OWLAPY
Release 0.1.2

Ontolearn Team

May 24, 2024

Contents:

OWLAPY[1]: Representation of OWL objects in python.

---
[1] https://github.com/dice-group/owlapy

1 About owlapy

Version: owlapy 1.0.2

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

Publisher and maintainer: DICE[2] - data science research group of Paderborn University[3].

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[4] and OntoSample[5].

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[6] including:

    - Entities, Literals, and Anonymous Individuals

    - 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.

---

[2] https://dice-research.org/
[3] https://www.uni-paderborn.de/en/university
[4] https://github.com/dice-group/Ontolearn
[5] https://github.com/alkidbaci/OntoSample
[6] https://www.w3.org/TR/owl2-syntax/

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␣
→temp_owlapy && pip3 install -e .
```

or using PyPI:

```
pip3 install owlapy
```

2 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
```
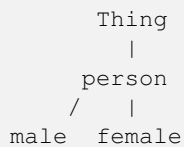
It contains only one object property which is `hasChild` and in total there are six persons (individuals), of which four are males and two are females.

2.1 Atomic Classes

To represent the classes `male`, `female`, and `person` we can simply use the class OWLClass[7]:

```python
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[8] 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:

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

---

[7] https://dice-group.github.io/owlapy/autoapi/owlapy/class_expression/owl_class/index.html#owlapy.class_expression.owl_class.OWLClass

[8] https://dice-group.github.io/owlapy/autoapi/owlapy/iri/index.html#owlapy.iri.IRI

2.2 Object Property

To represent the object property `hasChild` we can use the class OWLObjectProperty[9]:

```python
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[10]. In owlapy, as we said, we simply add the word *"OWL"* upfront to find the correct class:

```python
from owlapy.class_expression import OWLObjectMinCardinality

has_at_least_one_child = OWLObjectMinCardinality(
    cardinality = 1,
    property = hasChild,
    filler = person
)
```

As you can see, to create an object of class OWLObjectMinCardinality[11] 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[12]:

```python
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:

```python
from owlapy import owl_expression_to_sparql, owl_expression_to_dl, owl_expression_to_
↪manchester

print(owl_expression_to_dl(ce))
# Result: male ⊓ (≥ 1 hasChild.person)
```

(continues on next page)

---

[9] https://dice-group.github.io/owlapy/autoapi/owlapy/owl_property/index.html#owlapy.owl_property.OWLObjectProperty

[10] https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality

[11] https://dice-group.github.io/owlapy/autoapi/owlapy/class_expression/restriction/index.html#owlapy.class_expression.restriction.OWLObjectMinCardinality

[12] https://dice-group.github.io/owlapy/autoapi/owlapy/class_expression/nary_boolean_expression/index.html#owlapy.class_expression.nary_boolean_expression.OWLObjectIntersectionOf

```
print(owl_expression_to_sparql(ce))
# Result: SELECT DISTINCT ?x WHERE { ?x a <http://example.com/family#male> . { SELECT
→?x WHERE { ?x <http://example.com/family#hasChild> ?s_1 . ?s_1 a <http://example.
→com/family#person> .  } GROUP BY ?x HAVING ( COUNT ( ?s_1 ) >= 1 ) } }

print(owl_expression_to_manchester(ce))
# Result: male and (hasChild min 1 person)
```

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

```
from owlapy import dl_to_owl_expression, manchester_to_owl_expression

print(dl_to_owl_expression("∃ hasChild.male", namespace))
# Result: OWLObjectSomeValuesFrom(property=OWLObjectProperty(IRI('http://example.com/
→family#','hasChild')),filler=OWLClass(IRI('http://example.com/family#','male')))

print(manchester_to_owl_expression("female and (hasChild max 2 person)", namespace))
# Result: OWLObjectIntersectionOf((OWLClass(IRI('http://example.com/family#','female
→')), OWLObjectMaxCardinality(property=OWLObjectProperty(IRI('http://example.com/
→family#','hasChild')),2,filler=OWLClass(IRI('http://example.com/family#','person
→')))))
```

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[13] directory.

3 owlapy

3.1 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

---

[13] https://github.com/dice-group/owlapy/tree/develop/examples

Submodules

owlapy.class_expression.class_expression

OWL Base Classes Expressions

Classes

OWLPropertyRange	OWL Objects that can be the ranges of properties.
HasOperands	An interface to objects that have a collection of operands.
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

class owlapy.class_expression.class_expression.**OWLPropertyRange**

　　Bases: *owlapy.owl_object.OWLObject*

　　OWL Objects that can be the ranges of properties.

class owlapy.class_expression.class_expression.**HasOperands**

　　Bases: Generic[_T]

　　An interface to objects that have a collection of operands.

　　　　Parameters
　　　　　　_T – Operand type.

　　__slots__ = ()

　　abstract operands() → 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.class_expression.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)

　　__slots__ = ()

abstract is_owl_thing() → 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() → 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.

class owlapy.class_expression.class_expression.**OWLAnonymousClassExpression**

    Bases: *OWLClassExpression*

A Class Expression which is not a named Class.

is_owl_nothing() → 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() → 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() → *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.

```
__slots__ = ()
```

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

Bases: *OWLBooleanClassExpression*, *owlapy.meta_classes.*
HasOperands[*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() → 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

OWLClassExpression	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
OWLBooleanClassExpression	Represent an anonymous boolean class expression.
HasOperands	An interface to objects that have a collection of operands.
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.**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)
>
> **__slots__ = ()**
>
> **abstract is_owl_thing**() → 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**() → 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.

class owlapy.class_expression.nary_boolean_expression.**OWLBooleanClassExpression**

> Bases: OWLAnonymousClassExpression
>
> Represent an anonymous boolean class expression.
>
> **__slots__ = ()**

class owlapy.class_expression.nary_boolean_expression.**HasOperands**

> Bases: Generic[_T]
>
> An interface to objects that have a collection of operands.
>
> **Parameters**
> > **_T** – Operand type.
>
> **__slots__ = ()**
>
> **abstract operands**() → 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.class_expression.nary_boolean_expression.
  OWLNaryBooleanClassExpression(
   operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])

Bases: *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.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 ≤ i ≤ 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 ≤ i ≤ 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

OWLClassExpression	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
OWLObjectComplementOf	Represents an ObjectComplementOf class expression in the OWL 2 Specification.
OWLEntity	Represents Entities in the OWL 2 Specification.
IRI	An IRI, consisting of a namespace and a remainder.
OWLClass	An OWL 2 named Class. Classes can be understood as sets of individuals.

Module Contents

class owlapy.class_expression.owl_class.**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)

    __slots__ = ()

    abstract is_owl_thing() → 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() → 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.

class owlapy.class_expression.owl_class.**OWLObjectComplementOf**(
      op: OWLClassExpression)

    Bases: OWLBooleanClassExpression,   *owlapy.meta_classes.*
    HasOperands[*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() → 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.owl_class.**OWLEntity**

    Bases: OWLNamedObject

    Represents Entities in the OWL 2 Specification.

    __slots__ = ()

    to_string_id() → str

    is_anonymous() → bool

class owlapy.class_expression.owl_class.**IRI**(
      namespace: str | owlapy.namespaces.Namespaces, *remainder: str*)

    Bases: *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*) → *IRI*

    static create(*namespace: str*, *remainder: str*) → *IRI*

    static create(*string: str*) → *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 <http://www.w3.org/2002/07/owl#Nothing> 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 <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

is_reserved_vocabulary() → bool

> Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http://www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.
>
> > **Returns**
> >
> > > True if the IRI is in the reserved vocabulary, otherwise False.

as_iri() → *IRI*

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

as_str() → 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.

get_short_form() → str

> Gets the short form.
>
> > **Returns**
> >
> > > A string that represents the short form.

get_namespace() → str

> > **Returns**
> >
> > > The namespace as string.

get_remainder() → str

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

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() → 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() → 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() → *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

HasFiller	An interface to objects that have a filler.
HasCardinality	An interface to objects that have a cardinality.
HasOperands	An interface to objects that have a collection of operands.
OWLObjectIntersectionOf	An intersection class expression ObjectIntersectionOf(CE1 ... CEn) contains all individuals that are instances
OWLObjectUnionOf	A union class expression ObjectUnionOf(CE1 ... CEn) contains all individuals that are instances
OWLAnonymousClassExpression	A Class Expression which is not a named Class.
OWLClassExpression	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
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.
OWLPropertyRange	OWL Objects that can be the ranges of properties.
OWLDataRange	Represents a DataRange in the OWL 2 Specification.
OWLLiteral	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
OWLIndividual	Represents a named or anonymous individual.
OWLDatatype	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
OWLObject	Base interface for OWL objects
OWLFacet	Enumerations for OWL facets.
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

Table 1 – continued from previous page

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 $1 \leq i \leq n$.
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

class owlapy.class_expression.restriction.**HasFiller**

Bases: Generic[_T]

An interface to objects that have a filler.

> **Parameters**
> **_T** – Filler type.

__slots__ = ()

abstract get_filler() → _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.**HasCardinality**

> An interface to objects that have a cardinality.
>
> **__slots__ = ()**
>
> **abstract get_cardinality**() → int
>
>> Gets the cardinality of a restriction.
>>
>> **Returns**
>>> The cardinality. A non-negative integer.

class owlapy.class_expression.restriction.**HasOperands**

> Bases: Generic[_T]
>
> An interface to objects that have a collection of operands.
>
> **Parameters**
>> **_T** – Operand type.
>
> **__slots__ = ()**
>
> **abstract operands**() → 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.class_expression.restriction.**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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Intersection_of_Class_Expressions)
>
> **__slots__ = '_operands'**
>
> **type_index: Final = 3001**

class owlapy.class_expression.restriction.**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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Union_of_Class_Expressions)
>
> **__slots__ = '_operands'**
>
> **type_index: Final = 3002**

class owlapy.class_expression.restriction.**OWLAnonymousClassExpression**

> Bases: *OWLClassExpression*
>
> A Class Expression which is not a named Class.
>
> **is_owl_nothing**() → 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() → 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() → *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.restriction.**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)

__slots__ = ()

abstract is_owl_thing() → 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() → 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.

class owlapy.class_expression.restriction.**OWLPropertyExpression**

> Bases: *owlapy.owl_object.OWLObject*

Represents a property or possibly the inverse of a property.

__slots__ = ()

is_data_property_expression() → bool

> > **Returns**
> >> True if this is a data property.

is_object_property_expression() → bool

> > **Returns**
> >> True if this is an object property.

is_owl_top_object_property() → bool

> Determines if this is the owl:topObjectProperty.

> > **Returns**
> >> topObjectProperty.

> > **Return type**
> >> True if this property is the owl

is_owl_top_data_property() → bool

> Determines if this is the owl:topDataProperty.

> > **Returns**
> >> topDataProperty.

> > **Return type**
> >> True if this property is the owl

class owlapy.class_expression.restriction.**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() → bool

> > **Returns**
> >> True if this is an object property.

class owlapy.class_expression.restriction.**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.class_expression.restriction.**OWLPropertyRange**

    Bases: *owlapy.owl_object.OWLObject*

    OWL Objects that can be the ranges of properties.

class owlapy.class_expression.restriction.**OWLDataRange**

    Bases: *OWLPropertyRange*

    Represents a DataRange in the OWL 2 Specification.

class owlapy.class_expression.restriction.**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)

    __slots__ = ()

    type_index: Final = 4008

    get_literal() → 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() → bool

        Whether this literal is typed as boolean.

    parse_boolean() → 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() → bool

        Whether this literal is typed as double.

    parse_double() → 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() → bool

> Whether this literal is typed as integer.

parse_integer() → 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() → bool

> Whether this literal is typed as string.

parse_string() → 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() → bool

> Whether this literal is typed as date.

parse_date() → 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() → bool

> Whether this literal is typed as dateTime.

parse_datetime() → 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.

is_duration() → bool

> Whether this literal is typed as duration.

parse_duration() → 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() → bool

> > **Returns**
> > true if the annotation value is a literal

as_literal() → *OWLLiteral*

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

to_python() → Literals

abstract get_datatype() → *owlapy.owl_datatype.OWLDatatype*

>   Gets the OWLDatatype which types this literal.

>   > **Returns**
>   > The OWLDatatype that types this literal.

class owlapy.class_expression.restriction.**OWLIndividual**

>   Bases: *owlapy.owl_object.OWLObject*

>   Represents a named or anonymous individual.

>   **__slots__ = ()**

class owlapy.class_expression.restriction.**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 IRI as string

class owlapy.class_expression.restriction.**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**() → bool

class owlapy.class_expression.restriction.**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

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() → bool

Determines if this is a data restriction.

> **Returns**
> True if this is a data restriction.

is_object_restriction() → bool

Determines if this is an object restriction.

> **Returns**
> True if this is an object restriction.

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

            _T – The value type.

    __slots__ = ()

    __eq__(*other*)

        Return self==value.

    __hash__()

        Return hash(self).

    get_filler() → _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() → 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__ = ()

class owlapy.class_expression.restriction.**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.

`__slots__ = ()`

`get_cardinality``()` → int

> Gets the cardinality of a restriction.
>
> > **Returns**
> > > The cardinality. A non-negative integer.

`get_filler``()` → _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 constant (data value). For quantified 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.

`__slots__ = ()`

`get_filler``()` → *[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](#))

Bases: *[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, 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()
        → *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]*, *OWLObjectRestriction*

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() → *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.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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals)

__slots__ = '_values'

type_index: Final = 3004

individuals() → 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() → *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}, … , {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__ = ()**
>
> **is_data_restriction**() → 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 constant (data value). For quantified restriction this will be a class expression or a data range.
> >
> > > **Returns**
> > >
> > > > the value

class owlapy.class_expression.restriction.**OWLDataCardinalityRestriction**(
        cardinality: int, *property:* *owlapy.owl_property.OWLDataPropertyExpression*,
        filler: *owlapy.owl_data_ranges.OWLDataRange*)

> 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).

`class` owlapy.class_expression.restriction.**`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.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 expressions DPEi, $1 \leq i \leq n$, 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, $1 \leq i \leq n$, 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, $1 \leq i \leq n$, 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, $1 \leq i \leq n$, 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]*, *OWLDataRestriction*

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() → *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.restriction.**OWLDataOneOf**(
> *values: owlapy.owl_literal.OWLLiteral | Iterable[owlapy.owl_literal.OWLLiteral]*)

Bases: *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() → Iterable[*owlapy.owl_literal.OWLLiteral*]

> Gets the values that are in the oneOf.
>
> #### Returns
>
> > The values of this {@code DataOneOf} class expression.

operands() → 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.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*

get_facet_restrictions() → 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

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).

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.
OWLClass	An OWL 2 named Class. Classes can be understood as sets of individuals.
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
OWLRestriction	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
OWLQuantifiedRestriction	Represents a quantified restriction.
OWLQuantifiedObjectRestriction	Represents a quantified object restriction.
OWLObjectRestriction	Represents an Object Property Restriction in the OWL 2 specification.
OWLHasValueRestriction	Represent a HasValue 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.
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
OWLQuantifiedDataRestriction	Represents a quantified data restriction.
OWLDataCardinalityRestriction	Represents Data Property Cardinality Restrictions.

Table 2 – continued from previous page

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 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,
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 $1 \leq i \leq n$.
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)

__slots__ = ()

abstract is_owl_thing() → 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() → 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.

class owlapy.class_expression.**OWLAnonymousClassExpression**

> Bases: *OWLClassExpression*

> A Class Expression which is not a named Class.

> **is_owl_nothing**() → 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**() → 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**() → *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.* *HasOperands*[*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() → 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() → 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() → 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() → *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.

class owlapy.class_expression.**OWLNaryBooleanClassExpression**(
> *operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression]*)

Bases: *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 $1 \leq i \leq n$. (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 $1 \leq i \leq n$. (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*

        Returns
            Property being restricted.

    is_data_restriction() → bool

        Determines if this is a data restriction.

        Returns
            True if this is a data restriction.

    is_object_restriction() → 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

    __slots__ = ()

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.

    __slots__ = ()

    get_filler() → *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.**OWLObjectRestriction**

    Bases: *OWLRestriction*

    Represents an Object Property Restriction in the OWL 2 specification.

    __slots__ = ()

    is_object_restriction() → 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.**OWLHasValueRestriction**(*value: _T*)

    Bases: Generic[_T], *OWLRestriction*, *owlapy.meta_classes.HasFiller*[_T]

    Represent a HasValue restriction in the OWL 2

        Parameters

            _T – The value type.

    __slots__ = ()

    __eq__(*other*)

        Return self==value.

    __hash__()

        Return hash(self).

    get_filler() → _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.**OWLDataRestriction**

    Bases: *OWLRestriction*

    Represents a Data Property Restriction.

    __slots__ = ()

    is_data_restriction() → 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.

__slots__ = ()

get_cardinality() → int

> Gets the cardinality of a restriction.
>
> > **Returns**
> >
> > > The cardinality. A non-negative integer.

get_filler() → _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 constant (data value). For quantified restriction this will be a class expression or a data range.
>
> > **Returns**
> >
> > > the value

class owlapy.class_expression.**OWLObjectCardinalityRestriction**(*cardinality: int*, *property: owlapy.owl_property.OWLObjectPropertyExpression*, *filler: owlapy.class_expression.class_expression.OWLClassExpression*)

Bases: *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.**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](#)]*)

Bases: `[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 ≤ i ≤ n. The resulting data range has arity one. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Literals)

type_index: Final = 4003

values() → Iterable[[*owlapy.owl_literal.OWLLiteral*](#)]

>   Gets the values that are in the oneOf.

>   **Returns**
>   The values of this {@code DataOneOf} class expression.

operands() → 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](#)*)

Bases: `[OWLQuantifiedRestriction](#)[owlapy.owl_data_ranges.OWLDataRange]`,   `[OWLDataRestriction](#)`

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)

    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).

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).

get_property() → *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`], `OWLObjectRestriction`

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() → *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*

`get_facet_restrictions`() → Sequence[*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_Cardinality]())

__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()
> → [*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.**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 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).

__eq__(*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).

__eq__(*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*], *OWLDataRestriction*

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() → *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 $1 \leq i \leq n$. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals)

`__slots__ = '_values'`

`type_index: Final = 3004`

`individuals`() → 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`() → *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}, … , {an})

`__hash__``()`

> Return hash(self).

`__eq__``(`*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.

3.2 Submodules

owlapy.converter

Format converter.

Attributes

TopOWLDatatype	
converter	

Classes

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 $1 \leq i \leq n$.
OWLDatatypeRestriction	A datatype restriction DatatypeRestriction(DT F1 lt1 ... Fn ltn) consists of a unary datatype DT and n pairs
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
OWLClass	An OWL 2 named Class. Classes can be understood as sets of individuals.
OWLClassExpression	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
OWLObjectIntersectionOf	An intersection class expression ObjectIntersectionOf(CE1 ... CEn) contains all individuals that are instances
OWLObjectUnionOf	A union class expression ObjectUnionOf(CE1 ... CEn) contains all individuals that are instances
OWLObjectComplementOf	Represents an ObjectComplementOf class expression in the OWL 2 Specification.
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
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
OWLDataCardinalityRestriction	Represents Data Property Cardinality Restrictions.

Table 3 – continued from previous page

OWLObjectHasSelf	A self-restriction ObjectHasSelf(OPE) consists of an object property expression OPE,
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
OWLNamedIndividual	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
OWLLiteral	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
OWLObjectProperty	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
OWLDataProperty	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
OWLEntity	Represents Entities in the OWL 2 Specification.
OWLDatatype	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
OWLFacet	Enumerations for OWL facets.
OWLRDFVocabulary	Enumerations for OWL/RDF vocabulary.
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

class owlapy.converter.**OWLObjectHasValue**(
       property: owlapy.owl_property.OWLObjectPropertyExpression,
       individual: owlapy.owl_individual.OWLIndividual)

    Bases: `OWLHasValueRestriction[`*owlapy.owl_individual.OWLIndividual*`]`, `OWLObjectRestriction`

    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)

52

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

`type_index: Final = 3007`

`get_property``()` → *owlapy.owl_property.OWLObjectPropertyExpression*

> **Returns**
>> Property being restricted.

`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.

> **Returns**
>> The existential equivalent of this value restriction. simp(HasValue(p a)) = some(p {a}).

`__repr__``()`

> Return repr(self).

class owlapy.converter.**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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals)

`__slots__ = '_values'`

`type_index: Final = 3004`

`individuals``()` → 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``()` → *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}, … , {an})

`__hash__``()`

> Return hash(self).

`__eq__``(`*other*`)`

> Return self==value.

`__repr__``()`

> Return repr(self).

class owlapy.converter.**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*

get_facet_restrictions() → Sequence[*OWLFacetRestriction*]

__eq__(*other*)
  Return self==value.

__hash__()
  Return hash(self).

__repr__()
  Return repr(self).

class owlapy.converter.**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.converter.**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.converter.**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.converter.`**`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`() → 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`() → 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() → *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.

class owlapy.converter.**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)
>
> **__slots__ = ()**
>
> **abstract is_owl_thing**() → 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**() → 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.

class owlapy.converter.**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 $1 \leq i \leq n$. (https://www.w3.org/TR/owl2-syntax/#Intersection_of_Class_Expressions)
>
> **__slots__ = '_operands'**
>
> **type_index: Final = 3001**

class owlapy.converter.**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 $1 \leq i \leq n$. (https://www.w3.org/TR/owl2-syntax/#Union_of_Class_Expressions)

__slots__ = '_operands'

type_index: Final = 3002

class owlapy.converter.**OWLObjectComplementOf**(*op: OWLClassExpression*)

Bases: OWLBooleanClassExpression, *owlapy.meta_classes. HasOperands*[*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() → 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.converter.**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.converter.**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.converter.**OWLObjectCardinalityRestriction**(*cardinality: int*,
        property: owlapy.owl_property.OWLObjectPropertyExpression,
        filler: owlapy.class_expression.class_expression.OWLClassExpression)

Bases: 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.converter.**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.converter.**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.converter.**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()
```
→ *[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.converter.**OWLDataCardinalityRestriction**(*cardinality: int*,
property: [owlapy.owl_property.OWLDataPropertyExpression](#),
filler: [owlapy.owl_data_ranges.OWLDataRange](#))

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).

class owlapy.converter.**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.converter.**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 expressions DPEi, $1 \leq i \leq n$, 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, $1 \leq i \leq n$, 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.converter.**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.converter.**OWLDataHasValue**(
        property: owlapy.owl_property.OWLDataPropertyExpression,
        value: owlapy.owl_literal.OWLLiteral)

Bases: `OWLHasValueRestriction[`*owlapy.owl_literal.OWLLiteral*`]`, `OWLDataRestriction`

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() → *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.converter.**OWLDataOneOf**(
>   *values: owlapy.owl_literal.OWLLiteral | Iterable[owlapy.owl_literal.OWLLiteral]*)

Bases: *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() → Iterable[*owlapy.owl_literal.OWLLiteral*]

> Gets the values that are in the oneOf.
>
> > **Returns**
> >
> > > The values of this {@code DataOneOf} class expression.

operands() → 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.converter.**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

class owlapy.converter.**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)

> **__slots__ = ()**

> **type_index: Final = 4008**

> **get_literal**() → 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**() → bool

>> Whether this literal is typed as boolean.

> **parse_boolean**() → 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**() → bool

>> Whether this literal is typed as double.

> **parse_double**() → 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**() → bool

>> Whether this literal is typed as integer.

> **parse_integer**() → 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**() → bool

>> Whether this literal is typed as string.

> **parse_string**() → 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() → bool

> Whether this literal is typed as date.

parse_date() → 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() → bool

> Whether this literal is typed as dateTime.

parse_datetime() → 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.

is_duration() → bool

> Whether this literal is typed as duration.

parse_duration() → 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() → bool

> > **Returns**
> >
> > true if the annotation value is a literal

as_literal() → *OWLLiteral*

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

to_python() → Literals

abstract get_datatype() → *owlapy.owl_datatype.OWLDatatype*

> Gets the OWLDatatype which types this literal.
>
> > **Returns**
> >
> > The OWLDatatype that types this literal.

owlapy.converter.**TopOWLDatatype: Final**

class owlapy.converter.**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`

`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.

`get_inverse_property``()` → *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``()` → bool

> Determines if this is the owl:topObjectProperty.

> > **Returns**
> > topObjectProperty.

> > **Return type**
> > True if this property is the owl

class `owlapy.converter.`**`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

class `owlapy.converter.`**`OWLEntity`**

> Bases: `OWLNamedObject`

> Represents Entities in the OWL 2 Specification.

> **`__slots__ = ()`**

> **`to_string_id`**`()` → str

> **`is_anonymous`**`()` → bool

class `owlapy.converter.`**`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 IRI as string

class `owlapy.converter.`**`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.converter.`**`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.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

new_individual_variable() → str

new_property_variable() → 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 () → 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) → 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*) → 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 (¬(∃r.¬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

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.
Namespaces	Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup
IRI	An IRI, consisting of a namespace and a remainder.

Module Contents

class owlapy.iri.**OWLAnnotationSubject**

Bases: OWLAnnotationObject

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

__slots__ = ()

class owlapy.iri.**OWLAnnotationValue**

Bases: OWLAnnotationObject

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

__slots__ = ()

is_literal() → bool

> **Returns**
> > true if the annotation value is a literal

as_literal() → *[OWLLiteral](#)* | None

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

class owlapy.iri.**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.

class owlapy.iri.**IRI**(*namespace: str | owlapy.namespaces.Namespaces*, *remainder: str*)

> Bases: *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*) → *IRI*
>
> **static create**(*namespace: str*, *remainder: str*) → *IRI*
>
> **static create**(*string: str*) → *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 <http://www.w3.org/2002/07/owl#Nothing> 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 <http://www.w3.org/2002/07/owl#Thing> and otherwise False.
>
> **is_reserved_vocabulary**() → bool
>
> > Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http://www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.

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

as_iri() → *IRI*

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

as_str() → 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.

get_short_form() → str

> Gets the short form.

> **Returns**
>> A string that represents the short form.

get_namespace() → str

> **Returns**
>> The namespace as string.

get_remainder() → 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__ = ()**
>
> **abstract operands**() → 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__ = ()**
>
> **abstract get_filler**() → _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.meta_classes.**HasCardinality**

> An interface to objects that have a cardinality.
>
> **__slots__ = ()**
>
> **abstract get_cardinality**() → 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 element 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.namespaces.**XSD: Final**

owlapy.owl_annotation

OWL Annotations

Classes

OWLObject	Base interface for OWL objects
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

class owlapy.owl_annotation.**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() → bool

class owlapy.owl_annotation.**OWLAnnotationObject**

   Bases: *owlapy.owl_object.OWLObject*

   A marker interface for the values (objects) of annotations.

   __slots__ = ()

   as_iri() → *IRI* | None

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

   as_anonymous_individual()

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

class owlapy.owl_annotation.**OWLAnnotationSubject**

   Bases: *OWLAnnotationObject*

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

   __slots__ = ()

class owlapy.owl_annotation.**OWLAnnotationValue**

    Bases: *OWLAnnotationObject*

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

    __slots__ = ()

    is_literal() → bool

        Returns
            true if the annotation value is a literal

    as_literal() → *OWLLiteral* | None

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

owlapy.owl_axiom

OWL Axioms

Attributes

OWLNothing	
OWLThing	

Classes

OWLDataPropertyExpression	A high level interface to describe different types of data properties.
OWLObjectPropertyExpression	A high level interface to describe different types of object properties.
OWLObject	Base interface for OWL objects
OWLEntity	Represents Entities in the OWL 2 Specification.
OWLDatatype	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
OWLDataRange	Represents a DataRange in the OWL 2 Specification.
HasOperands	An interface to objects that have a collection of operands.
OWLPropertyExpression	Represents a property or possibly the inverse of a property.
OWLProperty	A base class for properties that aren't expression i.e. named properties. By definition, properties
OWLClassExpression	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
OWLClass	An OWL 2 named Class. Classes can be understood as sets of individuals.

Table 4 – continued from previous page

OWLObjectUnionOf	A union class expression ObjectUnionOf(CE1 ... CEn) contains all individuals that are instances
OWLIndividual	Represents a named or anonymous individual.
IRI	An IRI, consisting of a namespace and a remainder.
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.
OWLLiteral	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
OWLAxiom	Represents Axioms in the OWL 2 Specification.
OWLLogicalAxiom	A base interface of all axioms that affect the logical meaning 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 Specification. A declaration axiom declares an entity in an ontology.
OWLDatatypeDefinitionAxiom	A datatype definition DatatypeDefinition(DT DR) defines 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, $1 \leq i \leq n$,
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,
OWLSameIndividualAxiom	An individual equality axiom SameIndividual(a1 ... an) states that all of the individuals ai, $1 \leq i \leq n$,
OWLNaryPropertyAxiom	Represents an axiom that contains two or more operands that could also be represented with
OWLEquivalentObjectPropertiesAxiom	An equivalent object properties axiom EquivalentObjectProperties(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

Table 4 – continued from previous page

`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
`OWLNegativeObjectPropertyAssertionAxiom`	A negative object property assertion NegativeObjectPropertyAssertion(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
`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
`OWLInverseFunctionalObjectPropertyAxiom`	An object property inverse functionality axiom InverseFunctionalObjectProperty(OPE)
`OWLIrreflexiveObjectPropertyAxiom`	An object property irreflexivity axiom IrreflexiveObjectProperty(OPE) states that the
`OWLReflexiveObjectPropertyAxiom`	An object property reflexivity axiom ReflexiveObjectProperty(OPE) states that the

77

Table 4 – continued from previous page

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 FunctionalDataProp-erty(DPE) states that
OWLPropertyDomainAxiom	Base class for Property Domain axioms.
OWLPropertyRangeAxiom	Base class for Property Range axioms.
OWLObjectPropertyDomainAxiom	An object property domain axiom ObjectPropertyDo-main(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

class owlapy.owl_axiom.**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_axiom.**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() → bool

> **Returns**
> True if this is an object property.

78

class owlapy.owl_axiom.**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() → bool

class owlapy.owl_axiom.**OWLEntity**

    Bases: OWLNamedObject

    Represents Entities in the OWL 2 Specification.

    __slots__ = ()

    to_string_id() → str

    is_anonymous() → bool

class owlapy.owl_axiom.**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 IRI as string

class owlapy.owl_axiom.**OWLDataRange**

    Bases: OWLPropertyRange

    Represents a DataRange in the OWL 2 Specification.

class owlapy.owl_axiom.**HasOperands**

> Bases: `Generic[_T]`
>
> An interface to objects that have a collection of operands.
>
> > **Parameters**
> > **_T** – Operand type.
>
> **__slots__ = ()**
>
> **abstract operands**() → 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.owl_axiom.**OWLPropertyExpression**

> Bases: *owlapy.owl_object.OWLObject*
>
> Represents a property or possibly the inverse of a property.
>
> **__slots__ = ()**
>
> **is_data_property_expression**() → bool
>
> > > **Returns**
> > > True if this is a data property.
>
> **is_object_property_expression**() → bool
>
> > > **Returns**
> > > True if this is an object property.
>
> **is_owl_top_object_property**() → bool
>
> > Determines if this is the owl:topObjectProperty.
> >
> > > **Returns**
> > > topObjectProperty.
> >
> > > **Return type**
> > > True if this property is the owl
>
> **is_owl_top_data_property**() → bool
>
> > Determines if this is the owl:topDataProperty.
> >
> > > **Returns**
> > > topDataProperty.
> >
> > > **Return type**
> > > True if this property is the owl

class owlapy.owl_axiom.**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_axiom.**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)

> **__slots__ = ()**

> **abstract is_owl_thing**() → 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**() → 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.

class owlapy.owl_axiom.**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() → 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() → 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() → *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.owl_axiom.**OWLNothing: Final**

owlapy.owl_axiom.**OWLThing: Final**

class owlapy.owl_axiom.**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 $1 \leq i \leq n$. (https://www.w3.org/TR/owl2-syntax/#Union_of_Class_Expressions)

>   **__slots__ = '_operands'**

>   **type_index: Final = 3002**

class owlapy.owl_axiom.**OWLIndividual**

>   Bases: *owlapy.owl_object.OWLObject*

>   Represents a named or anonymous individual.

`__slots__ = ()`

class owlapy.owl_axiom.**IRI**(*namespace: str | owlapy.namespaces.Namespaces*, *remainder: str*)

Bases: *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*) → *IRI*

static create(*namespace: str*, *remainder: str*) → *IRI*

static create(*string: str*) → *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 <http://www.w3.org/2002/07/owl#Nothing> 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 <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

`is_reserved_vocabulary`() → bool

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

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

`as_iri`() → *IRI*

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

`as_str`() → 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.

get_short_form() → str

> Gets the short form.
>
> > **Returns**
> >
> > > A string that represents the short form.

get_namespace() → str

> > **Returns**
> >
> > > The namespace as string.

get_remainder() → str

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

class owlapy.owl_axiom.**OWLAnnotationSubject**

> Bases: OWLAnnotationObject
>
> A marker interface for annotation subjects, which can either be IRIs or anonymous individuals
>
> **__slots__ = ()**

class owlapy.owl_axiom.**OWLAnnotationValue**

> Bases: OWLAnnotationObject
>
> A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.
>
> **__slots__ = ()**
>
> **is_literal**() → bool
>
> > **Returns**
> >
> > > true if the annotation value is a literal
>
> **as_literal**() → *OWLLiteral* | None
>
> > **Returns**
> >
> > > if the value is a literal, returns it. Return None otherwise

class owlapy.owl_axiom.**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)
>
> **__slots__ = ()**
>
> **type_index: Final = 4008**
>
> **get_literal**() → 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**() → bool
>
> > Whether this literal is typed as boolean.

parse_boolean() → 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() → bool

> Whether this literal is typed as double.

parse_double() → 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() → bool

> Whether this literal is typed as integer.

parse_integer() → 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() → bool

> Whether this literal is typed as string.

parse_string() → 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() → bool

> Whether this literal is typed as date.

parse_date() → 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() → bool

> Whether this literal is typed as dateTime.

parse_datetime() → 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.

is_duration() → bool

> Whether this literal is typed as duration.

parse_duration() → 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() → bool

>   **Returns**
>
> >   true if the annotation value is a literal

as_literal() → *OWLLiteral*

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

to_python() → Literals

abstract get_datatype() → *owlapy.owl_datatype.OWLDatatype*

>   Gets the OWLDatatype which types this literal.
>
>   **Returns**
>
> >   The OWLDatatype that types this literal.

class owlapy.owl_axiom.**OWLAxiom**(*annotations: Iterable[OWLAnnotation] | None = 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**() → List[*OWLAnnotation*] | None
>
>   **is_annotated**() → bool
>
>   **is_logical_axiom**() → bool
>
>   **is_annotation_axiom**() → 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**() → 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*

    __eq__(*other*)

        Return self==value.

    __hash__()

        Return hash(self).

    __repr__()

        Return repr(self).

class owlapy.owl_axiom.**OWLDatatypeDefinitionAxiom**(
    datatype: owlapy.owl_datatype.OWLDatatype, *datarange: owlapy.owl_datatype.OWLDataRange*,
    annotations: Iterable[OWLAnnotation] | None = None)

    Bases: *OWLLogicalAxiom*

    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.

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

```
__slots__ = ('_datatype', '_datarange')
```

get_datatype() → *owlapy.owl_datatype.OWLDatatype*

get_datarange() → *owlapy.owl_datatype.OWLDataRange*

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

class owlapy.owl_axiom.**OWLHasKeyAxiom**(
        class_expression: owlapy.class_expression.OWLClassExpression,
        property_expressions: List[owlapy.owl_property.OWLPropertyExpression],
        annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLLogicalAxiom, owlapy.meta_classes.HasOperands[owlapy.owl_property.OWLPropertyExpression]*

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*]

operands() → Iterable[*owlapy.owl_property.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() → 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() → 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.

        Returns

            This axiom as a set of pairwise axioms.

__eq__(*other*)

    Return self==value.

__hash__()

    Return hash(self).

__repr__()

    Return repr(self).

class owlapy.owl_axiom.**OWLEquivalentClassesAxiom**(
       class_expressions: List[owlapy.class_expression.OWLClassExpression],
       annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLNaryClassAxiom*

An equivalent classes axiom EquivalentClasses(CE1 … CEn) states that all of the class expressions CEi, $1 \le i \le n$, are semantically equivalent to each other. This axiom allows one to use each CEi as a synonym for each CEj — 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() → bool

contains_owl_nothing() → bool

contains_owl_thing() → 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: *OWLNaryClassAxiom*

A disjoint classes axiom DisjointClasses(CE1 … CEn) states that all of the class expressions CEi, $1 \leq i \leq n$, are pairwise disjoint; that is, no individual can be at the same time an instance of both CEi and CEj for $i \neq j$.

(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() → Iterable[*OWLNaryIndividualAxiom*]

__eq__(*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, $1 \leq i \leq n$, 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, $1 \leq i \leq n$, are equal to each other. This axiom allows one to use each ai as a synonym for each aj — 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() → Iterable[_P]

        Get all the properties that appear in the axiom.

            Returns

                Generator containing the properties.

    as_pairwise_axioms() → 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 ≤ i ≤ n, are semantically equivalent to each other. This axiom allows one to use each OPEi as a synonym for each OPEj — 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 ≤ i ≤ n, are pairwise disjoint; that is, no individual x can be connected to an individual y by both OPEi and OPEj for i ≠ 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 ≤ i ≤ n, are semantically equivalent to each other. This axiom allows one to use each DPEi as a synonym for each DPEj — 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)

Bases: *OWLNaryPropertyAxiom*[*owlapy.owl_property.OWLDataPropertyExpression*], *OWLDataPropertyAxiom*

A disjoint data properties axiom DisjointDataProperties(DPE1 … DPEn) states that all of the data property expressions DPEi, 1 ≤ i ≤ n, are pairwise disjoint; that is, no individual x can be connected to a literal y by both

    DPEi and DPEj for i ≠ 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*

__eq__(*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 expressions CEi, 1 ≤ i ≤ 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')
```

get_owl_class() → *owlapy.class_expression.OWLClass*

get_class_expressions() → Iterable[*owlapy.class_expression.OWLClassExpression*]

get_owl_equivalent_classes_axiom() → *OWLEquivalentClassesAxiom*

get_owl_disjoint_classes_axiom() → *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*

get_class_expression() → *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.
>
> > **Returns**
> > 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.

get_value() → *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() → 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')

    get_subject() → *owlapy.owl_annotation.OWLAnnotationSubject*

        Gets the subject of this object.

            Returns
                The subject.

    get_property() → *OWLAnnotationProperty*

        Gets the property.

            Returns
                The property.

    get_value() → *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(AP U) 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')

get_property() → *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')

get_property() → *OWLAnnotationProperty*

get_range() → *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')

　　get_sub_property() → _P

　　get_super_property() → _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)

　　Bases:　*OWLSubPropertyAxiom[owlapy.owl_property.OWLObjectPropertyExpression]*,
　　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)

　　Bases:　　*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*

get_property() → _P

get_object() → _C

__eq__(*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)

Bases: `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)

Bases: *[OWLPropertyAssertionAxiom]()[[owlapy.owl_property.OWLDataPropertyExpression](),
[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'

get_property() → _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 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 expressionOPE 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*

`__eq__`(*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'`

`get_range`() → _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 expression 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)

  Bases: *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 | DataIntersectionOf | DataUnionOf | DataComplementOf | DataOneOf | DatatypeRestriction

Classes

OWLObject	Base interface for OWL objects
HasOperands	An interface to objects that have a collection of operands.
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.**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() → bool

class owlapy.owl_data_ranges.**HasOperands**

 Bases: Generic[_T]

 An interface to objects that have a collection of operands.

 Parameters

 _T – Operand type.

 __slots__ = ()

 abstract operands() → 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.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 ≤ i ≤ 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 ≤ i ≤ 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

OWLEntity	Represents Entities in the OWL 2 Specification.
OWLDataRange	Represents a DataRange in the OWL 2 Specification.
IRI	An IRI, consisting of a namespace and a remainder.
HasIRI	Simple class to access the IRI.
OWLDatatype	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,

Module Contents

class owlapy.owl_datatype.**OWLEntity**

    Bases: OWLNamedObject

    Represents Entities in the OWL 2 Specification.

    __slots__ = ()

    to_string_id() → str

    is_anonymous() → bool

class owlapy.owl_datatype.**OWLDataRange**

    Bases: OWLPropertyRange

    Represents a DataRange in the OWL 2 Specification.

class owlapy.owl_datatype.**IRI**(*namespace: str | owlapy.namespaces.Namespaces*, *remainder: str*)

    Bases: *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*) → *IRI*

    static create(*namespace: str*, *remainder: str*) → *IRI*

    static create(*string: str*) → *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 <http://www.w3.org/2002/07/owl#Nothing> 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 <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

is_reserved_vocabulary() → bool

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

>> **Returns**

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

as_iri() → *IRI*

>> **Returns**

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

as_str() → 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.

get_short_form() → str

> Gets the short form.

>> **Returns**

>>> A string that represents the short form.

get_namespace() → str

>> **Returns**

>>> The namespace as string.

get_remainder() → str

>> **Returns**

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

class owlapy.owl_datatype.**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.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 IRI as string

owlapy.owl_hierarchy

Classes representing hierarchy in OWL.

Attributes

OWLThing
OWLNothing
OWLTopObjectProperty
OWLBottomObjectProperty
OWLTopDataProperty
OWLBottomDataProperty

Classes

OWLClass	An OWL 2 named Class. Classes can be understood as sets of individuals.
HasIRI	Simple class to access the IRI.
OWLObjectProperty	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
OWLDataProperty	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
OWLReasoner	An OWLReasoner reasons over a set of axioms (the set of reasoner axioms) that is based on the imports closure of
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

class owlapy.owl_hierarchy.**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() → 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() → 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() → *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.owl_hierarchy.**OWLThing: Final**

owlapy.owl_hierarchy.**OWLNothing: Final**

class owlapy.owl_hierarchy.**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

owlapy.owl_hierarchy.**OWLTopObjectProperty: Final**

owlapy.owl_hierarchy.**OWLBottomObjectProperty: Final**

owlapy.owl_hierarchy.**OWLTopDataProperty: Final**

owlapy.owl_hierarchy.**OWLBottomDataProperty: Final**

class owlapy.owl_hierarchy.**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**

`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.

`get_inverse_property`() → *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`() → bool

> Determines if this is the owl:topObjectProperty.
>
> > **Returns**
> > topObjectProperty.
> >
> > **Return type**
> > True if this property is the owl

class owlapy.owl_hierarchy.**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

class owlapy.owl_hierarchy.**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*) → 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.

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 ObjectComplementOf(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 EquivalentDataProperties(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*)
> → 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).

abstract object_property_values(*ind: owlapy.owl_individual.OWLNamedIndividual*,
> *pe: owlapy.owl_property.OWLObjectPropertyExpression*, *direct: bool = True*)
> → 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).

abstract flush() → 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) → 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 sub-properties (descendants) should be retrieved (False).

> **Returns**
> If direct is True, each property P where the set of reasoner axioms entails DirectSubDataProp-ertyOf(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 sub-properties (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.

abstract 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).

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*) → 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_hierarchy.**AbstractHierarchy**(*factory: Type[_S]*, *hierarchy_down: Iterable[Tuple[_S, Iterable[_S]]]*)

class owlapy.owl_hierarchy.**AbstractHierarchy**(*factory: Type[_S]*, *reasoner: owlapy.owl_reasoner.OWLReasoner*)

Bases: `Generic[_S]`

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',...

classmethod get_top_entity() → _S

Abstractmethod

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

classmethod get_bottom_entity() → _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*) → _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*) → _U

> Restrict this hierarchy.
>
> See restrict for more info.

parents(*entity: _S*, *direct: bool = True*) → 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*) → bool

> if A is a parent of B.
>
> ---
>
> **Note:** A is always a parent of A.
>
> ---

is_child_of(*a: _S*, *b: _S*) → bool

> If A is a child of B.
>
> ---
>
> **Note:** A is always a child of A.
>
> ---

children(*entity: _S*, *direct: bool = True*) → Iterable[_S]

> Children of an entity.
>
> > **Parameters**
> >
> > - **entity** – Entity for which to query child entities.
> >
> > - **direct** – False to return transitive children.
> >
> > **Returns**
> > Sub-entities.

siblings(*entity: _S*) → Iterable[_S]

items() → Iterable[_S]

roots(*of: _S | None = None*) → Iterable[_S]

leaves(*of: _S | None = None*) → Iterable[_S]

__contains__(*item: _S*) → 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 sub-classes.

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) → 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() → *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](#)*) → bool

most_general_roles() → Iterable[*[owlapy.owl_property.OWLObjectProperty](#)*]

most_special_roles() → 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.

    classmethod get_bottom_entity() → *[owlapy.owl_property.OWLDataProperty](#)*
        The most specific entity in this hierarchy, which contains none of the entities.

    sub_data_properties(*entity: [owlapy.owl_property.OWLDataProperty](#), direct: bool = True*)

    super_data_properties(*entity: [owlapy.owl_property.OWLDataProperty](#), direct: bool = True*)

    more_general_roles(*role: [owlapy.owl_property.OWLDataProperty](#), direct: bool = True*)
        → Iterable[*[owlapy.owl_property.OWLDataProperty](#)*]

    more_special_roles(*role: [owlapy.owl_property.OWLDataProperty](#), direct: bool = True*)
        → Iterable[*[owlapy.owl_property.OWLDataProperty](#)*]

    is_sub_property_of(*sub_property: [owlapy.owl_property.OWLDataProperty](#),
        super_property: [owlapy.owl_property.OWLDataProperty](#)*) → bool

    most_general_roles() → Iterable[*[owlapy.owl_property.OWLDataProperty](#)*]

    most_special_roles() → Iterable[*[owlapy.owl_property.OWLDataProperty](#)*]

owlapy.owl_individual

OWL Individuals

Classes

OWLObject	Base interface for OWL objects
OWLEntity	Represents Entities in the OWL 2 Specification.
IRI	An IRI, consisting of a namespace and a remainder.
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.**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() → bool

class owlapy.owl_individual.**OWLEntity**

    Bases: OWLNamedObject

    Represents Entities in the OWL 2 Specification.

    __slots__ = ()

    to_string_id() → str

    is_anonymous() → bool

class owlapy.owl_individual.**IRI**(*namespace: str | owlapy.namespaces.Namespaces*, *remainder: str*)

    Bases: *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*) → *IRI*
    static create(*namespace: str*, *remainder: str*) → *IRI*
    static create(*string: str*) → *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 <http://www.w3.org/2002/07/owl#Nothing> 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 <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

`is_reserved_vocabulary`() → bool

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

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

`as_iri`() → *IRI*

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

`as_str`() → 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.

`get_short_form`() → str

> Gets the short form.

>> **Returns**
>>   A string that represents the short form.

`get_namespace`() → str

>> **Returns**
>>   The namespace as string.

`get_remainder`() → str

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

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	
DateOWLDatatype	
DateTimeOWLDatatype	
DurationOWLDatatype	
TopOWLDatatype	
NUMERIC_DATATYPES	
TIME_DATATYPES	

Classes

OWLAnnotationValue	A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.
OWLDatatype	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
OWLRDFVocabulary	Enumerations for OWL/RDF vocabulary.
XSDVocabulary	Enumerations for XSD vocabulary.
OWLObjectProperty	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
OWLDataProperty	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
OWLLiteral	Literals represent data values such as particular strings or integers. They are analogous to typed RDF

Module Contents

class owlapy.owl_literal.**OWLAnnotationValue**

    Bases: OWLAnnotationObject

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

    __slots__ = ()

    is_literal() → bool

            Returns

                true if the annotation value is a literal

    as_literal() → *OWLLiteral* | None

            Returns

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

class owlapy.owl_literal.**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 IRI as string

class owlapy.owl_literal.**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.owl_literal.**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.owl_literal.**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

    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.

    get_inverse_property() → *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() → bool

    Determines if this is the owl:topObjectProperty.

> **Returns**
>   topObjectProperty.
>
> **Return type**
>   True if this property is the owl

class owlapy.owl_literal.**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_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)

    __slots__ = ()

    type_index: Final = 4008

    get_literal() → 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() → bool

        Whether this literal is typed as boolean.

    parse_boolean() → 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`() → bool

>   Whether this literal is typed as double.

`parse_double`() → 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`() → bool

>   Whether this literal is typed as integer.

`parse_integer`() → 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`() → bool

>   Whether this literal is typed as string.

`parse_string`() → 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`() → bool

>   Whether this literal is typed as date.

`parse_date`() → 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`() → bool

>   Whether this literal is typed as dateTime.

`parse_datetime`() → 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.

`is_duration`() → bool

>   Whether this literal is typed as duration.

`parse_duration`() → 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() → bool

> **Returns**
>> true if the annotation value is a literal

as_literal() → *OWLLiteral*

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

to_python() → 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.**OWLBottomObjectProperty: Final**

owlapy.owl_literal.**OWLTopDataProperty: 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.**NUMERIC_DATATYPES: Final[Set[*owlapy.owl_datatype.OWLDatatype*]]**

owlapy.owl_literal.**TIME_DATATYPES: Final[Set[*owlapy.owl_datatype.OWLDatatype*]]**

owlapy.owl_object

OWL Base classes

Classes

HasIRI	Simple class to access the IRI.
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.`**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.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 () → bool

class `owlapy.owl_object.`**OWLObjectRenderer**

Abstract class with a render method to render an OWL Object into a string.

abstract set_short_form_provider(*short_form_provider*) → None

   Configure a short form provider that shortens the OWL objects during rendering.

   Parameters
      short_form_provider – Short form provider.

abstract render(*o: OWLObject*) → 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*) → *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.

   __lt__(*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() → str

   is_anonymous() → bool

owlapy.owl_ontology

OWL Ontology

Attributes

IntegerOWLDatatype
DoubleOWLDatatype
BooleanOWLDatatype
StringOWLDatatype
DateOWLDatatype
DateTimeOWLDatatype
DurationOWLDatatype
OWLThing
logger
OWLREADY2_FACET_KEYS

Classes

OWLEquivalentClassesAxiom	An equivalent classes axiom EquivalentClasses(CE1 ... CEn) states that all of the class expressions CEi,
OWLClassAxiom	The base interface for class axioms.
OWLDataPropertyDomainAxiom	A data property domain axiom DataPropertyDomain(DPE CE) states that the domain of the
OWLDataPropertyRangeAxiom	A data property range axiom DataPropertyRange(DPE DR) states that the range of the data property
OWLObjectPropertyDomainAxiom	An object property domain axiom ObjectPropertyDomain(OPE CE) states that the domain of the
OWLObjectPropertyRangeAxiom	An object property range axiom ObjectPropertyRange(OPE CE) states that the range of the object property
OWLSubClassOfAxiom	A subclass axiom SubClassOf(CE1 CE2) states that the class expression CE1 is a subclass of the class
OWLAnnotationProperty	Represents an AnnotationProperty in the OWL 2 specification.
OWLDataRange	Represents a DataRange in the OWL 2 Specification.
OWLDataComplementOf	A complement data range DataComplementOf(DR) contains all tuples of literals that are not contained in the

Table 5 – continued from previous page

OWLDataUnionOf	A union data range DataUnionOf(DR1 ... DRn) contains all tuples of literals that are contained in the at least
OWLDataIntersectionOf	An intersection data range DataIntersectionOf(DR1 ... DRn) contains all tuples of literals that are contained
OWLDatatype	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
OWLNamedIndividual	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
OWLIndividual	Represents a named or anonymous individual.
OWLLiteral	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
OWLObject	Base interface for OWL objects
IRI	An IRI, consisting of a namespace and a remainder.
OWLClass	An OWL 2 named Class. Classes can be understood as sets of individuals.
OWLClassExpression	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
OWLObjectComplementOf	Represents an ObjectComplementOf class expression in the OWL 2 Specification.
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
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
OWLObjectOneOf	An enumeration of individuals ObjectOneOf(a1 ... an) contains exactly the individuals ai with $1 \leq i \leq n$.
OWLObjectExactCardinality	An exact cardinality expression ObjectExactCardinality(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
OWLObjectMinCardinality	A minimum cardinality expression ObjectMinCardinality(n OPE CE) consists of a nonnegative integer n, an object
OWLObjectHasValue	A has-value class expression ObjectHasValue(OPE a) consists of an object property expression OPE and 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,
OWLDataExactCardinality	An exact cardinality expression ObjectExactCardinality(n OPE CE) consists of a nonnegative integer n, an
OWLDataMaxCardinality	A maximum cardinality expression ObjectMaxCardinality(n OPE CE) consists of a nonnegative integer n, an object

Table 5 – continued from previous page

OWLDataMinCardinality	A minimum cardinality expression DataMinCardinality(n DPE DR) consists of a nonnegative integer n, a data
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
OWLRestriction	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
OWLObjectRestriction	Represents an Object Property Restriction in the OWL 2 specification.
OWLDataRestriction	Represents a Data Property Restriction.
OWLFacetRestriction	A facet restriction is used to restrict a particular datatype.
OWLDataProperty	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
OWLObjectProperty	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
OWLPropertyExpression	Represents a property or possibly the inverse of a property.
OWLObjectInverseOf	Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression
OWLObjectPropertyExpression	A high level interface to describe different types of object properties.
OWLDataPropertyExpression	A high level interface to describe different types of data properties.
OWLFacet	Enumerations for OWL facets.
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

class owlapy.owl_ontology.**OWLEquivalentClassesAxiom**(
      class_expressions: List[owlapy.class_expression.OWLClassExpression],
      annotations: Iterable[OWLAnnotation] | None = None)

    Bases: OWLNaryClassAxiom

    An equivalent classes axiom EquivalentClasses(CE1 … CEn) states that all of the class expressions CEi, 1 ≤ i ≤ n, are semantically equivalent to each other. This axiom allows one to use each CEi as a synonym for each CEj — 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() → bool

contains_owl_nothing() → bool

contains_owl_thing() → bool

named_classes() → Iterable[*owlapy.class_expression.OWLClass*]

class owlapy.owl_ontology.**OWLClassAxiom**(
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLLogicalAxiom

The base interface for class axioms.

__slots__ = ()

class owlapy.owl_ontology.**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 expression 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_ontology.**OWLDataPropertyRangeAxiom**(
    property_: owlapy.owl_property.OWLDataPropertyExpression,
    range_: owlapy.owl_datatype.OWLDataRange,
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: 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__ = ()

class owlapy.owl_ontology.**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_ontology.**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_ontology.**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*

   `__eq__`(*other*)
      Return self==value.

   `__hash__`()
      Return hash(self).

   `__repr__`()
      Return repr(self).

class owlapy.owl_ontology.**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.

         Returns
               The IRI of this object.

property str: str

>   Gets the string representation of this object

>>   **Returns**

>>   The IRI as string

class owlapy.owl_ontology.**OWLDataRange**

>   Bases: OWLPropertyRange

>   Represents a DataRange in the OWL 2 Specification.

class owlapy.owl_ontology.**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).

class owlapy.owl_ontology.**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 ≤ i ≤ 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_ontology.**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 ≤ i ≤ 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_ontology.**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 IRI as string

class owlapy.owl_ontology.**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

class owlapy.owl_ontology.**OWLIndividual**

    Bases: *owlapy.owl_object.OWLObject*

    Represents a named or anonymous individual.

    __slots__ = ()

owlapy.owl_ontology.**IntegerOWLDatatype: Final**

owlapy.owl_ontology.**DoubleOWLDatatype: Final**

owlapy.owl_ontology.**BooleanOWLDatatype: Final**

owlapy.owl_ontology.**StringOWLDatatype: Final**

owlapy.owl_ontology.**DateOWLDatatype: Final**

owlapy.owl_ontology.**DateTimeOWLDatatype: Final**

owlapy.owl_ontology.**DurationOWLDatatype: Final**

class owlapy.owl_ontology.**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)

__slots__ = ()

type_index: Final = 4008

get_literal() → 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() → bool

Whether this literal is typed as boolean.

parse_boolean() → 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() → bool

Whether this literal is typed as double.

parse_double() → 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() → bool

Whether this literal is typed as integer.

parse_integer() → 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``()` → bool

    Whether this literal is typed as string.

`parse_string``()` → 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``()` → bool

    Whether this literal is typed as date.

`parse_date``()` → 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``()` → bool

    Whether this literal is typed as dateTime.

`parse_datetime``()` → 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.

`is_duration``()` → bool

    Whether this literal is typed as duration.

`parse_duration``()` → 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``()` → bool

        Returns

            true if the annotation value is a literal

`as_literal``()` → *OWLLiteral*

        Returns

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

`to_python``()` → Literals

abstract `get_datatype``()` → *owlapy.owl_datatype.OWLDatatype*

    Gets the OWLDatatype which types this literal.

        Returns

            The OWLDatatype that types this literal.

class owlapy.owl_ontology.**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` () → bool

class owlapy.owl_ontology.**IRI** (*namespace: str | owlapy.namespaces.Namespaces*, *remainder: str*)

    Bases: *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*) → *IRI*

    static `create` (*namespace: str*, *remainder: str*) → *IRI*

    static `create` (*string: str*) → *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 <http://www.w3.org/2002/07/owl#Nothing> 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 <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

    `is_reserved_vocabulary` () → bool

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

        Returns

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

    `as_iri` () → *IRI*

        Returns

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

as_str() → 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.

get_short_form() → str

> Gets the short form.
>
> > **Returns**
> > A string that represents the short form.

get_namespace() → str

> > **Returns**
> > The namespace as string.

get_remainder() → str

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

class owlapy.owl_ontology.**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() → 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() → 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() → *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.owl_ontology.**OWLThing: Final**

class owlapy.owl_ontology.**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)

> **__slots__ = ()**

> **abstract is_owl_thing**() → 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**() → 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.

class owlapy.owl_ontology.**OWLObjectComplementOf**(*op: OWLClassExpression*)

> Bases: OWLBooleanClassExpression, *owlapy.meta_classes.HasOperands*[*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() → 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.owl_ontology.**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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Union_of_Class_Expressions)

```
__slots__ = '_operands'
```

```
type_index: Final = 3002
```

class owlapy.owl_ontology.**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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Intersection_of_Class_Expressions)

```
__slots__ = '_operands'
```

```
type_index: Final = 3001
```

class owlapy.owl_ontology.**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.owl_ontology.**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.owl_ontology.**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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals)

__slots__ = '_values'

type_index: Final = 3004

individuals() → 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() → *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}, … , {an})

__hash__()

>   Return hash(self).

__eq__(*other*)

>   Return self==value.

__repr__()

>   Return repr(self).

class owlapy.owl_ontology.**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()
>   >   → *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.owl_ontology.**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.owl_ontology.**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.owl_ontology.**OWLObjectHasValue**(
      property: [owlapy.owl_property.OWLObjectPropertyExpression](#),
      individual: [owlapy.owl_individual.OWLIndividual](#))

Bases: OWLHasValueRestriction[*[owlapy.owl_individual.OWLIndividual](#)*], *OWLObjectRestriction*

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() → *[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.owl_ontology.**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 expressions DPEi, $1 \leq i \leq n$, 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, $1 \leq i \leq n$, 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.owl_ontology.**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 ≤ i ≤ 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 ≤ i ≤ 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.owl_ontology.**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.owl_ontology.**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.owl_ontology.**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.owl_ontology.**OWLDataHasValue**(
  property: *owlapy.owl_property.OWLDataPropertyExpression*,
  value: *owlapy.owl_literal.OWLLiteral*)

Bases: OWLHasValueRestriction[*owlapy.owl_literal.OWLLiteral*], *OWLDataRestriction*

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() → *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.owl_ontology.**OWLDataOneOf**(
  values: owlapy.owl_literal.OWLLiteral | Iterable[owlapy.owl_literal.OWLLiteral])

Bases: `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 ≤ i ≤ n. The resulting data range has arity one. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Literals)

type_index: Final = 4003

values() → Iterable[*owlapy.owl_literal.OWLLiteral*]

Gets the values that are in the oneOf.

> **Returns**
>> The values of this {@code DataOneOf} class expression.

operands() → 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.owl_ontology.**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*

get_facet_restrictions() → Sequence[*OWLFacetRestriction*]

__eq__(*other*)

    Return self==value.

__hash__()

    Return hash(self).

__repr__()

    Return repr(self).

class owlapy.owl_ontology.**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() → bool

        Determines if this is a data restriction.

        Returns

            True if this is a data restriction.

    is_object_restriction() → bool

        Determines if this is an object restriction.

        Returns

            True if this is an object restriction.

class owlapy.owl_ontology.**OWLObjectRestriction**

    Bases: *OWLRestriction*

    Represents an Object Property Restriction in the OWL 2 specification.

    __slots__ = ()

    is_object_restriction() → 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.owl_ontology.**OWLDataRestriction**

    Bases: *OWLRestriction*

    Represents a Data Property Restriction.

    __slots__ = ()

is_data_restriction() → bool

> Determines if this is a data restriction.

> > **Returns**
> > True if this is a data restriction.

class owlapy.owl_ontology.**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.owl_ontology.**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

class owlapy.owl_ontology.**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**

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.

get_inverse_property() → *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() → bool

> Determines if this is the owl:topObjectProperty.

> > **Returns**
> > > topObjectProperty.

> > **Return type**
> > > True if this property is the owl

class owlapy.owl_ontology.**OWLPropertyExpression**

> Bases: *owlapy.owl_object.OWLObject*

> Represents a property or possibly the inverse of a property.

> **__slots__ = ()**

> **is_data_property_expression**() → bool

> > **Returns**
> > > True if this is a data property.

> **is_object_property_expression**() → bool

> > **Returns**
> > > True if this is an object property.

> **is_owl_top_object_property**() → bool

> > Determines if this is the owl:topObjectProperty.

> > > **Returns**
> > > > topObjectProperty.

> > > **Return type**
> > > > True if this property is the owl

> **is_owl_top_data_property**() → bool

> > Determines if this is the owl:topDataProperty.

> > > **Returns**
> > > > topDataProperty.

> > > **Return type**
> > > > True if this property is the owl

class owlapy.owl_ontology.**OWLObjectInverseOf**(*property: OWLObjectProperty*)

> Bases: *OWLObjectPropertyExpression*

> Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression Object-InverseOf(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`() → *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.

`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.

`__repr__`()

> Return repr(self).

`__eq__`(*other*)

> Return self==value.

`__hash__`()

> Return hash(self).

class owlapy.owl_ontology.**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`() → bool

> > **Returns**
> >
> > > True if this is an object property.

class owlapy.owl_ontology.**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_ontology.**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**

owlapy.owl_ontology.**logger**

class owlapy.owl_ontology.**OWLOntologyID**(*ontology_iri: owlapy.iri.IRI | None = None*, *version_iri: owlapy.iri.IRI | None = None*)

>   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**() → *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 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.

abstract 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.

abstract get_owl_ontology_manager() → _M

Gets the manager that manages this ontology.

abstract get_ontology_id() → _OWLOntologyID_

Gets the OWLOntologyID belonging to this object.

> **Returns**
> The OWLOntologyID.

is_anonymous() → bool

Check whether this ontology does contain an IRI or not.

class owlapy.owl_ontology.**Ontology**(_manager: OntologyManager_, _ontology_iri: owlapy.iri.IRI_,
load: bool)

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() → 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() → 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.

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.

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() → 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.

get_owl_ontology_manager() → *OntologyManager*

Gets the manager that manages this ontology.

get_ontology_id() → *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*)
→ 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

Attributes

OWLThing	

Classes

IRI	An IRI, consisting of a namespace and a remainder.
HasIRI	Simple class to access the IRI.
OWLObject	Base interface for OWL objects
OWLClass	An OWL 2 named Class. Classes can be understood as sets of individuals.
OWLQuantifiedDataRestriction	Represents a quantified data restriction.
OWLDataHasValue	A has-value class expression DataHasValue(DPE lt) consists of a data property expression DPE and a literal lt,
OWLNaryBooleanClassExpression	OWLNaryBooleanClassExpression.
OWLObjectOneOf	An enumeration of individuals ObjectOneOf(a1 ... an) contains exactly the individuals ai with $1 \leq i \leq n$.
OWLObjectComplementOf	Represents an ObjectComplementOf class expression in the OWL 2 Specification.
OWLObjectHasValue	A has-value class expression ObjectHasValue(OPE a) consists of an object property expression OPE and an
OWLQuantifiedObjectRestriction	Represents a quantified object restriction.
OWLObjectPropertyRangeAxiom	An object property range axiom ObjectPropertyRange(OPE CE) states that the range of the object property
OWLAxiom	Represents Axioms in the OWL 2 Specification.
OWLSubClassOfAxiom	A subclass axiom SubClassOf(CE1 CE2) states that the class expression CE1 is a subclass of the class
OWLEquivalentClassesAxiom	An equivalent classes axiom EquivalentClasses(CE1 ... CEn) states that all of the class expressions CEi,
OWLDisjointUnionAxiom	A disjoint union axiom DisjointUnion(C CE1 ... CEn) states that a class C is a disjoint union of the class
OWLAnnotationAssertionAxiom	An annotation assertion AnnotationAssertion(AP as av) states that the annotation subject as — an IRI or an
OWLAnnotationProperty	Represents an AnnotationProperty in the OWL 2 specification.
OWLSubPropertyAxiom	Base interface for object and data sub-property axioms.
OWLPropertyRangeAxiom	Base class for Property Range axioms.
OWLClassAssertionAxiom	A class assertion ClassAssertion(CE a) states that the individual a is an instance of the class expression CE.
OWLDeclarationAxiom	Represents a Declaration axiom in the OWL 2 Specification. A declaration axiom declares an entity in an ontology.
OWLObjectPropertyAssertionAxiom	A positive object property assertion ObjectPropertyAssertion(OPE a1 a2) states that the individual a1 is

Table 6 – continued from previous page

OWLSymmetricObjectPropertyAxiom	An object property symmetry axiom SymmetricObject-Property(OPE) states that
OWLTransitiveObjectPropertyAxiom	An object property transitivity axiom TransitiveObject-Property(OPE) states that the
OWLPropertyDomainAxiom	Base class for Property Domain axioms.
OWLAsymmetricObjectPropertyAxiom	An object property asymmetry axiom AsymmetricOb-jectProperty(OPE) states that
OWLDataPropertyCharacteristicAxiom	Base interface for Functional data property axiom.
OWLFunctionalDataPropertyAxiom	A data property functionality axiom FunctionalDataProp-erty(DPE) states that
OWLReflexiveObjectPropertyAxiom	An object property reflexivity axiom ReflexiveObject-Property(OPE) states that the
OWLDataPropertyAssertionAxiom	A positive data property assertion DataPropertyAsser-tion(DPE a lt) states that the individual a is connected
OWLFunctionalObjectPropertyAxiom	An object property functionality axiom FunctionalOb-jectProperty(OPE) states that
OWLObjectPropertyCharacteristicAxiom	Base interface for functional object property axiom.
OWLIrreflexiveObjectPropertyAxiom	An object property irreflexivity axiom IrreflexiveObject-Property(OPE) states that the
OWLInverseFunctionalObjectPropertyAx-iom	An object property inverse functionality axiom Inverse-FunctionalObjectProperty(OPE)
OWLDisjointDataPropertiesAxiom	A disjoint data properties axiom DisjointDataProperties(DPE1 ... DPEn) states that all of the data property
OWLDisjointObjectPropertiesAxiom	A disjoint object properties axiom DisjointObjectProp-erties(OPE1 ... OPEn) states that all of the object
OWLEquivalentDataPropertiesAxiom	An equivalent data properties axiom EquivalentDataProp-erties(DPE1 ... DPEn) states that all the data property
OWLEquivalentObjectPropertiesAxiom	An equivalent object properties axiom EquivalentObject-Properties(OPE1 ... OPEn) states that all of the object
OWLInverseObjectPropertiesAxiom	An inverse object properties axiom InverseObjectProp-erties(OPE1 OPE2) states that the object property
OWLNaryPropertyAxiom	Represents an axiom that contains two or more operands that could also be represented with
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,
OWLDisjointClassesAxiom	A disjoint classes axiom DisjointClasses(CE1 ... CEn) states that all of the class expressions CEi, $1 \leq i \leq n$,
OWLSameIndividualAxiom	An individual equality axiom SameIndividual(a1 ... an) states that all of the individuals ai, $1 \leq i \leq n$,
OWLNamedIndividual	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
OWLIndividual	Represents a named or anonymous individual.
OWLOntology	Represents an OWL 2 Ontology in the OWL 2 specifica-tion.
Ontology	Represents an OWL 2 Ontology in the OWL 2 specifica-tion.
ToOwlready2	
OWLDataProperty	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.

Table 6 – continued from previous page

OWLObjectInverseOf	Represents the inverse of a property expression (Object-InverseOf). An inverse object property expression
OWLObjectProperty	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
OWLProperty	A base class for properties that aren't expression i.e. named properties. By definition, properties
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

class owlapy.owl_ontology_manager.**IRI**(*namespace: str | owlapy.namespaces.Namespaces*,
    remainder: str)

  Bases: *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*) → *IRI*

  static create(*namespace: str*, *remainder: str*) → *IRI*

  static create(*string: str*) → *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 <http://www.w3.org/2002/07/owl#Nothing> 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 <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

is_reserved_vocabulary() → bool

> Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http://www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.
>
> > **Returns**
> >
> > True if the IRI is in the reserved vocabulary, otherwise False.

as_iri() → *IRI*

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

as_str() → 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.

get_short_form() → str

> Gets the short form.
>
> > **Returns**
> >
> > A string that represents the short form.

get_namespace() → str

> > **Returns**
> >
> > The namespace as string.

get_remainder() → str

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

class owlapy.owl_ontology_manager.**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.owl_ontology_manager.**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() → bool

owlapy.owl_ontology_manager.**OWLThing: Final**

class owlapy.owl_ontology_manager.**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() → 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() → 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() → *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.

class owlapy.owl_ontology_manager.**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 constant (data value). For quantified restriction this will be a class expression
    or a data range.

        Returns

        the value

class owlapy.owl_ontology_manager.**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() → *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.owl_ontology_manager.**OWLNaryBooleanClassExpression**(
> *operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression]*)

Bases: *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.owl_ontology_manager.**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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals)

__slots__ = '_values'

type_index: Final = 3004

individuals() → 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() → *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}, … , {an})

__hash__()

> Return hash(self).

__eq__(*other*)

> Return self==value.

__repr__()

> Return repr(self).

class owlapy.owl_ontology_manager.**OWLObjectComplementOf**(*op: OWLClassExpression*)

> Bases: OWLBooleanClassExpression, *owlapy.meta_classes.*
> *HasOperands*[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() → 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.owl_ontology_manager.**OWLObjectHasValue**(
> *property: owlapy.owl_property.OWLObjectPropertyExpression*,
> *individual: owlapy.owl_individual.OWLIndividual*)

Bases: OWLHasValueRestriction[*owlapy.owl_individual.OWLIndividual*], OWLObjectRestriction

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() → *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.owl_ontology_manager.**OWLQuantifiedObjectRestriction**(
> *filler: owlapy.class_expression.class_expression.OWLClassExpression*)

Bases: `OWLQuantifiedRestriction[`*`owlapy.class_expression.class_expression.`*
`OWLClassExpression``]`, `OWLObjectRestriction`

Represents a quantified object restriction.

__slots__ = ()

get_filler() → *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.owl_ontology_manager.**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_ontology_manager.**OWLAxiom**(
> *annotations: Iterable[OWLAnnotation] | None = 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() → List[*OWLAnnotation*] | None

is_annotated() → bool

is_logical_axiom() → bool

is_annotation_axiom() → bool

class owlapy.owl_ontology_manager.**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](#)*

__eq__(*other*)

> Return self==value.

__hash__()

> Return hash(self).

__repr__()

> Return repr(self).

class owlapy.owl_ontology_manager.**OWLEquivalentClassesAxiom**(
      class_expressions: List[[owlapy.class_expression.OWLClassExpression](#)],
      annotations: Iterable[[OWLAnnotation](#)] | None = None)

Bases: `OWLNaryClassAxiom`

An equivalent classes axiom EquivalentClasses(CE1 … CEn) states that all of the class expressions CEi, $1 \leq i \leq$ n, are semantically equivalent to each other. This axiom allows one to use each CEi as a synonym for each CEj — 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() → bool

contains_owl_nothing() → bool

contains_owl_thing() → bool

named_classes() → Iterable[*[owlapy.class_expression.OWLClass](#)*]

class owlapy.owl_ontology_manager.**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 expressions CEi, $1 \leq i \leq$ 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')`

`get_owl_class``()` → *owlapy.class_expression.OWLClass*

`get_class_expressions``()` → Iterable[*owlapy.class_expression.OWLClassExpression*]

`get_owl_equivalent_classes_axiom``()` → *OWLEquivalentClassesAxiom*

`get_owl_disjoint_classes_axiom``()` → *OWLDisjointClassesAxiom*

`__eq__``(other)`

> Return self==value.

`__hash__``()`

> Return hash(self).

`__repr__``()`

> Return repr(self).

`class` owlapy.owl_ontology_manager.**`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')`

`get_subject``()` → *owlapy.owl_annotation.OWLAnnotationSubject*

> Gets the subject of this object.
>
> > **Returns**
> > The subject.

`get_property``()` → *OWLAnnotationProperty*

> Gets the property.
>
> > **Returns**
> > The property.

`get_value``()` → *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_ontology_manager.**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.

            Returns

                The IRI of this object.

    property str: str

        Gets the string representation of this object

            Returns

                The IRI as string

class owlapy.owl_ontology_manager.**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')

    get_sub_property() → _P

    get_super_property() → _P

    __eq__(*other*)

        Return self==value.

    __hash__()

        Return hash(self).

    __repr__()

        Return repr(self).

class owlapy.owl_ontology_manager.**OWLPropertyRangeAxiom**(*property_: _P*, *range_: _R*, *annotations: Iterable[OWLAnnotation] | None = None*)

    Bases: Generic[_P, _R], OWLUnaryPropertyAxiom[_P]

    Base class for Property Range axioms.

    __slots__ = '_range'

    get_range() → _R

    __eq__(*other*)

        Return self==value.

    __hash__()

        Return hash(self).

    __repr__()

        Return repr(self).

class owlapy.owl_ontology_manager.**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]()*

get_class_expression() → *[owlapy.class_expression.OWLClassExpression]()*

__eq__(*other*)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.owl_ontology_manager.**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]()*

__eq__(*other*)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.owl_ontology_manager.**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_ontology_manager.**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_ontology_manager.**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 expressionOPE 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_ontology_manager.**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*

    __eq__(*other*)

      Return self==value.

    __hash__()

      Return hash(self).

    __repr__()

      Return repr(self).

class owlapy.owl_ontology_manager.**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_ontology_manager.**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_ontology_manager.**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_ontology_manager.**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_ontology_manager.**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_ontology_manager.**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_ontology_manager.**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_ontology_manager.**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_ontology_manager.**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 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_ontology_manager.**OWLDisjointDataPropertiesAxiom**(
    properties: List[owlapy.owl_property.OWLDataPropertyExpression],
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLNaryPropertyAxiom[owlapy.owl_property.OWLDataPropertyExpression]*,
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 $i \ne j$.
>
> (https://www.w3.org/TR/owl2-syntax/#Disjoint_Data_Properties)

__slots__ = ()

class owlapy.owl_ontology_manager.**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 OPEj for $i \ne j$.

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

__slots__ = ()

class owlapy.owl_ontology_manager.**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 DPEj — 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_ontology_manager.**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 OPEj — 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_ontology_manager.**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_ontology_manager.**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() → Iterable[_P]
        Get all the properties that appear in the axiom.

            Returns
                Generator containing the properties.

    as_pairwise_axioms() → Iterable[*OWLNaryPropertyAxiom*]

    __eq__(*other*)
        Return self==value.

    __hash__()
        Return hash(self).

    __repr__()
        Return repr(self).

class owlapy.owl_ontology_manager.**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() → Iterable[*OWLNaryIndividualAxiom*]

__eq__(*other*)

> Return self==value.

__hash__()

> Return hash(self).

__repr__()

> Return repr(self).

class owlapy.owl_ontology_manager.**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 ≤ i ≤ n, are different from each other; that is, no individuals ai and aj with i ≠ 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_ontology_manager.**OWLDisjointClassesAxiom**(
> *class_expressions: List[owlapy.class_expression.OWLClassExpression]*,
> *annotations: Iterable[OWLAnnotation] | None = None*)

> Bases: OWLNaryClassAxiom

> A disjoint classes axiom DisjointClasses(CE1 … CEn) states that all of the class expressions CEi, 1 ≤ i ≤ 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_ontology_manager.**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 ≤ i ≤ n, are equal to each other. This axiom allows one to use each ai as a synonym for each aj — 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_ontology_manager.**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

`class` owlapy.owl_ontology_manager.**`OWLIndividual`**

>   Bases: *owlapy.owl_object.OWLObject*

>   Represents a named or anonymous individual.

>   **`__slots__ = ()`**

`class` owlapy.owl_ontology_manager.**`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.

abstract 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.

abstract get_owl_ontology_manager() → _M

    Gets the manager that manages this ontology.

abstract get_ontology_id() → *OWLOntologyID*

    Gets the OWLOntologyID belonging to this object.

> **Returns**
>   The OWLOntologyID.

is_anonymous() → bool

    Check whether this ontology does contain an IRI or not.

class owlapy.owl_ontology_manager.**Ontology**(*manager: OntologyManager,*
    ontology_iri: owlapy.iri.IRI, load: bool)

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() → 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() → 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.

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.

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() → 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.

get_owl_ontology_manager() → *OntologyManager*

 Gets the manager that manages this ontology.

get_ontology_id() → *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).

class owlapy.owl_ontology_manager.**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*)
→ owlready2.ClassConstruct | type

Map owlapy data range classes.

class owlapy.owl_ontology_manager.**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

class owlapy.owl_ontology_manager.**OWLObjectInverseOf**(*property: OWLObjectProperty*)

Bases: `OWLObjectPropertyExpression`

Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression Object-InverseOf(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() → *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.

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.

__repr__()

Return repr(self).

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

class owlapy.owl_ontology_manager.**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

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.

get_inverse_property() → *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() → bool

> Determines if this is the owl:topObjectProperty.

> > **Returns**
> > topObjectProperty.

> > **Return type**
> > True if this property is the owl

class owlapy.owl_ontology_manager.**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_ontology_manager.**OWLOntologyChange**(
> > *ontology: owlapy.owl_ontology.OWLOntology*)

> 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.

abstract `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.

abstract `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.

abstract `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.

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

class owlapy.owl_ontology_manager.**AddImport**(*ontology: owlapy.owl_ontology.OWLOntology*, *import_declaration: OWLImportsDeclaration*)

> 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*) → *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

[*OWLObject*](#)	Base interface for OWL objects
[*OWLEntity*](#)	Represents Entities in the OWL 2 Specification.
[*IRI*](#)	An IRI, consisting of a namespace and a remainder.
[*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 (ObjectInverseOf). 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.**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**() → bool

class owlapy.owl_property.**OWLEntity**

> Bases: OWLNamedObject
>
> Represents Entities in the OWL 2 Specification.
>
> **__slots__ = ()**
>
> **to_string_id**() → str
>
> **is_anonymous**() → bool

class owlapy.owl_property.**IRI**(*namespace: str | [owlapy.namespaces.Namespaces](#), remainder: str*)

> Bases: [*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*) → *IRI*

static create(*namespace: str*, *remainder: str*) → *IRI*

static create(*string: str*) → *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 <http://www.w3.org/2002/07/owl#Nothing> 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 <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

is_reserved_vocabulary() → bool

> Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http://www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.
>
> > **Returns**
> >
> > > True if the IRI is in the reserved vocabulary, otherwise False.

as_iri() → *IRI*

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

as_str() → 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.

get_short_form() → str

> Gets the short form.
>
> > **Returns**
> >
> > > A string that represents the short form.

get_namespace() → str

> **Returns**
>> The namespace as string.

get_remainder() → str

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

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**() → bool

>> **Returns**
>>> True if this is a data property.

> **is_object_property_expression**() → bool

>> **Returns**
>>> True if this is an object property.

> **is_owl_top_object_property**() → bool

>> Determines if this is the owl:topObjectProperty.

>> **Returns**
>>> topObjectProperty.

>> **Return type**
>>> True if this property is the owl

> **is_owl_top_data_property**() → 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() → 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**

> **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.

> **get_inverse_property**() → *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() → 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 Object-InverseOf(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**() → *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.

> **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.

> **__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

OWLClassExpression	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
OWLObjectSomeValuesFrom	An existential class expression ObjectSomeValuesFrom(OPE CE) consists of an object property expression OPE and
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
OWLObjectComplementOf	Represents an ObjectComplementOf class expression in the OWL 2 Specification.
OWLObjectAllValuesFrom	A universal class expression ObjectAllValuesFrom(OPE CE) consists of an object property expression OPE and a
OWLObjectOneOf	An enumeration of individuals ObjectOneOf(a1 ... an) contains exactly the individuals ai with $1 \leq i \leq n$.
OWLObjectHasValue	A has-value class expression ObjectHasValue(OPE a) consists of an object property expression OPE and an
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

Table 7 – continued from previous page

OWLObjectExactCardinality	An exact cardinality expression ObjectExactCardinality(n OPE CE) consists of a nonnegative integer n, an object
OWLObjectCardinalityRestriction	Represents Object Property Cardinality Restrictions in the OWL 2 specification.
OWLDataSomeValuesFrom	An existential class expression DataSomeValuesFrom(DPE1 ... DPEn DR) consists of n data property expressions
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.
OWLDataHasValue	A has-value class expression DataHasValue(DPE lt) consists of a data property expression DPE and a literal lt,
OWLDataAllValuesFrom	A universal class expression DataAllValuesFrom(DPE1 ... DPEn DR) consists of n data property expressions DPEi,
OWLClass	An OWL 2 named Class. Classes can be understood as sets of individuals.
IRI	An IRI, consisting of a namespace and a remainder.
OWLAxiom	Represents Axioms in the OWL 2 Specification.
OWLSubClassOfAxiom	A subclass axiom SubClassOf(CE1 CE2) states that the class expression CE1 is a subclass of the class
OWLDataRange	Represents a DataRange in the OWL 2 Specification.
OWLDataComplementOf	A complement data range DataComplementOf(DR) contains all tuples of literals that are not contained in the
OWLDataUnionOf	A union data range DataUnionOf(DR1 ... DRn) contains all tuples of literals that are contained in the at least
OWLDataIntersectionOf	An intersection data range DataIntersectionOf(DR1 ... DRn) contains all tuples of literals that are contained
OWLDatatype	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
OWLOntology	Represents an OWL 2 Ontology in the OWL 2 specification.
Ontology	Represents an OWL 2 Ontology in the OWL 2 specification.
ToOwlready2	
OntologyManager	An OWLOntologyManager manages a set of ontologies. It is the main point for creating, loading and accessing
OWLObjectPropertyExpression	A high level interface to describe different types of object properties.
OWLDataProperty	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
OWLObjectProperty	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
OWLObjectInverseOf	Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression
OWLPropertyExpression	Represents a property or possibly the inverse of a property.
OWLDataPropertyExpression	A high level interface to describe different types of data properties.

Table 7 – continued from previous page

OWLNamedIndividual	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
OWLLiteral	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
LRUCache	Constants shares by all lru cache instances.
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

class owlapy.owl_reasoner.**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)

__slots__ = ()

abstract is_owl_thing() → 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() → 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.

class owlapy.owl_reasoner.**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.owl_reasoner.`**`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.owl_reasoner.`**`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.owl_reasoner.`**`OWLObjectComplementOf`**(*op: OWLClassExpression*)
  Bases: `OWLBooleanClassExpression`, *[owlapy.meta_classes.](owlapy.meta_classes.HasOperands)*
  [HasOperands](owlapy.meta_classes.HasOperands)[*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() → 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.owl_reasoner.**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.owl_reasoner.**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() → 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() → *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}, … , {an})

__hash__()

> Return hash(self).

__eq__(*other*)

> Return self==value.

__repr__()

> Return repr(self).

class owlapy.owl_reasoner.**OWLObjectHasValue**(
> *property: owlapy.owl_property.OWLObjectPropertyExpression,*
> *individual: owlapy.owl_individual.OWLIndividual*)

Bases: `OWLHasValueRestriction[`*`owlapy.owl_individual.OWLIndividual`*`]`, `OWLObjectRestriction`

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() → *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.owl_reasoner.**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.owl_reasoner.**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.owl_reasoner.**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()
        → *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.owl_reasoner.**OWLObjectCardinalityRestriction**(*cardinality: int*,
       property: owlapy.owl_property.OWLObjectPropertyExpression,
       filler: owlapy.class_expression.class_expression.OWLClassExpression)

    Bases: 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.owl_reasoner.**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 expressions DPEi, $1 \leq i \leq n$, 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, $1 \leq i \leq n$, 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.owl_reasoner.**OWLDataOneOf**(
> *values: owlapy.owl_literal.OWLLiteral | Iterable[owlapy.owl_literal.OWLLiteral]*)

Bases: *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 $1 \leq i \leq n$. The resulting data range has arity one. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Literals)

```
type_index: Final = 4003
```

values() → Iterable[*owlapy.owl_literal.OWLLiteral*]
> Gets the values that are in the oneOf.

> **Returns**
>> The values of this {@code DataOneOf} class expression.

operands() → 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.owl_reasoner.**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*

get_facet_restrictions() → Sequence[*OWLFacetRestriction*]

__eq__(*other*)

    Return self==value.

__hash__()

    Return hash(self).

__repr__()

    Return repr(self).

class owlapy.owl_reasoner.**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.owl_reasoner.**OWLDataHasValue**(
>   *property: owlapy.owl_property.OWLDataPropertyExpression*,
>   *value: owlapy.owl_literal.OWLLiteral*)

Bases: `OWLHasValueRestriction`[*owlapy.owl_literal.OWLLiteral*], `OWLDataRestriction`

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`() → *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.owl_reasoner.**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 ≤ i ≤ 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 ≤ i ≤ 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.owl_reasoner.**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**() → 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**() → 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.

205

get_nnf() → *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.

class owlapy.owl_reasoner.**IRI**(*namespace: str | owlapy.namespaces.Namespaces*, *remainder: str*)

    Bases: *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*) → *IRI*

    static create(*namespace: str*, *remainder: str*) → *IRI*

    static create(*string: str*) → *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 <http://www.w3.org/2002/07/owl#Nothing> 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 <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

    is_reserved_vocabulary() → bool

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

        Returns

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

    as_iri() → *IRI*

        Returns

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

    as_str() → 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.

get_short_form() → str

>   Gets the short form.

>   >   **Returns**
>   >   A string that represents the short form.

get_namespace() → str

>   >   **Returns**
>   >   The namespace as string.

get_remainder() → str

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

class owlapy.owl_reasoner.**OWLAxiom**(*annotations: Iterable[OWLAnnotation] | None = 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() → List[*OWLAnnotation*] | None

is_annotated() → bool

is_logical_axiom() → bool

is_annotation_axiom() → bool

class owlapy.owl_reasoner.**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*

__eq__(*other*)

>   Return self==value.

__hash__()

>   Return hash(self).

__repr__()

> Return repr(self).

class owlapy.owl_reasoner.**OWLDataRange**

> Bases: `OWLPropertyRange`
>
> Represents a DataRange in the OWL 2 Specification.

class owlapy.owl_reasoner.**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).

class owlapy.owl_reasoner.**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 ≤ i ≤ 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_reasoner.**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 ≤ i ≤ 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_reasoner.**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 IRI as string

class owlapy.owl_reasoner.**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.

abstract 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.

abstract get_owl_ontology_manager() → _M

    Gets the manager that manages this ontology.

abstract get_ontology_id() → *OWLOntologyID*

    Gets the OWLOntologyID belonging to this object.

        Returns

           The OWLOntologyID.

is_anonymous() → bool

    Check whether this ontology does contain an IRI or not.

class owlapy.owl_reasoner.**Ontology**(*manager: OntologyManager*, *ontology_iri: owlapy.iri.IRI*,
      load: bool)

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() → 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() → 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.

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.

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() → 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.

get_owl_ontology_manager() → *OntologyManager*

Gets the manager that manages this ontology.

get_ontology_id() → *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).

class owlapy.owl_reasoner.**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*)
            → owlready2.ClassConstruct | type

        Map owlapy data range classes.

class owlapy.owl_reasoner.**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*) → *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.

class owlapy.owl_reasoner.**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() → bool

> > **Returns**
> > > True if this is an object property.

class owlapy.owl_reasoner.**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

class owlapy.owl_reasoner.**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**

> **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.

> **get_inverse_property**() → *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**() → bool

> > Determines if this is the owl:topObjectProperty.

> > > **Returns**
> > > > topObjectProperty.

> > > **Return type**
> > > > True if this property is the owl

class owlapy.owl_reasoner.**OWLObjectInverseOf**(*property: OWLObjectProperty*)

    Bases: *OWLObjectPropertyExpression*

    Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression Object-InverseOf(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() → *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.

    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.

    __repr__()

        Return repr(self).

    __eq__(*other*)

        Return self==value.

    __hash__()

        Return hash(self).

class owlapy.owl_reasoner.**OWLPropertyExpression**

    Bases: *owlapy.owl_object.OWLObject*

    Represents a property or possibly the inverse of a property.

    __slots__ = ()

    is_data_property_expression() → bool

        Returns

            True if this is a data property.

    is_object_property_expression() → bool

        Returns

            True if this is an object property.

is_owl_top_object_property() → bool

> Determines if this is the owl:topObjectProperty.
>
> > **Returns**
> > topObjectProperty.
> >
> > **Return type**
> > True if this property is the owl

is_owl_top_data_property() → bool

> Determines if this is the owl:topDataProperty.
>
> > **Returns**
> > topDataProperty.
> >
> > **Return type**
> > True if this property is the owl

class owlapy.owl_reasoner.**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_reasoner.**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

class owlapy.owl_reasoner.**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)

`__slots__ = ()`

`type_index: Final = 4008`

`get_literal``() →` 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``() →` bool

    Whether this literal is typed as boolean.

`parse_boolean``() →` 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``() →` bool

    Whether this literal is typed as double.

`parse_double``() →` 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``() →` bool

    Whether this literal is typed as integer.

`parse_integer``() →` 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``() →` bool

    Whether this literal is typed as string.

`parse_string``() →` 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``() →` bool

    Whether this literal is typed as date.

`parse_date``() →` 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() → bool

    Whether this literal is typed as dateTime.

parse_datetime() → 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.

is_duration() → bool

    Whether this literal is typed as duration.

parse_duration() → 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() → bool

        Returns

            true if the annotation value is a literal

as_literal() → *OWLLiteral*

        Returns

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

to_python() → Literals

abstract get_datatype() → *owlapy.owl_datatype.OWLDatatype*

    Gets the OWLDatatype which types this literal.

        Returns

            The OWLDatatype that types this literal.

class owlapy.owl_reasoner.**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*) → bool

__getitem__(*item: _K*) → _V

__setitem__(*key: _K*, *value: _V*)

cache_info()

> Report cache statistics.

cache_clear()

> Clear the cache and cache statistics.

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) → 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.

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 ObjectComplementOf(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 EquivalentDataProperties(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*)
> → 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).

abstract object_property_values (*ind:* *owlapy.owl_individual.OWLNamedIndividual*,
   pe: *owlapy.owl_property.OWLObjectPropertyExpression*, *direct: bool = True*)
      → 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).

abstract flush() → 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) → 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 sub-properties (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.

abstract 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).

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) → 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

data_property_ranges(*pe: owlapy.owl_property.OWLDataProperty*, *direct: bool = False*) → Iterable[*owlapy.owl_data_ranges.OWLDataRange*]

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.

ind_data_properties(*ind: [owlapy.owl_individual.OWLNamedIndividual](), direct: bool = True*)
> → Iterable[*[owlapy.owl_property.OWLDataProperty]()*]

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.

ind_object_properties(*ind: [owlapy.owl_individual.OWLNamedIndividual](), direct: bool = True*)
> → Iterable[*[owlapy.owl_property.OWLObjectProperty]()*]

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.

class owlapy.owl_reasoner.**OntologyReasoner**(*ontology: [owlapy.owl_ontology.Ontology](), isolate: bool = False*)

Bases: *[OWLReasonerEx]()*

Extra convenience methods for OWL Reasoners

__slots__ = ('_ontology', '_world')

update_isolated_ontology(*axioms_to_add: List[[owlapy.owl_axiom.OWLAxiom]()] = None, axioms_to_remove: List[[owlapy.owl_axiom.OWLAxiom]()] = None*)

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.

227

- **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).

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.

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*) → 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).

flush () → 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*) → 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*) → 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)).

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.

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.

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.

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 sub-properties (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.

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).

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.

class owlapy.owl_reasoner.**FastInstanceCheckerReasoner**(
ontology: [owlapy.owl_ontology.OWLOntology](#), base_reasoner: [OWLReasoner](#), *,
property_cache: bool = True, negation_default: bool = True, sub_properties: bool = False*)

Bases: *[OWLReasonerEx](#)*

Tries to check instances fast (but maybe incomplete).

**__slots__ = ('_ontology', '_base_reasoner', '_ind_set', '_cls_to_ind',
'_has_prop',...**

reset()

> The reset method shall reset any cached state.

is_isolated()

> Return True if this reasoner is using an isolated ontology.

is_using_triplestore()

data_property_domains(*pe: [owlapy.owl_property.OWLDataProperty](#), direct: bool = False*)
$\rightarrow$ 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 su-
> per_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).

data_property_ranges(*pe: [owlapy.owl_property.OWLDataProperty](#), direct: bool = False*)
$\rightarrow$ Iterable[*[owlapy.owl_data_ranges.OWLDataRange](#)*]

> 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:

object_property_domains(*pe: [owlapy.owl_property.OWLObjectProperty](#), direct: bool = False*)
$\rightarrow$ 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.

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)).

different_individuals(*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).

same_individuals(*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.

object_property_values(*ind: owlapy.owl_individual.OWLNamedIndividual*,
pe: owlapy.owl_property.OWLObjectPropertyExpression, *direct: bool = True*)
→ 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() → 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) → 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) → 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.

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 Equivalen-tObjectProperties(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)).

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.

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.

super_object_properties(*op: owlapy.owl_property.OWLObjectProperty*, *direct: bool = False*)
        → Iterable[*owlapy.owl_property.OWLDataProperty*]

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.

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 sub-properties (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.

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.

class owlapy.owl_reasoner.**SyncReasoner**(*ontology: [owlapy.owl_ontology.Ontology](),*
> *base_reasoner: [BaseReasoner]() | None = None, infer_property_values: bool = True,*
> *infer_data_property_values: bool = True, isolate: bool = False*)

Bases: `OntologyReasoner`

Extra convenience methods for OWL Reasoners

__slots__ = ('_cnt', '_conv', '_base_reasoner')

update_isolated_ontology(*axioms_to_add: List[[owlapy.owl_axiom.OWLAxiom]()] = None,*
> *axioms_to_remove: List[[owlapy.owl_axiom.OWLAxiom]()] = None*)

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.

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.

__del__()

owlapy.parser

String to OWL parsers.

Attributes

IntegerOWLDatatype
BooleanOWLDatatype
DoubleOWLDatatype
StringOWLDatatype
DateOWLDatatype
DateTimeOWLDatatype
DurationOWLDatatype
MANCHESTER_GRAMMAR
DL_GRAMMAR
DLparser
ManchesterParser

Classes

IRI	An IRI, consisting of a namespace and a remainder.
OWLNamedIndividual	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
OWLLiteral	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
OWLObjectPropertyExpression	A high level interface to describe different types of object properties.
OWLObjectProperty	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.

Table 8 – continued from previous page

OWLDataProperty	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
OWLObjectParser	Abstract class with a parse method to parse a string to an OWL Object.
Namespaces	Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup
OWLDatatype	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
OWLFacet	Enumerations for OWL facets.
OWLRDFVocabulary	Enumerations for OWL/RDF vocabulary.
OWLObjectHasSelf	A self-restriction ObjectHasSelf(OPE) consists of an object property expression OPE,
OWLObjectIntersectionOf	An intersection class expression ObjectIntersectionOf(CE1 ... CEn) contains all individuals that are instances
OWLObjectMinCardinality	A minimum cardinality expression ObjectMinCardinality(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
OWLObjectUnionOf	A union class expression ObjectUnionOf(CE1 ... CEn) contains all individuals that are instances
OWLClass	An OWL 2 named Class. Classes can be understood as sets of individuals.
OWLObjectOneOf	An enumeration of individuals ObjectOneOf(a1 ... an) contains exactly the individuals ai with $1 \leq i \leq n$.
OWLClassExpression	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
OWLObjectComplementOf	Represents an ObjectComplementOf class expression in the OWL 2 Specification.
OWLObjectExactCardinality	An exact cardinality expression ObjectExactCardinality(n OPE CE) consists of a nonnegative integer n, an object
OWLQuantifiedDataRestriction	Represents a quantified data restriction.
OWLQuantifiedObjectRestriction	Represents a quantified object restriction.
OWLFacetRestriction	A facet restriction is used to restrict a particular datatype.
OWLDataSomeValuesFrom	An existential class expression DataSomeValuesFrom(DPE1 ... DPEn DR) consists of n data property expressions
OWLDataExactCardinality	An exact cardinality expression ObjectExactCardinality(n OPE CE) consists of a nonnegative integer n, an
OWLObjectHasValue	A has-value class expression ObjectHasValue(OPE a) consists of an object property expression OPE and an
OWLDataMaxCardinality	A maximum cardinality expression ObjectMaxCardinality(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
OWLDataMinCardinality	A minimum cardinality expression DataMinCardinality(n DPE DR) consists of a nonnegative integer n, a data

Table 8 – continued from previous page

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
OWLObjectCardinalityRestriction	Represents Object Property Cardinality Restrictions in the OWL 2 specification.
OWLDatatypeRestriction	A datatype restriction DatatypeRestriction(DT F1 lt1 ... Fn ltn) consists of a unary datatype DT and n pairs
OWLDataCardinalityRestriction	Represents Data Property Cardinality Restrictions.
OWLObjectAllValuesFrom	A universal class expression ObjectAllValuesFrom(OPE CE) consists of an object property expression OPE and a
OWLDataAllValuesFrom	A universal class expression DataAllValuesFrom(DPE1 ... DPEn DR) consists of n data property expressions DPEi,
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
OWLDataRange	Represents a DataRange in the OWL 2 Specification.
ManchesterOWLSyntaxParser	Manchester Syntax parser to parse strings to OWLClassExpressions.
DLSyntaxParser	Description Logic Syntax parser to parse strings to OWLClassExpressions.

Functions

dl_to_owl_expression(dl_expression, namespace)	
manchester_to_owl_expression(manchester_ex ...)	

Module Contents

class owlapy.parser.**IRI** (*namespace: str | owlapy.namespaces.Namespaces*, *remainder: str*)

Bases: *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*) → *IRI*

static create (*namespace: str*, *remainder: str*) → *IRI*

static create (*string: str*) → *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 <http://www.w3.org/2002/07/owl#Nothing> 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 <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

`is_reserved_vocabulary``()` → bool

> Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http://www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.
>
> > **Returns**
> >
> > > True if the IRI is in the reserved vocabulary, otherwise False.

`as_iri``()` → *IRI*

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

`as_str``()` → 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.

`get_short_form``()` → str

> Gets the short form.
>
> > **Returns**
> >
> > > A string that represents the short form.

`get_namespace``()` → str

> > **Returns**
> >
> > > The namespace as string.

`get_remainder``()` → str

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

class owlapy.parser.**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.parser.**IntegerOWLDatatype: Final**

owlapy.parser.**BooleanOWLDatatype: Final**

owlapy.parser.**DoubleOWLDatatype: Final**

owlapy.parser.**StringOWLDatatype: Final**

owlapy.parser.**DateOWLDatatype: Final**

owlapy.parser.**DateTimeOWLDatatype: Final**

owlapy.parser.**DurationOWLDatatype: Final**

class owlapy.parser.**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)

> **__slots__ = ()**

> **type_index: Final = 4008**

> **get_literal**() → 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**() → bool
>> Whether this literal is typed as boolean.

parse_boolean() → 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() → bool

> Whether this literal is typed as double.

parse_double() → 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() → bool

> Whether this literal is typed as integer.

parse_integer() → 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() → bool

> Whether this literal is typed as string.

parse_string() → 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() → bool

> Whether this literal is typed as date.

parse_date() → 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() → bool

> Whether this literal is typed as dateTime.

parse_datetime() → 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.

is_duration() → bool

> Whether this literal is typed as duration.

parse_duration() → 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() → bool

> > **Returns**
> >
> > > true if the annotation value is a literal

as_literal() → *OWLLiteral*

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

to_python() → Literals

abstract get_datatype() → *owlapy.owl_datatype.OWLDatatype*

> Gets the OWLDatatype which types this literal.
>
> > **Returns**
> >
> > > The OWLDatatype that types this literal.

class owlapy.parser.**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**() → bool
>
> > > **Returns**
> > >
> > > > True if this is an object property.

class owlapy.parser.**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**

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.

get_inverse_property() → *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() → bool

    Determines if this is the owl:topObjectProperty.

        Returns

            topObjectProperty.

        Return type

            True if this property is the owl

class owlapy.parser.**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

class owlapy.parser.**OWLObjectParser**

    Abstract class with a parse method to parse a string to an OWL Object.

    abstract parse_expression(*expression_str: str*) → *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.parser.**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.

class owlapy.parser.**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 IRI as string

class owlapy.parser.**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.parser.**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.parser.**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.parser.**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 $1 \leq i \leq n$. (https://www.w3.org/TR/owl2-syntax/#Intersection_of_Class_Expressions)

    __slots__ = '_operands'

    type_index: Final = 3001

class owlapy.parser.**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.parser.**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.parser.**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 $1 \leq i \leq n$. (https://www.w3.org/TR/owl2-syntax/#Union_of_Class_Expressions)

```
__slots__ = '_operands'
```

```
type_index: Final = 3002
```

class owlapy.parser.**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() → 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() → 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() → *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.

class owlapy.parser.**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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals)

    __slots__ = '_values'

    type_index: Final = 3004

    individuals() → 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() → *[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}, … , {an})

    __hash__()

        Return hash(self).

    __eq__(*other*)

        Return self==value.

    __repr__()

        Return repr(self).

class owlapy.parser.**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)

    __slots__ = ()

    abstract is_owl_thing() → 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**() → 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.

class owlapy.parser.**OWLObjectComplementOf**(*op: OWLClassExpression*)

>   Bases: OWLBooleanClassExpression,   *owlapy.meta_classes.*
>   *HasOperands*[*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**() → 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.parser.**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()
    → *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.parser.**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 constant (data value). For quantified restriction this will be a class expression
       or a data range.

       Returns
           the value

class owlapy.parser.**OWLQuantifiedObjectRestriction**(
    filler: owlapy.class_expression.class_expression.OWLClassExpression)

   Bases: OWLQuantifiedRestriction[*owlapy.class_expression.class_expression.*
   OWLClassExpression], OWLObjectRestriction

   Represents a quantified object restriction.

   __slots__ = ()

   get_filler() → *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.parser.**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.parser.**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 expressions DPEi, $1 \leq i \leq n$, 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, $1 \leq i \leq n$, 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.parser.**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.parser.**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() → *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.parser.**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.parser.**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.parser.**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.parser.**OWLDataHasValue**(
         property: owlapy.owl_property.OWLDataPropertyExpression,
         value: owlapy.owl_literal.OWLLiteral)

Bases: OWLHasValueRestriction[*owlapy.owl_literal.OWLLiteral*], OWLDataRestriction

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() → *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.parser.**OWLDataOneOf**(
         values: owlapy.owl_literal.OWLLiteral | Iterable[owlapy.owl_literal.OWLLiteral])

Bases: *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 ≤ i ≤ n. The resulting data range has arity one. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Literals)

type_index: Final = 4003

values() → Iterable[*owlapy.owl_literal.OWLLiteral*]

> Gets the values that are in the oneOf.

> > **Returns**
> > > The values of this {@code DataOneOf} class expression.

operands() → 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.parser.**OWLObjectCardinalityRestriction**(*cardinality: int*,
property: owlapy.owl_property.OWLObjectPropertyExpression,
filler: owlapy.class_expression.class_expression.OWLClassExpression)

Bases: 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.parser.**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*

get_facet_restrictions() → Sequence[*OWLFacetRestriction*]

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

class owlapy.parser.**OWLDataCardinalityRestriction**(*cardinality: int*,
property: owlapy.owl_property.OWLDataPropertyExpression,
filler: owlapy.owl_data_ranges.OWLDataRange)

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).

class owlapy.parser.**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.parser.**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 ≤ i ≤ 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 ≤ i ≤ 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.parser.**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 ≤ i ≤ 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.parser.**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 ≤ i ≤ 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.parser.**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).

class owlapy.parser.**OWLDataRange**

Bases: OWLPropertyRange

Represents a DataRange in the OWL 2 Specification.

owlapy.parser.**MANCHESTER_GRAMMAR**

class owlapy.parser.**ManchesterOWLSyntaxParser**(
>> *namespace: str | owlapy.namespaces.Namespaces | None = None*, *grammar=None*)

Bases: parsimonious.nodes.NodeVisitor, *owlapy.owl_object.OWLObjectParser*

Manchester Syntax parser to parse strings to OWLClassExpressions. Following: https://www.w3.org/TR/owl2-manchester-syntax.

slots = ('ns', 'grammar')

ns: str | owlapy.namespaces.Namespaces | None

parse_expression(*expression_str: str*) → *owlapy.class_expression.OWLClassExpression*

> Parse a string to an OWL Object.

>> **Parameters**
>>> **expression_str** (*str*) – Expression string.

>> **Returns**
>>> The OWL Object which is represented by the string.

visit_union(*node*, *children*) → *owlapy.class_expression.OWLClassExpression*

visit_intersection(*node*, *children*) → *owlapy.class_expression.OWLClassExpression*

visit_primary(*node*, *children*) → *owlapy.class_expression.OWLClassExpression*

visit_some_only_res(*node*, *children*) → *owlapy.class_expression.OWLQuantifiedObjectRestriction*

visit_cardinality_res(*node*, *children*)
→ *owlapy.class_expression.OWLObjectCardinalityRestriction*

visit_value_res(*node*, *children*) → *owlapy.class_expression.OWLObjectHasValue*

visit_has_self(*node*, *children*) → *owlapy.class_expression.OWLObjectHasSelf*

visit_object_property(*node*, *children*) → *owlapy.owl_property.OWLObjectPropertyExpression*

visit_class_expression(*node*, *children*) → *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*) → *owlapy.class_expression.OWLDataHasValue*

visit_data_union(*node*, *children*) → *owlapy.owl_data_ranges.OWLDataRange*

visit_data_intersection(*node*, *children*) → *owlapy.owl_data_ranges.OWLDataRange*

visit_literal_list(*node*, *children*) → *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*) → *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*) → str

visit_float_literal(*node*, *children*) → *owlapy.owl_literal.OWLLiteral*

visit_decimal_literal(*node*, *children*) → *owlapy.owl_literal.OWLLiteral*

visit_integer_literal(*node*, *children*) → *owlapy.owl_literal.OWLLiteral*

visit_boolean_literal(*node*, *children*) → *owlapy.owl_literal.OWLLiteral*

visit_datetime_literal(*node*, *children*) → *owlapy.owl_literal.OWLLiteral*

visit_duration_literal(*node*, *children*) → *owlapy.owl_literal.OWLLiteral*

visit_date_literal(*node*, *children*) → *[owlapy.owl_literal.OWLLiteral](#)*

visit_non_negative_integer(*node*, *children*) → int

visit_datatype_iri(*node*, *children*) → str

visit_datatype(*node*, *children*) → *[owlapy.owl_datatype.OWLDatatype](#)*

visit_facet(*node*, *children*) → *[owlapy.vocab.OWLFacet](#)*

visit_class_iri(*node*, *children*) → *[owlapy.class_expression.OWLClass](#)*

visit_individual_iri(*node*, *children*) → *[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*) → *[owlapy.iri.IRI](#)*

visit_full_iri(*node*, *children*) → *[owlapy.iri.IRI](#)*

abstract visit_abbreviated_iri(*node*, *children*)

visit_simple_iri(*node*, *children*) → *[owlapy.iri.IRI](#)*

visit_parentheses(*node*, *children*) → *[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.**DL_GRAMMAR**

class owlapy.parser.**DLSyntaxParser**(
    namespace: str | [owlapy.namespaces.Namespaces](#) | None = None, *grammar=None*)

> Bases: parsimonious.nodes.NodeVisitor, *[owlapy.owl_object.OWLObjectParser](#)*
>
> Description Logic Syntax parser to parse strings to OWLClassExpressions.
>
> **slots = ('ns', 'grammar')**
>
> **ns: str | [owlapy.namespaces.Namespaces](#) | None**
>
> **parse_expression**(*expression_str: str*) → *[owlapy.class_expression.OWLClassExpression](#)*
>
> > Parse a string to an OWL Object.
> >
> > > **Parameters**
> > > **expression_str** (*str*) – Expression string.
> > >
> > > **Returns**
> > > The OWL Object which is represented by the string.
>
> **visit_union**(*node*, *children*) → *[owlapy.class_expression.OWLClassExpression](#)*

visit_intersection(*node*, *children*) → *owlapy.class_expression.OWLClassExpression*

visit_primary(*node*, *children*) → *owlapy.class_expression.OWLClassExpression*

visit_some_only_res(*node*, *children*) → *owlapy.class_expression.OWLQuantifiedObjectRestriction*

visit_cardinality_res(*node*, *children*)
→ *owlapy.class_expression.OWLObjectCardinalityRestriction*

visit_value_res(*node*, *children*) → *owlapy.class_expression.OWLObjectHasValue*

visit_has_self(*node*, *children*) → *owlapy.class_expression.OWLObjectHasSelf*

visit_object_property(*node*, *children*) → *owlapy.owl_property.OWLObjectPropertyExpression*

visit_class_expression(*node*, *children*) → *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*) → *owlapy.class_expression.OWLDataHasValue*

visit_data_union(*node*, *children*) → *owlapy.owl_data_ranges.OWLDataRange*

visit_data_intersection(*node*, *children*) → *owlapy.owl_data_ranges.OWLDataRange*

visit_literal_list(*node*, *children*) → *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*) → *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*) → str

visit_float_literal(*node*, *children*) → *owlapy.owl_literal.OWLLiteral*

visit_decimal_literal(*node*, *children*) → *owlapy.owl_literal.OWLLiteral*

visit_integer_literal(*node*, *children*) → *owlapy.owl_literal.OWLLiteral*

visit_boolean_literal(*node*, *children*) → *owlapy.owl_literal.OWLLiteral*

visit_datetime_literal(*node*, *children*) → *owlapy.owl_literal.OWLLiteral*

visit_duration_literal(*node*, *children*) → *owlapy.owl_literal.OWLLiteral*

visit_date_literal(*node*, *children*) → *owlapy.owl_literal.OWLLiteral*

visit_non_negative_integer(*node*, *children*) → int

visit_datatype_iri(*node*, *children*) → str

visit_datatype(*node*, *children*) → *owlapy.owl_datatype.OWLDatatype*

visit_facet(*node*, *children*) → *owlapy.vocab.OWLFacet*

visit_class_iri(*node*, *children*) → *owlapy.class_expression.OWLClass*

visit_individual_iri(*node*, *children*) → *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*) → *owlapy.iri.IRI*

visit_full_iri(*node*, *children*) → *owlapy.iri.IRI*

abstract visit_abbreviated_iri(*node*, *children*)

visit_simple_iri(*node*, *children*) → *owlapy.iri.IRI*

visit_parentheses(*node*, *children*) → *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

Classes

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

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

class owlapy.providers.**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)

    __slots__ = ()

    type_index: Final = 4008

    get_literal() → 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() → bool

> Whether this literal is typed as boolean.

parse_boolean() → 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() → bool

> Whether this literal is typed as double.

parse_double() → 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() → bool

> Whether this literal is typed as integer.

parse_integer() → 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() → bool

> Whether this literal is typed as string.

parse_string() → 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() → bool

> Whether this literal is typed as date.

parse_date() → 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() → bool

> Whether this literal is typed as dateTime.

parse_datetime() → 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.

`is_duration``()` → bool

> Whether this literal is typed as duration.

`parse_duration``()` → 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``()` → bool

> > **Returns**
> >
> > > true if the annotation value is a literal

`as_literal``()` → *OWLLiteral*

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

`to_python``()` → Literals

abstract `get_datatype``()` → *owlapy.owl_datatype.OWLDatatype*

> Gets the OWLDatatype which types this literal.
>
> > **Returns**
> >
> > > The OWLDatatype that types this literal.

class `owlapy.providers.`**`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*

`get_facet_restrictions``()` → Sequence[*OWLFacetRestriction*]

`__eq__``(other)`

> Return self==value.

`__hash__``()`

> Return hash(self).

`__repr__``()`

> Return repr(self).

class `owlapy.providers.`**`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.providers.**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).

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

[IRI](#)	An IRI, consisting of a namespace and a remainder.
[OWLNamedIndividual](#)	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
[OWLLiteral](#)	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
[OWLObjectRenderer](#)	Abstract class with a render method to render an OWL Object into a string.
[OWLEntity](#)	Represents Entities in the OWL 2 Specification.
[OWLObject](#)	Base interface for OWL objects
[OWLObjectInverseOf](#)	Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression
[OWLPropertyExpression](#)	Represents a property or possibly the inverse of a property.
[OWLClassExpression](#)	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
[OWLBooleanClassExpression](#)	Represent an anonymous boolean class expression.

Table 9 – continued from previous page

OWLClass	An OWL 2 named Class. Classes can be understood as sets of individuals.
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
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
OWLObjectComplementOf	Represents an ObjectComplementOf class expression in the OWL 2 Specification.
OWLObjectMinCardinality	A minimum cardinality expression ObjectMinCardinality(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
OWLObjectMaxCardinality	A maximum cardinality expression ObjectMaxCardinality(n OPE CE) consists of a nonnegative integer n, an object
OWLObjectHasSelf	A self-restriction ObjectHasSelf(OPE) consists of an object property expression OPE,
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,
OWLDataMinCardinality	A minimum cardinality expression DataMinCardinality(n DPE DR) consists of a nonnegative integer n, a data
OWLDataExactCardinality	An exact cardinality expression ObjectExactCardinality(n OPE CE) consists of a nonnegative integer n, an
OWLDataMaxCardinality	A maximum cardinality expression ObjectMaxCardinality(n OPE CE) consists of a nonnegative integer n, an object
OWLDataOneOf	An enumeration of literals DataOneOf(lt1 ... ltn) contains exactly the explicitly specified literals lti with
OWLNaryBooleanClassExpression	OWLNaryBooleanClassExpression.
OWLRestriction	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
OWLFacet	Enumerations for OWL facets.
OWLNaryDataRange	OWLNaryDataRange.
OWLDataComplementOf	A complement data range DataComplementOf(DR) contains all tuples of literals that are not contained in the
OWLDataUnionOf	A union data range DataUnionOf(DR1 ... DRn) contains all tuples of literals that are contained in the at least
OWLDataIntersectionOf	An intersection data range DataIntersectionOf(DR1 ... DRn) contains all tuples of literals that are contained

Table 9 – continued from previous page

OWLObjectHasValue	A has-value class expression ObjectHasValue(OPE a) consists of an object property expression OPE and an
OWLFacetRestriction	A facet restriction is used to restrict a particular datatype.
OWLDatatypeRestriction	A datatype restriction DatatypeRestriction(DT F1 lt1 ... Fn ltn) consists of a unary datatype DT and n pairs
OWLObjectOneOf	An enumeration of individuals ObjectOneOf(a1 ... an) contains exactly the individuals ai with $1 \leq i \leq n$.
OWLDatatype	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
DLSyntaxObjectRenderer	DL Syntax renderer for OWL Objects.
ManchesterOWLSyntaxOWLObjectRenderer	Manchester Syntax renderer for OWL Objects

Functions

owl_expression_to_dl(→ str)

owl_expression_to_manchester(→ str)

Module Contents

class owlapy.render.**IRI**(*namespace: str | owlapy.namespaces.Namespaces*, *remainder: str*)

Bases: *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*) → *IRI*

static create(*namespace: str*, *remainder: str*) → *IRI*

static create(*string: str*) → *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 <http://www.w3.org/2002/07/owl#Nothing> and otherwise False.

273

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 <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

is_reserved_vocabulary() → bool

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

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

as_iri() → *IRI*

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

as_str() → 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.

get_short_form() → str

> Gets the short form.

> > **Returns**
> > A string that represents the short form.

get_namespace() → str

> > **Returns**
> > The namespace as string.

get_remainder() → str

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

class owlapy.render.**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

class owlapy.render.**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)

__slots__ = ()

type_index: Final = 4008

get_literal() → 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() → bool

> Whether this literal is typed as boolean.

parse_boolean() → 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() → bool

> Whether this literal is typed as double.

parse_double() → 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() → bool

> Whether this literal is typed as integer.

parse_integer() → 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() → bool

> Whether this literal is typed as string.

parse_string() → 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() → bool

> Whether this literal is typed as date.

parse_date() → 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() → bool

> Whether this literal is typed as dateTime.

parse_datetime() → 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.

is_duration() → bool

> Whether this literal is typed as duration.

parse_duration() → 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() → bool

> > **Returns**
> > true if the annotation value is a literal

as_literal() → *OWLLiteral*

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

to_python() → Literals

abstract get_datatype() → *owlapy.owl_datatype.OWLDatatype*

> Gets the OWLDatatype which types this literal.
>
> > **Returns**
> > The OWLDatatype that types this literal.

class owlapy.render.**OWLObjectRenderer**

> Abstract class with a render method to render an OWL Object into a string.

abstract set_short_form_provider(*short_form_provider*) → None

>   Configure a short form provider that shortens the OWL objects during rendering.

>   > **Parameters**
>   > **short_form_provider** – Short form provider.

abstract render(*o: OWLObject*) → str

>   Render OWL Object to string.

>   > **Parameters**
>   > **o** – OWL Object.

>   > **Returns**
>   > String rendition of OWL object.

class owlapy.render.**OWLEntity**

>   Bases: OWLNamedObject

>   Represents Entities in the OWL 2 Specification.

>   **__slots__ = ()**

>   **to_string_id**() → str

>   **is_anonymous**() → bool

class owlapy.render.**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**() → bool

class owlapy.render.**OWLObjectInverseOf**(*property: OWLObjectProperty*)

>   Bases: OWLObjectPropertyExpression

>   Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression Object-InverseOf(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() → *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.

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.

__repr__()

> Return repr(self).

__eq__(*other*)

> Return self==value.

__hash__()

> Return hash(self).

class owlapy.render.**OWLPropertyExpression**

> Bases: *owlapy.owl_object.OWLObject*
>
> Represents a property or possibly the inverse of a property.
>
> **__slots__ = ()**
>
> **is_data_property_expression**() → bool
>
> > **Returns**
> > True if this is a data property.
>
> **is_object_property_expression**() → bool
>
> > **Returns**
> > True if this is an object property.
>
> **is_owl_top_object_property**() → bool
>
> > Determines if this is the owl:topObjectProperty.
> >
> > **Returns**
> > topObjectProperty.
> >
> > **Return type**
> > True if this property is the owl
>
> **is_owl_top_data_property**() → bool
>
> > Determines if this is the owl:topDataProperty.
> >
> > **Returns**
> > topDataProperty.
> >
> > **Return type**
> > True if this property is the owl

class owlapy.render.**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)
>
> **__slots__ = ()**
>
> **abstract is_owl_thing**() → 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**() → 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.

class owlapy.render.**OWLBooleanClassExpression**

> Bases: OWLAnonymousClassExpression
>
> Represent an anonymous boolean class expression.
>
> **__slots__ = ()**

class owlapy.render.**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() → 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() → 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() → *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.

class owlapy.render.**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.render.**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.render.**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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Union_of_Class_Expressions)

__slots__ = '_operands'

type_index: Final = 3002

class owlapy.render.**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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Intersection_of_Class_Expressions)

__slots__ = '_operands'

type_index: Final = 3001

class owlapy.render.**OWLObjectComplementOf**(*op: OWLClassExpression*)

Bases: *OWLBooleanClassExpression,*   *owlapy.meta_classes.*
HasOperands[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() → 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.render.**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.render.**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()
   → *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.render.**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.render.`**`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.render.`**`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 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).

`__eq__`(*other*)
> Return self==value.

__hash__()

>   Return hash(self).

get_property() → *owlapy.owl_property.OWLDataPropertyExpression*

>   **Returns**
>
>   Property being restricted.

class owlapy.render.**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 ≤ i ≤ 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 ≤ i ≤ 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.render.**OWLDataHasValue**(
>   *property: owlapy.owl_property.OWLDataPropertyExpression*,
>   *value: owlapy.owl_literal.OWLLiteral*)

Bases: OWLHasValueRestriction[*owlapy.owl_literal.OWLLiteral*], OWLDataRestriction

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`() → *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.render.**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.render.**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.render.**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.render.**OWLDataOneOf**(
    values: [owlapy.owl_literal.OWLLiteral](#) | Iterable[[owlapy.owl_literal.OWLLiteral](#)])

Bases: [*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 $1 \leq i \leq$ n. The resulting data range has arity one. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Literals)

```
type_index: Final = 4003
```

values() → Iterable[[*owlapy.owl_literal.OWLLiteral*](#)]

> Gets the values that are in the oneOf.
>
> > **Returns**
> >
> > > The values of this {@code DataOneOf} class expression.

operands() → 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.render.**OWLNaryBooleanClassExpression**(
    operands: Iterable[[owlapy.class_expression.class_expression.OWLClassExpression](#)])

Bases: [*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.render.**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() → bool

        Determines if this is a data restriction.

            Returns

                True if this is a data restriction.

    is_object_restriction() → bool

        Determines if this is an object restriction.

            Returns

                True if this is an object restriction.

class owlapy.render.**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.render.**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.render.`**`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).

class `owlapy.render.`**`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 $1 \leq i \leq n$. 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.render.`**`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 $1 \leq i \leq n$. 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.render.**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() → *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.render.**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.render.**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*

get_facet_restrictions() → Sequence[*OWLFacetRestriction*]

__eq__(*other*)
> Return self==value.

__hash__()
> Return hash(self).

__repr__()
> Return repr(self).

class owlapy.render.**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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals)

__slots__ = '_values'

type_index: Final = 3004

individuals() → 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() → *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}, … , {an})

__hash__()
> Return hash(self).

__eq__(*other*)
> Return self==value.

__repr__()
> Return repr(self).

class owlapy.render.**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 IRI as string

class owlapy.render.**DLSyntaxObjectRenderer**(
> > *short_form_provider: Callable[[owlapy.owl_object.OWLEntity], str] = _simple_short_form_provider*)

> Bases: *owlapy.owl_object.OWLObjectRenderer*

> DL Syntax renderer for OWL Objects.

> **__slots__ = '_sfp'**

> **set_short_form_provider**(*short_form_provider: Callable[[owlapy.owl_object.OWLEntity], str]*)
> > → 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.

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]*)
    → 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 classes __module__ 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

OWLNothing
OWLThing
measurer

Classes

OWLNamedIndividual	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
HasIRI	Simple class to access the IRI.
HasFiller	An interface to objects that have a filler.
HasCardinality	An interface to objects that have a cardinality.
HasOperands	An interface to objects that have a collection of operands.
OWLLiteral	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
OWLObjectInverseOf	Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression
OWLObjectProperty	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
OWLDataProperty	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
OWLClassExpression	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
OWLClass	An OWL 2 named Class. Classes can be understood as sets of individuals.
OWLObjectCardinalityRestriction	Represents Object Property Cardinality Restrictions in the OWL 2 specification.
OWLObjectComplementOf	Represents an ObjectComplementOf class expression in the OWL 2 Specification.
OWLRestriction	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
OWLObjectSomeValuesFrom	An existential class expression ObjectSomeValuesFrom(OPE CE) consists of an object property expression OPE and
OWLObjectMinCardinality	A minimum cardinality expression ObjectMinCardinality(n OPE CE) consists of a nonnegative integer n, an object

Table 10 – continued from previous page

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
OWLObjectHasSelf	A self-restriction ObjectHasSelf(OPE) consists of an object property expression OPE,
OWLDataMaxCardinality	A maximum cardinality expression ObjectMaxCardinality(n OPE CE) consists of a nonnegative integer n, an object
OWLDataMinCardinality	A minimum cardinality expression DataMinCardinality(n DPE DR) consists of a nonnegative integer n, a data
OWLDataExactCardinality	An exact cardinality expression ObjectExactCardinality(n OPE CE) consists of a nonnegative integer n, an
OWLDataHasValue	A has-value class expression DataHasValue(DPE lt) consists of a data property expression DPE and a literal lt,
OWLDataAllValuesFrom	A universal class expression DataAllValuesFrom(DPE1 ... DPEn DR) consists of n data property expressions DPEi,
OWLDataSomeValuesFrom	An existential class expression DataSomeValuesFrom(DPE1 ... DPEn DR) consists of n data property expressions
OWLObjectAllValuesFrom	A universal class expression ObjectAllValuesFrom(OPE CE) consists of an object property expression OPE and a
OWLDataOneOf	An enumeration of literals DataOneOf(lt1 ... ltn) contains exactly the explicitly specified literals lti with
OWLObjectIntersectionOf	An intersection class expression ObjectIntersectionOf(CE1 ... CEn) contains all individuals that are instances
OWLDataCardinalityRestriction	Represents Data Property Cardinality Restrictions.
OWLNaryBooleanClassExpression	OWLNaryBooleanClassExpression.
OWLObjectUnionOf	A union class expression ObjectUnionOf(CE1 ... CEn) contains all individuals that are instances
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
OWLFacetRestriction	A facet restriction is used to restrict a particular datatype.
OWLObjectOneOf	An enumeration of individuals ObjectOneOf(a1 ... an) contains exactly the individuals ai with $1 \leq i \leq n$.
OWLDataComplementOf	A complement data range DataComplementOf(DR) contains all tuples of literals that are not contained in the
OWLDataUnionOf	A union data range DataUnionOf(DR1 ... DRn) contains all tuples of literals that are contained in the at least
OWLDataIntersectionOf	An intersection data range DataIntersectionOf(DR1 ... DRn) contains all tuples of literals that are contained
OWLNaryDataRange	OWLNaryDataRange.
OWLDataRange	Represents a DataRange in the OWL 2 Specification.
OWLPropertyRange	OWL Objects that can be the ranges of properties.
OWLObject	Base interface for OWL objects
OWLDatatype	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
OWLClassExpressionLengthMetric	Length calculation of OWLClassExpression

Table 10 – continued from previous page

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(→ HasIndex)	Cast OWL Object to HasIndex.

Module Contents

class owlapy.utils.**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

295

class owlapy.utils.**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.utils.**HasFiller**

> Bases: Generic[_T]

> An interface to objects that have a filler.

>> **Parameters**
>> **_T** – Filler type.

> **__slots__ = ()**

> **abstract get_filler**() → _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.utils.**HasCardinality**

> An interface to objects that have a cardinality.

> **__slots__ = ()**

> **abstract get_cardinality**() → int

>> Gets the cardinality of a restriction.

>>> **Returns**
>>> The cardinality. A non-negative integer.

class owlapy.utils.**HasOperands**

> Bases: Generic[_T]

> An interface to objects that have a collection of operands.

>> **Parameters**
>> **_T** – Operand type.

> **__slots__ = ()**

abstract operands`()` → 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.utils.**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)

__slots__ = ()

type_index: Final = 4008

get_literal`()` → 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`()` → bool

    Whether this literal is typed as boolean.

parse_boolean`()` → 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`()` → bool

    Whether this literal is typed as double.

parse_double`()` → 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`()` → bool

    Whether this literal is typed as integer.

parse_integer`()` → 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`()` → bool

    Whether this literal is typed as string.

parse_string() → 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() → bool

>Whether this literal is typed as date.

parse_date() → 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() → bool

>Whether this literal is typed as dateTime.

parse_datetime() → 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.

is_duration() → bool

>Whether this literal is typed as duration.

parse_duration() → 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() → bool

>>**Returns**

>>>true if the annotation value is a literal

as_literal() → *OWLLiteral*

>>**Returns**

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

to_python() → Literals

abstract get_datatype() → *owlapy.owl_datatype.OWLDatatype*

>Gets the OWLDatatype which types this literal.

>>**Returns**

>>>The OWLDatatype that types this literal.

class owlapy.utils.**OWLObjectInverseOf**(*property: OWLObjectProperty*)

>Bases: OWLObjectPropertyExpression

Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression Object-InverseOf(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`() → *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.

`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.

`__repr__`()

>   Return repr(self).

`__eq__`(*other*)

>   Return self==value.

`__hash__`()

>   Return hash(self).

`class` `owlapy.utils.`**`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`**

>   **`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.

>   **`get_inverse_property`**() → *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() → bool

> Determines if this is the owl:topObjectProperty.

> > **Returns**
> > > topObjectProperty.

> > **Return type**
> > > True if this property is the owl

class owlapy.utils.**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

class owlapy.utils.**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)

> **__slots__ = ()**

> **abstract is_owl_thing**() → 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**() → 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.

class owlapy.utils.**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**() → 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**() → 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**() → *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.

class owlapy.utils.**OWLObjectCardinalityRestriction**(*cardinality: int*,
        property: owlapy.owl_property.OWLObjectPropertyExpression,
        filler: owlapy.class_expression.class_expression.OWLClassExpression)

    Bases: 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.utils.**OWLObjectComplementOf**(*op: OWLClassExpression*)

    Bases: OWLBooleanClassExpression,   *owlapy.meta_classes.*
HasOperands[*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() → 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.utils.**OWLNothing: Final**

class owlapy.utils.**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`() → bool

> Determines if this is a data restriction.

> > **Returns**
> > > True if this is a data restriction.

`is_object_restriction`() → bool

> Determines if this is an object restriction.

> > **Returns**
> > > True if this is an object restriction.

owlapy.utils.**`OWLThing: Final`**

class owlapy.utils.**`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.utils.**`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.utils.**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.utils.**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()
```
→ *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.utils.**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.utils.**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.utils.**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.utils.**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.utils.**OWLDataHasValue**(
  property: owlapy.owl_property.OWLDataPropertyExpression,
  value: owlapy.owl_literal.OWLLiteral)

Bases: `OWLHasValueRestriction[`*owlapy.owl_literal.OWLLiteral*`], OWLDataRestriction`

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``()` → *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.utils.`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 ≤ i ≤ 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 ≤ i ≤ 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.utils.**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 expressions DPEi, $1 \leq i \leq n$, 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, $1 \leq i \leq n$, 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.utils.**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.utils.**OWLDataOneOf**(
        values: [owlapy.owl_literal.OWLLiteral](#) | Iterable[[owlapy.owl_literal.OWLLiteral](#)])

Bases: *[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 ≤ i ≤ n. The resulting data range has arity one. (https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Literals)

type_index: Final = 4003

values() → Iterable[*[owlapy.owl_literal.OWLLiteral](#)*]

        Gets the values that are in the oneOf.

                Returns
                        The values of this {@code DataOneOf} class expression.

operands() → 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.utils.**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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Intersection_of_Class_Expressions)

__slots__ = '_operands'

type_index: Final = 3001

class owlapy.utils.**OWLDataCardinalityRestriction**(*cardinality: int*,
        property: [owlapy.owl_property.OWLDataPropertyExpression](#),
        filler: [owlapy.owl_data_ranges.OWLDataRange](#))

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).

class owlapy.utils.**OWLNaryBooleanClassExpression**(
> *operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression]*)

Bases: *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.utils.**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 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Union_of_Class_Expressions)

__slots__ = '_operands'

type_index: Final = 3002

class owlapy.utils.**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() → *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.utils.**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*

get_facet_restrictions() → Sequence[*OWLFacetRestriction*]

__eq__(*other*)

> Return self==value.

__hash__()

> Return hash(self).

__repr__()

> Return repr(self).

class owlapy.utils.**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.utils.**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**() → 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**() → *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}, … , {an})

>**__hash__**()

>>Return hash(self).

>**__eq__**(*other*)

>>Return self==value.

>**__repr__**()

>>Return repr(self).

class owlapy.utils.**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).

class owlapy.utils.**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 ≤ i ≤ 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.utils.**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 ≤ i ≤ 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.utils.**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.utils.**OWLDataRange**

>   Bases: *OWLPropertyRange*
>
>   Represents a DataRange in the OWL 2 Specification.

class owlapy.utils.**OWLPropertyRange**

    Bases: *owlapy.owl_object.OWLObject*

    OWL Objects that can be the ranges of properties.

class owlapy.utils.**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() → bool

class owlapy.utils.**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 IRI as string

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, datatype_length: int, data_one_of_length: int, data_complement_length: int, data_intersection_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 $\sqcup$ B
- **`object_complement_length`** – Complement: ¬ C
- **`object_some_values_length`** – Obj. Some Values: $\exists$ 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: $\exists$ r.Self
- **`object_one_of_length`** – Obj. One of: $\exists$ r.{X,Y,Z}
- **`data_some_values_length`** – Data Some Values: $\exists$ p.t
- **`data_all_values_length`** – Data All Values: $\forall$ p.t
- **`data_has_value_length`** – Data Has Value: $\exists$ p.{V}
- **`data_cardinality_length`** – Data Cardinality restriction: ≤n r.t
- **`object_property_length`** – Obj. Property: $\exists$ r.C
- **`object_inverse_length`** – Inverse property: $\exists$ r⁻.C
- **`data_property_length`** – Data Property: $\exists$ p.t
- **`datatype_length`** – Datatype: ^^datatype
- **`data_one_of_length`** – Data One of: $\exists$ p.{U,V,W}
- **`data_complement_length`** – Data Complement: ¬datatype
- **`data_intersection_length`** – Data Intersection: datatype $\sqcap$ datatype
- **`data_union_length`** – Data Union: datatype $\sqcup$ 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*) → int

class owlapy.utils.**EvaluatedDescriptionSet**(*ordering: Callable[[_N], _O]*,
        max_size: int = 10)

Bases: `Generic[_N, _O]`

Abstract base class for generic types.

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:
    try:
        return mapping[key]
    except KeyError:
        return default
```

__slots__ = ('items', '_max_size', '_Ordering')

items: SortedSet[_N]

maybe_add(*node: _N*)

clean()

worst()

best()

best_quality_value() → float

__iter__() → Iterable[_N]

class owlapy.utils.**ConceptOperandSorter**

    abstract sort(*o: _O*) → _O

class owlapy.utils.**OperandSetTransform**

    simplify(*o: owlapy.class_expression.OWLClassExpression*)
                 → *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*)

        Return self==value.

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.

    o

        OWL object.

    __slots__ = ('o', '_chain')

    o: _HasIndex

    __lt__(*other*)

        Return self<value.

    __eq__(*other*)

        Return self==value.

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) → *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.

get_top_level_dnf(*ce: owlapy.class_expression.OWLClassExpression*)
→ *owlapy.class_expression.OWLClassExpression*

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*) → bool

__getitem__ (*item: _K*) → _V

__setitem__ (*key: _K*, *value: _V*)

cache_info ()

> Report cache statistics.

cache_clear ()

> Clear the cache and cache statistics.

owlapy.vocab

Enumerations.

Classes

HasIRI	Simple class to access the IRI.
IRI	An IRI, consisting of a namespace and a remainder.
Namespaces	Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup
OWLRDFVocabulary	Enumerations for OWL/RDF vocabulary.
XSDVocabulary	Enumerations for XSD vocabulary.
OWLFacet	Enumerations for OWL facets.

Module Contents

class owlapy.vocab.**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.vocab.**IRI**(*namespace: str | owlapy.namespaces.Namespaces*, *remainder: str*)

> Bases: *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*) → *IRI*
> **static create**(*namespace: str*, *remainder: str*) → *IRI*
> **static create**(*string: str*) → *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 <http://www.w3.org/2002/07/owl#Nothing> 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 <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

> **is_reserved_vocabulary**() → bool
> > Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http://www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.

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

> **as_iri**() → *IRI*

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

> **as_str**() → 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.

get_short_form() → str
> Gets the short form.
>
> > **Returns**
> > A string that represents the short form.

get_namespace() → str
>
> > **Returns**
> > The namespace as string.

get_remainder() → str
>
> > **Returns**
> > The remainder (coincident with NCName usually) for this IRI.

class owlapy.vocab.**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.

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*) → *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

3.3 Attributes

__version__

3.4 Functions

owl_expression_to_dl(→ str)	
owl_expression_to_manchester(→ str)	
dl_to_owl_expression(dl_expression, namespace)	
manchester_to_owl_expression(manchester_ex ...)	
owl_expression_to_sparql(→ str)	Convert an OWL Class Expression (https://www.w3.org/TR/owl2-syntax/#Class_Expressions) into a SPARQL query

3.5 Package Contents

owlapy.**owl_expression_to_dl**(*o: owlapy.owl_object.OWLObject*) → str

owlapy.**owl_expression_to_manchester**(*o: owlapy.owl_object.OWLObject*) → str

owlapy.**dl_to_owl_expression**(*dl_expression: str*, *namespace: str*)

owlapy.**manchester_to_owl_expression**(*manchester_expression: str*, *namespace: str*)

owlapy.**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*) → 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 (¬(∃r.¬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__** = '1.0.2'

Python Module Index

o

Index

Non-alphabetical

330

337

A

B

C

D

343

I

355

O

375

U

V

W

X