OWLAPY

Release 0.1.2

Ontolearn Team

Apr 08, 2024

Contents:

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

1 OWLAPY

placeholder

2 owlapy

2.1 Subpackages

owlapy.model

@TODO: CD: This is not a python code. We should refactor this model module.

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

Submodules

owlapy.model.providers

OWL Datatype restriction constructors.

Module Contents

Functions

OWLDatatypeMaxExclusiveRestriction(...)	Create a max exclusive restriction.
OWLDatatypeMinExclusiveRestriction(...)	Create a min exclusive restriction.
OWLDatatypeMaxInclusiveRestriction(...)	Create a max inclusive restriction.
OWLDatatypeMinInclusiveRestriction(...)	Create a min inclusive restriction.
OWLDatatypeMinMaxExclusiveRestriction(...)	Create a min-max exclusive restriction.
OWLDatatypeMinMaxInclusiveRestriction(...)	Create a min-max inclusive restriction.

Attributes

Restriction_Literals

owlapy.model.providers.**Restriction_Literals**

owlapy.model.providers.**OWLDatatypeMaxExclusiveRestriction**(*max_: Restriction_Literals*) → *owlapy.model.OWLDatatypeRestriction*

Create a max exclusive restriction.

owlapy.model.providers.**OWLDatatypeMinExclusiveRestriction**(*min_: Restriction_Literals*) → *owlapy.model.OWLDatatypeRestriction*

Create a min exclusive restriction.

owlapy.model.providers.**OWLDatatypeMaxInclusiveRestriction**(*max_: Restriction_Literals*) → *owlapy.model.OWLDatatypeRestriction*

Create a max inclusive restriction.

owlapy.model.providers.**OWLDatatypeMinInclusiveRestriction**(*min_: Restriction_Literals*) → *owlapy.model.OWLDatatypeRestriction*

Create a min inclusive restriction.

owlapy.model.providers.**OWLDatatypeMinMaxExclusiveRestriction**(*min_: Restriction_Literals*, *max_: Restriction_Literals*) → *owlapy.model.OWLDatatypeRestriction*

Create a min-max exclusive restriction.

owlapy.model.providers.**OWLDatatypeMinMaxInclusiveRestriction**(*min_: Restriction_Literals*, *max_: Restriction_Literals*) → *owlapy.model.OWLDatatypeRestriction*

Create a min-max inclusive restriction.

Package Contents

Classes

OWLRDFVocabulary	Enumerations for OWL/RDF vocabulary.
XSDVocabulary	Enumerations for XSD vocabulary.
OWLFacet	Enumerations for OWL facets.
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.
HasIRI	Simple class to access the IRI.
IRI	An IRI, consisting of a namespace and a remainder.
HasIndex	Interface for types with an index; this is used to group objects by type when sorting.
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.
OWLClassExpression	An OWL 2 Class Expression.
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.
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.
OWLClass	An OWL 2 named Class
OWLPropertyExpression	Represents a property or possibly the inverse of a property.
OWLRestriction	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
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 marker interface for properties that aren't expression i.e. named properties. By definition, properties
OWLDataProperty	Represents a Data Property in the OWL 2 Specification.
OWLObjectProperty	Represents an Object Property in the OWL 2 Specification.
OWLObjectInverseOf	Represents the inverse of a property expression (ObjectInverseOf). This can be used to refer to the inverse of
OWLDataRestriction	Represents a Data Property Restriction in the OWL 2 specification.
OWLObjectRestriction	Represents a Object Property Restriction in the OWL 2 specification.
HasFiller	An interface to objects that have a filler.
OWLHasValueRestriction	OWLHasValueRestriction.
OWLQuantifiedRestriction	Represents a quantified restriction.

continues on next page

Table 1 – continued from previous page

OWLQuantifiedObjectRestriction	Represents a quantified object restriction.
OWLObjectSomeValuesFrom	Represents an ObjectSomeValuesFrom class expression in the OWL 2 Specification.
OWLObjectAllValuesFrom	Represents an ObjectAllValuesFrom class expression in the OWL 2 Specification.
OWLNaryBooleanClassExpression	OWLNaryBooleanClassExpression.
OWLObjectUnionOf	Represents an ObjectUnionOf class expression in the OWL 2 Specification.
OWLObjectIntersectionOf	Represents an OWLObjectIntersectionOf class expression in the OWL 2 Specification.
HasCardinality	An interface to objects that have a cardinality.
OWLCardinalityRestriction	Base interface for owl min and max cardinality restriction.
OWLObjectCardinalityRestriction	Represents Object Property Cardinality Restrictions in the OWL 2 specification.
OWLObjectMinCardinality	Represents a ObjectMinCardinality restriction in the OWL 2 Specification.
OWLObjectMaxCardinality	Represents a ObjectMaxCardinality restriction in the OWL 2 Specification.
OWLObjectExactCardinality	Represents an ObjectExactCardinality restriction in the OWL 2 Specification.
OWLObjectHasSelf	Represents an ObjectHasSelf class expression in the OWL 2 Specification.
OWLIndividual	Represents a named or anonymous individual.
OWLObjectHasValue	Represents an ObjectHasValue class expression in the OWL 2 Specification.
OWLObjectOneOf	Represents an ObjectOneOf class expression in the OWL 2 Specification.
OWLNamedIndividual	Represents a Named Individual in the OWL 2 Specification.
OWLOntologyID	An object that identifies an ontology. Since OWL 2, ontologies do not have to have an ontology IRI, or if they
OWLAxiom	Represents Axioms in the OWL 2 Specification.
OWLDatatype	Represents a Datatype (named data range) in the OWL 2 Specification.
OWLDatatypeRestriction	Represents a DatatypeRestriction data range in the OWL 2 Specification.
OWLFacetRestriction	A facet restriction is used to restrict a particular datatype.
OWLLiteral	Represents a Literal in the OWL 2 Specification.
OWLQuantifiedDataRestriction	Represents a quantified data restriction.
OWLDataCardinalityRestriction	Represents Data Property Cardinality Restrictions in the OWL 2 specification.
OWLDataAllValuesFrom	Represents DataAllValuesFrom class expressions in the OWL 2 Specification.
OWLDataComplementOf	Represents DataComplementOf in the OWL 2 Specification.
OWLDataExactCardinality	Represents DataExactCardinality restrictions in the OWL 2 Specification.
OWLDataHasValue	Represents DataHasValue restrictions in the OWL 2 Specification.
OWLDataMaxCardinality	Represents DataMaxCardinality restrictions in the OWL 2 Specification.

continues on next page

Table 1 – continued from previous page

OWLDataMinCardinality	Represents DataMinCardinality restrictions in the OWL 2 Specification.
OWLDataOneOf	Represents DataOneOf in the OWL 2 Specification.
OWLDataSomeValuesFrom	Represents a DataSomeValuesFrom restriction in the OWL 2 Specification.
OWLNaryDataRange	OWLNaryDataRange.
OWLDataUnionOf	Represents a DataUnionOf data range in the OWL 2 Specification.
OWLDataIntersectionOf	Represents DataIntersectionOf in the OWL 2 Specification.
OWLImportsDeclaration	Represents an import statement in an ontology.
OWLLogicalAxiom	A base interface of all axioms that affect the logical meaning of an ontology. This excludes declaration axioms
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	Represents a DatatypeDefinition axiom in the OWL 2 Specification.
OWLHasKeyAxiom	Represents a HasKey axiom in the OWL 2 Specification.
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	Represents an EquivalentClasses axiom in the OWL 2 Specification.
OWLDisjointClassesAxiom	Represents a DisjointClasses axiom in the OWL 2 Specification.
OWLNaryIndividualAxiom	Represents an axiom that contains two or more operands that could also be represented with
OWLDifferentIndividualsAxiom	Represents a DifferentIndividuals axiom in the OWL 2 Specification.
OWLSameIndividualAxiom	Represents a SameIndividual axiom in the OWL 2 Specification.
OWLNaryPropertyAxiom	Represents an axiom that contains two or more operands that could also be represented with
OWLEquivalentObjectPropertiesAxiom	Represents EquivalentObjectProperties axioms in the OWL 2 Specification.
OWLDisjointObjectPropertiesAxiom	Represents DisjointObjectProperties axioms in the OWL 2 Specification.
OWLInverseObjectPropertiesAxiom	Represents InverseObjectProperties axioms in the OWL 2 Specification.
OWLEquivalentDataPropertiesAxiom	Represents EquivalentDataProperties axioms in the OWL 2 Specification.
OWLDisjointDataPropertiesAxiom	Represents DisjointDataProperties axioms in the OWL 2 Specification.
OWLSubClassOfAxiom	Represents an SubClassOf axiom in the OWL 2 Specification.

continues on next page

Table 1 – continued from previous page

OWLDisjointUnionAxiom	Represents a DisjointUnion axiom in the OWL 2 Specification.
OWLClassAssertionAxiom	Represents ClassAssertion axioms in the OWL 2 Specification.
OWLAnnotationAxiom	A super interface for annotation axioms.
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
OWLAnnotationAssertionAxiom	Represents AnnotationAssertion axioms in the OWL 2 specification.
OWLSubAnnotationPropertyOfAxiom	Represents an SubAnnotationPropertyOf axiom in the OWL 2 specification.
OWLAnnotationPropertyDomainAxiom	Represents an AnnotationPropertyDomain axiom in the OWL 2 specification.
OWLAnnotationPropertyRangeAxiom	Represents an AnnotationPropertyRange axiom in the OWL 2 specification.
OWLSubPropertyAxiom	Base interface for object and data sub-property axioms.
OWLSubObjectPropertyOfAxiom	Represents a SubObjectPropertyOf axiom in the OWL 2 specification.
OWLSubDataPropertyOfAxiom	Represents a SubDataPropertyOf axiom in the OWL 2 specification.
OWLPropertyAssertionAxiom	Represents a PropertyAssertion axiom in the OWL 2 specification.
OWLObjectPropertyAssertionAxiom	Represents an ObjectPropertyAssertion axiom in the OWL 2 specification.
OWLNegativeObjectPropertyAssertionAxiom	Represents a NegativeObjectPropertyAssertion axiom in the OWL 2 specification.
OWLDataPropertyAssertionAxiom	Represents an DataPropertyAssertion axiom in the OWL 2 specification.
OWLNegativeDataPropertyAssertionAxiom	Represents an NegativeDataPropertyAssertion axiom in the OWL 2 specification.
OWLUnaryPropertyAxiom	Unary property axiom.
OWLObjectPropertyCharacteristicAxiom	Base interface for functional object property axiom.
OWLFunctionalObjectPropertyAxiom	Represents FunctionalObjectProperty axioms in the OWL 2 specification.
OWLAsymmetricObjectPropertyAxiom	Represents AsymmetricObjectProperty axioms in the OWL 2 specification.
OWLInverseFunctionalObjectPropertyAxiom	Represents InverseFunctionalObjectProperty axioms in the OWL 2 specification.
OWLIrreflexiveObjectPropertyAxiom	Represents IrreflexiveObjectProperty axioms in the OWL 2 specification.
OWLReflexiveObjectPropertyAxiom	Represents ReflexiveObjectProperty axioms in the OWL 2 specification.
OWLSymmetricObjectPropertyAxiom	Represents SymmetricObjectProperty axioms in the OWL 2 specification.
OWLTransitiveObjectPropertyAxiom	Represents TransitiveObjectProperty axioms in the OWL 2 specification.
OWLDataPropertyCharacteristicAxiom	Base interface for Functional data property axiom.
OWLFunctionalDataPropertyAxiom	Represents FunctionalDataProperty axioms in the OWL 2 specification.

continues on next page

Table 1 – continued from previous page

OWLPropertyDomainAxiom	Represents ObjectPropertyDomain axioms in the OWL 2 specification.
OWLPropertyRangeAxiom	Represents ObjectPropertyRange axioms in the OWL 2 specification.
OWLObjectPropertyDomainAxiom	Represents a ObjectPropertyDomain axiom in the OWL 2 Specification.
OWLDataPropertyDomainAxiom	Represents a DataPropertyDomain axiom in the OWL 2 Specification.
OWLObjectPropertyRangeAxiom	Represents a ObjectPropertyRange axiom in the OWL 2 Specification.
OWLDataPropertyRangeAxiom	Represents a DataPropertyRange axiom in the OWL 2 Specification.
OWLOntology	Represents an OWL 2 Ontology in the OWL 2 specification.
OWLOntologyChange	Represents an ontology change.
AddImport	Represents an ontology change where an import statement is added to an ontology.
OWLOntologyManager	An OWLOntologyManager manages a set of ontologies. It is the main point for creating, loading and accessing
OWLReasoner	An OWLReasoner reasons over a set of axioms (the set of reasoner axioms) that is based on the imports closure of

Functions

MOVE(*args)	"Move" an imported class to the current module by setting the classes __module__ attribute.

Attributes

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

class owlapy.model.**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.model.**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.model.**OWLFacet**(*remainder: str*, *symbolic_form: str*, *operator: Callable[[_X, _X], bool]*)

Bases: _Vocabulary, enum.Enum

Enumerations for OWL facets.

property symbolic_form

property operator

MIN_INCLUSIVE: Final = ('minInclusive', '>=')

MIN_EXCLUSIVE: Final = ('minExclusive', '>')

MAX_INCLUSIVE: Final = ('maxInclusive', '<=')

MAX_EXCLUSIVE: Final = ('maxExclusive', '<')

LENGTH: Final = ('length', 'length')

MIN_LENGTH: Final = ('minLength', 'minLength')

MAX_LENGTH: Final = ('maxLength', 'maxLength')

PATTERN: Final = ('pattern', 'pattern')

TOTAL_DIGITS: Final = ('totalDigits', 'totalDigits')

FRACTION_DIGITS: Final = ('fractionDigits', 'fractionDigits')

static from_str(*name: str*) → *OWLFacet*

owlapy.model.**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.

class owlapy.model.**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.model.**OWLAnnotationObject**

Bases: *OWLObject*

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

__slots__ = ()

as_iri() → owlapy.model._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.model.**OWLAnnotationSubject**

Bases: *OWLAnnotationObject*

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

__slots__ = ()

class owlapy.model.**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() → *owlapy.model.OWLLiteral* | None

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

class owlapy.model.**HasIRI**

Simple class to access the IRI.

__slots__ = ()

abstract get_iri() → *IRI*

Gets the IRI of this object.

Returns
The IRI of this object.

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

Bases: owlapy.model._base.OWLAnnotationSubject, owlapy.model._base.OWLAnnotationValue

An IRI, consisting of a namespace and a remainder.

property str: str

Returns: The string that specifies the IRI.

property reminder: str

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

__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.

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.model.**Literals**

class owlapy.model.**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.model.**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.model.**OWLPropertyRange**

Bases: _base.OWLObject

OWL Objects that can be the ranges of properties.

class owlapy.model.**OWLDataRange**

Bases: *OWLPropertyRange*

Represents a DataRange in the OWL 2 Specification.

class owlapy.model.**OWLClassExpression**

Bases: *OWLPropertyRange*

An OWL 2 Class Expression.

__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.model.**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.model.**OWLBooleanClassExpression**

Bases: *OWLAnonymousClassExpression*

Represent an anonymous boolean class expression.

__slots__ = ()

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

Bases: *OWLBooleanClassExpression*, *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.model.**OWLNamedObject**

Bases: _base.OWLObject, _iri.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.model.**OWLEntity**

Bases: *OWLNamedObject*

Represents Entities in the OWL 2 Specification.

__slots__ = ()

to_string_id() → str

is_anonymous() → bool

class owlapy.model.**OWLClass**(*iri: _iri.IRI*)

Bases: *OWLClassExpression*, *OWLEntity*

An OWL 2 named Class

property str

property reminder: str

The reminder of the IRI

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

type_index: Final = 1001

get_iri() → _iri.IRI

Gets the IRI of this object.

Returns

The IRI of this object.

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() → *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.model.**OWLPropertyExpression**

Bases: _base.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.model.**OWLRestriction**

Bases: *OWLAnonymousClassExpression*

Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.

__slots__ = ()

abstract get_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.model.**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.model.**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.model.**OWLProperty**

Bases: *OWLPropertyExpression*, *OWLEntity*

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

__slots__ = ()

class owlapy.model.**OWLDataProperty**(*iri: _iri.IRI*)

Bases: *OWLDataPropertyExpression*, *OWLProperty*

Represents a Data Property in the OWL 2 Specification.

__slots__ = '_iri'

type_index: Final = 1004

get_iri() → _iri.IRI

Gets the IRI of this object.

Returns
The IRI of this object.

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.model.**OWLObjectProperty**(*iri: _iri.IRI*)

Bases: *OWLObjectPropertyExpression*, *OWLProperty*

Represents an Object Property in the OWL 2 Specification.

property str: str

__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.

get_iri() → _iri.IRI

Gets the IRI of this object.

Returns
The IRI of this object.

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.model.**OWLObjectInverseOf**(*property: OWLObjectProperty*)

Bases: *OWLObjectPropertyExpression*

Represents the inverse of a property expression (ObjectInverseOf). 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.

__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.model.**OWLDataRestriction**

Bases: *OWLRestriction*

Represents a Data Property Restriction in the OWL 2 specification.

__slots__ = ()

is_data_restriction() → bool

Determines if this is a data restriction.

Returns
True if this is a data restriction.

class owlapy.model.**OWLObjectRestriction**

Bases: *OWLRestriction*

Represents a 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() → *OWLObjectPropertyExpression*

Returns
Property being restricted.

class owlapy.model.**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.model.**OWLHasValueRestriction**(*value: _T*)

Bases: Generic[_T], *OWLRestriction*, *HasFiller*[_T]

OWLHasValueRestriction.

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.model.**OWLQuantifiedRestriction**

Bases: Generic[_T], *OWLRestriction*, *HasFiller*[_T]

Represents a quantified restriction.

Parameters
_T – value type

__slots__ = ()

class owlapy.model.**OWLQuantifiedObjectRestriction**(*filler: OWLClassExpression*)

Bases: *OWLQuantifiedRestriction*[*OWLClassExpression*], *OWLObjectRestriction*

Represents a quantified object restriction.

__slots__ = ()

get_filler() → *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.model.**OWLObjectSomeValuesFrom**(*property: OWLObjectPropertyExpression*, *filler: OWLClassExpression*)

Bases: *OWLQuantifiedObjectRestriction*

Represents an ObjectSomeValuesFrom class expression in the OWL 2 Specification.

__slots__ = ('_property', '_filler')

type_index: Final = 3005

__repr__()
Return repr(self).

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

__hash__()
Return hash(self).

get_property() → *OWLObjectPropertyExpression*

Returns
Property being restricted.

class owlapy.model.**OWLObjectAllValuesFrom**(*property: OWLObjectPropertyExpression*, *filler: OWLClassExpression*)

Bases: *OWLQuantifiedObjectRestriction*

Represents an ObjectAllValuesFrom class expression in the OWL 2 Specification.

__slots__ = ('_property', '_filler')

type_index: Final = 3006

__repr__()
Return repr(self).

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

__hash__()
Return hash(self).

get_property() → *OWLObjectPropertyExpression*

Returns
Property being restricted.

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

Bases: *OWLBooleanClassExpression*, *HasOperands*[*OWLClassExpression*]

OWLNaryBooleanClassExpression.

__slots__ = ()

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.model.**OWLObjectUnionOf**(*operands: Iterable[OWLClassExpression]*)

Bases: *OWLNaryBooleanClassExpression*

Represents an ObjectUnionOf class expression in the OWL 2 Specification.

__slots__ = '_operands'

type_index: Final = 3002

class owlapy.model.**OWLObjectIntersectionOf**(*operands: Iterable[OWLClassExpression]*)

Bases: *OWLNaryBooleanClassExpression*

Represents an OWLObjectIntersectionOf class expression in the OWL 2 Specification.

__slots__ = '_operands'

type_index: Final = 3001

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

Bases: Generic[_F], *OWLQuantifiedRestriction*[_F], *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.model.**OWLObjectCardinalityRestriction**(*cardinality: int*, *property: OWLObjectPropertyExpression*, *filler: OWLClassExpression*)

Bases: *OWLCardinalityRestriction*[*OWLClassExpression*], *OWLQuantifiedObjectRestriction*

Represents Object Property Cardinality Restrictions in the OWL 2 specification.

__slots__ = ()

get_property() → *OWLObjectPropertyExpression*

Returns
Property being restricted.

__repr__()

Return repr(self).

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

class owlapy.model.**OWLObjectMinCardinality**(*cardinality: int*, *property: OWLObjectPropertyExpression*, *filler: OWLClassExpression*)

Bases: *OWLObjectCardinalityRestriction*

Represents a ObjectMinCardinality restriction in the OWL 2 Specification.

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

type_index: Final = 3008

class owlapy.model.**OWLObjectMaxCardinality**(*cardinality: int*, *property: OWLObjectPropertyExpression*, *filler: OWLClassExpression*)

Bases: *OWLObjectCardinalityRestriction*

Represents a ObjectMaxCardinality restriction in the OWL 2 Specification.

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

type_index: Final = 3010

class owlapy.model.**OWLObjectExactCardinality**(*cardinality: int*, *property: OWLObjectPropertyExpression*, *filler: OWLClassExpression*)

Bases: *OWLObjectCardinalityRestriction*

Represents an ObjectExactCardinality restriction in the OWL 2 Specification.

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

type_index: Final = 3009

as_intersection_of_min_max() → *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.model.**OWLObjectHasSelf**(*property: OWLObjectPropertyExpression*)

Bases: *OWLObjectRestriction*

Represents an ObjectHasSelf class expression in the OWL 2 Specification.

__slots__ = '_property'

type_index: Final = 3011

get_property() → *OWLObjectPropertyExpression*

Returns

Property being restricted.

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

class owlapy.model.**OWLIndividual**

Bases: _base.OWLObject

Represents a named or anonymous individual.

__slots__ = ()

class owlapy.model.**OWLObjectHasValue**(*property: OWLObjectPropertyExpression*, *individual: OWLIndividual*)

Bases: *OWLHasValueRestriction*[*OWLIndividual*], *OWLObjectRestriction*

Represents an ObjectHasValue class expression in the OWL 2 Specification.

__slots__ = ('_property', '_v')

type_index: Final = 3007

get_property() → *OWLObjectPropertyExpression*

Returns

Property being restricted.

as_some_values_from() → *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.model.**OWLObjectOneOf**(*values: OWLIndividual | Iterable[OWLIndividual]*)

Bases: *OWLAnonymousClassExpression*, *HasOperands*[*OWLIndividual*]

Represents an ObjectOneOf class expression in the OWL 2 Specification.

__slots__ = '_values'

type_index: Final = 3004

individuals() → Iterable[*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[*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() → *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.model.**OWLNamedIndividual**(*iri: _iri.IRI*)

Bases: *OWLIndividual*, *OWLEntity*

Represents a Named Individual in the OWL 2 Specification.

property iri

property str

__slots__ = '_iri'

type_index: Final = 1005

get_iri() → _iri.IRI

Gets the IRI of this object.

Returns

The IRI of this object.

class owlapy.model.**OWLOntologyID**(*ontology_iri: _iri.IRI | None = None*, *version_iri: _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() → _iri.IRI | None

Gets the ontology IRI.

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

get_version_iri() → _iri.IRI | None

Gets the version IRI.

Returns
Version IRI or None.

get_default_document_iri() → _iri.IRI | None

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

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

is_anonymous() → bool

__repr__()

Return repr(self).

__eq__(*other*)

Return self==value.

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

Bases: _base.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.model.**OWLDatatype**(*iri: _iri.IRI | _iri.HasIRI*)

Bases: *OWLEntity*, *OWLDataRange*

Represents a Datatype (named data range) in the OWL 2 Specification.

__slots__ = '_iri'

type_index: Final = 4001

get_iri() → _iri.IRI

Gets the IRI of this object.

Returns

The IRI of this object.

class owlapy.model.**OWLDatatypeRestriction**(*type_: OWLDatatype*, *facet_restrictions: OWLFacetRestriction | Iterable[OWLFacetRestriction]*)

Bases: *OWLDataRange*

Represents a DatatypeRestriction data range in the OWL 2 Specification.

__slots__ = ('_type', '_facet_restrictions')

type_index: Final = 4006

get_datatype() → *OWLDatatype*

get_facet_restrictions() → Sequence[*OWLFacetRestriction*]

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

class owlapy.model.**OWLFacetRestriction**(*facet: owlapy.vocab.OWLFacet*, *literal: Literals*)

Bases: _base.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() → *OWLLiteral*

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

class owlapy.model.**OWLLiteral**

Bases: _base.OWLAnnotationValue

Represents a Literal in the OWL 2 Specification.

__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() → *OWLDatatype*

Gets the OWLDatatype which types this literal.

Returns
The OWLDatatype that types this literal.

class owlapy.model.**OWLQuantifiedDataRestriction**(*filler: OWLDataRange*)

Bases: *OWLQuantifiedRestriction*[*OWLDataRange*], *OWLDataRestriction*

Represents a quantified data restriction.

__slots__ = ()

get_filler() → *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.model.**OWLDataCardinalityRestriction**(*cardinality: int*, *property: OWLDataPropertyExpression*, *filler: OWLDataRange*)

Bases: *OWLCardinalityRestriction*[*OWLDataRange*], *OWLQuantifiedDataRestriction*, *OWLDataRestriction*

Represents Data Property Cardinality Restrictions in the OWL 2 specification.

__slots__ = ()

get_property() → *OWLDataPropertyExpression*

Returns
Property being restricted.

__repr__()
Return repr(self).

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

__hash__()
Return hash(self).

class owlapy.model.**OWLDataAllValuesFrom**(*property: OWLDataPropertyExpression*, *filler: OWLDataRange*)

Bases: *OWLQuantifiedDataRestriction*

Represents DataAllValuesFrom class expressions in the OWL 2 Specification.

__slots__ = '_property'

type_index: Final = 3013

__repr__()
Return repr(self).

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

__hash__()
Return hash(self).

get_property() → *OWLDataPropertyExpression*

Returns
Property being restricted.

class owlapy.model.**OWLDataComplementOf**(*data_range: OWLDataRange*)

Bases: *OWLDataRange*

Represents DataComplementOf in the OWL 2 Specification.

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.model.**OWLDataExactCardinality**(*cardinality: int*, *property: OWLDataPropertyExpression*, *filler: OWLDataRange*)

Bases: *OWLDataCardinalityRestriction*

Represents DataExactCardinality restrictions in the OWL 2 Specification.

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

type_index: Final = 3016

as_intersection_of_min_max() → *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.model.**OWLDataHasValue**(*property: OWLDataPropertyExpression*, *value: OWLLiteral*)

Bases: *OWLHasValueRestriction*[*OWLLiteral*], *OWLDataRestriction*

Represents DataHasValue restrictions in the OWL 2 Specification.

__slots__ = '_property'

type_index: Final = 3014

__repr__()

Return repr(self).

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

as_some_values_from() → *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() → *OWLDataPropertyExpression*

Returns

Property being restricted.

class owlapy.model.**OWLDataMaxCardinality**(*cardinality: int*, *property: OWLDataPropertyExpression*, *filler: OWLDataRange*)

Bases: *OWLDataCardinalityRestriction*

Represents DataMaxCardinality restrictions in the OWL 2 Specification.

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

type_index: Final = 3017

class owlapy.model.**OWLDataMinCardinality**(*cardinality: int*, *property: OWLDataPropertyExpression*, *filler: OWLDataRange*)

Bases: *OWLDataCardinalityRestriction*

Represents DataMinCardinality restrictions in the OWL 2 Specification.

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

type_index: Final = 3015

class owlapy.model.**OWLDataOneOf**(*values: OWLLiteral | Iterable[OWLLiteral]*)

Bases: *OWLDataRange*, *HasOperands*[*OWLLiteral*]

Represents DataOneOf in the OWL 2 Specification.

type_index: Final = 4003

values() → Iterable[*OWLLiteral*]

Gets the values that are in the oneOf.

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

operands() → Iterable[*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.model.**OWLDataSomeValuesFrom**(*property: OWLDataPropertyExpression*, *filler: OWLDataRange*)

Bases: *OWLQuantifiedDataRestriction*

Represents a DataSomeValuesFrom restriction in the OWL 2 Specification.

__slots__ = '_property'

type_index: Final = 3012

__repr__()

Return repr(self).

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

get_property() → *OWLDataPropertyExpression*

Returns
Property being restricted.

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

Bases: *OWLDataRange*, *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.model.**OWLDataUnionOf**(*operands: Iterable[OWLDataRange]*)

Bases: *OWLNaryDataRange*

Represents a DataUnionOf data range in the OWL 2 Specification.

__slots__ = '_operands'

type_index: Final = 4005

class owlapy.model.**OWLDataIntersectionOf**(*operands: Iterable[OWLDataRange]*)

Bases: *OWLNaryDataRange*

Represents DataIntersectionOf in the OWL 2 Specification.

__slots__ = '_operands'

type_index: Final = 4004

class owlapy.model.**OWLImportsDeclaration**(*import_iri: _iri.IRI*)

Bases: _iri.HasIRI

Represents an import statement in an ontology.

__slots__ = '_iri'

get_iri() → _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.

class owlapy.model.**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.model.**OWLPropertyAxiom**(*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLLogicalAxiom*

The base interface for property axioms.

__slots__ = ()

class owlapy.model.**OWLObjectPropertyAxiom**(
annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLPropertyAxiom*

The base interface for object property axioms.

__slots__ = ()

class owlapy.model.**OWLDataPropertyAxiom**(
annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLPropertyAxiom*

The base interface for data property axioms.

__slots__ = ()

class owlapy.model.**OWLIndividualAxiom**(*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLLogicalAxiom*

The base interface for individual axioms.

__slots__ = ()

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

Bases: *OWLLogicalAxiom*

The base interface for class axioms.

__slots__ = ()

class owlapy.model.**OWLDeclarationAxiom**(*entity: 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() → *OWLEntity*

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

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

Bases: *OWLLogicalAxiom*

Represents a DatatypeDefinition axiom in the OWL 2 Specification.

__slots__ = ('_datatype', '_datarange')

get_datatype() → *OWLDatatype*

get_datarange() → *OWLDataRange*

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

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

Bases: *OWLLogicalAxiom*, *HasOperands*[*OWLPropertyExpression*]

Represents a HasKey axiom in the OWL 2 Specification.

__slots__ = ('_class_expression', '_property_expressions')

get_class_expression() → *OWLClassExpression*

get_property_expressions() → List[*OWLPropertyExpression*]

operands() → Iterable[*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.model.**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.model.**OWLNaryClassAxiom**(*class_expressions: List[OWLClassExpression]*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLClassAxiom*, *OWLNaryAxiom*[*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[*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.model.**OWLEquivalentClassesAxiom**(*class_expressions: List[OWLClassExpression]*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLNaryClassAxiom*

Represents an EquivalentClasses axiom in the OWL 2 Specification.

__slots__ = ()

contains_named_equivalent_class() → bool

contains_owl_nothing() → bool

contains_owl_thing() → bool

named_classes() → Iterable[*OWLClass*]

class owlapy.model.**OWLDisjointClassesAxiom**(*class_expressions: List[OWLClassExpression]*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLNaryClassAxiom*

Represents a DisjointClasses axiom in the OWL 2 Specification.

__slots__ = ()

class owlapy.model.**OWLNaryIndividualAxiom**(*individuals: List[OWLIndividual]*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLIndividualAxiom*, *OWLNaryAxiom*[*OWLIndividual*]

Represents an axiom that contains two or more operands that could also be represented with multiple pairwise individual axioms.

__slots__ = '_individuals'

individuals() → Iterable[*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.model.**OWLDifferentIndividualsAxiom**(*individuals: List[OWLIndividual]*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLNaryIndividualAxiom*

Represents a DifferentIndividuals axiom in the OWL 2 Specification.

__slots__ = ()

class owlapy.model.**OWLSameIndividualAxiom**(*individuals: List[OWLIndividual]*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLNaryIndividualAxiom*

Represents a SameIndividual axiom in the OWL 2 Specification.

__slots__ = ()

class owlapy.model.**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.model.**OWLEquivalentObjectPropertiesAxiom**(
properties: List[OWLObjectPropertyExpression],
annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLNaryPropertyAxiom*[*OWLObjectPropertyExpression*], *OWLObjectPropertyAxiom*

Represents EquivalentObjectProperties axioms in the OWL 2 Specification.

__slots__ = ()

class owlapy.model.**OWLDisjointObjectPropertiesAxiom**(
properties: List[OWLObjectPropertyExpression],
annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLNaryPropertyAxiom*[*OWLObjectPropertyExpression*], *OWLObjectPropertyAxiom*

Represents DisjointObjectProperties axioms in the OWL 2 Specification.

__slots__ = ()

class owlapy.model.**OWLInverseObjectPropertiesAxiom**(
first: OWLObjectPropertyExpression, *second: OWLObjectPropertyExpression*,
annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLNaryPropertyAxiom*[*OWLObjectPropertyExpression*], *OWLObjectPropertyAxiom*

Represents InverseObjectProperties axioms in the OWL 2 Specification.

__slots__ = ('_first', '_second')

get_first_property() → *OWLObjectPropertyExpression*

get_second_property() → *OWLObjectPropertyExpression*

__repr__()

Return repr(self).

class owlapy.model.**OWLEquivalentDataPropertiesAxiom**(
properties: List[OWLDataPropertyExpression],
annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLNaryPropertyAxiom*[*OWLDataPropertyExpression*], *OWLDataPropertyAxiom*

Represents EquivalentDataProperties axioms in the OWL 2 Specification.

__slots__ = ()

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

Bases: *OWLNaryPropertyAxiom*[*OWLDataPropertyExpression*], *OWLDataPropertyAxiom*

Represents DisjointDataProperties axioms in the OWL 2 Specification.

__slots__ = ()

class owlapy.model.**OWLSubClassOfAxiom**(*sub_class: OWLClassExpression*,
super_class: OWLClassExpression, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLClassAxiom*

Represents an SubClassOf axiom in the OWL 2 Specification.

__slots__ = ('_sub_class', '_super_class')

get_sub_class() → *OWLClassExpression*

get_super_class() → *OWLClassExpression*

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

class owlapy.model.**OWLDisjointUnionAxiom**(*cls_: OWLClass*, *class_expressions: List[OWLClassExpression]*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLClassAxiom*

Represents a DisjointUnion axiom in the OWL 2 Specification.

__slots__ = ('_cls', '_class_expressions')

get_owl_class() → *OWLClass*

get_class_expressions() → Iterable[*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.model.**OWLClassAssertionAxiom**(*individual: OWLIndividual*, *class_expression: OWLClassExpression*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLIndividualAxiom*

Represents ClassAssertion axioms in the OWL 2 Specification.

__slots__ = ('_individual', '_class_expression')

get_individual() → *OWLIndividual*

get_class_expression() → *OWLClassExpression*

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

class owlapy.model.**OWLAnnotationAxiom**(*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLAxiom*

A super interface for annotation axioms.

__slots__ = ()

is_annotation_axiom() → bool

class owlapy.model.**OWLAnnotationProperty**(*iri: _iri.IRI*)

Bases: *OWLProperty*

Represents an AnnotationProperty in the OWL 2 specification.

__slots__ = '_iri'

get_iri() → _iri.IRI

Gets the IRI of this object.

Returns

The IRI of this object.

class owlapy.model.**OWLAnnotation**(*property: OWLAnnotationProperty*, *value: _base.OWLAnnotationValue*)

Bases: _base.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() → _base.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.model.**OWLAnnotationAssertionAxiom**(*subject: _base.OWLAnnotationSubject*, *annotation: OWLAnnotation*)

Bases: *OWLAnnotationAxiom*

Represents AnnotationAssertion axioms in the OWL 2 specification.

__slots__ = ('_subject', '_annotation')

get_subject() → _base.OWLAnnotationSubject

Gets the subject of this object.

Returns
The subject.

get_property() → *OWLAnnotationProperty*

Gets the property.

Returns
The property.

get_value() → _base.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.model.**OWLSubAnnotationPropertyOfAxiom**(*sub_property: OWLAnnotationProperty*, *super_property: OWLAnnotationProperty*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLAnnotationAxiom*

Represents an SubAnnotationPropertyOf axiom in the OWL 2 specification.

__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.model.**OWLAnnotationPropertyDomainAxiom**(*property_: OWLAnnotationProperty*, *domain: _iri.IRI*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLAnnotationAxiom*

Represents an AnnotationPropertyDomain axiom in the OWL 2 specification.

__slots__ = ('_property', '_domain')

get_property() → *OWLAnnotationProperty*

get_domain() → _iri.IRI

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

class owlapy.model.**OWLAnnotationPropertyRangeAxiom**(*property_: OWLAnnotationProperty*, *range_: _iri.IRI*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLAnnotationAxiom*

Represents an AnnotationPropertyRange axiom in the OWL 2 specification.

__slots__ = ('_property', '_range')

get_property() → *OWLAnnotationProperty*

get_range() → _iri.IRI

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

class owlapy.model.**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.model.**OWLSubObjectPropertyOfAxiom**(
sub_property: OWLObjectPropertyExpression, *super_property: OWLObjectPropertyExpression*,
annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLSubPropertyAxiom*[*OWLObjectPropertyExpression*], *OWLObjectPropertyAxiom*

Represents a SubObjectPropertyOf axiom in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLSubDataPropertyOfAxiom**(*sub_property: OWLDataPropertyExpression*,
super_property: OWLDataPropertyExpression,
annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLSubPropertyAxiom*[*OWLDataPropertyExpression*], *OWLDataPropertyAxiom*

Represents a SubDataPropertyOf axiom in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLPropertyAssertionAxiom**(*subject: OWLIndividual*, *property_: _P*,
object_: _C, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: Generic[_P, _C], *OWLIndividualAxiom*

Represents a PropertyAssertion axiom in the OWL 2 specification.

__slots__ = ('_subject', '_property', '_object')

get_subject() → *OWLIndividual*

get_property() → _P

get_object() → _C

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

class owlapy.model.**OWLObjectPropertyAssertionAxiom**(*subject: OWLIndividual*,
property_: OWLObjectPropertyExpression, *object_: OWLIndividual*,
annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLPropertyAssertionAxiom*[*OWLObjectPropertyExpression*, *OWLIndividual*]

Represents an ObjectPropertyAssertion axiom in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLNegativeObjectPropertyAssertionAxiom**(
subject: OWLIndividual, *property_: OWLObjectPropertyExpression*, *object_: OWLIndividual*,
annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLPropertyAssertionAxiom*[*OWLObjectPropertyExpression*, *OWLIndividual*]

Represents a NegativeObjectPropertyAssertion axiom in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLDataPropertyAssertionAxiom**(*subject: OWLIndividual*, *property_: OWLDataPropertyExpression*, *object_: OWLLiteral*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLPropertyAssertionAxiom*[*OWLDataPropertyExpression*, *OWLLiteral*]

Represents an DataPropertyAssertion axiom in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLNegativeDataPropertyAssertionAxiom**(*subject: OWLIndividual*, *property_: OWLDataPropertyExpression*, *object_: OWLLiteral*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLPropertyAssertionAxiom*[*OWLDataPropertyExpression*, *OWLLiteral*]

Represents an NegativeDataPropertyAssertion axiom in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLUnaryPropertyAxiom**(*property_: _P*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: Generic[_P], *OWLPropertyAxiom*

Unary property axiom.

__slots__ = '_property'

get_property() → _P

class owlapy.model.**OWLObjectPropertyCharacteristicAxiom**(*property_: OWLObjectPropertyExpression*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLUnaryPropertyAxiom*[*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.model.**OWLFunctionalObjectPropertyAxiom**(*property_: OWLObjectPropertyExpression*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLObjectPropertyCharacteristicAxiom*

Represents FunctionalObjectProperty axioms in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLAsymmetricObjectPropertyAxiom**(*property_: OWLObjectPropertyExpression*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLObjectPropertyCharacteristicAxiom*

Represents AsymmetricObjectProperty axioms in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLInverseFunctionalObjectPropertyAxiom**(
property_: OWLObjectPropertyExpression, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLObjectPropertyCharacteristicAxiom*

Represents InverseFunctionalObjectProperty axioms in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLIrreflexiveObjectPropertyAxiom**(
property_: OWLObjectPropertyExpression, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLObjectPropertyCharacteristicAxiom*

Represents IrreflexiveObjectProperty axioms in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLReflexiveObjectPropertyAxiom**(
property_: OWLObjectPropertyExpression, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLObjectPropertyCharacteristicAxiom*

Represents ReflexiveObjectProperty axioms in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLSymmetricObjectPropertyAxiom**(
property_: OWLObjectPropertyExpression, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLObjectPropertyCharacteristicAxiom*

Represents SymmetricObjectProperty axioms in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLTransitiveObjectPropertyAxiom**(
property_: OWLObjectPropertyExpression, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLObjectPropertyCharacteristicAxiom*

Represents TransitiveObjectProperty axioms in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLDataPropertyCharacteristicAxiom**(
property_: OWLDataPropertyExpression, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLUnaryPropertyAxiom*[*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.model.**OWLFunctionalDataPropertyAxiom**(
property_: OWLDataPropertyExpression, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLDataPropertyCharacteristicAxiom*

Represents FunctionalDataProperty axioms in the OWL 2 specification.

__slots__ = ()

class owlapy.model.**OWLPropertyDomainAxiom**(*property_: _P*, *domain: OWLClassExpression*, *annotations: Iterable[OWLAnnotation] | None = None*)

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

Represents ObjectPropertyDomain axioms in the OWL 2 specification.

__slots__ = '_domain'

get_domain() → *OWLClassExpression*

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

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

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

Represents ObjectPropertyRange axioms in the OWL 2 specification.

__slots__ = '_range'

get_range() → _R

__eq__(*other*)

Return self==value.

__hash__()

Return hash(self).

__repr__()

Return repr(self).

class owlapy.model.**OWLObjectPropertyDomainAxiom**(
property_: OWLObjectPropertyExpression, *domain: OWLClassExpression*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLPropertyDomainAxiom*[*OWLObjectPropertyExpression*]

Represents a ObjectPropertyDomain axiom in the OWL 2 Specification.

__slots__ = ()

class owlapy.model.**OWLDataPropertyDomainAxiom**(*property_: OWLDataPropertyExpression*, *domain: OWLClassExpression*, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLPropertyDomainAxiom*[*OWLDataPropertyExpression*]

Represents a DataPropertyDomain axiom in the OWL 2 Specification.

__slots__ = ()

class owlapy.model.**OWLObjectPropertyRangeAxiom**(
property_: OWLObjectPropertyExpression, *range_: OWLClassExpression*,
annotations: Iterable[OWLAnnotation] | None = None)

Bases: *OWLPropertyRangeAxiom*[*OWLObjectPropertyExpression*, *OWLClassExpression*]

Represents a ObjectPropertyRange axiom in the OWL 2 Specification.

__slots__ = ()

class owlapy.model.**OWLDataPropertyRangeAxiom**(*property_: OWLDataPropertyExpression*,
range_: OWLDataRange, *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLPropertyRangeAxiom*[*OWLDataPropertyExpression*, *OWLDataRange*]

Represents a DataPropertyRange axiom in the OWL 2 Specification.

__slots__ = ()

class owlapy.model.**OWLOntology**

Bases: _base.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[*OWLClass*]

Gets the classes in the signature of this object.

Returns
Classes in the signature of this object.

abstract data_properties_in_signature() → Iterable[*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[*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[*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: OWLClass*) → Iterable[*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[*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: OWLDataProperty*) → Iterable[*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: OWLDataProperty*) → Iterable[*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: OWLObjectProperty*) → Iterable[*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: OWLObjectProperty*) → Iterable[*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.model.**OWLOntologyChange**(*ontology: OWLOntology*)

Represents an ontology change.

__slots__ = ()

get_ontology() → *OWLOntology*

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

Returns

The ontology that the change is applicable to.

class owlapy.model.**AddImport**(*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.model.**OWLOntologyManager**

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

abstract create_ontology(*iri: _iri.IRI*) → *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: _iri.IRI*) → *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: OWLOntology*, *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: OWLOntology*, *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: OWLOntology*, *document_iri: _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.model.**OWLReasoner**(*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: OWLDataProperty*, *direct: bool = False*)
→ Iterable[*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: OWLObjectProperty*, *direct: bool = False*) → Iterable[*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: OWLObjectProperty*, *direct: bool = False*) → Iterable[*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: OWLClassExpression*, *only_named: bool = True*) → Iterable[*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: OWLClassExpression*, *only_named: bool = True*) → Iterable[*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: OWLNamedIndividual*) → Iterable[*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: OWLNamedIndividual*) → Iterable[*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: OWLObjectPropertyExpression*) → Iterable[*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: OWLDataProperty*) → Iterable[*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: OWLNamedIndividual*, *pe: OWLDataProperty*, *direct: bool = True*) → Iterable[*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: OWLNamedIndividual*, *pe: OWLObjectPropertyExpression*, *direct: bool = True*) → Iterable[*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: OWLClassExpression*, *direct: bool = False*) → Iterable[*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: OWLClassExpression*, *direct: bool = False*, *only_named: bool = True*) → Iterable[*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: OWLObjectPropertyExpression*) → Iterable[*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: OWLDataProperty*) → Iterable[*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: OWLDataProperty*, *direct: bool = False*) → Iterable[*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: OWLDataProperty*, *direct: bool = False*) → Iterable[*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: OWLObjectPropertyExpression*, *direct: bool = False*) → Iterable[*OWLObjectPropertyExpression*]

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

Parameters

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

Returns

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

abstract **super_object_properties**(*op: OWLObjectPropertyExpression*, *direct: bool = False*) → Iterable[*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: OWLNamedIndividual*, *direct: bool = False*) → Iterable[*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() → *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 is_using_triplestore()

Return True if this reasoner is using a triplestore to retrieve instances.

abstract super_classes(*ce: OWLClassExpression*, *direct: bool = False*, *only_named: bool = True*) → Iterable[*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.

owlapy.model.**OWLThing: Final**

owlapy.model.**OWLNothing: Final**

owlapy.model.**OWLTopObjectProperty: Final**

owlapy.model.**OWLBottomObjectProperty: Final**

owlapy.model.**OWLTopDataProperty: Final**

owlapy.model.**OWLBottomDataProperty: Final**

owlapy.model.**DoubleOWLDatatype: Final**

owlapy.model.**IntegerOWLDatatype: Final**

owlapy.model.**BooleanOWLDatatype: Final**

owlapy.model.**StringOWLDatatype: Final**

owlapy.model.**DateOWLDatatype: Final**

owlapy.model.**DateTimeOWLDatatype: Final**

owlapy.model.**DurationOWLDatatype: Final**

owlapy.model.**TopOWLDatatype: Final**

owlapy.model.**NUMERIC_DATATYPES: Final[Set[*OWLDatatype*]]**

owlapy.model.**TIME_DATATYPES: Final[Set[*OWLDatatype*]]**

owlapy.owl2sparql

OWL-to-SPARQL converter.

Submodules

owlapy.owl2sparql.converter

Format converter.

Module Contents

Classes

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

Functions

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

Attributes

converter

owlapy.owl2sparql.converter.**peek**(*x*)

Peek the last element of an array.

Returns

The last element arr[-1].

class owlapy.owl2sparql.converter.**VariablesMapping**

Helper class for owl-to-sparql conversion.

__slots__ = ('class_cnt', 'prop_cnt', 'ind_cnt', 'dict')

get_variable(*e: owlapy.model.OWLEntity*) → str

new_individual_variable() → str

new_property_variable() → str

__contains__(*item: owlapy.model.OWLEntity*) → bool

__getitem__(*item: owlapy.model.OWLEntity*) → str

class owlapy.owl2sparql.converter.**Owl2SparqlConverter**

Convert owl (owlapy model class expressions) to SPARQL.

property modal_depth

property current_variable

__slots__ = ('ce', 'sparql', 'variables', 'parent', 'parent_var', 'properties', 'variable_entities', 'cnt',...

ce: *owlapy.model.OWLClassExpression*

sparql: List[str]

variables: List[str]

parent: List[*owlapy.model.OWLClassExpression*]

parent_var: List[str]

variable_entities: Set[*owlapy.model.OWLEntity*]

properties: Dict[int, List[*owlapy.model.OWLEntity*]]

mapping: *VariablesMapping*

grouping_vars: Dict[*owlapy.model.OWLClassExpression*, Set[str]]

having_conditions: Dict[*owlapy.model.OWLClassExpression*, Set[str]]

cnt: int

convert(*root_variable: str*, *ce: owlapy.model.OWLClassExpression*, *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]

abstract render(*e*)

stack_variable(*var*)

stack_parent(*parent: owlapy.model.OWLClassExpression*)

abstract process(*ce: owlapy.model.OWLClassExpression*)

new_count_var() → str

append_triple(*subject*, *predicate*, *object_*)

append(*frag*)

triple(*subject*, *predicate*, *object_*)

as_query(*root_variable: str*, *ce: owlapy.model.OWLClassExpression*, *count: bool = False*, *values: Iterable[owlapy.model.OWLNamedIndividual] | None = None*, *named_individuals: bool = False*)

owlapy.owl2sparql.converter.**converter**

owlapy.owl2sparql.converter.**owl_expression_to_sparql**(*root_variable: str = '?x'*, *ce: owlapy.model.OWLClassExpression = None*, *count: bool = False*, *values: Iterable[owlapy.model.OWLNamedIndividual] | None = None*, *named_individuals: bool = False*) → str

Convert an OWL Class Expression (https://www.w3.org/TR/owl2-syntax/#Class_Expressions) into a SPARQL query

2.2 Submodules

owlapy.io

Abstract renderer and parser classes.

Module Contents

Classes

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.

class owlapy.io.**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: owlapy.model.OWLObject*) → str

Render OWL Object to string.

Parameters
o – OWL Object.

Returns
String rendition of OWL object.

class owlapy.io.**OWLObjectParser**

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

abstract parse_expression(*expression_str: str*) → *owlapy.model.OWLObject*

Parse a string to an OWL Object.

Parameters
expression_str (*str*) – Expression string.

Returns
The OWL Object which is represented by the string.

owlapy.namespaces

Namespaces.

Module Contents

Classes

Namespaces	A Namespace and its prefix.

Attributes

OWL
RDFS
RDF
XSD

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

A Namespace and its prefix.

property ns: str

property prefix: str

__slots__ = ('_prefix', '_ns')

__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.parser

String to OWL parsers.

Module Contents

Classes

ManchesterOWLSyntaxParser	Manchester Syntax parser to parse strings to OWLClass-Expressions.
DLSyntaxParser	Description Logic Syntax parser to parse strings to OWL-ClassExpressions.

Functions

dl_to_owl_expression(dl_expression)
manchester_to_owl_expression(manchester_ex

Attributes

MANCHESTER_GRAMMAR
DL_GRAMMAR
DLparser
ManchesterParser

owlapy.parser.**MANCHESTER_GRAMMAR**

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

Bases: parsimonious.nodes.NodeVisitor, *owlapy.io.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.model.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.model.OWLClassExpression*

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

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

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

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

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

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

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

visit_class_expression(*node*, *children*) → *owlapy.model.OWLClassExpression*

visit_individual_list(*node*, *children*) → *owlapy.model.OWLObjectOneOf*

visit_data_primary(*node*, *children*) → *owlapy.model.OWLDataRange*

visit_data_some_only_res(*node*, *children*) → *owlapy.model.OWLQuantifiedDataRestriction*

visit_data_cardinality_res(*node*, *children*) → *owlapy.model.OWLDataCardinalityRestriction*

visit_data_value_res(*node*, *children*) → *owlapy.model.OWLDataHasValue*

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

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

visit_literal_list(*node*, *children*) → *owlapy.model.OWLDataOneOf*

visit_data_parentheses(*node*, *children*) → *owlapy.model.OWLDataRange*

visit_datatype_restriction(*node*, *children*) → *owlapy.model.OWLDatatypeRestriction*

visit_facet_restrictions(*node*, *children*) → List[*owlapy.model.OWLFacetRestriction*]

visit_facet_restriction(*node*, *children*) → *owlapy.model.OWLFacetRestriction*

visit_literal(*node*, *children*) → *owlapy.model.OWLLiteral*

visit_typed_literal(*node*, *children*) → *owlapy.model.OWLLiteral*

abstract visit_string_literal_language(*node*, *children*)

visit_string_literal_no_language(*node*, *children*) → *owlapy.model.OWLLiteral*

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

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

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

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

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

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

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

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

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

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

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

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

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

visit_individual_iri(*node*, *children*) → *owlapy.model.OWLNamedIndividual*

visit_object_property_iri(*node*, *children*) → *owlapy.model.OWLObjectProperty*

visit_data_property_iri(*node*, *children*) → *owlapy.model.OWLDataProperty*

visit_iri(*node*, *children*) → *owlapy.model.IRI*

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

abstract visit_abbreviated_iri(*node*, *children*)

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

visit_parentheses(*node*, *children*) → *owlapy.model.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.io.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.model.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.model.OWLClassExpression*

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

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

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

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

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

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

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

visit_class_expression(*node*, *children*) → *owlapy.model.OWLClassExpression*

visit_individual_list(*node*, *children*) → *owlapy.model.OWLObjectOneOf*

visit_data_primary(*node*, *children*) → *owlapy.model.OWLDataRange*

visit_data_some_only_res(*node*, *children*) → *owlapy.model.OWLQuantifiedDataRestriction*

visit_data_cardinality_res(*node*, *children*) → *owlapy.model.OWLDataCardinalityRestriction*

visit_data_value_res(*node*, *children*) → *owlapy.model.OWLDataHasValue*

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

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

visit_literal_list(*node*, *children*) → *owlapy.model.OWLDataOneOf*

visit_data_parentheses(*node*, *children*) → *owlapy.model.OWLDataRange*

visit_datatype_restriction(*node*, *children*) → *owlapy.model.OWLDatatypeRestriction*

visit_facet_restrictions(*node*, *children*) → List[*owlapy.model.OWLFacetRestriction*]

visit_facet_restriction(*node*, *children*) → *owlapy.model.OWLFacetRestriction*

visit_literal(*node*, *children*) → *owlapy.model.OWLLiteral*

visit_typed_literal(*node*, *children*) → *owlapy.model.OWLLiteral*

abstract visit_string_literal_language(*node*, *children*)

visit_string_literal_no_language(*node*, *children*) → *owlapy.model.OWLLiteral*

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

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

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

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

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

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

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

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

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

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

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

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

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

visit_individual_iri(*node*, *children*) → *owlapy.model.OWLNamedIndividual*

visit_object_property_iri(*node*, *children*) → *owlapy.model.OWLObjectProperty*

visit_data_property_iri(*node*, *children*) → *owlapy.model.OWLDataProperty*

visit_iri(*node*, *children*) → *owlapy.model.IRI*

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

abstract visit_abbreviated_iri(*node*, *children*)

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

visit_parentheses(*node*, *children*) → *owlapy.model.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*)

owlapy.parser.**manchester_to_owl_expression**(*manchester_expression: str*)

owlapy.render

Renderers for different syntax.

Module Contents

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)

Attributes

DLrenderer
ManchesterRenderer

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

Bases: *owlapy.io.OWLObjectRenderer*

DL Syntax renderer for OWL Objects.

__slots__ = '_sfp'

set_short_form_provider(*short_form_provider: Callable[[owlapy.model.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.model.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.model.OWLEntity], str] = _simple_short_form_provider*, *no_render_thing=False*)

Bases: *owlapy.io.OWLObjectRenderer*

Manchester Syntax renderer for OWL Objects

__slots__ = ('_sfp', '_no_render_thing')

set_short_form_provider(*short_form_provider: Callable[[owlapy.model.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.model.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.model.OWLObject*) → str

owlapy.render.**owl_expression_to_manchester**(*o: owlapy.model.OWLObject*) → str

owlapy.util

Owlapy utils.

Module Contents

Classes

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

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(→ owlapy.model.HasIndex)	Cast OWL Object to HasIndex.

class owlapy.util.**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.util.**NNF**

This class contains functions to transform a Class Expression into Negation Normal Form.

abstract get_class_nnf(*ce: owlapy.model.OWLClassExpression*, *negated: bool = False*) → *owlapy.model.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.util.**TopLevelCNF**

This class contains functions to transform a class expression into Top-Level Conjunctive Normal Form.

get_top_level_cnf(*ce: owlapy.model.OWLClassExpression*) → *owlapy.model.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.util.**TopLevelDNF**

This class contains functions to transform a class expression into Top-Level Disjunctive Normal Form.

get_top_level_dnf(*ce: owlapy.model.OWLClassExpression*) → *owlapy.model.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.util.**combine_nary_expressions**(*ce: owlapy.model.OWLClassExpression*) → *owlapy.model.OWLClassExpression*

owlapy.util.**combine_nary_expressions**(*ce: owlapy.model.OWLDataRange*) → *owlapy.model.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.util.**iter_count**(*i: Iterable*) → int

Count the number of elements in an iterable.

owlapy.util.**as_index**(*o: owlapy.model.OWLObject*) → *owlapy.model.HasIndex*

Cast OWL Object to HasIndex.

class owlapy.util.**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.

Module Contents

Classes

OWLRDFVocabulary	Enumerations for OWL/RDF vocabulary.
XSDVocabulary	Enumerations for XSD vocabulary.
OWLFacet	Enumerations for OWL facets.

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

MIN_INCLUSIVE: Final = ('minInclusive', '>=')

MIN_EXCLUSIVE: Final = ('minExclusive', '>')

MAX_INCLUSIVE: Final = ('maxInclusive', '<=')

MAX_EXCLUSIVE: Final = ('maxExclusive', '<')

LENGTH: Final = ('length', 'length')

MIN_LENGTH: Final = ('minLength', 'minLength')

MAX_LENGTH: Final = ('maxLength', 'maxLength')

PATTERN: Final = ('pattern', 'pattern')

TOTAL_DIGITS: Final = ('totalDigits', 'totalDigits')

FRACTION_DIGITS: Final = ('fractionDigits', 'fractionDigits')

static from_str(*name: str*) → *OWLFacet*

2.3 Package Contents

owlapy.**__version__ = '0.1.3'**

Python Module Index

O

Index

Non-alphabetical

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

V

X