctive Normal Form.

```
get_top_level_cnf (ce: owlapy.class_expression.OWLClassExpression)

→ owlapy.class_expression.OWLClassExpression
```

Convert a class expression into Top-Level Conjunctive Normal Form. Operands will be sorted.

### **Parameters**

ce - Class Expression.

# Returns

Class Expression in Top-Level Conjunctive Normal Form.

```
class owlapy.utils.TopLevelDNF
```

This class contains functions to transform a class expression into Top-Level Disjunctive Normal Form.

Convert a class expression into Top-Level Disjunctive Normal Form. Operands will be sorted.

#### **Parameters**

**ce** – Class Expression.

# Returns

Class Expression in Top-Level Disjunctive Normal Form.

```
owlapy.utils.combine_nary_expressions (ce: owlapy.class_expression.OWLClassExpression)
            → owlapy.class_expression.OWLClassExpression
owlapy.utils.combine_nary_expressions(ce: owlapy.owl_data_ranges.OWLDataRange)
            → owlapy.owl_data_ranges.OWLDataRange
     Shortens an OWLClassExpression or OWLDataRange by combining all nested nary expressions of the same type.
     Operands will be sorted.
     E.g.\ OWLObjectUnionOf(A,\ OWLObjectUnionOf(C,\ B)) \ -> \ OWLObjectUnionOf(A,\ B,\ C).
owlapy.utils.iter_count (i: Iterable) → int
     Count the number of elements in an iterable.
owlapy.utils.as_index(o: owlapy.owl_object.OWLObject) → HasIndex
     Cast OWL Object to HasIndex.
class owlapy.utils.LRUCache (maxsize: int | None = None)
     Bases: Generic[_K, _V]
     Constants shares by all lru cache instances.
     Adapted from functools.lru_cache.
     sentinel
          Unique object used to signal cache misses.
     PREV
          Name for the link field 0.
     NEXT
          Name for the link field 1.
     KEY
          Name for the link field 2.
     RESULT
          Name for the link field 3.
     sentinel
     __contains__(item: \_K) \rightarrow bool
     \_getitem\_(item: \_K) \rightarrow \_V
     __setitem__(key: _K, value: _V)
     cache_info()
          Report cache statistics.
     cache_clear()
```

Clear the cache and cache statistics.

# owlapy.vocab

Enumerations.

# **Classes**

OWLRDFVocabulary	Enumerations for OWL/RDF vocabulary.
XSDVocabulary	Enumerations for XSD vocabulary.
OWLFacet	Enumerations for OWL facets.

# **Module Contents**

```
class owlapy.vocab.OWLRDFVocabulary (namespace: owlapy.namespaces.Namespaces,
         remainder: str)
    Bases: _Vocabulary, enum. Enum
    Enumerations for OWL/RDF vocabulary.
    OWL_THING
    OWL_NOTHING
    OWL_CLASS
    OWL_NAMED_INDIVIDUAL
    OWL_TOP_OBJECT_PROPERTY
    OWL_BOTTOM_OBJECT_PROPERTY
    OWL_TOP_DATA_PROPERTY
    OWL_BOTTOM_DATA_PROPERTY
    RDFS_LITERAL
class owlapy.vocab.XSDVocabulary(remainder: str)
    Bases: _Vocabulary, enum.Enum
    Enumerations for XSD vocabulary.
    DECIMAL: Final = 'decimal'
    INTEGER: Final = 'integer'
    LONG: Final = 'long'
    DOUBLE: Final = 'double'
    FLOAT: Final = 'float'
    BOOLEAN: Final = 'boolean'
    STRING: Final = 'string'
```

```
DATE: Final = 'date'

DATE_TIME: Final = 'dateTime'

DATE_TIME_STAMP: Final = 'dateTimeStamp'

DURATION: Final = 'duration'

class owlapy.vocab.OWLFacet (remainder: str, symbolic_form: str, operator: Callable[[_X, _X], bool])

Bases: _Vocabulary, enum.Enum

Enumerations for OWL facets.

property symbolic_form
```

property operator

 $static from\_str(name: str) \rightarrow OWLFacet$ 

MIN\_INCLUSIVE: Final
MIN\_EXCLUSIVE: Final
MAX\_INCLUSIVE: Final
MAX\_EXCLUSIVE: Final

LENGTH: Final

MIN\_LENGTH: Final MAX\_LENGTH: Final

PATTERN: Final

TOTAL\_DIGITS: Final

FRACTION\_DIGITS: Final

# 7.3 Attributes

\_\_version\_\_

#### 7.4 Functions

```
owl\_expression\_to\_dl(\rightarrow str)
owl\_expression\_to\_manchester(\rightarrow str)
dl\_to\_owl\_expression(dl\_expression, names-pace)
manchester\_to\_owl\_expression(manchester\_ex ...)
owl\_expression\_to\_sparql(\rightarrow str)
Convert an OWL Class Expression (https://www.w3.org/TR/owl2-syntax/#Class\_Expressions) into a SPARQL query
```

#### 7.5 Package Contents

Convert an OWL Class Expression (https://www.w3.org/TR/owl2-syntax/#Class\_Expressions) into a SPARQL query root variable: the variable that will be projected expression: the class expression to be transformed to a SPARQL query

values: positive or negative examples from a class expression problem. Unclear for\_all\_de\_morgan: if set to True, the SPARQL mapping will use the mapping containing the nested FILTER NOT EXISTS patterns for the universal quantifier (¬(¬¬C)), instead of the counting query named\_individuals: if set to True, the generated SPARQL query will return only entities that are instances of owl:NamedIndividual

```
owlapy.__version__ = '1.1.1'
```

# **Python Module Index**

#### 0

```
owlapy, 18
owlapy.class_expression, 18
owlapy.class_expression.class_expression,
owlapy.class_expression.nary_boolean_expression,
       20
owlapy.class_expression.owl_class,21
owlapy.class expression.restriction, 23
owlapy.converter, 52
owlapy.entities, 52
owlapy.iri,55
owlapy.meta_classes, 56
owlapy.namespaces, 58
owlapy.owl annotation, 59
owlapy.owl_axiom, 60
owlapy.owl_data_ranges, 80
owlapy.owl_datatype,82
owlapy.owl_hierarchy,82
owlapy.owl_individual,86
owlapy.owl_literal,87
owlapy.owl_object, 90
owlapy.owl_ontology,92
owlapy.owl_ontology_manager,98
owlapy.owl_property, 101
owlapy.owl_reasoner, 105
owlapy.owlapi_adaptor, 126
owlapy.parser, 128
owlapy.providers, 134
owlapy.render, 135
owlapy.static_funcs, 136
owlapy.utils, 137
owlapy.vocab, 143
```

#### Index

## Non-alphabetical

```
__contains__() (owlapy.converter.VariablesMapping method), 53
__contains__() (owlapy.owl_hierarchy.AbstractHierarchy method), 84
 _contains__() (owlapy.utils.LRUCache method), 142
__del__() (owlapy.owl_reasoner.SyncReasoner method), 126
__eq__() (owlapy.class_expression.class_expression.OWLObjectComplementOf method), 20
__eq__() (owlapy.class_expression.nary_boolean_expression.OWLNaryBooleanClassExpression method), 21
  _eq__ () (owlapy.class_expression.OWLDataAllValuesFrom method), 49
__eq__() (owlapy.class_expression.OWLDataCardinalityRestriction method), 44
__eq__() (owlapy.class_expression.OWLDataHasValue method), 50
  _eq__() (owlapy.class_expression.OWLDataOneOf method), 44
__eq__() (owlapy.class_expression.OWLDataSomeValuesFrom method), 48
  _eq__() (owlapy.class_expression.OWLDatatypeRestriction method), 46
__eq__() (owlapy.class_expression.OWLFacetRestriction method), 47
__eq__() (owlapy.class_expression.OWLHasValueRestriction method), 42
__eq__() (owlapy.class_expression.OWLNaryBooleanClassExpression method), 40
__eq__() (owlapy.class_expression.OWLObjectAllValuesFrom method), 45
  _eq__() (owlapy.class_expression.OWLObjectCardinalityRestriction method), 43
__eq__() (owlapy.class_expression.OWLObjectComplementOf method), 38
__eq__() (owlapy.class_expression.OWLObjectHasSelf method), 43
__eq__() (owlapy.class_expression.OWLObjectOneOf method), 51
__eq__() (owlapy.class_expression.OWLObjectSomeValuesFrom method), 45
 _eq__() (owlapy.class_expression.restriction.OWLDataAllValuesFrom method), 33
__eq__() (owlapy.class_expression.restriction.OWLDataCardinalityRestriction method), 31
__eq__() (owlapy.class_expression.restriction.OWLDataHasValue method), 33
__eq__() (owlapy.class_expression.restriction.OWLDataOneOf method), 34
__eq__() (owlapy.class_expression.restriction.OWLDataSomeValuesFrom method), 32
 eq () (owlapy.class expression.restriction.OWLDatatypeRestriction method), 35
__eq__() (owlapy.class_expression.restriction.OWLFacetRestriction method), 35
 _eq__() (owlapy.class_expression.restriction.OWLHasValueRestriction method), 25
__eq__() (owlapy.class_expression.restriction.OWLObjectAllValuesFrom method), 29
       () (owlapy.class_expression.restriction.OWLObjectCardinalityRestriction method), 27
__eq__() (owlapy.class_expression.restriction.OWLObjectHasSelf method), 29
__eq__() (owlapy.class_expression.restriction.OWLObjectOneOf method), 30
__eq__() (owlapy.class_expression.restriction.OWLObjectSomeValuesFrom method), 28
__eq__() (owlapy.iri.IRI method), 55
  _eq__ () (owlapy.namespaces.Namespaces method), 59
__eq__() (owlapy.owl_axiom.OWLAnnotation method), 71
__eq__() (owlapy.owl_axiom.OWLAnnotationAssertionAxiom method), 72
__eq__() (owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom method), 73
__eq__() (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom method), 73
  _eq__() (owlapy.owl_axiom.OWLClassAssertionAxiom method), 70
__eq__() (owlapy.owl_axiom.OWLDataPropertyCharacteristicAxiom method), 78
__eq__() (owlapy.owl_axiom.OWLDatatypeDefinitionAxiom method), 64
__eq__() (owlapy.owl_axiom.OWLDeclarationAxiom method), 63
__eq__() (owlapy.owl_axiom.OWLDisjointUnionAxiom method), 70
  _eq__() (owlapy.owl_axiom.OWLHasKeyAxiom method), 65
__eq__() (owlapy.owl_axiom.OWLNaryClassAxiom method), 65
__eq__() (owlapy.owl_axiom.OWLNaryIndividualAxiom method), 66
__eq__() (owlapy.owl_axiom.OWLNaryPropertyAxiom method), 67
__eq__() (owlapy.owl_axiom.OWLObjectPropertyCharacteristicAxiom method), 76
__eq__() (owlapy.owl_axiom.OWLPropertyAssertionAxiom method), 74
__eq__() (owlapy.owl_axiom.OWLPropertyDomainAxiom method), 78
__eq__() (owlapy.owl_axiom.OWLPropertyRangeAxiom method), 78
__eq__() (owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom method), 72
__eq__() (owlapy.owl_axiom.OWLSubClassOfAxiom method), 69
__eq__() (owlapy.owl_axiom.OWLSubPropertyAxiom method), 73
__eq__() (owlapy.owl_data_ranges.OWLDataComplementOf method), 81
__eq__() (owlapy.owl_data_ranges.OWLNaryDataRange method), 81
__eq__() (owlapy.owl_object.OWLNamedObject method), 91
  _eq___() (owlapy.owl_object.OWLObject method), 90
__eq__() (owlapy.owl_ontology.Ontology method), 97
__eq__() (owlapy.owl_ontology.OWLOntologyID method), 93
 __eq__() (owlapy.owl_property.OWLObjectInverseOf method), 104
```

```
_eq__() (owlapy.utils.HasIndex method), 140
__eq__() (owlapy.utils.OrderedOWLObject method), 141
  _getitem__() (owlapy.converter.VariablesMapping method), 53
__getitem__() (owlapy.utils.LRUCache method), 142
  _hash__() (owlapy.class_expression.class_expression.OWLObjectComplementOf method), 20
__hash__() (owlapy.class_expression.nary_boolean_expression.OWLNaryBooleanClassExpression method), 21
  _hash___() (owlapy.class_expression.OWLDataAllValuesFrom method), 49
  hash () (owlapy.class expression.OWLDataCardinalityRestriction method), 44
__hash__() (owlapy.class_expression.OWLDataHasValue method), 50
  _hash___() (owlapy.class_expression.OWLDataOneOf method), 44
  _hash___() (owlapy.class_expression.OWLDataSomeValuesFrom method), 49
  _hash___() (owlapy.class_expression.OWLDatatypeRestriction method), 46
__hash__() (owlapy.class_expression.OWLFacetRestriction method), 47
__hash__() (owlapy.class_expression.OWLHasValueRestriction method), 42
  _hash___() (owlapy.class_expression.OWLNaryBooleanClassExpression method), 40
  _hash___() (owlapy.class_expression.OWLObjectAllValuesFrom method), 45
  _hash___() (owlapy.class_expression.OWLObjectCardinalityRestriction method), 43
__hash__() (owlapy.class_expression.OWLObjectComplementOf method), 38
__hash__() (owlapy.class_expression.OWLObjectHasSelf method), 43
  _hash___() (owlapy.class_expression.OWLObjectOneOf method), 51
  _hash___() (owlapy.class_expression.OWLObjectSomeValuesFrom method), 45
  _hash___() (owlapy.class_expression.restriction.OWLDataAllValuesFrom method), 33
__hash__ () (owlapy.class_expression.restriction.OWLDataCardinalityRestriction method), 31
__hash__() (owlapy.class_expression.restriction.OWLDataHasValue method), 33
  _hash___() (owlapy.class_expression.restriction.OWLDataOneOf method), 34
  hash___() (owlapy.class_expression.restriction.OWLDataSomeValuesFrom method), 32
  _hash__ () (owlapy.class_expression.restriction.OWLDatatypeRestriction method), 35
__hash__() (owlapy.class_expression.restriction.OWLFacetRestriction method), 35
  _hash___() (owlapy.class_expression.restriction.OWLHasValueRestriction method), 25
  _hash__ () (owlapy.class_expression.restriction.OWLObjectAllValuesFrom method), 29
  hash__() (owlapy.class_expression.restriction.OWLObjectCardinalityRestriction method), 27
  hash () (owlapy.class expression.restriction.OWLObjectHasSelf method), 29
__hash__() (owlapy.class_expression.restriction.OWLObjectOneOf method), 30
  _hash___() (owlapy.class_expression.restriction.OWLObjectSomeValuesFrom method), 28
__hash__() (owlapy.iri.IRI method), 55
  _hash___() (owlapy.namespaces.Namespaces method), 59
__hash__() (owlapy.owl_axiom.OWLAnnotation method), 71
__hash__() (owlapy.owl_axiom.OWLAnnotationAssertionAxiom method), 72
  _hash__() (owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom method), 73
  _hash___() (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom method), 73
  _hash___() (owlapy.owl_axiom.OWLClassAssertionAxiom method), 70
__hash__() (owlapy.owl_axiom.OWLDataPropertyCharacteristicAxiom method), 78
__hash__() (owlapy.owl_axiom.OWLDatatypeDefinitionAxiom method), 64
  _hash___() (owlapy.owl_axiom.OWLDeclarationAxiom method), 64
  _hash___() (owlapy.owl_axiom.OWLDisjointUnionAxiom method), 70
  _hash___() (owlapy.owl_axiom.OWLHasKeyAxiom method), 65
__hash__() (owlapy.owl_axiom.OWLNaryClassAxiom method), 65
__hash__ () (owlapy.owl_axiom.OWLNaryIndividualAxiom method), 66
  _hash___() (owlapy.owl_axiom.OWLNaryPropertyAxiom method), 67
  _hash___() (owlapy.owl_axiom.OWLObjectPropertyCharacteristicAxiom method), 76
  _hash___() (owlapy.owl_axiom.OWLPropertyAssertionAxiom method), 74
__hash__() (owlapy.owl_axiom.OWLPropertyDomainAxiom method), 78
__hash__() (owlapy.owl_axiom.OWLPropertyRangeAxiom method), 78
  _hash___() (owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom method), 72
  _hash___() (owlapy.owl_axiom.OWLSubClassOfAxiom method), 69
__hash__ () (owlapy.owl_axiom.OWLSubPropertyAxiom method), 73
__hash__() (owlapy.owl_data_ranges.OWLDataComplementOf method), 81
  _hash___() (owlapy.owl_data_ranges.OWLNaryDataRange method), 81
  _hash___() (owlapy.owl_object.OWLNamedObject method), 91
  _hash___() (owlapy.owl_object.OWLObject method), 90
  hash () (owlapy.owl ontology.Ontology method), 97
__hash__() (owlapy.owl_property.OWLObjectInverseOf method), 104
  _iter__() (owlapy.utils.EvaluatedDescriptionSet method), 140
__len__() (owlapy.owl_hierarchy.AbstractHierarchy method), 84
  _lt___() (owlapy.owl_object.OWLNamedObject method), 91
__lt__() (owlapy.utils.OrderedOWLObject method), 141
__repr__() (owlapy.class_expression.class_expression.OWLObjectComplementOf method), 20
repr_() (owlapy.class_expression.nary_boolean_expression.OWLNaryBooleanClassExpression method), 21
```

```
__repr__() (owlapy.class_expression.OWLDataAllValuesFrom method), 49
__repr__() (owlapy.class_expression.OWLDataCardinalityRestriction method), 44
 _repr__ () (owlapy.class_expression.OWLDataHasValue method), 49
__repr__() (owlapy.class_expression.OWLDataOneOf method), 44
__repr__() (owlapy.class_expression.OWLDataSomeValuesFrom method), 48
__repr__() (owlapy.class_expression.OWLDatatypeRestriction method), 46
  repr_() (owlapy.class_expression.OWLFacetRestriction method), 47
repr () (owlapy.class expression.OWLNaryBooleanClassExpression method), 40
__repr__() (owlapy.class_expression.OWLObjectAllValuesFrom method), 45
__repr__() (owlapy.class_expression.OWLObjectCardinalityRestriction method), 43
__repr__() (owlapy.class_expression.OWLObjectComplementOf method), 38
__repr__() (owlapy.class_expression.OWLObjectHasSelf method), 43
__repr__() (owlapy.class_expression.OWLObjectHasValue method), 46
__repr__() (owlapy.class_expression.OWLObjectOneOf method), 51
__repr__() (owlapy.class_expression.OWLObjectSomeValuesFrom method), 45
__repr__() (owlapy.class_expression.restriction.OWLDataAllValuesFrom method), 33
 repr_() (owlapy.class_expression.restriction.OWLDataCardinalityRestriction method), 31
__repr__() (owlapy.class_expression.restriction.OWLDataHasValue method), 33
__repr__() (owlapy.class_expression.restriction.OWLDataOneOf method), 34
__repr__() (owlapy.class_expression.restriction.OWLDataSomeValuesFrom method), 32
__repr__ () (owlapy.class_expression.restriction.OWLDatatypeRestriction method), 35
  repr_() (owlapy.class_expression.restriction.OWLFacetRestriction method), 35
__repr__() (owlapy.class_expression.restriction.OWLObjectAllValuesFrom method), 29
__repr__() (owlapy.class_expression.restriction.OWLObjectCardinalityRestriction method), 27
__repr__() (owlapy.class_expression.restriction.OWLObjectHasSelf method), 29
__repr__() (owlapy.class_expression.restriction.OWLObjectHasValue method), 30
__repr__() (owlapy.class_expression.restriction.OWLObjectOneOf method), 30
__repr__() (owlapy.class_expression.restriction.OWLObjectSomeValuesFrom method), 28
__repr__() (owlapy.iri.IRI method), 55
__repr__ () (owlapy.namespaces.Namespaces method), 59
__repr__() (owlapy.owl_axiom.OWLAnnotation method), 71
__repr__() (owlapy.owl_axiom.OWLAnnotationAssertionAxiom method), 72
__repr__() (owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom method), 73
__repr__() (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom method), 73
__repr__() (owlapy.owl_axiom.OWLClassAssertionAxiom method), 70
          () (owlapy.owl_axiom.OWLDataPropertyCharacteristicAxiom method), 78
repr () (owlapy.owl axiom.OWLDatatypeDefinitionAxiom method), 64
__repr__() (owlapy.owl_axiom.OWLDeclarationAxiom method), 64
__repr__() (owlapy.owl_axiom.OWLDisjointUnionAxiom method), 70
__repr__() (owlapy.owl_axiom.OWLHasKeyAxiom method), 65
__repr__() (owlapy.owl_axiom.OWLInverseObjectPropertiesAxiom method), 68
__repr__() (owlapy.owl_axiom.OWLNaryClassAxiom method), 65
__repr__() (owlapy.owl_axiom.OWLNaryIndividualAxiom method), 66
__repr__() (owlapy.owl_axiom.OWLNaryPropertyAxiom method), 67
__repr__() (owlapy.owl_axiom.OWLObjectPropertyCharacteristicAxiom method), 76
 repr_() (owlapy.owl_axiom.OWLPropertyAssertionAxiom method), 74
__repr__() (owlapy.owl_axiom.OWLPropertyDomainAxiom method), 78
__repr__() (owlapy.owl_axiom.OWLPropertyRangeAxiom method), 79
__repr__() (owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom method), 72
__repr__() (owlapy.owl_axiom.OWLSubClassOfAxiom method), 69
 _repr__() (owlapy.owl_axiom.OWLSubPropertyAxiom method). 73
__repr__() (owlapy.owl_data_ranges.OWLDataComplementOf method), 81
__repr__() (owlapy.owl_data_ranges.OWLNaryDataRange method), 80
__repr__() (owlapy.owl_object.OWLNamedObject method), 91
__repr__() (owlapy.owl_object.OWLObject method), 91
__repr__() (owlapy.owl_ontology.Ontology method), 97
__repr__() (owlapy.owl_ontology.OWLOntologyID method), 93
__repr__() (owlapy.owl_property.OWLObjectInverseOf method), 104
__setitem__() (owlapy.utils.LRUCache method), 142
__slots__ (owlapy.class_expression.class_expression.OWLBooleanClassExpression attribute), 19
__slots__ (owlapy.class_expression.class_expression.OWLClassExpression attribute), 18
\verb|\__slots__(owlapy.class\_expression.class\_expression.OWLObjectComplementOf \ attribute), 20
__slots__ (owlapy.class_expression.nary_boolean_expression.OWLNaryBooleanClassExpression attribute), 20
__slots__ (owlapy.class_expression.nary_boolean_expression.OWLObjectIntersectionOf attribute), 21
  _slots__ (owlapy.class_expression.nary_boolean_expression.OWLObjectUnionOf attribute), 21
__slots__ (owlapy.class_expression.owl_class.OWLClass attribute), 22
__slots__ (owlapy.class_expression.OWLBooleanClassExpression attribute), 38
__slots__ (owlapy.class_expression.OWLCardinalityRestriction attribute), 42
```

```
__slots__ (owlapy.class_expression.OWLClass attribute), 39
__slots__ (owlapy.class_expression.OWLClassExpression attribute), 37
 slots (owlapy.class expression.OWLDataAllValuesFrom attribute), 49
__slots__ (owlapy.class_expression.OWLDataCardinalityRestriction attribute), 44
__slots__ (owlapy.class_expression.OWLDataExactCardinality attribute), 50
__slots__ (owlapy.class_expression.OWLDataHasValue attribute), 49
  _slots__ (owlapy.class_expression.OWLDataMaxCardinality attribute), 50
slots (owlapy.class expression.OWLDataMinCardinality attribute), 50
__slots__ (owlapy.class_expression.OWLDataRestriction attribute), 42
__slots__ (owlapy.class_expression.OWLDataSomeValuesFrom attribute), 48
__slots__ (owlapy.class_expression.OWLDatatypeRestriction attribute), 46
__slots__(owlapy.class_expression.OWLFacetRestriction attribute), 47
__slots__(owlapy.class_expression.OWLHasValueRestriction attribute), 42
__slots__ (owlapy.class_expression.OWLNaryBooleanClassExpression attribute), 40
__slots__ (owlapy.class_expression.OWLObjectAllValuesFrom attribute), 45
__slots__(owlapy.class_expression.OWLObjectCardinalityRestriction attribute), 43
__slots__ (owlapy.class_expression.OWLObjectComplementOf attribute), 38
__slots__ (owlapy.class_expression.OWLObjectExactCardinality attribute), 48
__slots__(owlapy.class_expression.OWLObjectHasSelf attribute), 43
__slots__ (owlapy.class_expression.OWLObjectHasValue attribute), 46
__slots__ (owlapy.class_expression.OWLObjectIntersectionOf attribute), 40
 _slots__ (owlapy.class_expression.OWLObjectMaxCardinality attribute). 48
__slots__ (owlapy.class_expression.OWLObjectMinCardinality attribute), 47
__slots__ (owlapy.class_expression.OWLObjectOneOf attribute), 51
__slots__ (owlapy.class_expression.OWLObjectRestriction attribute), 41
__slots__ (owlapy.class_expression.OWLObjectSomeValuesFrom attribute), 45
 _slots__ (owlapy.class_expression.OWLObjectUnionOf attribute), 40
__slots__ (owlapy.class_expression.OWLQuantifiedDataRestriction attribute), 44
__slots__ (owlapy.class_expression.OWLQuantifiedObjectRestriction attribute), 41
__slots__ (owlapy.class_expression.OWLQuantifiedRestriction attribute), 41
__slots__ (owlapy.class_expression.OWLRestriction attribute), 40
slots (owlapy.class expression.restriction.OWLCardinalityRestriction attribute), 26
__slots__ (owlapy.class_expression.restriction.OWLDataAllValuesFrom attribute), 33
__slots__ (owlapy.class_expression.restriction.OWLDataCardinalityRestriction attribute), 31
__slots__ (owlapy.class_expression.restriction.OWLDataExactCardinality attribute), 32
  _slots__ (owlapy.class_expression.restriction.OWLDataHasValue attribute), 33
slots (owlapy.class expression.restriction.OWLDataMaxCardinality attribute), 32
__slots__ (owlapy.class_expression.restriction.OWLDataMinCardinality attribute), 31
__slots__ (owlapy.class_expression.restriction.OWLDataRestriction attribute), 30
__slots__ (owlapy.class_expression.restriction.OWLDataSomeValuesFrom attribute), 32
__slots__ (owlapy.class_expression.restriction.OWLDatatypeRestriction attribute), 34
__slots__ (owlapy.class_expression.restriction.OWLFacetRestriction attribute), 35
__slots__ (owlapy.class_expression.restriction.OWLHasValueRestriction attribute), 25
__slots__ (owlapy.class_expression.restriction.OWLObjectAllValuesFrom attribute). 28
__slots__ (owlapy.class_expression.restriction.OWLObjectCardinalityRestriction attribute), 27
 _slots__ (owlapy.class_expression.restriction.OWLObjectExactCardinality attribute), 28
__slots__ (owlapy.class_expression.restriction.OWLObjectHasSelf attribute), 29
__slots__ (owlapy.class_expression.restriction.OWLObjectHasValue attribute), 29
__slots__ (owlapy.class_expression.restriction.OWLObjectMaxCardinality attribute), 27
__slots__ (owlapy.class_expression.restriction.OWLObjectMinCardinality attribute), 27
 _slots__ (owlapy.class_expression.restriction.OWLObjectOneOf attribute). 30
__slots__ (owlapy.class_expression.restriction.OWLObjectRestriction attribute), 25
__slots__ (owlapy.class_expression.restriction.OWLObjectSomeValuesFrom attribute), 28
__slots__ (owlapy.class_expression.restriction.OWLQuantifiedDataRestriction attribute), 31
__slots__ (owlapy.class_expression.restriction.OWLQuantifiedObjectRestriction attribute), 26
slots (owlapy.class expression.restriction.OWLQuantifiedRestriction attribute), 26
__slots__ (owlapy.class_expression.restriction.OWLRestriction attribute), 25
__slots__ (owlapy.converter.Owl2SparqlConverter attribute), 53
__slots__ (owlapy.converter.VariablesMapping attribute), 53
__slots__ (owlapy.iri.IRI attribute), 55
slots (owlapy.meta classes.HasCardinality attribute), 58
__slots__(owlapy.meta_classes.HasFiller attribute), 57
__slots__(owlapy.meta_classes.HasIRI attribute), 57
__slots__ (owlapy.meta_classes.HasOperands attribute), 57
  _slots__ (owlapy.namespaces.Namespaces attribute), 58
__slots__ (owlapy.owl_annotation.OWLAnnotationObject attribute), 59
__slots__ (owlapy.owl_annotation.OWLAnnotationSubject attribute), 59
__slots__ (owlapy.owl_annotation.OWLAnnotationValue attribute), 60
```

```
__slots__ (owlapy.owl_axiom.OWLAnnotation attribute), 71
__slots__ (owlapy.owl_axiom.OWLAnnotationAssertionAxiom attribute), 71
 slots (owlapy.owl axiom.OWLAnnotationAxiom attribute), 71
__slots__ (owlapy.owl_axiom.OWLAnnotationProperty attribute), 70
__slots__ (owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom attribute), 72
__slots__ (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom attribute), 73
  _slots__ (owlapy.owl_axiom.OWLAsymmetricObjectPropertyAxiom attribute), 76
slots (owlapy.owl axiom.OWLAxiom attribute), 62
__slots__ (owlapy.owl_axiom.OWLClassAssertionAxiom attribute), 70
__slots__ (owlapy.owl_axiom.OWLClassAxiom attribute), 63
__slots__ (owlapy.owl_axiom.OWLDataPropertyAssertionAxiom attribute), 75
__slots__(owlapy.owl_axiom.OWLDataPropertyAxiom attribute), 63
__slots__ (owlapy.owl_axiom.OWLDataPropertyCharacteristicAxiom attribute), 78
__slots__ (owlapy.owl_axiom.OWLDataPropertyDomainAxiom attribute), 79
__slots__ (owlapy.owl_axiom.OWLDataPropertyRangeAxiom attribute), 80
  _slots__ (owlapy.owl_axiom.OWLDatatypeDefinitionAxiom attribute), 64
__slots__ (owlapy.owl_axiom.OWLDeclarationAxiom attribute), 63
__slots__ (owlapy.owl_axiom.OWLDifferentIndividualsAxiom attribute), 67
__slots__ (owlapy.owl_axiom.OWLDisjointClassesAxiom attribute), 66
__slots__ (owlapy.owl_axiom.OWLDisjointDataPropertiesAxiom attribute), 69
__slots__ (owlapy.owl_axiom.OWLDisjointObjectPropertiesAxiom attribute), 68
 _slots__ (owlapy.owl_axiom.OWLDisjointUnionAxiom attribute). 69
__slots__ (owlapy.owl_axiom.OWLEquivalentClassesAxiom attribute), 66
__slots__ (owlapy.owl_axiom.OWLEquivalentDataPropertiesAxiom attribute), 68
__slots__ (owlapy.owl_axiom.OWLEquivalentObjectPropertiesAxiom attribute), 68
__slots__ (owlapy.owl_axiom.OWLFunctionalDataPropertyAxiom attribute), 78
 slots (owlapy.owl axiom.OWLFunctionalObjectPropertyAxiom attribute), 76
__slots__ (owlapy.owl_axiom.OWLHasKeyAxiom attribute), 64
__slots__ (owlapy.owl_axiom.OWLIndividualAxiom attribute), 63
__slots__ (owlapy.owl_axiom.OWLInverseFunctionalObjectPropertyAxiom attribute), 76
__slots__ (owlapy.owl_axiom.OWLInverseObjectPropertiesAxiom attribute), 68
__slots__ (owlapy.owl_axiom.OWLIrreflexiveObjectPropertyAxiom attribute), 77
__slots__ (owlapy.owl_axiom.OWLLogicalAxiom attribute), 63
__slots__ (owlapy.owl_axiom.OWLNaryAxiom attribute), 65
__slots__ (owlapy.owl_axiom.OWLNaryClassAxiom attribute), 65
  _slots__ (owlapy.owl_axiom.OWLNaryIndividualAxiom attribute), 66
slots (owlapy.owl axiom.OWLNaryPropertyAxiom attribute), 67
__slots__ (owlapy.owl_axiom.OWLNegativeDataPropertyAssertionAxiom attribute), 75
__slots__ (owlapy.owl_axiom.OWLNegativeObjectPropertyAssertionAxiom attribute). 75
__slots__ (owlapy.owl_axiom.OWLObjectPropertyAssertionAxiom attribute), 75
__slots__ (owlapy.owl_axiom.OWLObjectPropertyAxiom attribute), 63
__slots__ (owlapy.owl_axiom.OWLObjectPropertyCharacteristicAxiom attribute), 76
__slots__ (owlapy.owl_axiom.OWLObjectPropertyDomainAxiom attribute), 79
__slots__ (owlapy.owl_axiom.OWLObjectPropertyRangeAxiom attribute). 79
__slots__ (owlapy.owl_axiom.OWLPropertyAssertionAxiom attribute), 74
  _slots__ (owlapy.owl_axiom.OWLPropertyAxiom attribute), 63
__slots__ (owlapy.owl_axiom.OWLPropertyDomainAxiom attribute), 78
__slots__ (owlapy.owl_axiom.OWLPropertyRangeAxiom attribute), 78
__slots__ (owlapy.owl_axiom.OWLReflexiveObjectPropertyAxiom attribute), 77
  _slots__ (owlapy.owl_axiom.OWLSameIndividualAxiom attribute), 67
 slots (owlapy.owl axiom.OWLSubAnnotationPropertyOfAxiom attribute), 72
__slots__ (owlapy.owl_axiom.OWLSubClassOfAxiom attribute), 69
__slots__ (owlapy.owl_axiom.OWLSubDataPropertyOfAxiom attribute), 74
__slots__ (owlapy.owl_axiom.OWLSubObjectPropertyOfAxiom attribute), 74
__slots__ (owlapy.owl_axiom.OWLSubPropertyAxiom attribute), 73
__slots__ (owlapy.owl_axiom.OWLSymmetricObjectPropertyAxiom attribute), 77
__slots__ (owlapy.owl_axiom.OWLTransitiveObjectPropertyAxiom attribute), 77
__slots__ (owlapy.owl_axiom.OWLUnaryPropertyAxiom attribute), 75
__slots__ (owlapy.owl_data_ranges.OWLDataIntersectionOf attribute), 81
  _slots__ (owlapy.owl_data_ranges.OWLDataUnionOf attribute), 81
__slots__(owlapy.owl_data_ranges.OWLNaryDataRange attribute), 80
__slots__ (owlapy.owl_datatype.OWLDatatype attribute), 82
__slots__ (owlapy.owl_hierarchy.AbstractHierarchy attribute), 83
__slots__ (owlapy.owl_individual.OWLIndividual attribute), 86
  _slots__ (owlapy.owl_individual.OWLNamedIndividual attribute), 86
__slots__ (owlapy.owl_literal.OWLLiteral attribute), 88
__slots__(owlapy.owl_object.OWLEntity attribute), 91
__slots__ (owlapy.owl_object.OWLNamedObject attribute), 91
```

```
__slots__ (owlapy.owl_object.OWLObject attribute), 90
__slots__ (owlapy.owl_ontology_manager.AddImport attribute), 100
 _slots__ (owlapy.owl_ontology_manager.OntologyManager attribute), 100
__slots__ (owlapy.owl_ontology_manager.OWLImportsDeclaration attribute), 99
__slots__(owlapy.owl_ontology_manager.OWLOntologyChange attribute), 98
__slots__ (owlapy.owl_ontology.FromOwlready2 attribute), 97
  _slots__ (owlapy.owl_ontology.Ontology attribute), 95
__slots__ (owlapy.owl_ontology.OWLOntology attribute), 93
__slots__(owlapy.owl_ontology.OWLOntologyID attribute), 92
__slots__ (owlapy.owl_ontology.ToOwlready2 attribute), 97
__slots__ (owlapy.owl_property.OWLDataProperty attribute), 104
  _slots__ (owlapy.owl_property.OWLDataPropertyExpression attribute), 102
__slots__(owlapy.owl_property.OWLObjectInverseOf attribute), 104
__slots__(owlapy.owl_property.OWLObjectProperty attribute), 103
__slots__ (owlapy.owl_property.OWLObjectPropertyExpression attribute). 102
__slots__ (owlapy.owl_property.OWLProperty attribute), 103
  _slots__ (owlapy.owl_property.OWLPropertyExpression attribute), 101
__slots__ (owlapy.owl_reasoner.FastInstanceCheckerReasoner attribute), 119
__slots__ (owlapy.owl_reasoner.OntologyReasoner attribute), 112
__slots__ (owlapy.owl_reasoner.OWLReasoner attribute), 105
__slots__ (owlapy.owl_reasoner.SyncReasoner attribute), 125
  _slots__ (owlapy.render.DLSyntaxObjectRenderer attribute), 135
__slots__ (owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer attribute), 136
__slots__ (owlapy.utils.EvaluatedDescriptionSet attribute), 140
__slots__ (owlapy.utils.OrderedOWLObject attribute), 141
__slots__ (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
__version__ (in module owlapy), 145
Α
AbstractHierarchy (class in owlapy.owl_hierarchy), 83
add_axiom() (owlapy.owl_ontology_manager.OntologyManager method), 100
add_axiom() (owlapy.owl_ontology_manager.OWLOntologyManager method), 99
AddImport (class in owlapy.owl_ontology_manager), 99
all_data_property_values() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 121
all_data_property_values() (owlapy.owl_reasoner.OntologyReasoner method), 115
all_data_property_values() (owlapy.owl_reasoner.OWLReasonerEx method), 112
annotations() (owlapy.owl_axiom.OWLAxiom method), 62
append() (owlapy.converter.Owl2SparglConverter method), 54
append_triple() (owlapy.converter.Owl2SparqlConverter method), 54
\verb"apply_change" () \textit{ (owlapy.owl\_ontology\_manager.OntologyManager method)}, 100
apply_change() (owlapy.owl_ontology_manager.OWLOntologyManager method), 98
as_anonymous_individual() (owlapy.owl_annotation.OWLAnnotationObject method), 59
as_index() (in module owlapy.utils), 142
as_intersection_of_min_max() (owlapy.class_expression.OWLDataExactCardinality method), 51
\verb|as_intersection_of_min_max|| (owlapy. class\_expression. OWLObject Exact Cardinality method)|, 48| \\
as_intersection_of_min_max() (owlapy.class_expression.restriction.OWLDataExactCardinality method), 32
as_intersection_of_min_max() (owlapy.class_expression.restriction.OWLObjectExactCardinality method), 28
as iri() (owlapy.iri.IRI method), 56
as_iri() (owlapy.owl_annotation.OWLAnnotationObject method), 59
\verb|as_literal()| (owlapy.owl_annotation.OWLAnnotationValue\ method), 60
as_literal() (owlapy.owl_literal.OWLLiteral method), 89
\verb|as_object_union_of()| (owlapy.class\_expression.OWLObjectOneOf method), 51|
as_object_union_of() (owlapy.class_expression.restriction.OWLObjectOneOf method), 30
as_pairwise_axioms() (owlapy.owl_axiom.OWLNaryAxiom method), 65
as_pairwise_axioms() (owlapy.owl_axiom.OWLNaryClassAxiom method), 65
as_pairwise_axioms() (owlapy.owl_axiom.OWLNaryIndividualAxiom method), 66
as_pairwise_axioms() (owlapy.owl_axiom.OWLNaryPropertyAxiom method), 67
as_query() (owlapy.converter.Owl2SparqlConverter method), 54
as_some_values_from() (owlapy.class_expression.OWLDataHasValue method), 50
as_some_values_from() (owlapy.class_expression.OWLObjectHasValue method), 46
as_some_values_from() (owlapy.class_expression.restriction.OWLDataHasValue method), 34
as_some_values_from() (owlapy.class_expression.restriction.OWLObjectHasValue method), 29
as_str() (owlapy.iri.IRI method), 56
В
BaseReasoner (class in owlapy.owl reasoner), 111
best () (owlapy.utils.EvaluatedDescriptionSet method), 140
```

```
best quality value () (owlapy.utils. Evaluated Description Set method), 140
BOOLEAN (owlapy.vocab.XSDVocabulary attribute), 143
BooleanOWLDatatype (in module owlapy.owl_literal), 90
cache_clear() (owlapy.utils.LRUCache method), 142
cache_info() (owlapy.utils.LRUCache method), 142
ce (owlapy.converter.Owl2SparqlConverter attribute), 53
children() (owlapy.owl_hierarchy.AbstractHierarchy method), 84
class_expressions() (owlapy.owl_axiom.OWLNaryClassAxiom method), 65
class_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
classes_in_signature() (owlapy.owl_ontology.Ontology method), 95
classes_in_signature() (owlapy.owl_ontology.OWLOntology method), 93
ClassHierarchy (class in owlapy.owl_hierarchy), 84
clean() (owlapy.utils.EvaluatedDescriptionSet method), 140
cnt (owlapy.converter.Owl2SparqlConverter attribute), 53
combine_nary_expressions() (in module owlapy.utils), 141
ConceptOperandSorter (class in owlapy.utils), 140
contains named equivalent class() (owlapy.owl axiom.OWLEquivalentClassesAxiom method), 66
contains_owl_nothing() (owlapy.owl_axiom.OWLEquivalentClassesAxiom method), 66
contains_owl_thing() (owlapy.owl_axiom.OWLEquivalentClassesAxiom method), 66
convert () (owlapy.converter.Owl2SparqlConverter method), 54
convert_from_owlapi() (owlapy.owlapi_adaptor.OWLAPIAdaptor method), 128
convert_to_owlapi() (owlapy.owlapi_adaptor.OWLAPIAdaptor method), 127
converter (in module owlapy.converter), 54
create() (owlapy.iri.IRI static method), 55
create_ontology() (owlapy.owl_ontology_manager.OntologyManager method), 100
create_ontology() (owlapy.owl_ontology_manager.OWLOntologyManager method), 98
current_variable (owlapy.converter.Owl2SparqlConverter property), 54
D
data_all_values_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
data_cardinality_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
data_complement_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
data_has_value_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
data_intersection_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
data_one_of_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
data_properties_in_signature() (owlapy.owl_ontology.Ontology method), 95
data_properties_in_signature() (owlapy.owl_ontology.OWLOntology method), 93
data_property_domain_axioms()(owlapy.owl_ontology.Ontology method), 96
data_property_domain_axioms() (owlapy.owl_ontology.OWLOntology method), 94
data_property_domains() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 119
data_property_domains() (owlapy.owl_reasoner.OntologyReasoner method), 113
data_property_domains()(owlapy.owl_reasoner.OWLReasoner method), 105
data_property_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
{\tt data\_property\_range\_axioms()} \ (\textit{owlapy.owl\_ontology.Ontology method}), 96
data_property_range_axioms() (owlapy.owl_ontology.OWLOntology method), 94
data_property_ranges() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 119
data_property_ranges() (owlapy.owl_reasoner.OWLReasonerEx method), 111
data_property_values() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 121
data_property_values() (owlapy.owl_reasoner.OntologyReasoner method), 115
data_property_values() (owlapy.owl_reasoner.OWLReasoner method), 108
data_some_values_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
data_union_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
datatype_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
DatatypePropertyHierarchy (class in owlapy.owl_hierarchy), 85
DATE (owlapy.vocab.XSDVocabulary attribute), 143
DATE_TIME (owlapy.vocab.XSDVocabulary attribute), 144
DATE_TIME_STAMP (owlapy.vocab.XSDVocabulary attribute), 144
DateOWLDatatype (in module owlapy.owl_literal), 90
DateTimeOWLDatatype (in module owlapy.owl_literal), 90
DECIMAL (owlapy.vocab.XSDVocabulary attribute), 143
{\tt different\_individuals()} \ (\textit{owlapy.owl\_reasoner.FastInstanceCheckerReasoner method}), 121
different_individuals() (owlapy.owl_reasoner.OntologyReasoner method), 114
different_individuals() (owlapy.owl_reasoner.OWLReasoner method), 107
disjoint_classes() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 121
disjoint_classes() (owlapy.owl_reasoner.OntologyReasoner method), 114
```

```
disjoint classes() (owlapy.owl reasoner.OWLReasoner method), 106
disjoint_data_properties() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 124
disjoint_data_properties() (owlapy.owl_reasoner.OntologyReasoner method), 117
disjoint_data_properties() (owlapy.owl_reasoner.OWLReasoner method), 109
disjoint_object_properties() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 124
disjoint_object_properties() (owlapy.owl_reasoner.OntologyReasoner method), 117
disjoint_object_properties() (owlapy.owl_reasoner.OWLReasoner method), 109
DL GRAMMAR (in module owlapy.parser), 131
dl_to_owl_expression() (in module owlapy), 145
dl_to_owl_expression() (in module owlapy.parser), 133
DLparser (in module owlapy.parser), 133
DLrenderer (in module owlapy.render), 136
DLSyntaxObjectRenderer (class in owlapy.render), 135
DLSyntaxParser (class in owlapy.parser), 131
DOUBLE (owlapy.vocab.XSDVocabulary attribute), 143
DoubleOWLDatatype (in module owlapy.owl_literal), 90
download_external_files() (in module owlapy.static_funcs), 137
DURATION (owlapy.vocab.XSDVocabulary attribute), 144
DurationOWLDatatype (in module owlapy.owl_literal), 90
equivalent_classes() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 120
equivalent_classes() (owlapy.owl_reasoner.OntologyReasoner method), 114
equivalent_classes() (owlapy.owl_reasoner.OWLReasoner method), 106
equivalent_classes_axioms() (owlapy.owl_ontology.Ontology method), 95
equivalent_classes_axioms() (owlapy.owl_ontology.OWLOntology method), 94
equivalent_data_properties() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 123
equivalent_data_properties() (owlapy.owl_reasoner.OntologyReasoner method), 117
equivalent_data_properties() (owlapy.owl_reasoner.OWLReasoner method), 107
equivalent_object_properties() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 123
equivalent_object_properties()(owlapy.owl_reasoner.OntologyReasoner method), 116
equivalent_object_properties() (owlapy.owl_reasoner.OWLReasoner method), 107
EvaluatedDescriptionSet (class in owlapy.utils), 139
F
FastInstanceCheckerReasoner (class in owlapy.owl_reasoner), 119
FLOAT (owlapy.vocab.XSDVocabulary attribute), 143
flush() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 122
flush() (owlapy.owl_reasoner.OntologyReasoner method), 115
flush() (owlapy.owl_reasoner.OWLReasoner method), 108
for_all_de_morgan (owlapy.converter.Owl2SparqlConverter attribute), 54
forAll() (owlapy.converter.Owl2SparqlConverter method), 54
forAllDeMorgan() (owlapy.converter.Owl2SparqlConverter method), 54
{\tt FRACTION\_DIGITS}~(owlapy. class\_expression. OWL Facet~attribute),~47
FRACTION_DIGITS (owlapy.vocab.OWLFacet attribute), 144
from_str() (owlapy.class_expression.OWLFacet static method), 47
from_str() (owlapy.vocab.OWLFacet static method), 144
FromOwlready2 (class in owlapy.owl_ontology), 97
G
general_class_axioms() (owlapy.owl_ontology.Ontology method), 96
general_class_axioms() (owlapy.owl_ontology.OWLOntology method), 94
generate_inferred_class_assertion_axioms() (owlapy.owlapi_adaptor.OWLAPIAdaptor method), 127
generic_visit() (owlapy.parser.DLSyntaxParser method), 133
generic_visit() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
get_bottom_entity() (owlapy.owl_hierarchy.AbstractHierarchy class method), 83
get_bottom_entity() (owlapy.owl_hierarchy.ClassHierarchy class method), 84
get_bottom_entity() (owlapy.owl_hierarchy.DatatypePropertyHierarchy class method), 85
get_bottom_entity() (owlapy.owl_hierarchy.ObjectPropertyHierarchy class method), 85
get_cardinality() (owlapy.class_expression.OWLCardinalityRestriction method), 42
get cardinality() (owlapy.class expression.restriction.OWLCardinalityRestriction method), 26
get_cardinality() (owlapy.meta_classes.HasCardinality method), 58
get_class_expression() (owlapy.owl_axiom.OWLClassAssertionAxiom method), 70
get_class_expression() (owlapy.owl_axiom.OWLHasKeyAxiom method), 64
\verb|get_class_expressions()| \textit{(owlapy.owl\_axiom.OWLDisjointUnionAxiom method)}, 69
get_class_nnf() (owlapy.utils.NNF method), 141
```

```
get data range() (owlapy.owl data ranges.OWLDataComplementOf method), 81
get_datarange() (owlapy.owl_axiom.OWLDatatypeDefinitionAxiom method), 64
get datatype() (owlapy.class expression.OWLDatatypeRestriction method), 46
get_datatype() (owlapy.class_expression.restriction.OWLDatatypeRestriction method), 34
get_datatype() (owlapy.owl_axiom.OWLDatatypeDefinitionAxiom method). 64
get_datatype() (owlapy.owl_literal.OWLLiteral method), 89
get_default() (owlapy.utils.OWLClassExpressionLengthMetric static method), 139
get default document iri() (owlapy.owl ontology.OWLOntologyID method), 93
get_domain() (owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom method), 72
get_domain() (owlapy.owl_axiom.OWLPropertyDomainAxiom method), 78
get_entity() (owlapy.owl_axiom.OWLDeclarationAxiom method), 63
get_expression_length() (in module owlapy.utils), 139
get_facet() (owlapy.class_expression.OWLFacetRestriction method), 47
get_facet() (owlapy.class_expression.restriction.OWLFacetRestriction method), 35
get_facet_restrictions() (owlapy.class_expression.OWLDatatypeRestriction method), 46
get_facet_restrictions() (owlapy.class_expression.restriction.OWLDatatypeRestriction method), 34
get_facet_value() (owlapy.class_expression.OWLFacetRestriction method), 47
get_facet_value() (owlapy.class_expression.restriction.OWLFacetRestriction method), 35
get_filler() (owlapy.class_expression.OWLCardinalityRestriction method), 42
get_filler() (owlapy.class_expression.OWLHasValueRestriction method), 42
get_filler() (owlapy.class_expression.OWLQuantifiedDataRestriction method), 44
get filler() (owlapy.class expression.OWLQuantifiedObjectRestriction method), 41
get_filler() (owlapy.class_expression.restriction.OWLCardinalityRestriction method), 26
get_filler() (owlapy.class_expression.restriction.OWLHasValueRestriction method), 25
get_filler() (owlapy.class_expression.restriction.OWLQuantifiedDataRestriction method), 31
get_filler() (owlapy.class_expression.restriction.OWLQuantifiedObjectRestriction method), 26
get_filler() (owlapy.meta_classes.HasFiller method), 57
get_first_property() (owlapy.owl_axiom.OWLInverseObjectPropertiesAxiom method), 68
get_import_declaration() (owlapy.owl_ontology_manager.AddImport method), 100
get_individual() (owlapy.owl_axiom.OWLClassAssertionAxiom method), 70
get_inverse() (owlapy.owl_property.OWLObjectInverseOf method), 104
get_inverse_property() (owlapy.owl_property.OWLObjectInverseOf method), 104
get_inverse_property() (owlapy.owl_property.OWLObjectProperty method), 103
get_inverse_property() (owlapy.owl_property.OWLObjectPropertyExpression method), 102
get_literal() (owlapy.owl_literal.OWLLiteral method), 88
get_named_property() (owlapy.owl_property.OWLObjectInverseOf method), 104
get named property() (owlapy.owl property.OWLObjectProperty method), 103
get_named_property() (owlapy.owl_property.OWLObjectPropertyExpression method), 102
get_namespace() (owlapy.iri.IRI method), 56
get_nnf() (owlapy.class_expression.class_expression.OWLAnonymousClassExpression method), 19
get_nnf() (owlapy.class_expression.class_expression.OWLClassExpression method), 19
get_nnf() (owlapy.class_expression.owl_class.OWLClass method), 22
get_nnf() (owlapy.class_expression.OWLAnonymousClassExpression method), 38
get_nnf() (owlapy.class_expression.OWLClass method), 39
get_nnf() (owlapy.class_expression.OWLClassExpression method), 37
get_object() (owlapy.owl_axiom.OWLPropertyAssertionAxiom method), 74
qet_object_complement_of() (owlapy.class_expression.class_expression.OWLAnonymousClassExpression method), 19
get_object_complement_of() (owlapy.class_expression.class_expression.OWLClassExpression method), 19
get_object_complement_of() (owlapy.class_expression.owl_class.OWLClass method), 22
get_object_complement_of() (owlapy.class_expression.OWLAnonymousClassExpression method), 38
get_object_complement_of() (owlapy.class_expression.OWLClass method), 39
get_object_complement_of() (owlapy.class_expression.OWLClassExpression method), 37
get_ontology() (owlapy.owl_ontology_manager.OWLOntologyChange method), 98
get_ontology_id() (owlapy.owl_ontology.Ontology method), 96
get_ontology_id() (owlapy.owl_ontology.OWLOntology method), 95
get_ontology_iri() (owlapy.owl_ontology.OWLOntologyID method), 92
get_operand() (owlapy.class_expression.class_expression.OWLObjectComplementOf method), 20
get_operand() (owlapy.class_expression.OWLObjectComplementOf method), 38
get_original_iri() (owlapy.owl_ontology.Ontology method), 97
get_owl_class() (owlapy.owl_axiom.OWLDisjointUnionAxiom method), 69
get owl disjoint classes axiom() (owlapy.owl axiom.OWLDisjointUnionAxiom method), 70
get_owl_equivalent_classes_axiom() (owlapy.owl_axiom.OWLDisjointUnionAxiom method), 69
get_owl_ontology_manager() (owlapy.owl_ontology.Ontology method), 96
get_owl_ontology_manager() (owlapy.owl_ontology.OWLOntology method), 95
get_property() (owlapy.class_expression.OWLDataAllValuesFrom method), 49
get_property() (owlapy.class_expression.OWLDataCardinalityRestriction method), 44
get_property() (owlapy.class_expression.OWLDataHasValue method), 50
get_property() (owlapy.class_expression.OWLDataSomeValuesFrom method), 49
```

```
get property() (owlapy.class expression.OWLObjectAllValuesFrom method), 45
get_property() (owlapy.class_expression.OWLObjectCardinalityRestriction method), 43
get property() (owlapy.class expression.OWLObjectHasSelf method), 43
get_property() (owlapy.class_expression.OWLObjectHasValue method), 46
get_property() (owlapy.class_expression.OWLObjectRestriction method), 41
get_property() (owlapy.class_expression.OWLObjectSomeValuesFrom method), 45
\verb"get_property" () \textit{ (owlapy.class\_expression.OWLR estriction method)}, 40
get property() (owlapy.class expression.restriction.OWLDataAllValuesFrom method), 33
get_property() (owlapy.class_expression.restriction.OWLDataCardinalityRestriction method), 31
get_property() (owlapy.class_expression.restriction.OWLDataHasValue method), 34
get_property() (owlapy.class_expression.restriction.OWLDataSomeValuesFrom method), 32
get_property() (owlapy.class_expression.restriction.OWLObjectAllValuesFrom method), 29
get_property() (owlapy.class_expression.restriction.OWLObjectCardinalityRestriction method), 27
get_property() (owlapy.class_expression.restriction.OWLObjectHasSelf method), 29
get_property() (owlapy.class_expression.restriction.OWLObjectHasValue method), 29
get_property() (owlapy.class_expression.restriction.OWLObjectRestriction method), 26
get_property() (owlapy.class_expression.restriction.OWLObjectSomeValuesFrom method), 28
get_property() (owlapy.class_expression.restriction.OWLRestriction method), 25
get_property() (owlapy.owl_axiom.OWLAnnotation method), 71
get_property() (owlapy.owl_axiom.OWLAnnotationAssertionAxiom method), 71
get_property() (owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom method), 72
get_property() (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom method), 73
get_property() (owlapy.owl_axiom.OWLPropertyAssertionAxiom method), 74
get_property() (owlapy.owl_axiom.OWLUnaryPropertyAxiom method), 76
get_property_expressions() (owlapy.owl_axiom.OWLHasKeyAxiom method), 64
get_range() (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom method), 73
get_range() (owlapy.owl_axiom.OWLPropertyRangeAxiom method), 78
get_remainder() (owlapy.iri.IRI method), 56
\verb|get_root_ontology|()| \textit{(owlapy.owl\_reasoner.FastInstanceCheckerReasoner method)}, 125
get_root_ontology() (owlapy.owl_reasoner.OntologyReasoner method), 119
get_root_ontology() (owlapy.owl_reasoner.OWLReasoner method), 111
get_second_property() (owlapy.owl_axiom.OWLInverseObjectPropertiesAxiom method), 68
get_short_form() (owlapy.iri.IRI method), 56
get_sub_class() (owlapy.owl_axiom.OWLSubClassOfAxiom method), 69
get_sub_property() (owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom method), 72
get_sub_property() (owlapy.owl_axiom.OWLSubPropertyAxiom method), 73
get subject() (owlapy.owl axiom.OWLAnnotationAssertionAxiom method), 71
get_subject() (owlapy.owl_axiom.OWLPropertyAssertionAxiom method), 74
get_super_class() (owlapy.owl_axiom.OWLSubClassOfAxiom method), 69
get_super_property() (owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom method), 72
get_super_property() (owlapy.owl_axiom.OWLSubPropertyAxiom method), 73
get_top_entity() (owlapy.owl_hierarchy.AbstractHierarchy class method), 83
get_top_entity() (owlapy.owl_hierarchy.ClassHierarchy class method), 84
get_top_entity() (owlapy.owl_hierarchy.DatatypePropertyHierarchy class method), 85
get_top_entity() (owlapy.owl_hierarchy.ObjectPropertyHierarchy class method), 85
get_top_level_cnf() (owlapy.utils.TopLevelCNF method), 141
get_top_level_dnf() (owlapy.utils.TopLevelDNF method), 141
get_value() (owlapy.owl_axiom.OWLAnnotation method), 71
get_value() (owlapy.owl_axiom.OWLAnnotationAssertionAxiom method), 72
get_variable() (owlapy.converter.VariablesMapping method), 53
get version iri() (owlapy.owl ontology.OWLOntologyID method), 92
grouping_vars (owlapy.converter.Owl2SparqlConverter attribute), 53
has_consistent_ontology() (owlapy.owlapi_adaptor.OWLAPIAdaptor method), 128
HasCardinality (class in owlapy.meta_classes), 58
HasFiller (class in owlapy.meta_classes), 57
HasIndex (class in owlapy.utils), 140
HasIRI (class in owlapy.meta_classes), 57
HasOperands (class in owlapy.meta_classes), 57
having_conditions (owlapy.converter.Owl2SparqlConverter attribute), 53
HERMIT (owlapy.owl_reasoner.BaseReasoner attribute), 111
ind_data_properties() (owlapy.owl_reasoner.OWLReasonerEx method), 112
ind_object_properties() (owlapy.owl_reasoner.OWLReasonerEx method), 112
individuals () (owlapy.class_expression.OWLObjectOneOf method), 51
```

```
individuals () (owlapy.class expression.restriction.OWLObjectOneOf method), 30
individuals () (owlapy.owl_axiom.OWLNaryIndividualAxiom method), 66
individuals in signature() (owlapy.owl ontology.Ontology method), 95
\verb|individuals_in_signature()| (owlapy.owl_ontology.OWLOntology.method), 93
instances() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 122
instances() (owlapy.owl_reasoner.OntologyReasoner method), 115
instances() (owlapy.owl_reasoner.OWLReasoner method), 108
instances() (owlapy.owl reasoner.SyncReasoner method), 126
instances() (owlapy.owlapi_adaptor.OWLAPIAdaptor method), 128
INTEGER (owlapy.vocab.XSDVocabulary attribute), 143
IntegerOWLDatatype (in module owlapy.owl_literal), 90
IRI (class in owlapy.iri), 55
iri (owlapy.class_expression.owl_class.OWLClass property), 22
iri (owlapy.class_expression.OWLClass property), 39
iri (owlapy.meta_classes.HasIRI property), 57
iri (owlapy.owl_axiom.OWLAnnotationProperty property), 70
iri (owlapy.owl_datatype.OWLDatatype property), 82
iri (owlapy.owl_individual.OWLNamedIndividual property), 86
iri (owlapy.owl_ontology_manager.OWLImportsDeclaration property), 99
iri (owlapy.owl_property.OWLProperty property), 103
is_annotated() (owlapy.owl_axiom.OWLAxiom method), 62
is annotation axiom() (owlapy.owl axiom.OWLAnnotationAxiom method), 71
is_annotation_axiom() (owlapy.owl_axiom.OWLAxiom method), 62
is_anonymous() (owlapy.owl_object.OWLEntity method), 92
is_anonymous() (owlapy.owl_object.OWLObject method), 91
is_anonymous()(owlapy.owl_ontology.OWLOntology method), 95
is_anonymous() (owlapy.owl_ontology.OWLOntologyID method), 93
is_boolean() (owlapy.owl_literal.OWLLiteral method), 88
is_child_of() (owlapy.owl_hierarchy.AbstractHierarchy method), 84
is_data_property_expression() (owlapy.owl_property.OWLDataPropertyExpression method), 102
is_data_property_expression() (owlapy.owl_property.OWLPropertyExpression method), 101
is data restriction() (owlapy.class expression.OWLDataRestriction method), 42
\verb|is_data_restriction()| (owlapy.class\_expression.OWLR estriction method), 40
is_data_restriction() (owlapy.class_expression.restriction.OWLDataRestriction method), 30
is_data_restriction() (owlapy.class_expression.restriction.OWLRestriction method), 25
is_date() (owlapy.owl_literal.OWLLiteral method), 89
is datetime() (owlapy.owl literal.OWLLiteral method), 89
is_double() (owlapy.owl_literal.OWLLiteral method), 88
is_duration() (owlapy.owl_literal.OWLLiteral method), 89
is_integer() (owlapy.owl_literal.OWLLiteral method), 88
is_isolated() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 119
is_isolated() (owlapy.owl_reasoner.OntologyReasoner method), 119
is_isolated() (owlapy.owl_reasoner.OWLReasoner method), 111
is_literal() (owlapy.owl_annotation.OWLAnnotationValue method), 60
is_literal() (owlapy.owl_literal.OWLLiteral method), 89
is_logical_axiom() (owlapy.owl_axiom.OWLAxiom method), 62
is_logical_axiom() (owlapy.owl_axiom.OWLLogicalAxiom method), 63
is_nothing() (owlapy.iri.IRI method), 55
is_object_property_expression() (owlapy.owl_property.OWLObjectPropertyExpression method), 102
is_object_property_expression() (owlapy.owl_property.OWLPropertyExpression method), 102
is_object_restriction() (owlapy.class_expression.OWLObjectRestriction method), 41
is_object_restriction() (owlapy.class_expression.OWLRestriction method), 41
is_object_restriction() (owlapy.class_expression.restriction.OWLObjectRestriction method), 25
is_object_restriction() (owlapy.class_expression.restriction.OWLRestriction method), 25
is_owl_nothing() (owlapy.class_expression.class_expression.OWLAnonymousClassExpression method), 19
is_owl_nothing() (owlapy.class_expression.class_expression.OWLClassExpression method), 19
is_owl_nothing() (owlapy.class_expression.owl_class.OWLClass method), 22
is_owl_nothing() (owlapy.class_expression.OWLAnonymousClassExpression method), 37
is_owl_nothing() (owlapy.class_expression.OWLClass method), 39
is_owl_nothing() (owlapy.class_expression.OWLClassExpression method), 37
is owl thing() (owlapy.class expression.class expression.OWLAnonymousClassExpression method), 19
is_owl_thing() (owlapy.class_expression.class_expression.OWLClassExpression method), 18
is_owl_thing() (owlapy.class_expression.owl_class.OWLClass method), 22
is_owl_thing() (owlapy.class_expression.OWLAnonymousClassExpression method), 38
is_owl_thing() (owlapy.class_expression.OWLClass method).39
is_owl_thing() (owlapy.class_expression.OWLClassExpression method), 37
is_owl_top_data_property() (owlapy.owl_property.OWLDataProperty method), 104
is_owl_top_data_property() (owlapy.owl_property.OWLPropertyExpression method), 102
```

```
is_owl_top_object_property() (owlapy.owl_property.OWLObjectProperty method), 103
is_owl_top_object_property() (owlapy.owl_property.OWLPropertyExpression method), 102
is_parent_of() (owlapy.owl_hierarchy.AbstractHierarchy method), 83
is_reserved_vocabulary() (owlapy.iri.IRI method), 56
is_string() (owlapy.owl_literal.OWLLiteral method), 89
is_sub_property_of() (owlapy.owl_hierarchy.DatatypePropertyHierarchy method), 86
is_sub_property_of() (owlapy.owl_hierarchy.ObjectPropertyHierarchy method), 85
is subclass of () (owlapy.owl hierarchy.ClassHierarchy method), 85
is_thing() (owlapy.iri.IRI method), 55
is_using_triplestore() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 119
items (owlapy.utils.EvaluatedDescriptionSet attribute), 140
items()(owlapy.owl_hierarchy.AbstractHierarchy method), 84
iter_count() (in module owlapy.utils), 142
K
KEY (owlapy.utils.LRUCache attribute), 142
leaves () (owlapy.owl_hierarchy.AbstractHierarchy method), 84
LENGTH (owlapy.class_expression.OWLFacet attribute), 47
LENGTH (owlapy.vocab.OWLFacet attribute), 144
length() (owlapy.utils.OWLClassExpressionLengthMetric method), 139
Literals (in module owlapy.class expression.restriction), 25
Literals (in module owlapy.owl_literal), 88
load_ontology() (owlapy.owl_ontology_manager.OntologyManager method), 100
load_ontology() (owlapy.owl_ontology_manager.OWLOntologyManager method), 98
logger (in module owlapy.owl_ontology), 92
logger (in module owlapy.owl reasoner), 105
LONG (owlapy.vocab.XSDVocabulary attribute), 143
LRUCache (class in owlapy.utils), 142
Μ
manager (owlapy.owlapi_adaptor.OWLAPIAdaptor attribute), 126
MANCHESTER_GRAMMAR (in module owlapy.parser), 129
manchester_to_owl_expression() (in module owlapy), 145
manchester_to_owl_expression() (in module owlapy.parser), 133
ManchesterOWLSyntaxOWLObjectRenderer (class in owlapy.render), 136
ManchesterOWLSyntaxParser (class in owlapy.parser), 129
ManchesterParser (in module owlapy.parser), 133
ManchesterRenderer (in module owlapy.render), 136
map_concept() (owlapy.owl_ontology.FromOwlready2 method). 97
map_concept() (owlapy.owl_ontology.ToOwlready2 method), 97
map_datarange() (owlapy.owl_ontology.FromOwlready2 method), 97
map_datarange() (owlapy.owl_ontology.ToOwlready2 method), 97
map_object() (owlapy.owl_ontology.ToOwlready2 method), 97
mapping (owlapy.converter.Owl2SparglConverter attribute), 53
MAX_EXCLUSIVE (owlapy.class_expression.OWLFacet attribute), 47
MAX_EXCLUSIVE (owlapy.vocab.OWLFacet attribute), 144
MAX_INCLUSIVE (owlapy.class_expression.OWLFacet attribute), 47
MAX_INCLUSIVE (owlapy.vocab.OWLFacet attribute), 144
MAX_LENGTH (owlapy.class_expression.OWLFacet attribute), 47
MAX_LENGTH (owlapy.vocab.OWLFacet attribute), 144
maybe_add() (owlapy.utils.EvaluatedDescriptionSet method), 140
measurer (in module owlapy.utils), 139
MIN_EXCLUSIVE (owlapy.class_expression.OWLFacet attribute), 47
MIN_EXCLUSIVE (owlapy.vocab.OWLFacet attribute), 144
MIN_INCLUSIVE (owlapy.class_expression.OWLFacet attribute), 47
MIN_INCLUSIVE (owlapy.vocab.OWLFacet attribute), 144
MIN_LENGTH (owlapy.class_expression.OWLFacet attribute), 47
MIN_LENGTH (owlapy.vocab.OWLFacet attribute), 144
modal_depth (owlapy.converter.Owl2SparqlConverter property), 54
module
     owlapy, 18
     owlapy.class_expression, 18
     owlapy.class_expression.class_expression, 18
     \verb"owlapy.class_expression.nary_boolean_expression, 20"
```

```
owlapy.class expression.owl class, 21
     owlapy.class_expression.restriction, 23
     owlapy.converter,52
     owlapy.entities, 52
     owlapy.iri,55
     owlapy.meta_classes, 56
     owlapy.namespaces, 58
     owlapy.owl annotation, 59
     owlapy.owl_axiom, 60
     owlapy.owl_data_ranges,80
     owlapy.owl_datatype,82
     owlapy.owl_hierarchy,82
     owlapy.owl_individual,86
     owlapy.owl_literal,87
     \verb"owlapy.owl_object", 90"
     owlapy.owl_ontology,92
     owlapy.owl_ontology_manager,98
     owlapy.owl_property, 101
     owlapy.owl_reasoner, 105
     owlapy.owlapi_adaptor, 126
     owlapy.parser, 128
     owlapy.providers, 134
     owlapy.render, 135
     owlapy.static_funcs, 136
     owlapy.utils, 137
     owlapy.vocab, 143
more_general_roles() (owlapy.owl_hierarchy.DatatypePropertyHierarchy method), 86
more_general_roles() (owlapy.owl_hierarchy.ObjectPropertyHierarchy method), 85
more_special_roles() (owlapy.owl_hierarchy.DatatypePropertyHierarchy method), 86
more_special_roles() (owlapy.owl_hierarchy.ObjectPropertyHierarchy method), 85
most_general_roles() (owlapy.owl_hierarchy.DatatypePropertyHierarchy method), 86
most general roles() (owlapy.owl hierarchy.ObjectPropertyHierarchy method), 85
most_special_roles() (owlapy.owl_hierarchy.DatatypePropertyHierarchy method), 86
most_special_roles() (owlapy.owl_hierarchy.ObjectPropertyHierarchy method), 85
move () (in module owlapy.static_funcs), 137
Ν
name_reasoner (owlapy.owlapi_adaptor.OWLAPIAdaptor attribute), 126
named_classes() (owlapy.owl_axiom.OWLEquivalentClassesAxiom method), 66
named_individuals (owlapy.converter.Owl2SparqlConverter attribute), 54
Namespaces (class in owlapy.namespaces), 58
new_count_var() (owlapy.converter.Owl2SparqlConverter method), 54
new_individual_variable() (owlapy.converter.VariablesMapping method), 53
new_property_variable() (owlapy.converter.VariablesMapping method), 53
NEXT (owlapy.utils.LRUCache attribute), 142
NNF (class in owlapy.utils), 141
ns (owlapy.namespaces.Namespaces property), 58
ns (owlapy.parser.DLSyntaxParser attribute), 131
ns (owlapy.parser.ManchesterOWLSyntaxParser attribute), 129
NUMERIC_DATATYPES (in module owlapy.owl_literal), 90
O
o (owlapy.utils.OrderedOWLObject attribute), 141
object_all_values_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
object_cardinality_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
object_complement_length (owlapy.utils.OWLClassExpressionLengthMetric attribute). 139
object_has_self_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
object_has_value_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
object_intersection_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
object_inverse_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
object_one_of_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
object_properties_in_signature() (owlapy.owl_ontology.Ontology method), 95
object_properties_in_signature() (owlapy.owl_ontology.OWLOntology method), 93
object_property_domain_axioms() (owlapy.owl_ontology.Ontology method), 96
object_property_domain_axioms() (owlapy.owl_ontology.OWLOntology method), 94
object_property_domains() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 120
object_property_domains() (owlapy.owl_reasoner.OntologyReasoner method), 113
```

```
object property domains () (owlapy, owl reasoner. OWL Reasoner method), 105
object_property_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
object_property_range_axioms() (owlapy.owl_ontology.Ontology method), 96
object_property_range_axioms() (owlapy.owl_ontology.OWLOntology method), 94
object_property_ranges() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 120
object_property_ranges() (owlapy.owl_reasoner.OntologyReasoner method), 113
object_property_ranges() (owlapy.owl_reasoner.OWLReasoner method), 106
object property values () (owlapy.owl reasoner.FastInstanceCheckerReasoner method), 122
object_property_values() (owlapy.owl_reasoner.OntologyReasoner method), 115
object_property_values() (owlapy.owl_reasoner.OWLReasoner method), 108
object_some_values_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
object_union_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 139
ObjectPropertyHierarchy (class in owlapy.owl_hierarchy), 85
Ontology (class in owlapy.owl_ontology), 95
ontology (owlapy.owlapi_adaptor.OWLAPIAdaptor attribute), 127
OntologyManager (class in owlapy.owl_ontology_manager), 100
OntologyReasoner (class in owlapy.owl_reasoner), 112
operands () (owlapy.class_expression.class_expression.OWLObjectComplementOf method), 20
operands () (owlapy.class_expression.nary_boolean_expression.OWLNaryBooleanClassExpression method), 20
operands () (owlapy.class_expression.OWLDataOneOf method), 44
operands () (owlapy.class_expression.OWLNaryBooleanClassExpression method), 40
operands () (owlapy.class expression.OWLObjectComplementOf method), 38
operands () (owlapy.class_expression.OWLObjectOneOf method), 51
operands () (owlapy.class_expression.restriction.OWLDataOneOf method), 34
operands () (owlapy.class_expression.restriction.OWLObjectOneOf method), 30
operands () (owlapy.meta_classes.HasOperands method), 57
operands () (owlapy.owl_axiom.OWLHasKeyAxiom method), 64
operands () (owlapy.owl_data_ranges.OWLNaryDataRange method), 80
OperandSetTransform (class in owlapy.utils), 140
operator (owlapy.class_expression.OWLFacet property), 47
operator (owlapy.vocab.OWLFacet property), 144
OrderedOWLObject (class in owlapy.utils), 140
OWL (in module owlapy.namespaces), 59
Owl2SparqlConverter (class in owlapy.converter), 53
OWL_BOTTOM_DATA_PROPERTY (owlapy.class_expression.OWLRDFVocabulary attribute), 52
OWL_BOTTOM_DATA_PROPERTY (owlapy.vocab.OWLRDFVocabulary attribute), 143
OWL BOTTOM OBJECT PROPERTY (owlapy, class expression. OWLRDFVocabulary attribute), 52
OWL_BOTTOM_OBJECT_PROPERTY (owlapy.vocab.OWLRDFVocabulary attribute), 143
OWL_CLASS (owlapy.class_expression.OWLRDFVocabulary attribute), 51
OWL_CLASS (owlapy.vocab.OWLRDFVocabulary attribute), 143
owl_datatype_max_exclusive_restriction() (in module owlapy.providers), 134
owl_datatype_max_inclusive_restriction() (in module owlapy.providers), 134
\verb"owl_datatype_min_exclusive_restriction()" (\textit{in module owlapy.providers}), 134
owl_datatype_min_inclusive_restriction() (in module owlapy.providers), 134
owl_datatype_min_max_exclusive_restriction() (in module owlapy.providers), 134
owl_datatype_min_max_inclusive_restriction() (in module owlapy.providers), 134
owl_expression_to_dl() (in module owlapy), 145
owl_expression_to_dl() (in module owlapy.render), 136
owl_expression_to_manchester() (in module owlapy), 145
owl_expression_to_manchester() (in module owlapy.render), 136
owl_expression_to_sparql() (in module owlapy), 145
owl_expression_to_sparql() (in module owlapy.converter), 54
\verb"OWL_NAMED_INDIVIDUAL" (owlapy.class\_expression.OWLRDFV ocabulary attribute), 52
OWL_NAMED_INDIVIDUAL (owlapy.vocab.OWLRDFVocabulary attribute), 143
{\tt OWL\_NOTHING}\ (owlary.class\_expression.OWLRDFV ocabulary\ attribute),\,51
OWL NOTHING (owlapy.vocab.OWLRDFVocabulary attribute), 143
OWL_THING (owlapy.class_expression.OWLRDFVocabulary attribute), 51
OWL THING (owlapy.vocab.OWLRDFVocabulary attribute), 143
OWL_TOP_DATA_PROPERTY (owlapy.class_expression.OWLRDFVocabulary attribute), 52
OWL_TOP_DATA_PROPERTY (owlapy.vocab.OWLRDFVocabulary attribute), 143
OWL TOP OBJECT PROPERTY (owlapy.class expression.OWLRDFVocabulary attribute), 52
OWL_TOP_OBJECT_PROPERTY (owlapy.vocab.OWLRDFVocabulary attribute), 143
OWLAnnotation (class in owlapy.owl_axiom), 70
OWLAnnotationAssertionAxiom (class in owlapy.owl_axiom), 71
OWLAnnotationAxiom (class in owlapy.owl_axiom), 71
OWLAnnotationObject (class in owlapy.owl_annotation), 59
OWLAnnotationProperty (class in owlapy.owl_axiom), 70
OWLAnnotationPropertyDomainAxiom (class in owlapy.owl_axiom), 72
```

```
OWLAnnotationPropertyRangeAxiom (class in owlapy.owl_axiom), 73
OWLAnnotationSubject (class in owlapy.owl_annotation), 59
OWLAnnotationValue (class in owlapy.owl_annotation), 60
OWLAnonymousClassExpression (class in owlapy.class_expression), 37
OWLAnonymousClassExpression (class in owlapy.class_expression.class_expression), 19
OWLAPIAdaptor (class in owlapy.owlapi_adaptor), 126
owlapy
    module, 18
owlapy.class_expression
    module, 18
owlapy.class_expression.class_expression
    module, 18
owlapy.class_expression.nary_boolean_expression
    module, 20
owlapy.class_expression.owl_class
    module, 21
owlapy.class_expression.restriction
    module, 23
owlapy.converter
    module, 52
owlapy.entities
    module, 52
owlapy.iri
    module, 55
owlapy.meta_classes
    module, 56
owlapy.namespaces
    module, 58
owlapy.owl_annotation
    module, 59
owlapy.owl_axiom
    module, 60
owlapy.owl_data_ranges
    module, 80
owlapy.owl_datatype
    module, 82
owlapy.owl_hierarchy
    module, 82
owlapy.owl_individual
    module, 86
owlapy.owl_literal
   module, 87
owlapy.owl_object
    module, 90
owlapy.owl_ontology
    module, 92
owlapy.owl_ontology_manager
    module, 98
owlapy.owl_property
    module, 101
owlapy.owl_reasoner
    module, 105
owlapy.owlapi_adaptor
    module, 126
owlapy.parser
    module, 128
owlapy.providers
    module, 134
owlapy.render
    module, 135
owlapy.static_funcs
    module, 136
owlapy.utils
    module, 137
owlapy.vocab
   module, 143
OWLAsymmetricObjectPropertyAxiom (class in owlapy.owl_axiom), 76
OWLAxiom (class in owlapy.owl_axiom), 62
```

```
OWLBooleanClassExpression (class in owlapy.class expression), 38
OWLBooleanClassExpression (class in owlapy.class_expression.class_expression), 19
OWLBottomDataProperty (in module owlapy.owl literal), 90
OWLBottomObjectProperty (in module owlapy.owl_literal), 89
OWLCardinalityRestriction (class in owlapy.class_expression), 42
OWLCardinalityRestriction (class in owlapy.class_expression.restriction), 26
OWLClass (class in owlapy.class_expression), 39
OWLClass (class in owlapy.class expression.owl class), 22
OWLClassAssertionAxiom (class in owlapy.owl_axiom), 70
OWLClassAxiom (class in owlapy.owl_axiom), 63
OWLClassExpression (class in owlapy.class_expression), 37
OWLClassExpression (class in owlapy.class_expression.class_expression), 18
OWLClassExpressionLengthMetric (class in owlapy.utils), 138
OWLDataAllValuesFrom (class in owlapy.class_expression), 49
OWLDataAllValuesFrom (class in owlapy.class_expression.restriction), 33
OWLDataCardinalityRestriction (class in owlapy.class_expression), 44
OWLDataCardinalityRestriction (class in owlapy.class_expression.restriction), 31
OWLDataComplementOf (class in owlapy.owl_data_ranges), 81
OWLDataExactCardinality (class in owlapy.class_expression), 50
OWLDataExactCardinality (class in owlapy.class_expression.restriction), 32
OWLDataHasValue (class in owlapy.class_expression), 49
OWLDataHasValue (class in owlapy.class expression.restriction), 33
OWLDataIntersectionOf (class in owlapy.owl_data_ranges), 81
OWLDataMaxCardinality (class in owlapy.class_expression), 50
OWLDataMaxCardinality (class in owlapy.class_expression.restriction), 31
OWLDataMinCardinality (class in owlapy.class_expression), 50
OWLDataMinCardinality (class in owlapy.class_expression.restriction), 31
OWLDataOneOf (class in owlapy.class_expression), 43
OWLDataOneOf (class in owlapy.class_expression.restriction), 34
OWLDataProperty (class in owlapy.owl_property), 104
OWLDataPropertyAssertionAxiom (class in owlapy.owl_axiom), 75
OWLDataPropertyAxiom (class in owlapy.owl axiom), 63
OWLDataPropertyCharacteristicAxiom (class in owlapy.owl_axiom), 77
OWLDataPropertyDomainAxiom (class in owlapy.owl_axiom), 79
OWLDataPropertyExpression (class in owlapy.owl_property), 102
OWLDataPropertyRangeAxiom (class in owlapy.owl_axiom), 79
OWLDataRange (class in owlapy.owl data ranges), 80
OWLDataRestriction (class in owlapy.class_expression), 42
OWLDataRestriction (class in owlapy.class_expression.restriction), 30
OWLDataSomeValuesFrom (class in owlapy.class_expression), 48
OWLDataSomeValuesFrom (class in owlapy.class_expression.restriction), 32
OWLDatatype (class in owlapy.owl_datatype), 82
OWLDatatypeDefinitionAxiom (class in owlapy.owl_axiom), 64
OWLDatatypeRestriction (class in owlapy.class_expression), 46
OWLDatatypeRestriction (class in owlapy.class_expression.restriction), 34
OWLDataUnionOf (class in owlapy.owl_data_ranges), 81
OWLDeclarationAxiom (class in owlapy.owl_axiom), 63
OWLDifferentIndividualsAxiom (class in owlapy.owl_axiom), 66
OWLDisjointClassesAxiom (class in owlapy.owl_axiom), 66
OWLDisjointDataPropertiesAxiom (class in owlapy.owl_axiom), 68
OWLDisjointObjectPropertiesAxiom (class in owlapy.owl_axiom), 68
OWLDisjointUnionAxiom (class in owlapy.owl_axiom), 69
OWLEntity (class in owlapy.owl_object), 91
OWLEquivalentClassesAxiom (class in owlapy.owl_axiom), 65
OWLEquivalentDataPropertiesAxiom (class in owlapy.owl_axiom), 68
OWLEquivalentObjectPropertiesAxiom (class in owlapy.owl_axiom), 67
OWLFacet (class in owlapy.class_expression), 46
OWLFacet (class in owlapy.vocab), 144
OWLFacetRestriction (class in owlapy.class_expression), 47
OWLFacetRestriction (class in owlapy.class_expression.restriction), 35
OWLFunctionalDataPropertyAxiom (class in owlapy.owl axiom), 78
OWLFunctionalObjectPropertyAxiom (class in owlapy.owl_axiom), 76
OWLHasKeyAxiom (class in owlapy.owl_axiom), 64
OWLHasValueRestriction (class in owlapy.class_expression), 41
OWLHasValueRestriction (class in owlapy.class_expression.restriction), 25
OWLImportsDeclaration (class in owlapy.owl_ontology_manager), 99
OWLIndividual (class in owlapy.owl_individual), 86
OWLIndividualAxiom (class in owlapy.owl_axiom), 63
```

```
OWLInverseFunctionalObjectPropertyAxiom (class in owlapy.owl axiom), 76
OWLInverseObjectPropertiesAxiom (class in owlapy.owl_axiom), 68
OWLIrreflexiveObjectPropertyAxiom (class in owlapy.owl_axiom), 77
OWLLiteral (class in owlapy.owl_literal), 88
OWLLogicalAxiom (class in owlapy.owl_axiom), 62
OWLNamedIndividual (class in owlapy.owl_individual), 86
OWLNamedObject (class in owlapy.owl_object), 91
OWLNaryAxiom (class in owlapy.owl axiom), 65
OWLNaryBooleanClassExpression (class in owlapy.class_expression), 39
OWLNaryBooleanClassExpression (class in owlapy.class_expression.nary_boolean_expression), 20
OWLNaryClassAxiom (class in owlapy.owl_axiom), 65
OWLNaryDataRange (class in owlapy.owl_data_ranges), 80
OWLNaryIndividualAxiom (class in owlapy.owl_axiom), 66
OWLNaryPropertyAxiom (class in owlapy.owl_axiom), 67
OWLNegativeDataPropertyAssertionAxiom (class in owlapy.owl_axiom), 75
OWLNegativeObjectPropertyAssertionAxiom (class in owlapy.owl_axiom), 75
OWLNothing (in module owlapy.class_expression), 52
OWLObject (class in owlapy.owl_object), 90
OWLObjectAllValuesFrom (class in owlapy.class_expression), 45
OWLObjectAllValuesFrom (class in owlapy.class_expression.restriction), 28
OWLObjectCardinalityRestriction (class in owlapy.class_expression), 42
OWLObjectCardinalityRestriction (class in owlapy.class expression.restriction), 27
OWLObjectComplementOf (class in owlapy.class_expression), 38
OWLObjectComplementOf (class in owlapy.class_expression.class_expression), 19
OWLObjectExactCardinality (class in owlapy.class_expression), 48
OWLObjectExactCardinality (class in owlapy.class_expression.restriction), 27
OWLObjectHasSelf (class in owlapy.class_expression), 43
OWLObjectHasSelf (class in owlapy.class_expression.restriction), 29
OWLObjectHasValue (class in owlapy.class_expression), 45
OWLObjectHasValue (class in owlapy.class_expression.restriction), 29
OWLObjectIntersectionOf (class in owlapy.class_expression), 40
OWLObjectIntersectionOf (class in owlapy.class_expression.nary_boolean_expression), 21
OWLObjectInverseOf (class in owlapy.owl_property), 103
OWLObjectMaxCardinality (class in owlapy.class_expression), 48
OWLObjectMaxCardinality (class in owlapy.class_expression.restriction), 27
OWLObjectMinCardinality (class in owlapy.class_expression), 47
OWLObjectMinCardinality (class in owlapy.class expression.restriction), 27
OWLObjectOneOf (class in owlapy.class_expression), 51
OWLObjectOneOf (class in owlapy.class_expression.restriction), 30
OWLObjectParser (class in owlapy.owl_object), 91
OWLObjectProperty (class in owlapy.owl_property), 103
OWLObjectPropertyAssertionAxiom (class in owlapy.owl_axiom), 74
OWLObjectPropertyAxiom (class in owlapy.owl_axiom), 63
OWLObjectPropertyCharacteristicAxiom (class in owlapy.owl_axiom), 76
OWLObjectPropertyDomainAxiom (class in owlapy.owl_axiom), 79
OWLObjectPropertyExpression (class in owlapy.owl_property), 102
OWLObjectPropertyRangeAxiom (class in owlapy.owl_axiom), 79
OWLObjectRenderer (class in owlapy.owl_object), 91
OWLObjectRestriction (class in owlapy.class_expression), 41
OWLObjectRestriction (class in owlapy.class_expression.restriction), 25
OWLObjectSomeValuesFrom (class in owlapy.class expression), 45
OWLObjectSomeValuesFrom (class in owlapy.class_expression.restriction), 28
OWLObjectUnionOf (class in owlapy.class_expression), 40
OWLObjectUnionOf (class in owlapy.class_expression.nary_boolean_expression), 21
OWLOntology (class in owlapy.owl_ontology), 93
OWLOntologyChange (class in owlapy.owl_ontology_manager), 98
OWLOntologyID (class in owlapy.owl_ontology), 92
OWLOntologyManager (class in owlapy.owl_ontology_manager), 98
OWLProperty (class in owlapy.owl_property), 103
OWLPropertyAssertionAxiom (class in owlapy.owl_axiom), 74
OWLPropertyAxiom (class in owlapy.owl axiom), 63
OWLPropertyDomainAxiom (class in owlapy.owl_axiom), 78
OWLPropertyExpression (class in owlapy.owl_property), 101
OWLPropertyRange (class in owlapy.owl_data_ranges), 80
OWLPropertyRangeAxiom (class in owlapy.owl_axiom), 78
OWLQuantifiedDataRestriction (class in owlapy.class_expression), 44
OWLQuantifiedDataRestriction (class in owlapy.class_expression.restriction), 31
OWLQuantifiedObjectRestriction (class in owlapy.class_expression), 41
```

```
OWLOuantifiedObjectRestriction (class in owlapy.class expression.restriction), 26
OWLQuantifiedRestriction (class in owlapy.class_expression), 41
OWLQuantifiedRestriction (class in owlapy.class_expression.restriction), 26
OWLRDFVocabulary (class in owlapy.class_expression), 51
OWLRDFVocabulary (class in owlapy.vocab), 143
OWLREADY2_FACET_KEYS (in module owlapy.owl_ontology), 97
OWLReasoner (class in owlapy.owl_reasoner), 105
OWLReasonerEx (class in owlapy.owl reasoner), 111
OWLReflexiveObjectPropertyAxiom (class in owlapy.owl_axiom), 77
OWLRestriction (class in owlapy.class_expression), 40
OWLRestriction (class in owlapy.class_expression.restriction), 25
OWLSameIndividualAxiom (class in owlapy.owl_axiom), 67
OWLSubAnnotationPropertyOfAxiom (class in owlapy.owl_axiom), 72
OWLSubClassOfAxiom (class in owlapy.owl_axiom), 69
OWLSubDataPropertyOfAxiom (class in owlapy.owl_axiom), 74
OWLSubObjectPropertyOfAxiom (class in owlapy.owl_axiom), 73
OWLSubPropertyAxiom (class in owlapy.owl_axiom), 73
OWLSymmetricObjectPropertyAxiom (class in owlapy.owl_axiom), 77
OWLThing (in module owlapy.class_expression), 52
OWLTopDataProperty (in module owlapy.owl_literal), 90
OWLTopObjectProperty (in module owlapy.owl_literal), 89
OWLTransitiveObjectPropertyAxiom (class in owlapy.owl_axiom), 77
OWLUnaryPropertyAxiom (class in owlapy.owl_axiom), 75
Р
parent (owlapy.converter.Owl2SparalConverter attribute), 53
parent_var (owlapy.converter.Owl2SparqlConverter attribute), 53
parents() (owlapy.owl_hierarchy.AbstractHierarchy method), 83
parse_boolean() (owlapy.owl_literal.OWLLiteral method), 88
parse_date() (owlapy.owl_literal.OWLLiteral method), 89
parse_datetime() (owlapy.owl_literal.OWLLiteral method), 89
parse_double() (owlapy.owl_literal.OWLLiteral method), 88
parse_duration() (owlapy.owl_literal.OWLLiteral method), 89
parse_expression() (owlapy.owl_object.OWLObjectParser method), 91
parse_expression() (owlapy.parser.DLSyntaxParser method), 131
parse_expression() (owlapy.parser.ManchesterOWLSyntaxParser method), 129
parse_integer() (owlapy.owl_literal.OWLLiteral method), 88
parse_string() (owlapy.owl_literal.OWLLiteral method), 89
parser (owlapy.owlapi_adaptor.OWLAPIAdaptor attribute), 127
path (owlapy.owlapi_adaptor.OWLAPIAdaptor attribute), 126
PATTERN (owlapy.class_expression.OWLFacet attribute), 47
PATTERN (owlapy.vocab.OWLFacet attribute), 144
peek () (in module owlapy.converter), 53
PELLET (owlapy.owl_reasoner.BaseReasoner attribute), 111
prefix (owlapy.namespaces.Namespaces property), 58
PREV (owlapy.utils.LRUCache attribute), 142
process() (owlapy.converter.Owl2SparqlConverter method), 54
properties (owlapy.converter.Owl2SparqlConverter attribute), 53
properties() (owlapy.owl_axiom.OWLNaryPropertyAxiom method), 67
R
RDF (in module owlapy.namespaces), 59
RDFS (in module owlapy.namespaces), 59
RDFS_LITERAL (owlapy.class_expression.OWLRDFVocabulary attribute), 52
RDFS_LITERAL (owlapy.vocab.OWLRDFVocabulary attribute), 143
reasoner (owlapy.owlapi_adaptor.OWLAPIAdaptor attribute), 127
reminder (owlapy.class_expression.owl_class.OWLClass property), 22
reminder (owlapy.class_expression.OWLClass property), 39
reminder (owlapy.iri.IRI property), 56
remove_axiom() (owlapy.owl_ontology_manager.OntologyManager method), 100
remove_axiom() (owlapy.owl_ontology_manager.OWLOntologyManager method), 99
render() (owlapy.converter.Owl2SparglConverter method), 54
render() (owlapy.owl_object.OWLObjectRenderer method), 91
render() (owlapy.render.DLSyntaxObjectRenderer method), 135
render() (owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer method), 136
renderer (owlapy.owlapi adaptor.OWLAPIAdaptor attribute), 127
reset () (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 119
```

```
restrict() (owlapy.owl hierarchy.AbstractHierarchy static method), 83
restrict_and_copy() (owlapy.owl_hierarchy.AbstractHierarchy method), 83
Restriction_Literals (in module owlapy.providers), 134
RESULT (owlapy.utils.LRUCache attribute), 142
roots() (owlapy.owl_hierarchy.AbstractHierarchy method), 84
S
same_individuals() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 121
same individuals () (owlapy.owl reasoner.OntologyReasoner method), 114
same_individuals() (owlapy.owl_reasoner.OWLReasoner method), 107
save_ontology() (owlapy.owl_ontology_manager.OntologyManager method), 101
save_ontology() (owlapy.owl_ontology_manager.OWLOntologyManager method), 99
save_world() (owlapy.owl_ontology_manager.OntologyManager method), 101
sentinel (owlapy.utils.LRUCache attribute), 142
set_short_form_provider() (owlapy.owl_object.OWLObjectRenderer method), 91
set_short_form_provider() (owlapy.render.DLSyntaxObjectRenderer method), 135
set_short_form_provider() (owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer method), 136
siblings() (owlapy.owl_hierarchy.AbstractHierarchy method), 84
simplify() (owlapy.utils.OperandSetTransform method), 140
slots (owlapy.parser.DLSyntaxParser attribute), 131
slots (owlapy.parser.ManchesterOWLSyntaxParser attribute), 129
sort () (owlapy.utils.ConceptOperandSorter method), 140
sparql (owlapy.converter.Owl2SparqlConverter attribute), 53
stack_parent() (owlapy.converter.Owl2SparqlConverter method), 54
stack_variable() (owlapy.converter.Owl2SparqlConverter method), 54
stopJVM() (owlapy.owlapi_adaptor.OWLAPIAdaptor method), 127
str (owlapy.class_expression.owl_class.OWLClass property), 22
str (owlapy.class_expression.OWLClass property), 39
str (owlapy.iri.IRI property), 56
str (owlapy.meta_classes.HasIRI property), 57
str (owlapy.owl_axiom.OWLAnnotationProperty property), 70
str (owlapy.owl_datatype.OWLDatatype property), 82
str (owlapy.owl_individual.OWLNamedIndividual property), 87
str (owlapy.owl_ontology_manager.OWLImportsDeclaration property), 99
str (owlapy.owl_property.OWLProperty property), 103
STRING (owlapy.vocab.XSDVocabulary attribute), 143
StringOWLDatatype (in module owlapy.owl_literal), 90
sub_classes() (owlapy.owl_hierarchy.ClassHierarchy method), 85
sub_classes() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 122
sub_classes() (owlapy.owl_reasoner.OntologyReasoner method), 116
sub_classes() (owlapy.owl_reasoner.OWLReasoner method), 108
sub_data_properties() (owlapy.owl_hierarchy.DatatypePropertyHierarchy method), 86
sub_data_properties() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 124
sub_data_properties() (owlapy.owl_reasoner.OntologyReasoner method), 118
sub_data_properties() (owlapy.owl_reasoner.OWLReasoner method), 109
sub_object_properties() (owlapy.owl_hierarchy.ObjectPropertyHierarchy method), 85
sub_object_properties() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 125
sub_object_properties() (owlapy.owl_reasoner.OntologyReasoner method), 118
sub_object_properties() (owlapy.owl_reasoner.OWLReasoner method), 110
super_classes() (owlapy.owl_hierarchy.ClassHierarchy method), 85
super_classes() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 123
super_classes() (owlapy.owl_reasoner.OntologyReasoner method), 116
super_classes() (owlapy.owl_reasoner.OWLReasoner method), 111
super_data_properties() (owlapy.owl_hierarchy.DatatypePropertyHierarchy method), 86
super_data_properties() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method). 124
super_data_properties() (owlapy.owl_reasoner.OntologyReasoner method), 117
super_data_properties() (owlapy.owl_reasoner.OWLReasoner method), 110
super_object_properties() (owlapy.owl_hierarchy.ObjectPropertyHierarchy method), 85
super_object_properties() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 125
super_object_properties() (owlapy.owl_reasoner.OntologyReasoner method), 118
super_object_properties() (owlapy.owl_reasoner.OWLReasoner method), 110
symbolic_form (owlapy.class_expression.OWLFacet property), 46
symbolic_form (owlapy.vocab.OWLFacet property), 144
SyncReasoner (class in owlapy.owl_reasoner), 125
```

TIME\_DATATYPES (in module owlapy.owl\_literal), 90

```
to python () (owlapy.owl literal.OWLLiteral method), 89
to_string_id() (owlapy.owl_object.OWLEntity method), 92
ToOwlready2 (class in owlapy.owl_ontology), 97
TopLevelCNF (class in owlapy.utils), 141
TopLevelDNF (class in owlapy.utils), 141
TopOWLDatatype (in module owlapy.owl_literal), 90
TOTAL_DIGITS (owlapy.class_expression.OWLFacet attribute), 47
TOTAL DIGITS (owlapy.vocab.OWLFacet attribute), 144
triple() (owlapy.converter.Owl2SparqlConverter method), 54
type_index (owlapy.class_expression.class_expression.OWLObjectComplementOf attribute), 20
type_index (owlapy.class_expression.nary_boolean_expression.OWLObjectIntersectionOf attribute), 21
type_index (owlapy.class_expression.nary_boolean_expression.OWLObjectUnionOf attribute), 21
type_index (owlapy.class_expression.owl_class.OWLClass attribute), 22
type_index (owlapy.class_expression.OWLClass attribute), 39
type_index (owlapy.class_expression.OWLDataAllValuesFrom attribute), 49
type_index (owlapy.class_expression.OWLDataExactCardinality attribute), 50
type_index (owlapy.class_expression.OWLDataHasValue attribute), 49
type_index (owlapy.class_expression.OWLDataMaxCardinality attribute), 50
type_index (owlapy.class_expression.OWLDataMinCardinality attribute), 50
type_index (owlapy.class_expression.OWLDataOneOf attribute), 43
type_index (owlapy.class_expression.OWLDataSomeValuesFrom attribute), 48
type index (owlapy.class expression.OWLDatatypeRestriction attribute), 46
type_index (owlapy.class_expression.OWLFacetRestriction attribute), 47
type_index (owlapy.class_expression.OWLObjectAllValuesFrom attribute), 45
type_index (owlapy.class_expression.OWLObjectComplementOf attribute), 38
type_index (owlapy.class_expression.OWLObjectExactCardinality attribute), 48
type_index (owlapy.class_expression.OWLObjectHasSelf attribute), 43
type_index (owlapy.class_expression.OWLObjectHasValue attribute), 46
type_index (owlapy.class_expression.OWLObjectIntersectionOf attribute), 40
type_index (owlapy.class_expression.OWLObjectMaxCardinality attribute), 48
type_index (owlapy.class_expression.OWLObjectMinCardinality attribute), 47
type index (owlapy.class expression.OWLObjectOneOf attribute), 51
type_index (owlapy.class_expression.OWLObjectSomeValuesFrom attribute), 45
type_index (owlapy.class_expression.OWLObjectUnionOf attribute), 40
type_index (owlapy.class_expression.restriction.OWLDataAllValuesFrom attribute), 33
type_index (owlapy.class_expression.restriction.OWLDataExactCardinality attribute), 32
type index (owlapy.class expression.restriction.OWLDataHasValue attribute), 33
type_index (owlapy.class_expression.restriction.OWLDataMaxCardinality attribute), 32
type_index (owlapy.class_expression.restriction.OWLDataMinCardinality attribute), 31
type_index (owlapy.class_expression.restriction.OWLDataOneOf attribute), 34
type_index (owlapy.class_expression.restriction.OWLDataSomeValuesFrom attribute), 32
type_index (owlapy.class_expression.restriction.OWLDatatypeRestriction attribute), 34
type_index (owlapy.class_expression.restriction.OWLFacetRestriction attribute), 35
type_index (owlapy.class_expression.restriction.OWLObjectAllValuesFrom attribute), 28
type_index (owlapy.class_expression.restriction.OWLObjectExactCardinality attribute), 28
type_index (owlapy.class_expression.restriction.OWLObjectHasSelf attribute), 29
type_index (owlapy.class_expression.restriction.OWLObjectHasValue attribute), 29
type_index (owlapy.class_expression.restriction.OWLObjectMaxCardinality attribute), 27
type_index (owlapy.class_expression.restriction.OWLObjectMinCardinality attribute), 27
type_index (owlapy.class_expression.restriction.OWLObjectOneOf attribute), 30
type index (owlapy.class expression.restriction.OWLObjectSomeValuesFrom attribute), 28
type_index (owlapy.iri.IRI attribute), 55
{\tt type\_index}~(owlapy.owl\_data\_ranges.OWLDataComplementOf~attribute),~81
type_index (owlapy.owl_data_ranges.OWLDataIntersectionOf attribute), 81
type_index (owlapy.owl_data_ranges.OWLDataUnionOf attribute), 81
type_index (owlapy.owl_datatype.OWLDatatype attribute), 82
type_index (owlapy.owl_individual.OWLNamedIndividual attribute), 86
type_index (owlapy.owl_literal.OWLLiteral attribute), 88
type_index (owlapy.owl_ontology.OWLOntology attribute), 93
type_index (owlapy.owl_property.OWLDataProperty attribute), 104
type index (owlapy.owl property.OWLObjectInverseOf attribute), 104
type_index (owlapy.owl_property.OWLObjectProperty attribute), 103
type_index (owlapy.utils.HasIndex attribute), 140
types () (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 123
types () (owlapy.owl_reasoner.OntologyReasoner method), 118
types() (owlapy.owl_reasoner.OWLReasoner method), 110
```

## U

update\_isolated\_ontology() (owlapy.owl\_reasoner.OntologyReasoner method), 112 update\_isolated\_ontology() (owlapy.owl\_reasoner.SyncReasoner method), 125

### ٧

```
values () (owlapy.class_expression.OWLDataOneOf method), 43
values () (owlapy.class_expression.restriction.OWLDataOneOf method), 34
variable entities (owlapy.converter.Owl2SparglConverter attribute), 53
variables (owlapy.converter.Owl2SparqlConverter attribute), 53
VariablesMapping (class in owlapy.converter), 53
visit_abbreviated_iri() (owlapy.parser.DLSyntaxParser method), 133
visit_abbreviated_iri() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_boolean_literal() (owlapy.parser.DLSyntaxParser method), 133
visit_boolean_literal() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_cardinality_res() (owlapy.parser.DLSyntaxParser method), 132
visit_cardinality_res() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_class_expression() (owlapy.parser.DLSyntaxParser method), 132
visit_class_expression() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_class_iri() (owlapy.parser.DLSyntaxParser method), 133
visit_class_iri() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_data_cardinality_res() (owlapy.parser.DLSyntaxParser method), 132
visit_data_cardinality_res() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_data_intersection() (owlapy.parser.DLSyntaxParser method), 132
visit_data_intersection() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_data_parentheses() (owlapy.parser.DLSyntaxParser method), 132
visit_data_parentheses() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_data_primary() (owlapy.parser.DLSyntaxParser method), 132
visit_data_primary() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_data_property_iri() (owlapy.parser.DLSyntaxParser method), 133
visit_data_property_iri() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_data_some_only_res() (owlapy.parser.DLSyntaxParser method), 132
visit_data_some_only_res() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_data_union() (owlapy.parser.DLSyntaxParser method), 132
visit_data_union() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_data_value_res() (owlapy.parser.DLSyntaxParser method), 132
visit_data_value_res() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_datatype() (owlapy.parser.DLSyntaxParser method), 133
visit_datatype() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_datatype_iri() (owlapy.parser.DLSyntaxParser method), 133
visit_datatype_iri() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_datatype_restriction() (owlapy.parser.DLSyntaxParser method), 132
visit_datatype_restriction() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_date_literal() (owlapy.parser.DLSyntaxParser method), 133
visit_date_literal() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_datetime_literal() (owlapy.parser.DLSyntaxParser method), 133
visit_datetime_literal() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_decimal_literal() (owlapy.parser.DLSyntaxParser method), 133
visit_decimal_literal() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_duration_literal() (owlapy.parser.DLSyntaxParser method), 133
visit_duration_literal() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_facet() (owlapy.parser.DLSyntaxParser method), 133
visit_facet() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_facet_restriction() (owlapy.parser.DLSyntaxParser method), 132
visit_facet_restriction() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_facet_restrictions() (owlapy.parser.DLSyntaxParser method), 132
visit_facet_restrictions() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_float_literal() (owlapy.parser.DLSyntaxParser method), 133
visit_float_literal() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_full_iri() (owlapy.parser.DLSyntaxParser method), 133
\verb|visit_full_iri(|)| (owlapy.parser.ManchesterOWLSyntaxParser method), 131|
visit_has_self() (owlapy.parser.DLSyntaxParser method), 132
visit_has_self() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_individual_iri() (owlapy.parser.DLSyntaxParser method), 133
visit_individual_iri() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_individual_list() (owlapy.parser.DLSyntaxParser method), 132
visit_individual_list() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_integer_literal() (owlapy.parser.DLSyntaxParser method), 133
```

```
visit integer literal() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_intersection() (owlapy.parser.DLSyntaxParser method), 132
visit_intersection() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_iri() (owlapy.parser.DLSyntaxParser method), 133
visit_iri() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_literal() (owlapy.parser.DLSyntaxParser method), 132
visit_literal() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit literal list() (owlapy.parser.DLSyntaxParser method), 132
visit_literal_list() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_non_negative_integer() (owlapy.parser.DLSyntaxParser method), 133
visit_non_negative_integer() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_object_property() (owlapy.parser.DLSyntaxParser method), 132
visit_object_property() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_object_property_iri() (owlapy.parser.DLSyntaxParser method), 133
visit_object_property_iri() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_parentheses() (owlapy.parser.DLSyntaxParser method), 133
visit_parentheses() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_primary() (owlapy.parser.DLSyntaxParser method), 132
visit_primary() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_quoted_string() (owlapy.parser.DLSyntaxParser method), 133
visit_quoted_string() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_simple_iri() (owlapy.parser.DLSyntaxParser method), 133
visit_simple_iri() (owlapy.parser.ManchesterOWLSyntaxParser method), 131
visit_some_only_res() (owlapy.parser.DLSyntaxParser method), 132
visit_some_only_res() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_string_literal_language() (owlapy.parser.DLSyntaxParser method), 132
visit_string_literal_language() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_string_literal_no_language() (owlapy.parser.DLSyntaxParser method), 132
visit_string_literal_no_language() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_typed_literal() (owlapy.parser.DLSyntaxParser method), 132
visit_typed_literal() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
visit_union() (owlapy.parser.DLSyntaxParser method), 132
visit_union() (owlapy.parser.ManchesterOWLSyntaxParser method), 129
visit_value_res() (owlapy.parser.DLSyntaxParser method), 132
visit_value_res() (owlapy.parser.ManchesterOWLSyntaxParser method), 130
```

# W

worst () (owlapy.utils.EvaluatedDescriptionSet method), 140



XSD (in module owlapy.namespaces), 59 XSDVocabulary (class in owlapy.vocab), 143