Chowlk Visual Notation

A set of recommendations for ontology diagrams representation.

Contact:

María Poveda-Villalón
Contact email: chowlk@delicias.dia.fi.upm.es

Authors:

María Poveda-Villalón (Ontology Engineering Group, Universidad Politécnica de Madrid)
Raúl García-Castro (Ontology Engineering Group, Universidad Politécnica de Madrid)
Serge Chávez-Feria (Ontology Engineering Group, Universidad Politécnica de Madrid)
Sergio-Mario Carulli-Pérez (Ontology Engineering Group, Universidad Politécnica de Madrid)

Last update:

03-03-2025

Chowlk Diagrams.net Library (lightweight version):

Download

Chowlk Diagrams.net Library (complete version):

Download

Getting Started

The following video shows you how to start conceptualizing your ontology using diagrams.net and this notation.

1. Introduction

This document describes the Chowlk visual notation to construct ontology conceptualizations. It provides a set of visual blocks to represent each element from the OWL specification. The visual notation allows the representation of high-level as well as fine-grained constructs from the OWL language, giving the user the freedom to choose the level of expresiveness for their conceptualization.

2. Specification

This sections gives detailed information about the diagraming blocks used to represent the OWL elements used in the construction of an ontology. The specification is structured around the three main elements of an ontology: owl:Class, owl:ObjectProperty and owl:DatatypeProperty. Each table in the sub-sections contains not only the diagram block but also the equivalent owl code, and a description of the element.

2.1. Basic Elements

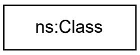
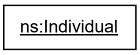
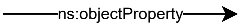
Diagram Block	Description	OWL Element
	Block to represent named and unnamed classes, as well as individual elements within the ontology conceptualization. The content of the block should be accompanied with the prefix and the name of the concept on order to fully identify it.	`owl:Class`
	Block to represent named and unnamed classes, as well as individual elements within the ontology conceptualization. The content of the block should be accompanied with the prefix and the name of the concept on order to fully identify it.	`owl:NamedIndividual`
	Standard way to represent object properties. Variations can apply to the type of line or the connections style depending on the range or domain specification. For	`owl:ObjectProperty`

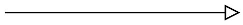
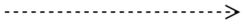
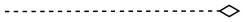
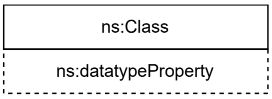
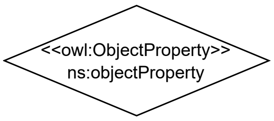
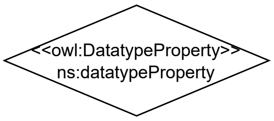
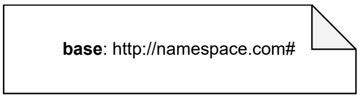
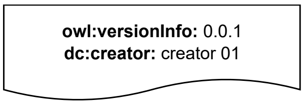
Diagram Block	Description	OWL Element
	more details see section 2.3.	
	Special arrow to indicate sub-class relationship between two classes.	`rdfs:subClassOf`
	Special arrow to represent `rdf:type` relationships.	`rdf:type`
	Special arrow to represent the elements involved in a list. For example, it connects a owl:unionOf axiom with all the concepts it is composed of.	
	Standard way to represent datatype properties attached to a specific owl:Class element. Variations can apply to the type of outer line depending on the domain and range specification. For more details see section 2.4.	`owl:DatatypeProperty`
	Alternative way to represent object properties.	`owl:ObjectProperty`
	Alternative way to represent datatype properties.	`owl:DatatypeProperty`
	Block to indicate all the namespaces used in the ontology. The first namespace is the URI used for the current ontology.	`@prefix base: <http://namespace.com#>`

Diagram Block	Description	OWL Element
	It is obligatory to include all the namespaces being used in order to use the ontology converter service.	
	Block to indicate the annotation properties describing the ontology. The annotations in use should include the prefix and the annotation name, as indicated in the figure. If custom annotations are utilized, the namespace block should include the prefixes and namespaces for those annotation properties.	`owl:AnnotationProperty`

2.2. Namespaces

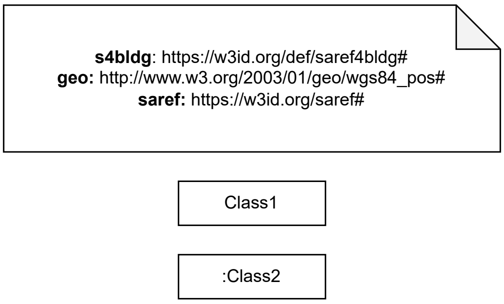
By default chowlk provides the following namespaces:

- **owl**: <http://www.w3.org/2002/07/owl#>
- **rdf**: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
- **rdfs**: <http://www.w3.org/2000/01/rdf-schema#>
- **xml**: <http://www.w3.org/XML/1998/namespace>
- **xsd**: <http://www.w3.org/2001/XMLSchema#>
- **dc**: <http://purl.org/dc/elements/1.1/>
- **dcterms**: <http://purl.org/dc/terms/>
- **vann**: <http://purl.org/vocab/vann/>

Note: In order to declare the ontology base, it is neccesary to use "@base". If it is used "base", a prefix base is created instead.

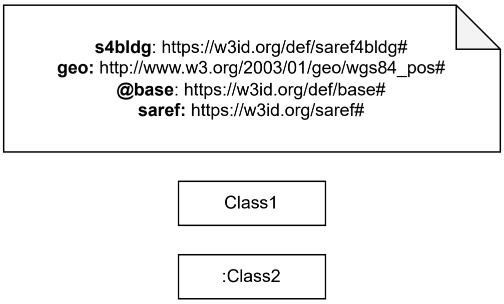
Definition of @base and empty prefix when they are not specified in the namespaces.

Diagram Block	OWL Element

 s4bldg: https://w3id.org/def/saref4bldg#
geo: http://www.w3.org/2003/01/geo/wgs84_pos#
saref: https://w3id.org/saref#
Class1
:Class2
	`prefix : <https://w3id.org/def/saref4bldg#> .`
	`...`
	`@base <https://w3id.org/def/saref4bldg#> .`
	`<https://w3id.org/def/saref4bldg#Class1> a owl:Class .`
	`:Class2 a owl:Class .`

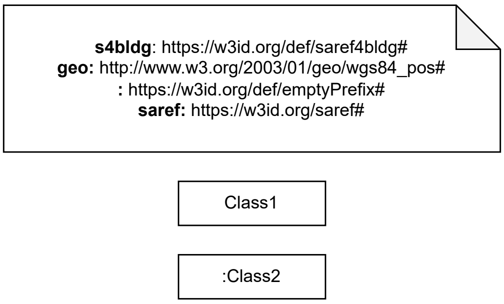
Definition of @base and empty prefix when a base is specified in the namespaces but the empty prefix is not specified.

Diagram Block	OWL Element

 s4bldg: https://w3id.org/def/saref4bldg#
geo: http://www.w3.org/2003/01/geo/wgs84_pos#
@base: https://w3id.org/def/base#
saref: https://w3id.org/saref#
Class1
:Class2
	`prefix : <https://w3id.org/def/base#> .`
	`...`
	`@base <https://w3id.org/def/base#> .`
	`<https://w3id.org/def/base#Class1> a owl:Class .`
	`:Class2 a owl:Class .`

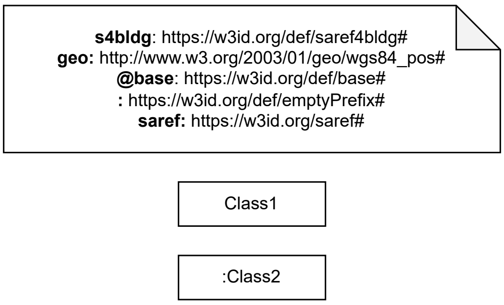
Definition of @base and empty prefix when a base is not specified in the namespaces but the empty prefix is specified.

Diagram Block	OWL Element

 s4bldg: https://w3id.org/def/saref4bldg#
geo: http://www.w3.org/2003/01/geo/wgs84_pos#
: https://w3id.org/def/emptyPrefix#
saref: https://w3id.org/saref#
Class1
:Class2
	`prefix : <https://w3id.org/def/emptyPrefix#> .`
	`...`
	`@base <https://w3id.org/def/saref4bldg#> .`
	`<https://w3id.org/def/saref4bldg#Class1> a owl:Class .`
	`:Class2 a owl:Class .`

Definition of @base and empty prefix when they are specified in namespaces.

Diagram Block	OWL Element

 s4bldg: https://w3id.org/def/saref4bldg#
geo: http://www.w3.org/2003/01/geo/wgs84_pos#
@base: https://w3id.org/def/base#
: https://w3id.org/def/emptyPrefix#
saref: https://w3id.org/saref#
Class1
:Class2
	`prefix : <https://w3id.org/def/emptyPrefix#> .`
	`...`
	`@base <https://w3id.org/def/base#> .`
	`<https://w3id.org/def/base#Class1> a owl:Class .`
	`:Class2 a owl:Class .`

2.3. Ontology Metadata

The metadata associated to the ontology itself is important in order to provide an overview and identify an ontology, understand its usage conditions and understand its provenance. We recommend the following properties. The optional properties are not critical to identify or reuse a target ontology. However, they provide additional information to understand the vocabulary.

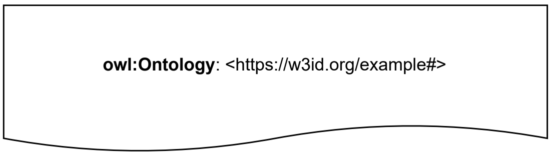
▶ More Information

Property Name	Annotation Property	Rationale	Guideline
License	`dcterms:license`	Usage conditions	Recommended
Creator	`dcterms:creator`	Provenance and attribution	Recommended
Contributor	`dcterms:contributor`	Provenance and attribution	Recommended
Creation date	`dcterms:created`	Provenance	Recommended
Previous version	`owl:priorVersion`	Provenance and comparison	Recommended
Namespace URI	`vann:preferredNamespaceUri`	Identifying the ontology	Recommended
Version IRI	`owl:versionIRI`	Versioning	Recommended
Prefix	`vann:preferredNamespacePrefix`	Identifying the ontology	Recommended
Title	`dcterms:title`	Understanding	Recommended
Description	`dcterms:description`	Understanding	Recommended
Citation	`dcterms:bibliographicCitation`	Credit	Recommended
Abstract	`dcterms:abstract`	Additional information	Optional
See also	`rdfs:seeAlso`	Additional information	Optional
Status	`sw:status`	Maturity information	Optional
Backward compatibility	`owl:backwardCompatibleWith`	Version compatibility	Optional
Incompatibility	`owl:incompatibleWith`	Version compatibility	Optional
Modification Date	`dcterms:modified`	Provenance and timeliness	Optional
Issued date	`dcterms:issued`	Provenance and timeliness	Optional
Source	`dcterms:source`	Provenance	Optional

Property Name	Annotation Property	Rationale	Guideline
Publisher	`dcterms:publisher`	Provenance	Optional
DOI	`bibo:doi`	Bibliographic information	Optional
Logo	`foaf:logo`	Identifying the ontology	Optional
Diagram	`foaf:depiction`	Visual documentation	Optional

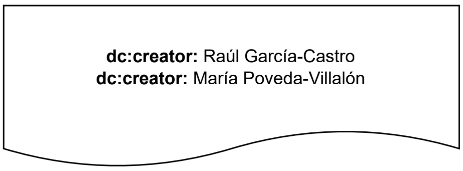
The ontology uri can be defined using the declaration **owl:Ontology:**. If the ontology uri is not defined, then the base uri is taken as the ontology uri. In the case that a base uri is undifined, the first defined prefixed is taken as the ontology uri.

Notation Example	OWL Code

 **owl:Ontology**: <https://w3id.org/example#>	`<https://w3id.org/example#>`
	`a owl:Ontology .`

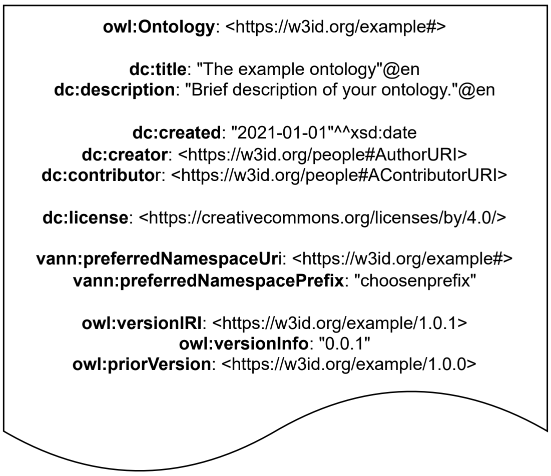
The same ontology metadata can be defined more than once if it has more than one value.

Notation Example	OWL Code

 **dc:creator:** Raúl García-Castro
dc:creator: María Poveda-Villalón
	`dc:creator "Raúl`
	`García-Castro",`
	`"María Poveda-`
	`Villalón" .`

An example providing the recommended ontology metadata is shown below:

Notation Example	OWL Code

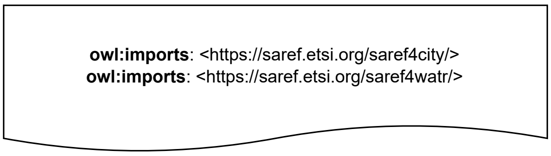
 **owl:Ontology**: <https://w3id.org/example#>
 	
dc:title: "The example ontology"@en
dc:description: "Brief description of your ontology."@en
 	
dc:created: "2021-01-01"^^xsd:date
dc:creator: <https://w3id.org/people#AuthorURI>
dc:contributor: <https://w3id.org/people#AContributorURI>
 	
dc:license: <https://creativecommons.org/licenses/by/4.0/>
 	
vann:preferredNamespaceUri: <https://w3id.org/example#>
vann:preferredNamespacePrefix: "choosenprefix"
 	
owl:versionIRI: <https://w3id.org/example/1.0.1>
owl:versionInfo: "0.0.1"
owl:priorVersion: <https://w3id.org/example/1.0.0>
	`<https://w3id.org/example#> a owl:Ontology ;`
	`dc:title "The example ontology"@en ;`
	`dc:description "Brief description of your`
	`ontology."@en ;`
	`dc:created "2021-01-01"^^xsd:date ;`
	`dc:creator <https://w3id.org/people#AuthorUR`
	`;`
	`dc:contributor`
	`<https://w3id.org/people#AContributorURI> ;`
	`dc:license`
	`<https://creativecommons.org/licenses/by/4.0`
	`;`
	`vann:preferredNamespaceUri`
	`<https://w3id.org/example#> ;`
	`vann:preferredNamespacePrefix "choosenprefix`
	`;`
	`owl:versionIRI`

Notation Example	OWL Code

	`<https://w3id.org/example/1.0.1> ;`
	`owl:versionInfo "0.0.1" .`
	`owl:priorVersion`
	`<https://w3id.org/example/1.0.0> ;`

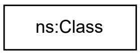
`owl:imports` can be defined too. If more than one ontology are going to be imported, they have to be in different lines.

Notation Example	OWL Code

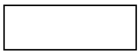
 **owl:imports**: <https://saref.etsi.org/saref4city/>
owl:imports: <https://saref.etsi.org/saref4watr/>
	`owl:imports`
	`<https://saref.etsi.org/saref4city/>,`
	`<https://saref.etsi.org/saref4watr/>`
	`.`

2.4. Class Definition

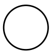
Definition of a named class.

Diagram Block	OWL Element
	`owl:Class`

Definition of an unnamed class to represent property restrictions and `owl:complementOf`.

Diagram Block	OWL Element
	`owl:Class`

Definition of an unnamed class to represent logical combinations between other classes, such as AND or OR operators.

Diagram Block	OWL Element
	`owl:Class`

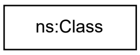
2.5. Class Descriptions

A class description describes an OWL class, either by a class name or by specifying the class extension of an unnamed anonymous class. OWL distinguishes six types of class descriptions:

1. a **class identifier** (a URI reference)
2. an exhaustive **enumeration** of individuals that together form the instances of a class
3. a property **restriction**
4. the **intersection** of two or more class descriptions
5. the **union** of two or more class descriptions
6. the **complement** of a class description

The first type is special in the sense that it describes a class through a **class name** (syntactically represented as a URI reference). The other five types of class descriptions describe an **anonymous class** by placing constraints on the class extension.

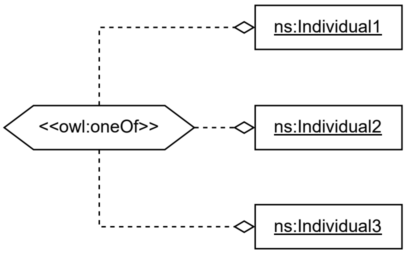
2.5.1. Class Identifier

Diagram Block	OWL Element
	`ns:Class a owl:Class`

2.5.2. Enumeration

A class description of the "enumeration" kind is defined with the `owl:oneOf` property. The value of this built-in OWL property must be a list of individuals that are the instances of the class. This enables a class to be described by exhaustively enumerating its instances. The class extension of a class described with owl:oneOf contains exactly the enumerated individuals, no more, no less.

Diagram Block	OWL Element

 ns:Individual1
<<owl:oneOf>>
ns:Individual2
ns:Individual3
	`owl:oneOf`
	`(ns:Individual1`
	`ns:Individual2`
	`ns:Individual3)`

2.5.3. Restriction

A property restriction is a special kind of class description. It describes an anonymous class, namely a class of all individuals that satisfy the restriction. OWL distinguishes two kinds of property restrictions: value constraints and cardinality constraints.

- A value constraint puts constraints on the range of the property when applied to this particular class description.
- A cardinality constraint puts constraints on the number of values a property can take, in the context of this particular class description.

Property restrictions can be applied both to datatype properties (properties for which the value is a data literal) and object properties (properties for which the value is an individual).

2.5.3.1. Value Constraints

The value constraint `owl:allValuesFrom` is a built-in OWL property that links a restriction class to either a class description or a data range. A restriction containing an `owl:allValuesFrom` constraint is used to describe a class of all individuals for which all values of the property under consideration are either members of the class extension of the class description or are data values within the specified data range. In other words, it defines a class of individuals x for which holds that if the pair (x,y) is an instance of P (the property concerned), then y should be an instance of the class description or a value in the data range, respectively.

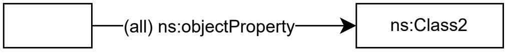
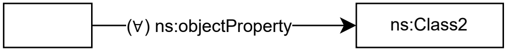
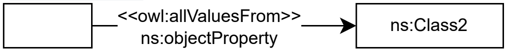
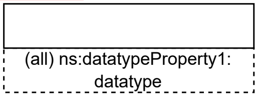
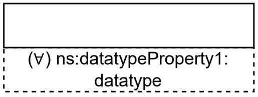
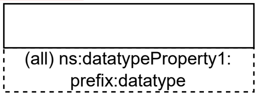
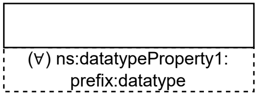
Diagram Block	OWL Element
Preferred 	`rdf:type owl:Restriction ; owl:onProperty ns:objectProperty ; owl:allValuesFrom ns:Class2 .`
Alternative 	
Alternative 	

Diagram Block	OWL Element
Preferred 	`rdf:type owl:Restriction ; owl:onProperty ns:datatypeProperty1 ; owl:allValuesFrom xsd:datatype .`
Alternative 	

Moreover, costum data values can be defined specifying the prefixes of the datatypes. By default the prefix is xsd.

Diagram Block	OWL Element
Preferred 	`rdf:type owl:Restriction ; owl:onProperty ns:datatypeProperty1 ; owl:allValuesFrom prefix:datatype .`
Alternative 	

The value constraint `owl:someValuesFrom` is a built-in OWL property that links a restriction class to a class description or a data range. A restriction containing an `owl:someValuesFrom` constraint describes a class of all individuals for which at least one value of the property concerned is an instance of the class description or a data value in the data range. In other words, it defines a class of individuals x for which there is at least one y (either an instance of the class description or value of the data range) such that the pair (x,y) is an instance of P. This does not exclude that there are other instances (x,y') of P for which y' does not belong to the class description or data range.

Diagram Block	OWL Element
Preferred	`rdf:type owl:Restriction ;`

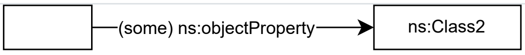
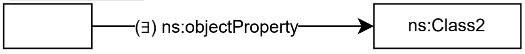
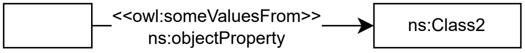
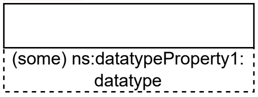
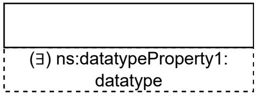
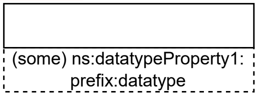
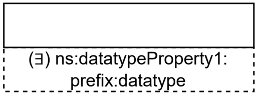
Diagram Block	OWL Element
	owl:onProperty ns:objectProperty ; owl:someValuesFrom ns:Class2 .
Alternative 	
Alternative  <<owl:someValuesFrom>> ns:objectProperty ns:Class2	

Diagram Block	OWL Element
Preferred 	rdf:type owl:Restriction ; owl:onProperty datatypeProperty1 ; owl:someValuesFrom xsd:datatype .
Alternative 	

Moreover, costum data values can be defined specifying the prefixes of the datatypes. By default the prefix is xsd.

Diagram Block	OWL Element
Preferred 	rdf:type owl:Restriction ; owl:onProperty datatypeProperty1 ; owl:someValuesFrom prefix:datatype .
Alternative 	

The value constraint `owl:hasValue` is a built-in OWL property that links a restriction class to a value V, which can be either an individual or a data value. A restriction containing a `owl:hasValue` constraint describes a class of all individuals for which the property concerned has at least one value semantically equal to V (it may have other values as well).

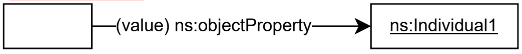
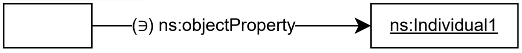
Diagram Block	OWL Element
Preferred 	rdf:type owl:Restriction ; owl:onProperty ns:objectProperty ; owl:hasValue ns:Individual1 .
Alternative 	
Alternative	

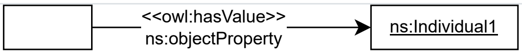
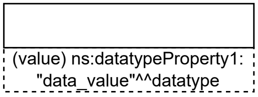
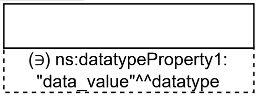
Diagram Block	OWL Element
	

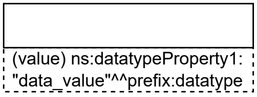
Diagram Block	OWL Element

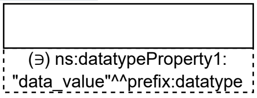
Preferred

	`rdf:type owl:Restriction ;`
	`owl:onProperty datatypeProperty1 ;`
	`owl:hasValue "data_value"^^xsd:datatype .`

Alternative 	

Moreover, costum data values can be defined specifying the prefixes of the datatypes. By default the prefix is xsd.

Diagram Block	OWL Element

Preferred
 (value) ns:datatypeProperty1:
"data_value"^^prefix:datatype
	`rdf:type owl:Restriction ;`
	`owl:onProperty datatypeProperty1 ;`
	`owl:hasValue "data_value"^^prefix:datatype .`

Alternative 	

2.5.3.2. Cardinality Constraints

Cardinality restriction of a concept on an object property. The `ns:Class1` class is subclass of an anonymus concept which has an object property `ns:objectProperty`, and should have at least N1 and at most N2 individuals for that property. If the N2 element is equal to the letter N, it means `owl:maxCardinality` does not exist. If the N1 element is equal to 0, it means `owl:minCardinality` does not exist. If the N1 is equal to N2, it means `owl:cardinality` does exist

Cardinality restriction of a concept on a datatype property. The `ns:Class` concept is subclass of an anonymus concept which has an datatype property `ns:datatypeProperty`, and shall have at least N1 and at most N2 values for that property. If the N2 element is equal to the letter N, it means `owl:maxCardinality` does not exist. If the N1 element is equal to 0, it means `owl:minCardinality` does not exist. If the N1 is equal to N2, it means `owl:cardinality` does exist

The cardinality constraint `owl:maxCardinality` is a built-in OWL property that links a restriction class to a data value belonging to the value space of the XML Schema datatype nonNegativeInteger. A restriction containing an `owl:maxCardinality` constraint describes a class of all individuals that have at most N semantically distinct values (individuals or data values) for the property concerned, where N is the value of the cardinality constraint.

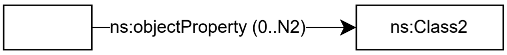
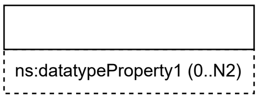
Diagram Block	OWL Element
	rdf:type owl:Restriction ; owl:onProperty ns:objectProperty ; owl:maxCardinality "N2"^^xsd:nonNegativeInteger

Diagram Block	OWL Element
	rdf:type owl:Restriction ; owl:onProperty ns:datatypeProperty1 ; owl:maxCardinality "N2"^^xsd:nonNegativeInteger

The cardinality constraint `owl:minCardinality` is a built-in OWL property that links a restriction class to a data value belonging to the value space of the XML Schema datatype nonNegativeInteger. A restriction containing an `owl:minCardinality` constraint describes a class of all individuals that have at least N semantically distinct values (individuals or data values) for the property concerned, where N is the value of the cardinality constraint.

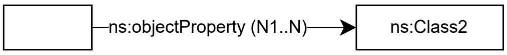
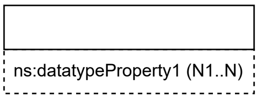
Diagram Block	OWL Element
	rdf:type owl:Restriction ; owl:onProperty ns:objectProperty ; owl:minCardinality "N1"^^xsd:nonNegativeInteger

Diagram Block	OWL Element
	rdf:type owl:Restriction ; owl:onProperty ns:datatypeProperty1 ; owl:minCardinality "N1"^^xsd:nonNegativeInteger

The cardinality constraint `owl:cardinality` is a built-in OWL property that links a restriction class to a data value belonging to the range of the XML Schema datatype nonNegativeInteger. A restriction containing an `owl:cardinality` constraint describes a class of all individuals that have exactly N semantically distinct values (individuals or data values) for the property concerned, where N is the value of the cardinality constraint.

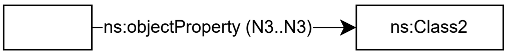
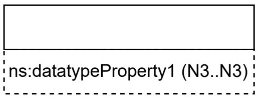
Diagram Block	OWL Element
	rdf:type owl:Restriction ; owl:onProperty ns:objectProperty ; owl:cardinality "N3"^^xsd:nonNegativeInteger

Diagram Block	OWL Element

	`rdf:type owl:Restriction ;`
	`owl:onProperty`
	`ns:datatypeProperty1 ;`
	`owl:cardinality`
	`"N3"^^xsd:nonNegativeInteger`

2.5.3.3. Qualified Cardinality Constraints

Qualified cardinality restriction of a concept on an object property. The `ns:Class1` class is subclass of an anonymus concept which has an object property `ns:objectProperty`, and should have at least N1 and at most N2 individuals from class `ns:Class2`. If the N2 element is equal to the letter N, it means `owl:maxQualifiedCardinality` does not exist. If the N1 element is equal to 0, it means `owl:minQualifiedCardinality` does not exist. If the N1 is equal to N2, it means `owl:qualifiedCardinality` does exist

Qualified cardinality restriction of a concept on a datatype property. The `ns:Class` concept is subclass of an anonymus concept which has an datatype property `ns:datatypeProperty`, and shall have at least N1 and at most N2 values. If the N2 element is equal to the letter N, it means `owl:maxQualifiedCardinality` does not exist. If the N1 element is equal to 0, it means `owl:minQualifiedCardinality` does not exist. If the N1 is equal to N2, it means `owl:qualifiedCardinality` does exist

The qualified cardinality constraint `owl:maxQualifiedCardinality` is a built-in OWL property that links a restriction class to a data value belonging to the value space of the XML Schema datatype nonNegativeInteger. A restriction containing an `owl:maxQualifiedCardinality` constraint describes a class of all individuals that have at most N values (individuals or data values) for the property concerned, where N is the value of the qualified cardinality constraint.

Diagram Block	OWL Element

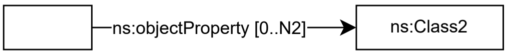
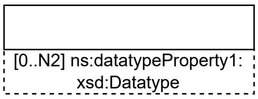
	`rdf:type owl:Restriction ;`
	`owl:onProperty`
	`ns:objectProperty ;`
	`owl:maxQualifiedCardinality`
	`"N2"^^xsd:nonNegativeInteger`
	`;`
	`owl:onClass ns:Class2 .`

Diagram Block	OWL Element

	`rdf:type owl:Restriction ;`
	`owl:onProperty`
	`ns:datatypeProperty1 ;`
	`owl:maxQualifiedCardinality`
	`"N2"^^xsd:nonNegativeInteger`
	`;`
	`owl:onDataRange xsd:Datatype`
	`.`

The cardinality constraint `owl:minQualifiedCardinality` is a built-in OWL property that links a restriction class to a data value belonging to the value space of the XML Schema datatype nonNegativeInteger. A restriction containing an `owl:minQualifiedCardinality` constraint describes a class of all individuals that have at least N values (individuals or data values) for the property concerned, where N is the value of the qualified cardinality constraint.

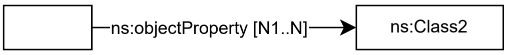
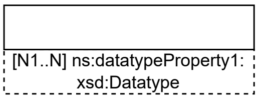
Diagram Block	OWL Element
	`rdf:type owl:Restriction ; owl:onProperty ns:objectProperty ; owl:minQualifiedCardinality "N1"^^xsd:nonNegativeInteger ; owl:onClass ns:Class2 .`

Diagram Block	OWL Element
	`rdf:type owl:Restriction ; owl:onProperty ns:datatypeProperty1 ; owl:minQualifiedCardinality "N1"^^xsd:nonNegativeInteger ; owl:onDataRange xsd:Datatype .`

The cardinality constraint `owl:qualifiedCardinality` is a built-in OWL property that links a restriction class to a data value belonging to the range of the XML Schema datatype nonNegativeInteger. A restriction containing an `owl:qualifiedCardinality` constraint describes a class of all individuals that have exactly N values (individuals or data values) for the property concerned, where N is the value of the qualified cardinality constraint.

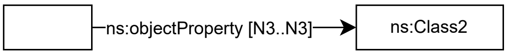
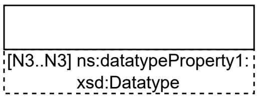
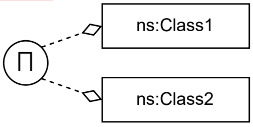
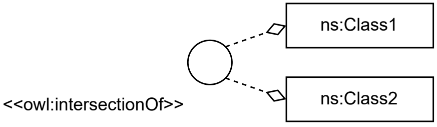
Diagram Block	OWL Element
	`rdf:type owl:Restriction ; owl:onProperty ns:objectProperty ; owl:qualifiedCardinality "N3"^^xsd:nonNegativeInteger ; owl:onClass ns:Class2 .`

Diagram Block	OWL Element
	`rdf:type owl:Restriction ; owl:onProperty ns:datatypeProperty1 ; owl:qualifiedCardinality "N3"^^xsd:nonNegativeInteger ; owl:onDataRange xsd:Datatype .`

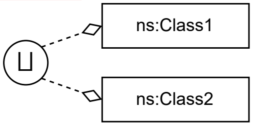
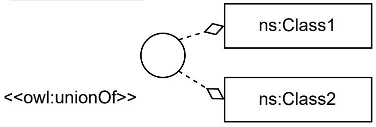
2.5.4. Intersection

It can be viewed as representing the AND operator on classes. The `owl:intersectionOf` property links a class to a list of class descriptions. An `owl:intersectionOf` statement describes a class for which the class extension contains precisely those individuals that are members of the class extension of all class descriptions in the list.

Diagram Block	OWL Element
Preferred 	`owl:intersectionOf (ns:Class1 ns:Class2)`
Alternative 	

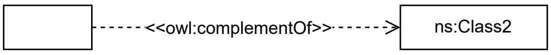
2.5.5. Union

It can be viewed as representing the OR operator on classes. The `owl:unionOf` property links a class to a list of class descriptions. An `owl:unionOf` statement describes an anonymous class for which the class extension contains those individuals that occur in at least one of the class extensions of the class descriptions in the list.

Diagram Block	OWL Element
Preferred 	`owl:unionOf (ns:Class1 ns:Class2)`
Alternative 	

2.5.6. Complement

It can be viewed as representing the NOR operator on classes. The `owl:complementOf` property links a class to precisely one class description. An `owl:complementOf` statement describes a class for which the class extension contains exactly those individuals that do not belong to the class extension of the class description that is the object of the statement. `owl:complementOf` is analogous to logical negation: the class extension consists of those individuals that are NOT members of the class extension of the complement class.

Diagram Block	OWL Element
	`owl:complementOf (ns:Class2)`

2.6. Class Axioms

Class descriptions form the building blocks for defining classes through class axioms. OWL contains three language constructs for combining class descriptions into class axioms:

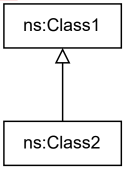
- **rdfs:subClassOf** allows one to say that the class extension of a class description is a subset of the class extension of another class description.
- **owl:equivalentClass** allows one to say that a class description has exactly the same class extension as another class description.
- **owl:disjointWith** allows one to say that the class extension of a class description has no members in common with the class extension of another class description.

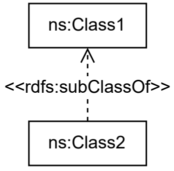
▶ More Information

2.6.1. Subclass

Graphical representations to indicate that `ns:Class2` concept is sub-class of `ns:Class1`

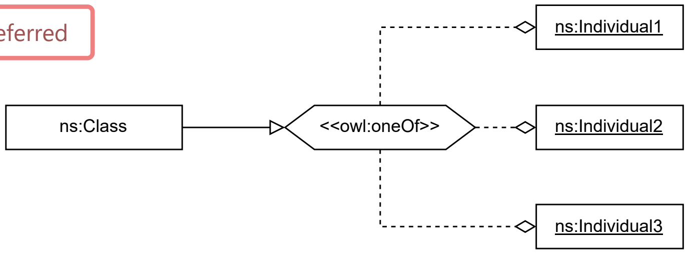
Notation Example	OWL Code

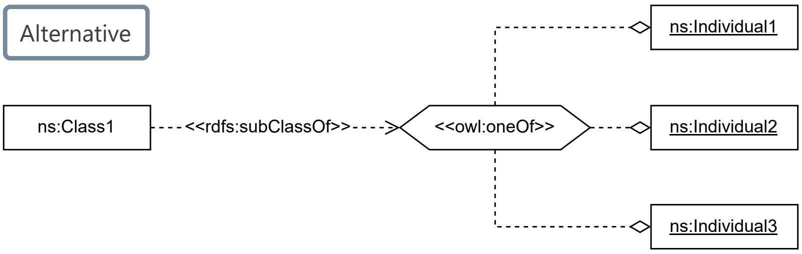
Preferred

	`ns:Class2 rdfs:subClassOf ns:Class1 .`
	
	`ns:Class1 a owl:Class .`

Alternative 	

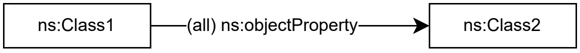
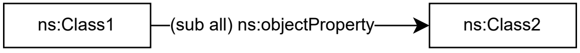
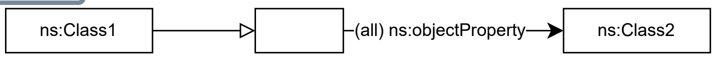
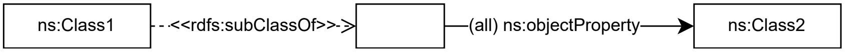
Graphical representations to indicate that `ns:Class` concept is sub-class of an anonymous class which represents an **enumeration**

Notation Example	OWL Code

Preferred
 ns:Individual1
ns:Class
<<owl:oneOf>>
ns:Individual2
ns:Individual3
	`ns:Class a owl:Class ; rdfs:subClassOf [ a owl:Class ; owl:oneOf ( ns:Individual1 ns:Individual2 ns:Individual3 ) ] .`
	
	`ns:Individual1 a`

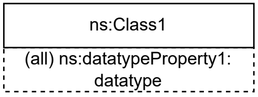
Notation Example	OWL Code
Alternative  ns:Class1 <<rdfs:subClassOf>> <<owl:oneOf>> ns:Individual1 ns:Individual2 ns:Individual3	owl:NamedIndividual . ns:Individual2 a owl:NamedIndividual . ns:Individual3 a owl:NamedIndividual .

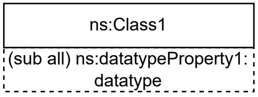
Graphical representations to indicate that `ns:Class` concept is subclass of an anonymous class which represents a property restriction. Note that all the **property restrictions explained above applied** with the subclass class axiom. However, just the `owl:allValuesFrom` notation was drawn in order to not overload this section.

Notation Example	OWL Code
Preferred 	ns:Class1 a owl:Class ; rdfs:subClassOf [a owl:Restriction ; owl:allValuesFrom ns:Class2 ; owl:onProperty ns:objectProperty] . ns:Class2 a owl:Class . ns:objectProperty a owl:ObjectProperty ; rdfs:domain ns:Class1 ; rdfs:range ns:Class2 .
Alternative 	
Alternative 	ns:Class1 a owl:Class ; rdfs:subClassOf [a owl:Restriction ; owl:allValuesFrom ns:Class2 ; owl:onProperty ns:objectProperty] . ns:Class2 a owl:Class . ns:objectProperty
Alternative  ns:Class1 <<rdfs:subClassOf>> (all) ns:objectProperty ns:Class2	

Notation Example	OWL Code
	`a owl:ObjectProperty ; rdfs:range ns:Class 2 .`

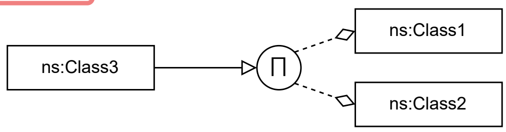
Notation Example	OWL Code

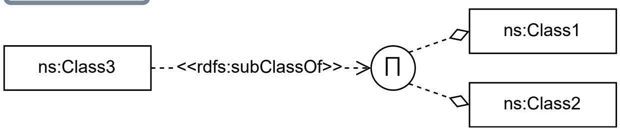
Preferred 	`ns:Class1 a owl:Class ; rdfs:subClassOf [ a owl:Restriction ; owl:allValuesFrom xsd:datatype ; owl:onProperty ns:datatypeProperty1 ] .`
	
	`ns:datatypeProperty1 a owl:DatatypeProperty ; rdfs:range xsd:datatype .`

Alternative 	

Graphical representations to indicate that `ns:Class` concept is sub-class of an anonymous class which represents an **intersection**

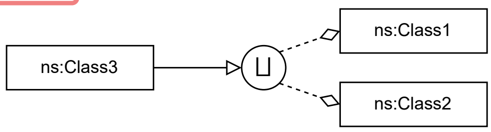
Notation Example	OWL Code

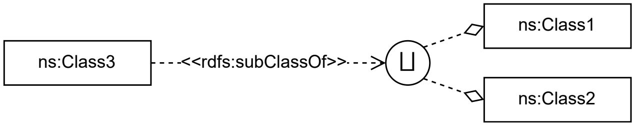
Preferred 	`ns:Class3 a owl:Class ; rdfs:subClassOf [ a owl:Class ; owl:intersectionOf ( ns:Class1 ns:Class2 ) ] .`
	
	`ns:Class1 a owl:Class .`
	
	`ns:Class2 a owl:Class .`

Alternative  ns:Class3 <<rdfs:subClassOf>> ⊓ ns:Class1 ns:Class2	

Graphical representations to indicate that `ns:Class` concept is sub-class of an anonymous class which represents an **union**

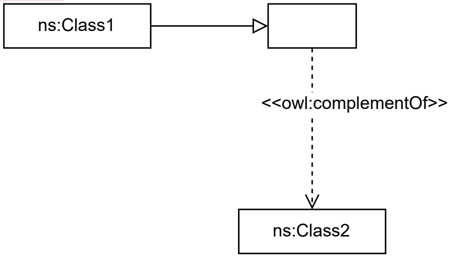
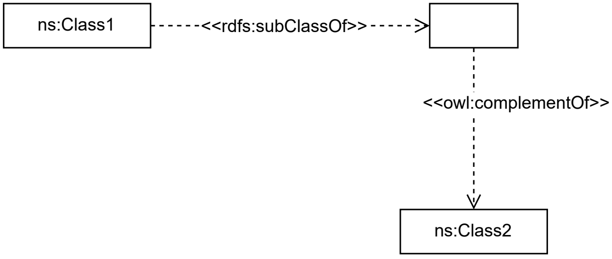
Notation Example	OWL Code

Preferred 	`ns:Class3 a owl:Class ; rdfs:subClassOf [ a owl:Class ; owl:unionOf ( ns:Class1 ns:Class2 ) ] .`
	
	`ns:Class1 a owl:Class .`

Alternative  ns:Class3 <<rdfs:subClassOf>> ⊔ ns:Class1 ns:Class2	

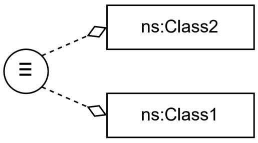
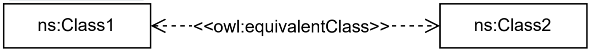
Notation Example	OWL Code
	ns:Class2 a owl:Class .

Graphical representations to indicate that ns:Class concept is subclass of an anonymous class which represents a **complement**

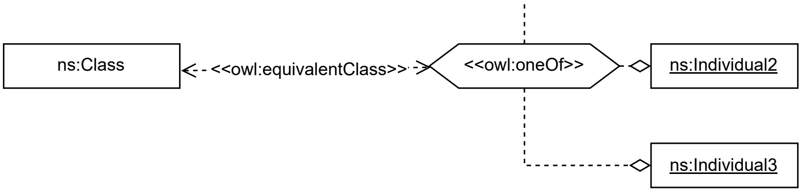
Notation Example	OWL Code
Preferred 	ns:Class1 a owl:Class ; rdfs:subClassOf [a owl:Class ; owl:complementOf ns:Class2] . ns:Class2 a owl:Class .
Alternative  ns:Class1 <<rdfs:subClassOf>> <<owl:complementOf>> ns:Class2	

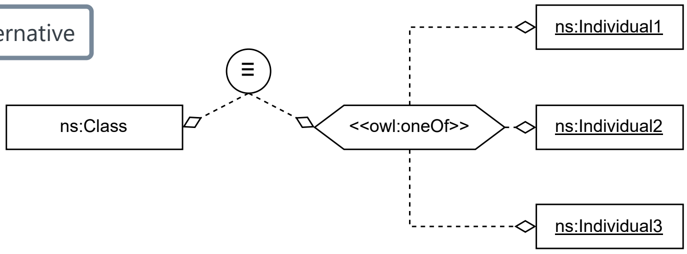
2.6.2. Equivalent

Graphical representations to indicate that ns:Class2 concept is equivalent to ns:Class1

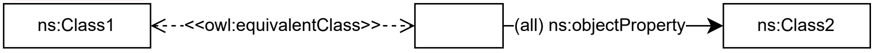
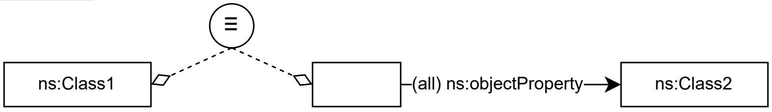
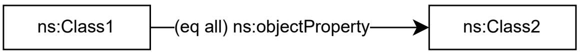
Notation Example	OWL Code
Preferred 	ns:Class1 owl:equivalentClass ns:Class2 . ns:Class2 a owl:Class .
Alternative 	

Graphical representations to indicate that ns:Class concept is equivalent to an anonymous class which represents an **enumeration**

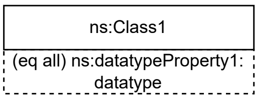
Notation Example	OWL Code
Preferred  ns:Individual1 ns:Class <<owl:equivalentClass>> <<owl:oneOf>> ns:Individual2 ns:Individual3	ns:Class a owl:Class ; owl:equivalentClass [a owl:Class ; owl:oneOf (ns:Individual1 ns:Individual2 ns:Individual3)]

Notation Example	OWL Code
	.
Alternative  ≡ ns:Class <<owl:oneOf>> ns:Individual1 ns:Individual2 ns:Individual3	ns:Individual1 a owl:NamedIndividual . ns:Individual2 a owl:NamedIndividual . ns:Individual3 a owl:NamedIndividual .

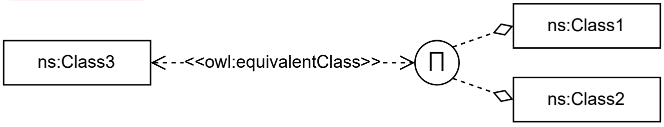
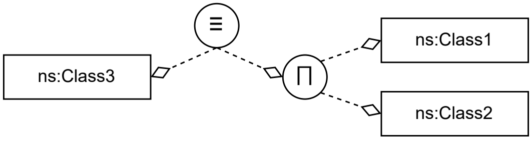
Graphical representations to indicate that `ns:Class` concept is equivalent to an anonymous class which represents a property restriction. Note that all the **property restrictions explained above applied** with the equivalent class axiom. However, just the `owl:allValuesFrom` notation was drawn in order to not overload this section.

Notation Example	OWL Code
Preferred  ns:Class1 <<owl:equivalentClass>> (all) ns:objectProperty ns:Class2	ns:Class1 a owl:Class ; owl:equivalentC [a owl:Restrict ; owl:allValuesFr ns:Class2 ; owl:onProperty ns:objectPropert . ns:Class2 a owl:Class . ns:objectPropert owl:ObjectProper ; rdfs:range ns:Class2 .
Alternative 	
Alternative 	ns:Class1 a owl:Class ; owl:equivalentC [a owl:Restrict ; owl:allValuesFr ns:Class2 ; owl:onProperty ns:objectPropert . ns:Class2 a owl:Class . ns:objectPropert

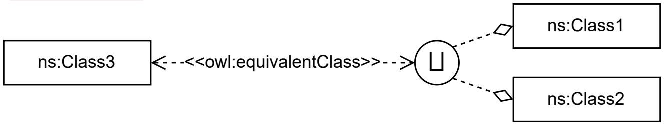
Notation Example	OWL Code
	owl:ObjectProper ; rdfs:domain ns:Class1 ; rdfs:range ns:Class2 .

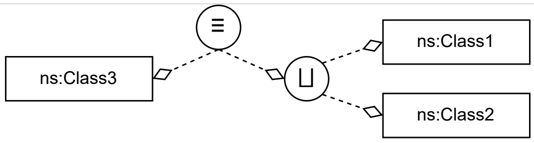
Notation Example	OWL Code
	ns:Class1 a owl:Class ; owl:equivalentClass [a owl:Restriction ; owl:allValuesFrom xsd:datatype ; owl:onProperty ns:datatypeProperty1] . ns:datatypeProperty1 a owl:DatatypeProperty ; rdfs:range xsd:datatype .

Graphical representations to indicate that `ns:Class` concept is equivalent to an anonymous class which represents an **intersection**

Notation Example	OWL Code
Preferred  ns:Class3 <<owl:equivalentClass>> ⊓ ns:Class1 ns:Class2	ns:Class3 a owl:Class ; owl:equivalentClass [a owl:Class ; owl:intersectionOf (ns:Class1 ns:Class2)] . ns:Class1 a owl:Class . ns:Class2 a owl:Class .
Alternative 	

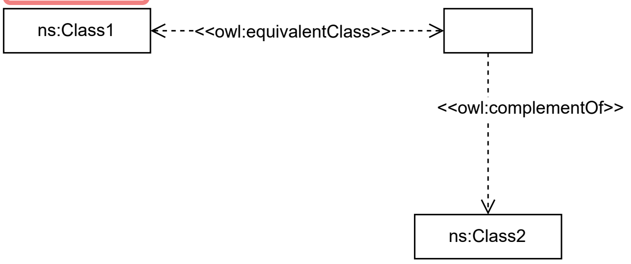
Graphical representations to indicate that `ns:Class` concept is equivalent to an anonymous class which represents an **union**

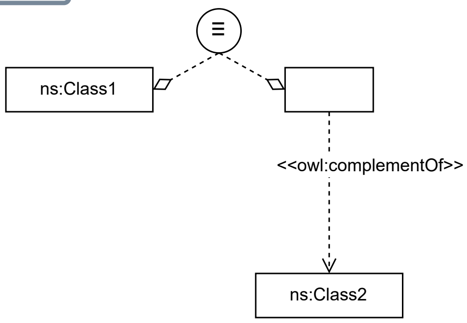
Notation Example	OWL Code
Preferred  ns:Class3 <<owl:equivalentClass>> ⊔ ns:Class1 ns:Class2	ns:Class3 a owl:Class ; owl:equivalentClass [a owl:Class ; owl:unionOf (ns:Class1 ns:Class2)] . ns:Class1 a owl:Class .
Alternative	

Notation Example	OWL Code
	`ns:Class2 a owl:Class .`

Graphical representations to indicate that `ns:Class` concept is equivalent to an anonymous class which represents a **complement**

Notation Example	OWL Code

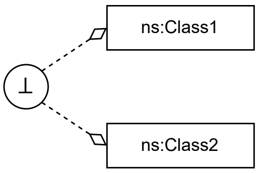
Preferred
 ns:Class1
<<owl:equivalentClass>>
<<owl:complementOf>>
ns:Class2
	`ns:Class1 a owl:Class ; owl:equivalentClass [ a owl:Class ; owl:complementOf ns:Class2 ] .`
	`ns:Class2 a owl:Class .`

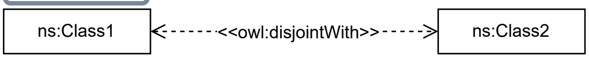
Alternative 	

2.6.3. Disjoint

Graphical representations to indicate that `ns:Class2` and `ns:Class1` are disjoint concepts

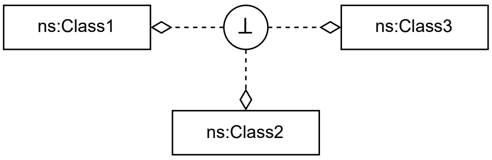
Notation Example	OWL Code

Preferred

	`ns:Class1 owl:disjointWith ns:Class2 .`
	`ns:Class 2 a owl:Class .`

Alternative 	

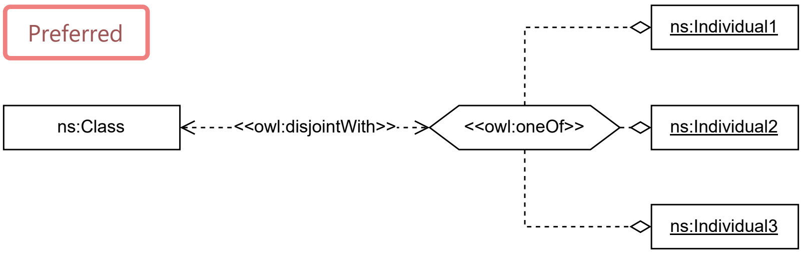
Graphical representations to indicate that `ns:Class3`, `ns:Class2` and `ns:Class1` are disjoint concepts

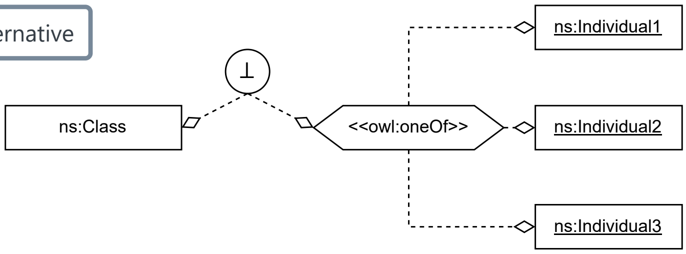
Notation Example	OWL Code

	`[rdf:type owl:AllDisjointClasses ;`
	`owl:members (ns:Class1 ns:Class2 ns:Class3)] .`
	
	`ns:Class1 a owl:Class .`
	
	`ns:Class2 a owl:Class .`
	
	`ns:Class3 a owl:Class .`

Graphical representations to indicate that `ns:Class` concept is disjoint with an anonymous class which represents an **enumeration**.

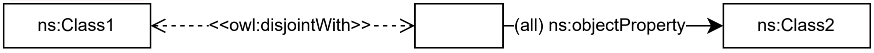
Notation Example	OWL Code

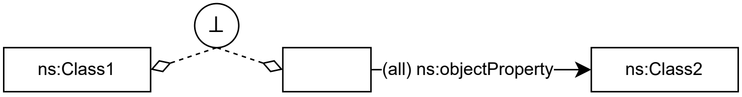
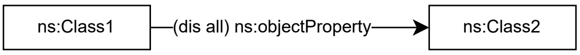
Preferred
 ns:Class
<<owl:disjointWith>>
<<owl:oneOf>>
ns:Individual1
ns:Individual2
ns:Individual3
	`ns:Class a owl:Class ;`
	`owl:disjointWith [`
	`a owl:Class ;`
	`owl:oneOf (`
	`ns:Individual1`
	`ns:Individual2`
	`ns:Individual3 ) ]`
	`.`
	
	`ns:Individual1 a owl:NamedIndividual .`
	
	`ns:Individual2 a owl:NamedIndividual .`
	
	`ns:Individual3 a owl:NamedIndividual .`

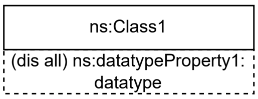
Alternative  ⊥ ns:Class <<owl:oneOf>> ns:Individual1 ns:Individual2 ns:Individual3	

Graphical representations to indicate that `ns:Class` concept is disjoint with an anonymous class which represents a **property restriction**. Note that all the **property restrictions explained above applied** with the subclass class axiom. However, just the `owl:allValuesFrom` notation was drawn in order to not overload this section.

Notation Example	OWL Code

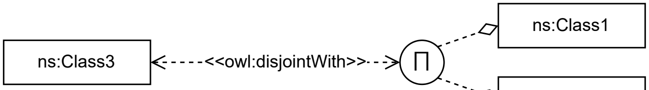
Preferred
 ns:Class1
<<owl:disjointWith>>
(all) ns:objectProperty
ns:Class2
	`ns:Class1 a owl:Class ;`
	`owl:disjointWit`
	`a owl:Restricti`
	`;`

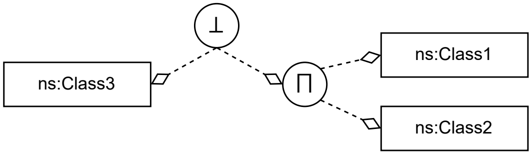
Notation Example	OWL Code
	owl:allValuesFr ns:Class2 ; owl:onProperty ns:objectPropert] .
Alternative 	ns:Class2 a owl:Class . ns:objectPropert a owl:ObjectProper ; rdfs:range ns:Class 2 .
Alternative 	ns:Class1 a owl:Class ; owl:disjointWit a owl:Restricti ; owl:allValuesFr ns:Class2 ; owl:onProperty ns:objectPropert] . ns:Class2 a owl:Class ns:objectPropert a owl:ObjectProper ; rdfs:domain ns:Class1 ; rdfs:range ns:Class 2 .

Notation Example	OWL Code
	ns:Class1 a owl:Class ; owl:disjointWith [a owl:Restriction ; owl:allValuesFrom xsd:datatype ; owl:onProperty ns:datatypeProperty1] . ns:Class1 a owl:Class . ns:datatypeProperty1 a owl:DatatypeProperty ; rdfs:range xsd:datatype .

Graphical representations to indicate that `ns:Class` concept is disjoint with an anonymous class which represents an **intersection**

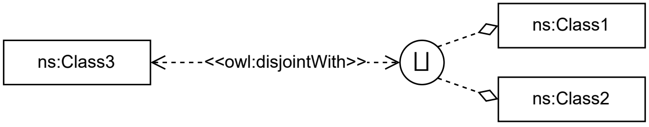
Notation Example	OWL Code

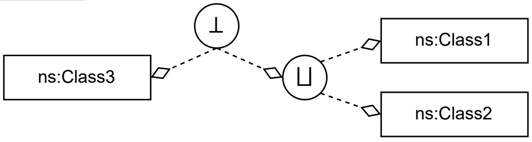
Preferred
 ns:Class3
<<owl:disjointWith>>
⊓
ns:Class1
ns:Class2
	`ns:Class3 a owl:Class ; owl:disjointWith [ a owl:Class ; owl:intersectionOf ( ns:Class1 ns:Class2 ) ] .`
	
	`ns:Class1 a owl:Class`
	
	`ns:Class2 a owl:Class`

Alternative 	

Graphical representations to indicate that `ns:Class` concept is disjoint with an anonymous class which represents an **union**

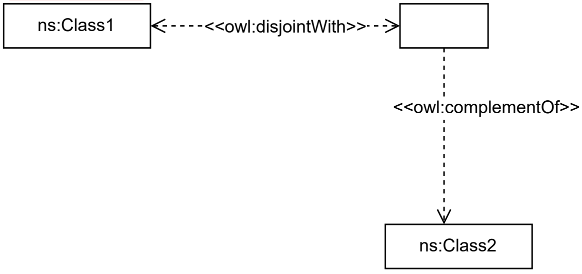
Notation Example	OWL Code

Preferred
 ns:Class3
<<owl:disjointWith>>
⊔
ns:Class1
ns:Class2
	`ns:Class3 a owl:Class ; owl:disjointWith [ a owl:Class ; owl:unionOf ( ns:Class1 ns:Class2 ) ] .`
	
	`ns:Class1 a owl:Class .`
	
	`ns:Class2 a owl:Class .`

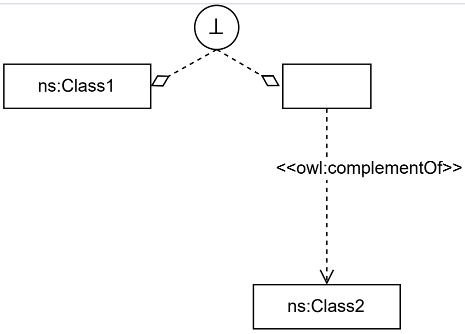
Alternative 	

Graphical representations to indicate that `ns:Class` concept is disjoint with an anonymous class which represents a **complement**

Notation Example	OWL Code

Preferred
 ns:Class1
<<owl:disjointWith>>
<<owl:complementOf>>
ns:Class2
	`ns:Class1 a owl:Class ; owl:disjointWith [ a owl:Class ; owl:complementOf ns:Class2 ] .`
	
	`ns:Class2 a owl:Class .`

Alternative	

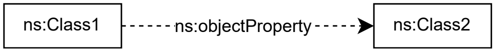
Notation Example	OWL Code
	

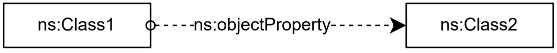
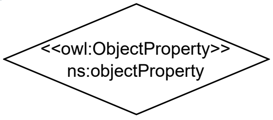
2.7. Object Properties

2.7.1. Domain and Range

Object properties without domain and range.

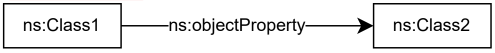
Notation Example	OWL Code

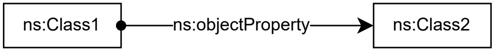
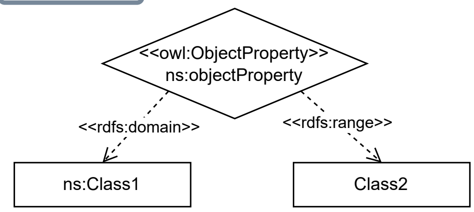
Preferred

	`ns:objectProperty rdf:type owl:ObjectProperty .`
	
	`ns:Class1 a owl:Class .`
	
	`ns:Class2 a owl:Class`

Alternative 	
Alternative 	`ns:objectProperty rdf:type owl:ObjectProperty .`

Object properties with domain and range.

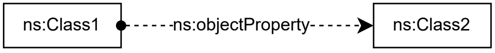
Notation Example	OWL Code

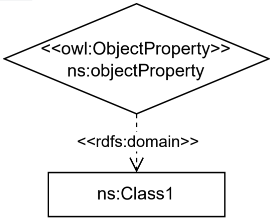
Preferred

	`ns:objectProperty rdf:type owl:ObjectProperty ;`
	`rdfs:domain ns:Class1 ;`
	`rdfs:range ns:Class2 .`
	
	`ns:Class1 a owl:Class .`
	
	`ns:Class2 a owl:Class .`

Alternative 	
Alternative  <<owl:ObjectProperty>> ns:objectProperty <<rdfs:domain>> <<rdfs:range>> ns:Class1 Class2	

Object properties with domain but without range.

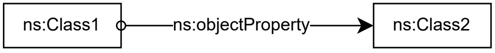
Notation Example	OWL Code

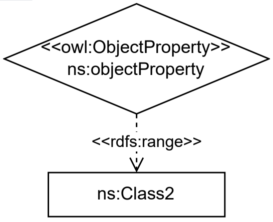
Preferred

 	
	`ns:objectProperty rdf:type owl:ObjectProperty ; rdfs:domain ns:Class1 .`
	
	`ns:Class1 a owl:Class .`
	
	`ns:Class2 a owl:Class .`

Alternative

 <<owl:ObjectProperty>>
ns:objectProperty
<<rdfs:domain>>
ns:Class1
	`ns:objectProperty rdf:type owl:ObjectProperty ; rdfs:domain ns:Class1 .`
	
	`ns:Class1 a owl:Class .`

Object properties without domain but with range.

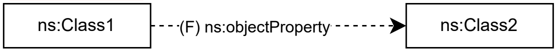
Notation Example	OWL Code

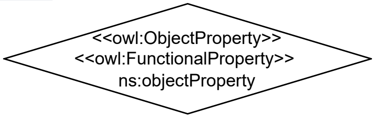
Preferred

 	
	`ns:objectProperty rdf:type owl:ObjectProperty ; rdfs:range ns:Class2 .`
	
	`ns:Class1 a owl:Class .`
	
	`ns:Class2 a owl:Class .`

Alternative

 <<owl:ObjectProperty>>
ns:objectProperty
<<rdfs:range>>
ns:Class2
	`ns:objectProperty rdf:type owl:ObjectProperty ; rdfs:range ns:Class2 .`
	
	`ns:Class2 a owl:Class .`

2.7.2. Functional Properties

Notation Example	OWL Code

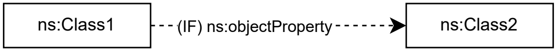
Preferred

	`ns:objectProperty rdf:type owl:ObjectProperty , owl:FunctionalProperty .`
	`ns:Class1 a owl:Class .`
	`ns:Class2 a owl:Class .`

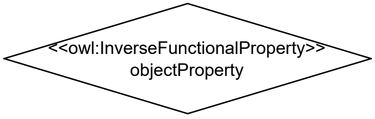
Alternative  <<owl:ObjectProperty>> <<owl:FunctionalProperty>> ns:objectProperty	`ns:objectProperty rdf:type owl:ObjectProperty , owl:FunctionalProperty .`

2.7.3. Inverse Functional Properties

A functional property can be an `owl:ObjectProperty` or an `owl:DatatypeProperty`. For that reason, it is neccesary to specify the type of the property.

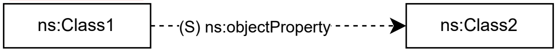
Notation Example	OWL Code

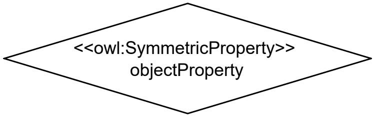
Preferred

	`ns:objectProperty rdf:type owl:ObjectProperty , owl:FunctionalProperty .`
	`ns:Class1 a owl:Class .`
	`ns:Class2 a owl:Class .`

Alternative 	`ns:objectProperty rdf:type owl:ObjectProperty , owl:FunctionalProperty .`

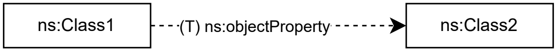
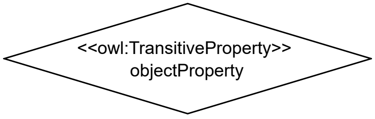
2.7.4. Symmetric Properties

Notation Example	OWL Code

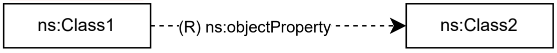
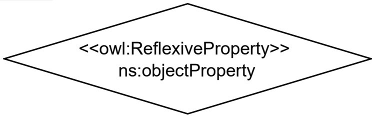
Preferred

	`ns:objectProperty rdf:type owl:ObjectProperty , owl:SymmetricProperty .`
	`ns:Class1 a owl:Class .`

Notation Example	OWL Code
	ns:Class2 a owl:Class .
Alternative 	ns:objectProperty rdf:type owl:ObjectProperty , owl:SymmetricProperty .

2.7.5. Transitive Properties

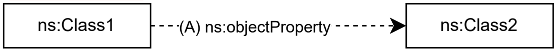
Notation Example	OWL Code
Preferred 	ns:objectProperty rdf:type owl:ObjectProperty , owl:TransitiveProperty . ns:Class1 a owl:Class . ns:Class2 a owl:Class .
Alternative 	ns:objectProperty rdf:type owl:ObjectProperty , owl:TransitiveProperty .

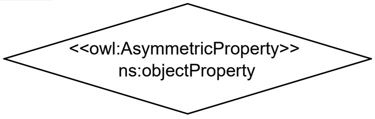
2.7.6. Reflexive Properties

Notation Example	OWL Code
Preferred 	ns:objectProperty rdf:type owl:ObjectProperty , owl:ReflexiveProperty . ns:Class1 a owl:Class . ns:Class2 a owl:Class .
Alternative 	ns:objectProperty rdf:type owl:ObjectProperty , owl:ReflexiveProperty .

2.7.7. Asymmetric Properties

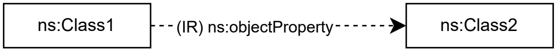
Notation Example	OWL Code

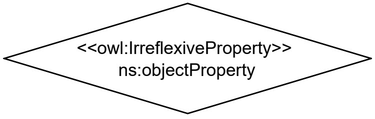
Preferred

	`ns:objectProperty rdf:type owl:ObjectProperty , owl:AsymmetricProperty .`
	`ns:Class1 a owl:Class .`
	`ns:Class2 a owl:Class .`

Alternative 	`ns:objectProperty rdf:type owl:ObjectProperty , owl:AsymmetricProperty .`

2.7.8. Irreflexive Properties

Notation Example	OWL Code

Preferred

	`ns:objectProperty rdf:type owl:ObjectProperty , owl:IrreflexiveProperty .`
	`ns:Class1 a owl:Class .`
	`ns:Class2 a owl:Class .`

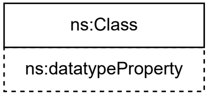
Alternative 	`ns:objectProperty rdf:type owl:ObjectProperty , owl:IrreflexiveProperty .`

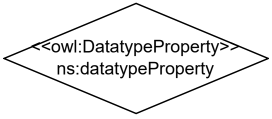
2.8. Datatype Properties

2.8.1. Domain and Range

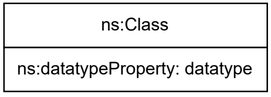
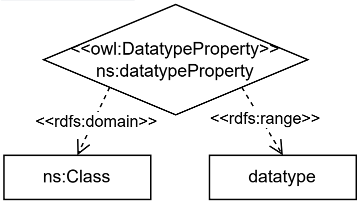
Datatype properties without domain and range.

Notation Example	OWL Code

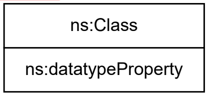
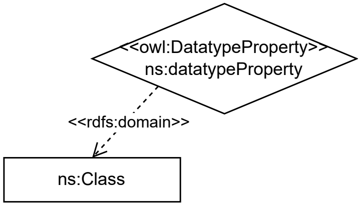
Preferred

	`ns:datatypeProperty rdf:type owl:DatatypeProperty .`
	`ns:Class a owl:Class .`

Alternative 	`ns:datatypeProperty rdf:type owl:DatatypeProperty .`

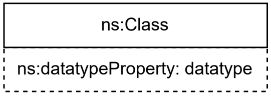
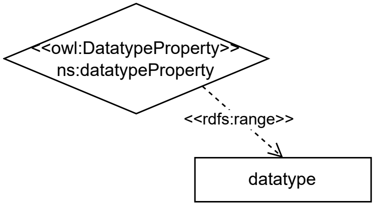
Datatype properties with domain and range.

Notation Example	OWL Code
Preferred  	ns:datatypeProperty rdf:type owl:DatatypeProperty ; rdfs:domain ns:Class ; rdfs:range xsd:datatype . ns:Class a owl:Class .
Alternative  <<owl:DatatypeProperty>> ns:datatypeProperty <<rdfs:domain>> <<rdfs:range>> ns:Class datatype	

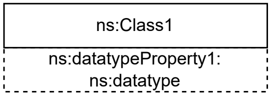
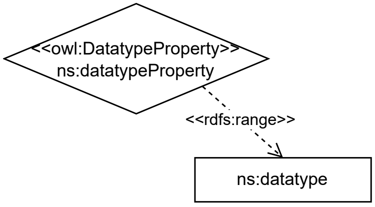
Datatype properties with domain and without range.

Notation Example	OWL Code
Preferred  	ns:datatypeProperty rdf:type owl:DatatypeProperty ; rdfs:domain ns:Class . ns:Class a owl:Class .
Alternative  <<owl:DatatypeProperty>> ns:datatypeProperty <<rdfs:domain>> ns:Class	

Datatype properties without domain but with range.

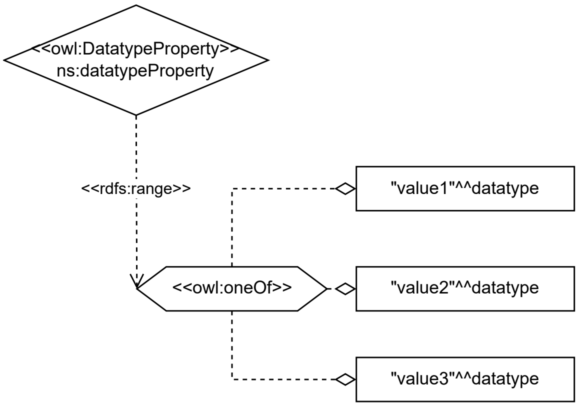
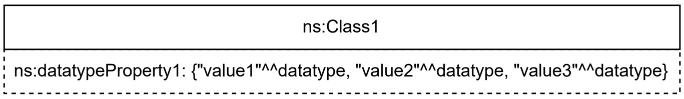
Notation Example	OWL Code
Preferred  	ns:datatypeProperty rdf:type owl:DatatypeProperty ; rdfs:range xsd:datatype . ns:Class a owl:Class .
Alternative  <<owl:DatatypeProperty>> ns:datatypeProperty <<rdfs:range>> datatype	ns:datatypeProperty rdf:type owl:DatatypeProperty ; rdfs:range xsd:datatype .

Definition of a customized datatype

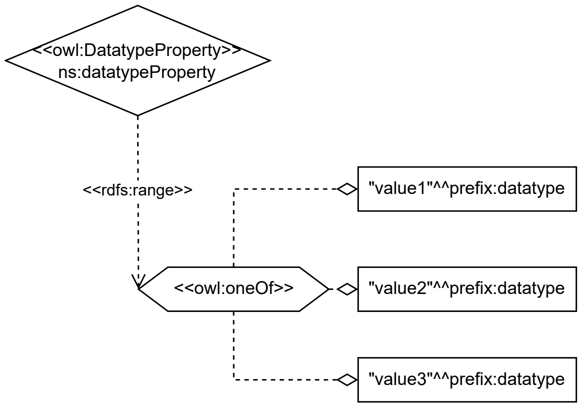
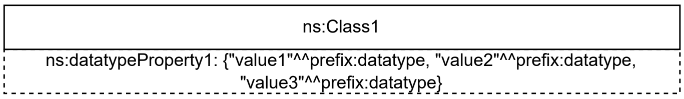
Notation Example	OWL Code
Preferred 	ns:datatypeProperty1 rdf:type owl:DatatypeProperty ; rdfs:range ns:datatype . ns:Class1 a owl:Class .
Alternative  <<owl:DatatypeProperty>> ns:datatypeProperty <<rdfs:range>> ns:datatype	ns:datatypeProperty1 rdf:type owl:DatatypeProperty ; rdfs:range ns:datatype .

2.8.2. Enumerated datatypes

In addition to the RDF datatypes, OWL provides one additional construct for defining a range of data values, namely an enumerated datatype. This datatype format makes use of the `owl:oneOf` construct. In the case of an enumerated datatype, the subject of `owl:oneOf` is a blank node of class owl:DataRange and the object is a list of literals.

Diagram Block	OWL Element
 <<owl:DatatypeProperty>> ns:datatypeProperty <<rdfs:range>> <<owl:oneOf>> "value1"^^datatype "value2"^^datatype "value3"^^datatype  ns:Class1 ns:datatypeProperty1: {"value1"^^datatype, "value2"^^datatype, "value3"^^datatype}	ns:datatypeProperty2 a owl:DatatypeProperty ; rdfs:range [a rdfs:Datatype ; owl:oneOf [a rdf:List ; rdf:first "value1"^^xsd:datatype ; rdf:rest [a rdf:List ; rdf:first "value2"^^xsd:datatype ; rdf:rest [a rdf:List ; rdf:first "value3"^^xsd:datatype ; rdf:rest ()]]] .

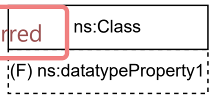
Moreover, costum data values can be defined specifying the prefixes of the datatypes. By default the prefix is xsd.

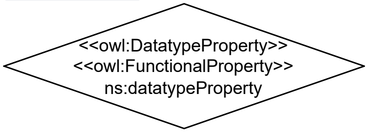
Diagram Block	OWL Element
 <<owl:DatatypeProperty>> ns:datatypeProperty <<rdfs:range>> <<owl:oneOf>> "value1"^^prefix:datatype "value2"^^prefix:datatype "value3"^^prefix:datatype  ns:Class1 ns:datatypeProperty1: {"value1"^^prefix:datatype, "value2"^^prefix:datatype, "value3"^^prefix:datatype}	`ns:datatypeProperty2 a owl:DatatypeProperty ; rdfs:range [ a rdfs:Datatype ; owl:oneOf [ a rdf:List ; rdf:first "value1"^^prefix:datatype ; rdf:rest [ a rdf:List ; rdf:first "value2"^^prefix:datatype ; rdf:rest [ a rdf:List ; rdf:first "value3"^^prefix:datatype ; rdf:rest () ] ] ] ] .`

2.8.3. Functional Properties

A functional property can be an `owl:ObjectProperty` or an `owl:DatatypeProperty`. For that reason, it is neccesary to specify the type of the property.

Notation Example	OWL Code

Preferred

	`ns:datatypeProperty rdf:type owl:DatatypeProperty , owl:FunctionalProperty .`
	`ns:Class a owl:Class .`

Alternative  <<owl:DatatypeProperty>> <<owl:FunctionalProperty>> ns:datatypeProperty	`ns:datatypeProperty rdf:type owl:DatatypeProperty , owl:FunctionalProperty .`

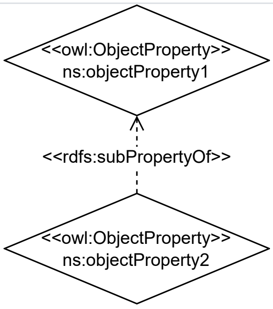
2.9. Relations between Object Properties

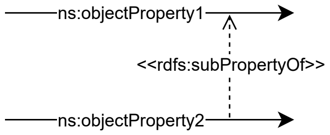
Currently, the converter only supports the "Preferred" version of the relations.

2.9.1. Sub-Property

Notation Example	OWL Code

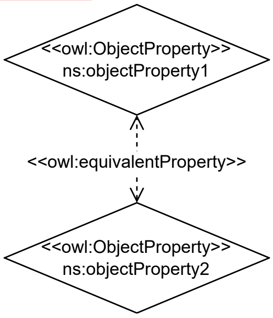
Preferred	`ns:objectProperty1 a owl:ObjectProperty ; rdfs:subPropertyOf ns:objectProperty2 .`
	`ns:objectProperty2 a owl:ObjectProperty .`

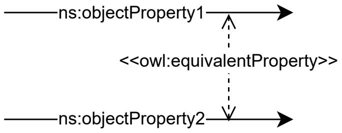
Notation Example	OWL Code
 <<owl:ObjectProperty>> ns:objectProperty1 <<rdfs:subPropertyOf>> <<owl:ObjectProperty>> ns:objectProperty2	

Alternative
Note that this is just visual notation
 ns:objectProperty1
<<rdfs:subPropertyOf>>
ns:objectProperty2
	`ns:objectProperty1 a owl:ObjectProperty .`
	
	`ns:objectProperty2 a owl:ObjectProperty .`

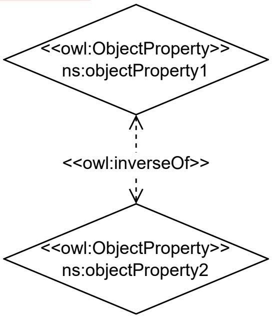
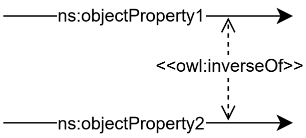
2.9.2. Equivalent Property

Notation Example	OWL Code

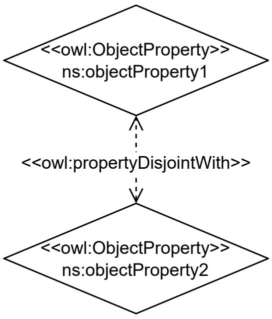
Preferred
 <<owl:ObjectProperty>>
ns:objectProperty1
<<owl:equivalentProperty>>
<<owl:ObjectProperty>>
ns:objectProperty2
	`ns:objectProperty1 a owl:ObjectProperty ; owl:equivalentProperty ns:objectProperty2 .`
	
	`ns:objectProperty2 a owl:ObjectProperty .`

Alternative
Note that this is just visual notation
 ns:objectProperty1
<<owl:equivalentProperty>>
ns:objectProperty2
	`ns:objectProperty1 a owl:ObjectProperty .`
	
	`ns:objectProperty2 a owl:ObjectProperty .`

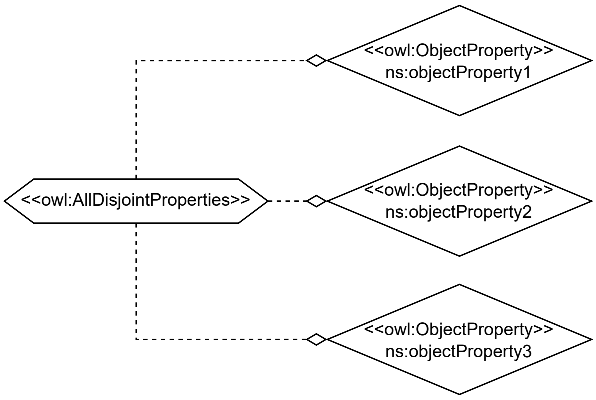
2.9.3. Inverse Property

Notation Example	OWL Code
Preferred  <<owl:ObjectProperty>> ns:objectProperty1 <<owl:inverseOf>> <<owl:ObjectProperty>> ns:objectProperty2	ns:objectProperty1 a owl:ObjectProperty ; owl:inverseOf ns:objectProperty2 . ns:ObjectProperty2 a owl:ObjectProperty .
Alternative **Note that this is just visual notation**  ns:objectProperty1 <<owl:inverseOf>> ns:objectProperty2	ns:objectProperty1 a owl:ObjectProperty . ns:ObjectProperty2 a owl:ObjectProperty .

2.9.4. Disjoint Property

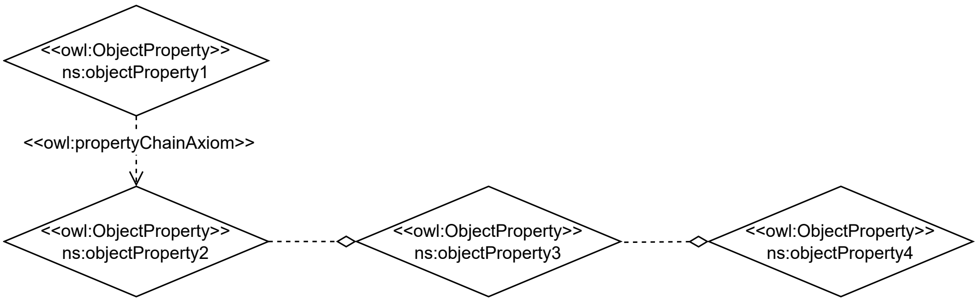
Notation Example	OWL Code
 <<owl:ObjectProperty>> ns:objectProperty1 <<owl:propertyDisjointWith>> <<owl:ObjectProperty>> ns:objectProperty2	ns:objectProperty2 a owl:ObjectProperty ; owl:propertyDisjointWith ns:objectProperty1 . ns:objectProperty1 a owl:ObjectProperty .

2.9.5. All Disjoint Properties

Notation Example	OWL Code
 <<owl:ObjectProperty>> ns:objectProperty1 <<owl:AllDisjointProperties>> <<owl:ObjectProperty>> ns:objectProperty2 <<owl:ObjectProperty>> ns:objectProperty3	[rdf:type owl:AllDisjointProperties ; owl:members (ns:objectProperty1 ns:objectProperty2 ns:objectProperty3)] . ns:objectProperty1 a owl:ObjectProperty . ns:objectProperty2 a owl:ObjectProperty . ns:objectProperty3 a owl:ObjectProperty .

2.9.6. Property Chain

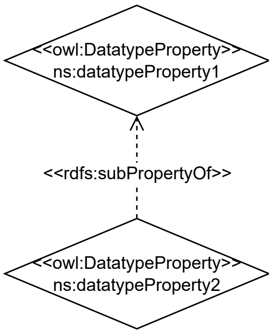
Notation Example	OWL C

 <<owl:ObjectProperty>>
ns:objectProperty1
<<owl:propertyChainAxiom>>
<<owl:ObjectProperty>>
ns:objectProperty2
<<owl:ObjectProperty>>
ns:objectProperty3
<<owl:ObjectProperty>>
ns:objectProperty4
	`ns:obj`
	`owl:Ob`
	`owl:pr`
	`( ns:o`
	`ns:obj`
	`ns:obj`
	
	`ns:obj`
	`owl:Ob`
	
	`ns:obj`
	`owl:Ob`
	
	`ns:obj`
	`owl:Ob`

2.10. Relations between Datatype Properties

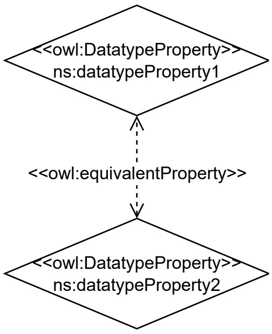
2.10.1. Sub-Property

Notation Example	OWL Code

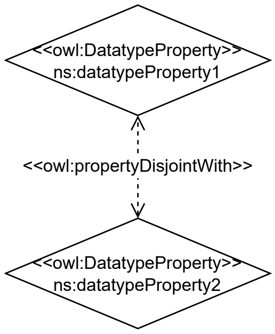
 <<owl:DatatypeProperty>>
ns:datatypeProperty1
<<rdfs:subPropertyOf>>
<<owl:DatatypeProperty>>
ns:datatypeProperty2
	`ns:datatypeProperty2 a`
	`owl:DatatypeProperty ;`
	`rdfs:subPropertyOf`
	`ns:datatypeProperty1 .`
	
	`ns:datatypeProperty1 a`
	`owl:DatatypeProperty .`

2.10.2. Equivalent Property

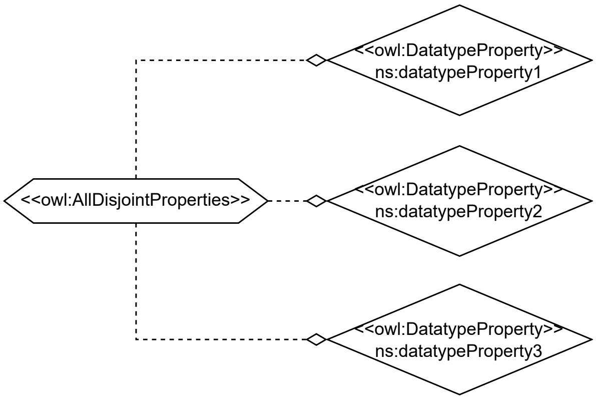
Notation Example	OWL Code

 <<owl:DatatypeProperty>>
ns:datatypeProperty1
<<owl:equivalentProperty>>
<<owl:DatatypeProperty>>
ns:datatypeProperty2
	`ns:datatypeProperty2 a`
	`owl:DatatypeProperty ;`
	`owl:equivalentProperty`
	`ns:datatypeProperty1 .`
	
	`ns:datatypeProperty1 a`
	`owl:DatatypeProperty .`

2.10.3. Disjoint Property

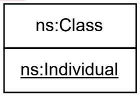
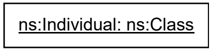
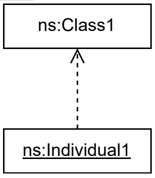
Notation Example	OWL Code
 <<owl:DatatypeProperty>> ns:datatypeProperty1 <<owl:propertyDisjointWith>> <<owl:DatatypeProperty>> ns:datatypeProperty2	ns:datatypeProperty2 a owl:DatatypeProperty ; owl:propertyDisjointWith ns:datatypeProperty1 . ns:datatypeProperty1 a owl:DatatypeProperty .

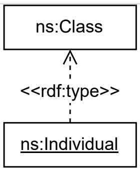
2.10.4. All Disjoint Properties

Notation Example	OWL Code
 <<owl:DatatypeProperty>> ns:datatypeProperty1 <<owl:AllDisjointProperties>> <<owl:DatatypeProperty>> ns:datatypeProperty2 <<owl:DatatypeProperty>> ns:datatypeProperty3	[rdf:type owl:AllDisjointProperties ; owl:members (ns:datatypeProperty1 ns:datatypeProperty2 ns:datatypeProperty3)] . ns:datatypeProperty1 a owl:DatatypeProperty . ns:datatypeProperty2 a owl:DatatypeProperty . ns:datatypeProperty3 a owl:DatatypeProperty .

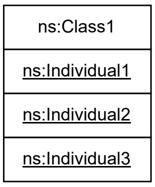
2.11. Individuals

2.11.1. Class Membership

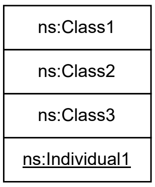
Notation Example	OWL Code
Preferred 	ns:Individual a owl:NamedIndividual ; rdf:type ns:Class . ns:Class a owl:Class .
Alternative 	
Alternative 	
Alternative	

Notation Example	OWL Code
	

In addition, the following shortcut is allowed when it is intented to represent that several individuals belongs to the same class.

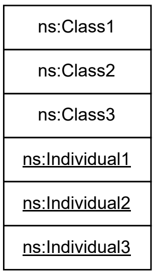
Notation Example	OWL Code
 ns:Class1 ns:Individual1 ns:Individual2 ns:Individual3	ns:Individual1 a ns:Class1, owl:NamedIndividual . ns:Individual2 a ns:Class1, owl:NamedIndividual . ns:Individual3 a ns:Class1, owl:NamedIndividual . ns:Class1 a owl:Class ; rdfs:label "Class1" .

Moreover, the following shortcut is allowed when it is intented to represent that an individual belongs to several classes.

Notation Example	OWL Code
	ns:Individual1 a ns:Class1, ns:Class2, ns:Class3, owl:NamedIndividual . ns:Class1 a owl:Class ; rdfs:label "Class1" . ns:Class2 a owl:Class ; rdfs:label "Class2" . ns:Class3 a owl:Class ; rdfs:label "Class3" .

Additionally, the following shortcut is allowed when it is intented to represent that several individuals belongs to the same several classes.

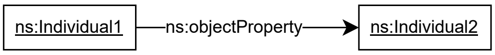
Notation Example	OWL Code

 ns:Class1
ns:Class2
ns:Class3
ns:Individual1
ns:Individual2
ns:Individual3
	`ns:Individual1 a`
	`ns:Class1,`
	`ns:Class2,`
	`ns:Class3,`
	`owl:NamedIndividual .`
	
	`ns:Individual2 a`
	`ns:Class1,`
	`ns:Class2,`
	`ns:Class3,`
	`owl:NamedIndividual .`
	
	`ns:Individual3 a`
	`ns:Class1,`
	`ns:Class2,`
	`ns:Class3,`
	`owl:NamedIndividual .`
	
	`ns:Class1 a owl:Class ;`
	`rdfs:label "Class1" .`
	
	`ns:Class2 a owl:Class ;`
	`rdfs:label "Class2" .`
	
	`ns:Class3 a owl:Class ;`
	`rdfs:label "Class3" .`

2.11.2. Property Values

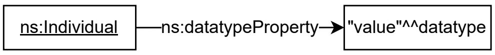
Association between individuals in RDF graphs.

Notation Example	OWL Code

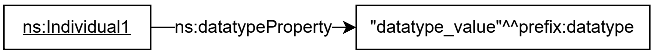
	`ns:Individual1 a`
	`owl:NamedIndividual`
	`;`
	`ns:objectProperty`
	`ns:Individual2 .`
	
	`ns:Individual2 a`
	`owl:NamedIndividual`
	`.`
	
	`ns:objectProperty a`
	`owl:ObjectProperty`
	`.`

Association between individuals and datatype values in RDF graphs.

Notation Example	OWL Code

	`ns:Individual a`
	`owl:NamedIndividual ;`
	`ns:datatypeProperty`
	`"value"^^xsd:datatype`
	`.`

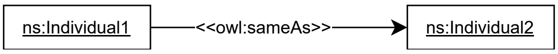
Moreover, costum data values can be defined specifying the prefixes of the datatypes. By default the prefix is xsd.

Notation Example	OWL Code
 ns:Individual1 ns:datatypeProperty "datatype_value"^^prefix:datatype	`ns:Individual a owl:NamedIndividual ; ns:datatypeProperty "datatype_value"^^prefix:datatype .`

2.11.3. Individual Identity

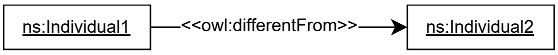
2.11.3.1. Same As

Notation Example	OWL Code

	`ns:Individual1 a owl:NamedIndividual ; owl:sameAs ns:Individual2 .`
	
	`ns:Individual2 a owl:NamedIndividual .`

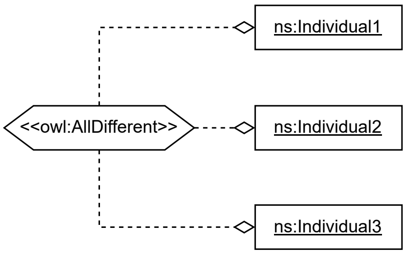
2.11.3.2. Different From

Notation Example	OWL Code

	`ns:Individual1 a owl:NamedIndividual ; owl:differentFrom ns:Individual2 .`
	
	`ns:Individual2 a owl:NamedIndividual .`

2.11.3.3. All Different

Notation Example	OWL Code

 ns:Individual1
<<owl:AllDifferent>>
ns:Individual2
ns:Individual3
	`[rdf:type owl:AllDifferent ; owl:distinctMembers (ns:Individual1 ns:Individual2 ns:Individual3)] .`
	
	`ns:Individual1 a owl:NamedIndividual .`
	
	`ns:Individual2 a owl:NamedIndividual .`

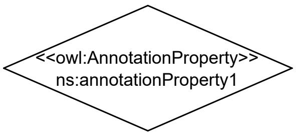
Notation Example	OWL Code
	`ns:Individual3 a owl:NamedIndividual .`

2.12. Annotation properties

Annotations are allowed on classes, properties, individuals and ontology headers. The object of an annotation property must be either a data literal, a URI reference, or an individual. **Annotation properties must have an explicit declaration to distinguish them from object properties**.

Five annotation properties are predefined and they do not need an explicit declaration:

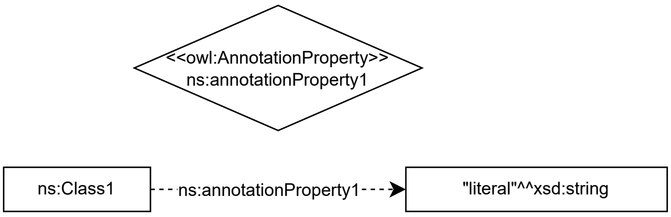
- owl:versionInfo
- rdfs:label
- rdfs:comment
- rdfs:seeAlso
- rdfs:isDefinedBy

Notation Example	OWL Code
	`ns:annotationProperty1 a owl:AnnotationProperty .`

2.12.1. Class Annotations

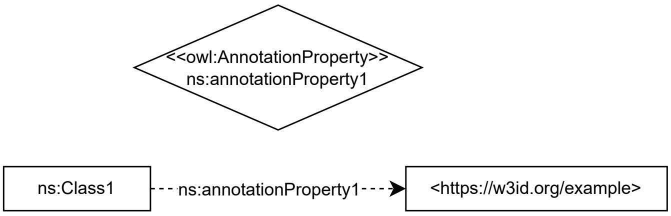
Class annotation whose object is a data literal.

Notation Example	OWL Code

 <<owl:AnnotationProperty>>
ns:annotationProperty1
ns:Class1
ns:annotationProperty1
"literal"^^xsd:string
	`ns:annotationProperty1 a owl:AnnotationProperty .`
	
	`ns:Class1 a owl:Class ; ns:annotationProperty1 "literal"^^xsd:string .`

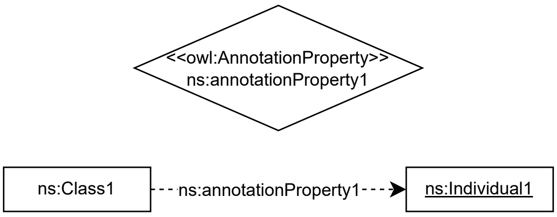
Class annotation whose object is an URI reference.

Notation Example	OWL Code

 <<owl:AnnotationProperty>>
ns:annotationProperty1
ns:Class1
ns:annotationProperty1
<https://w3id.org/example>
	`ns:annotationProperty1 a owl:AnnotationProperty .`
	
	`ns:Class1 a owl:Class ; ns:annotationProperty1 <https://w3id.org/example> .`

Class annotation whose object is an individual.

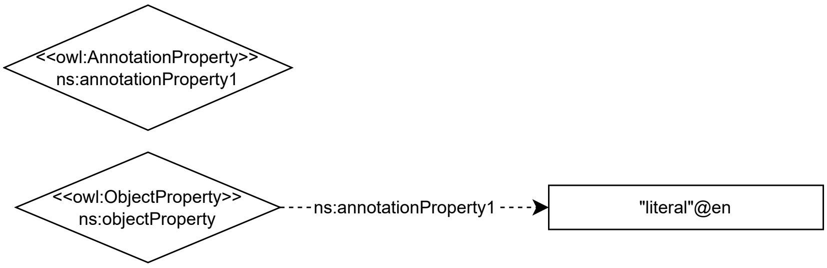
Notation Example	OWL Code

 <<owl:AnnotationProperty>>
ns:annotationProperty1
ns:Class1
ns:annotationProperty1
ns:Individual1
	`ns:annotationProperty1 a owl:AnnotationProperty .`
	
	`ns:Class1 a owl:Class ; ns:annotationProperty1 ns:Individual1 .`
	
	`ns:Individual1 a owl:NamedIndividual .`

2.12.2. Object Property Annotations

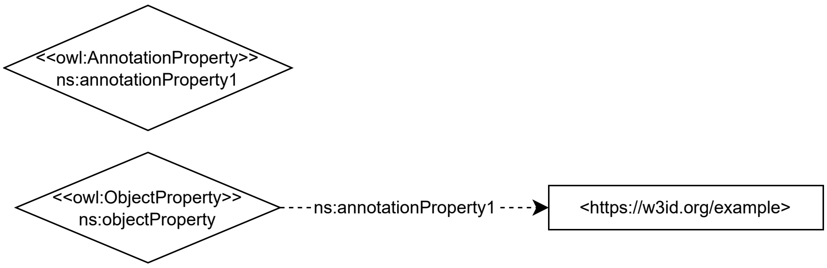
Object property annotation whose object is a data literal.

Notation Example	OWL Code

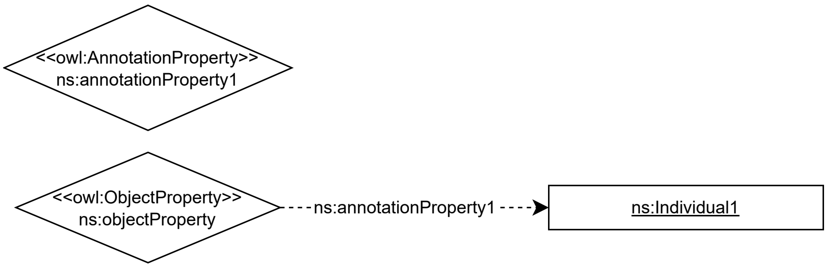
 <<owl:AnnotationProperty>>
ns:annotationProperty1
<<owl:ObjectProperty>>
ns:objectProperty
ns:annotationProperty1
"literal"@en
	`ns:annotationPropert a owl:AnnotationProper .`
	
	`ns:objectProperty a owl:ObjectProperty ; ns:annotationPropert "literal"@en .`

Object property annotation whose object is an URI reference.

Notation Example	OWL Code

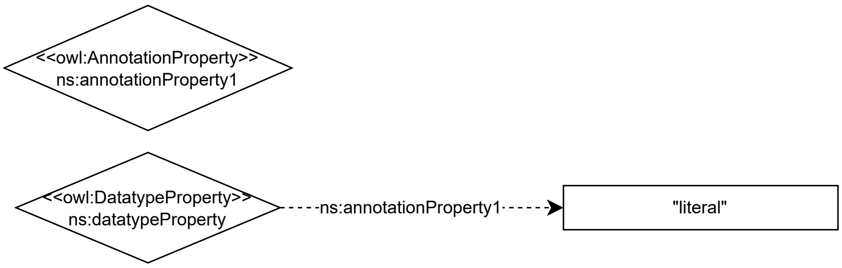
 <<owl:AnnotationProperty>>
ns:annotationProperty1
<<owl:ObjectProperty>>
ns:objectProperty
ns:annotationProperty1
<https://w3id.org/example>
	`ns:annotationPropert owl:AnnotationProper`
	
	`ns:objectProperty a owl:ObjectProperty ; ns:annotationPropert <https://w3id.org/e .`

Object property annotation whose object is an individual.

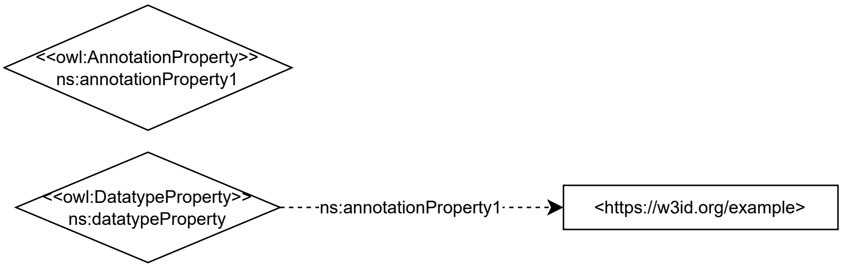
Notation Example	OWL Code
 <<owl:AnnotationProperty>> ns:annotationProperty1 <<owl:ObjectProperty>> ns:objectProperty ns:annotationProperty1 ns:Individual1	ns:annotationPropert a owl:AnnotationProper . ns:objectProperty a owl:ObjectProperty ; ns:annotationPropert ns:Individual1 . ns:Individual1 a owl:NamedIndividual

2.12.3. Datatype Property Annotations

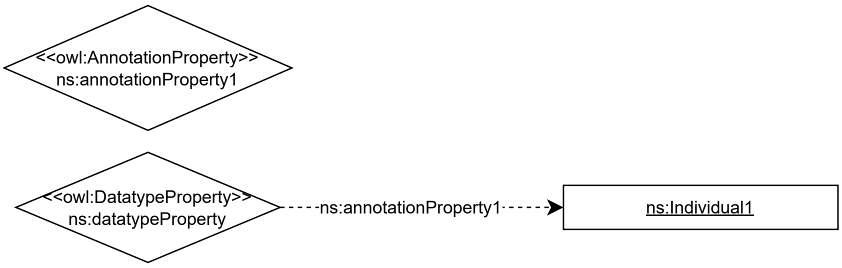
Datatype property annotation whose object is a data literal.

Notation Example	OWL Code
 <<owl:AnnotationProperty>> ns:annotationProperty1 <<owl:DatatypeProperty>> ns:datatypeProperty ns:annotationProperty1 "literal"	ns:annotationPrope a owl:AnnotationProp . ns:datatypePropert owl:DatatypeProper ns:annotationPrope "literal" .

Datatype property annotation whose object is an URI reference.

Notation Example	OWL Code
 <<owl:AnnotationProperty>> ns:annotationProperty1 <<owl:DatatypeProperty>> ns:datatypeProperty ns:annotationProperty1 <https://w3id.org/example>	ns:annotationPrope owl:AnnotationProp ns:datatypePropert owl:DatatypeProper ns:annotationPrope <https://w3id.org/ .

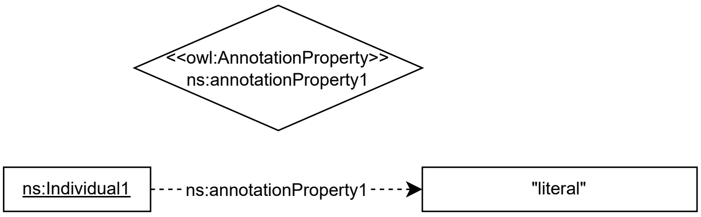
Datatype property annotation whose object is an individual.

Notation Example	OWL Code
 <<owl:AnnotationProperty>> ns:annotationProperty1 <<owl:DatatypeProperty>> ns:datatypeProperty ns:annotationProperty1 ns:Individual1	ns:annotationPrope a owl:AnnotationProp . ns:datatypePropert owl:DatatypeProper ns:annotationPrope ns:Individual1 .

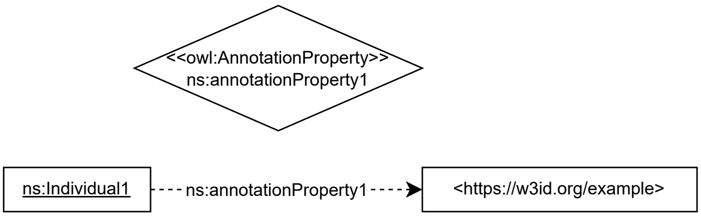
Notation Example	OWL Code
	ns:Individual1 a owl:NamedIndividua

2.12.4. Individual Annotations

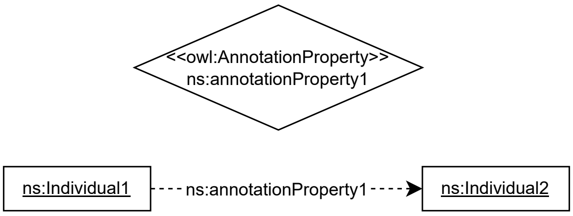
Individual annotation whose object is a data literal.

Notation Example	OWL Code
 <<owl:AnnotationProperty>> ns:annotationProperty1 ns:Individual1 ns:annotationProperty1 "literal"	ns:annotationProperty1 a owl:AnnotationProperty . ns:Individual1 a owl:NamedIndividual ; ns:annotationProperty1 "literal" .

Individual annotation whose object is an URI reference.

Notation Example	OWL Code
 <<owl:AnnotationProperty>> ns:annotationProperty1 ns:Individual1 ns:annotationProperty1 <https://w3id.org/example>	ns:annotationProperty1 a owl:AnnotationProperty . ns:Individual1 a owl:NamedIndividual ; ns:annotationProperty1 <https://w3id.org/example> .

Individual annotation whose object is an individual.

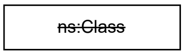
Notation Example	OWL Code
 <<owl:AnnotationProperty>> ns:annotationProperty1 ns:Individual1 ns:annotationProperty1 ns:Individual2	ns:annotationProperty1 a owl:AnnotationProperty . ns:Individual1 a owl:NamedIndividual ; ns:annotationProperty1 ns:Individual2 . ns:Individual2 a owl:NamedIndividual .

2.12.5. Deprecated class and property

Here, a specific identifier is said to be of type `owl:DeprecatedClass` or `owl:DeprecatedProperty`, where `owl:DeprecatedClass` is a subclass of `rdfs:Class` and `owl:DeprecatedProperty` is a subclass of `rdf:Property`. By deprecating a term, it means that the term should not be used in new documents that commit to the ontology. This allows an ontology to maintain backward-compatibility while phasing out an old vocabulary.

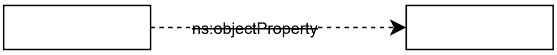
Deprecating a named class using `owl:DeprecatedClass`.

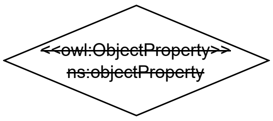
Notation Example	OWL Code

	`ns:Class a owl:Class,`
	`owl:DeprecatedClass.`

Deprecating an object property using `owl:DeprecatedProperty`. Note that in order to deprecate an object property it is only mandatory to crossed out the property name (but if you cross out more letters than the name, it works the same way).

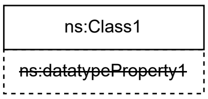
Notation Example	OWL Code

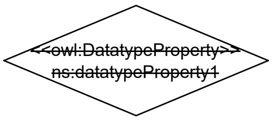
	`ns:objectProperty a`
	`owl:ObjectProperty,`
	`owl:DeprcatedProperty`
	`.`

Deprecating a datatype property using `owl:DeprecatedProperty`. Note that in order to deprecate a datatype property it is only mandatory to crossed out the property name (but if you cross out more letters than the name, it works the same way).

Notation Example	OWL Code

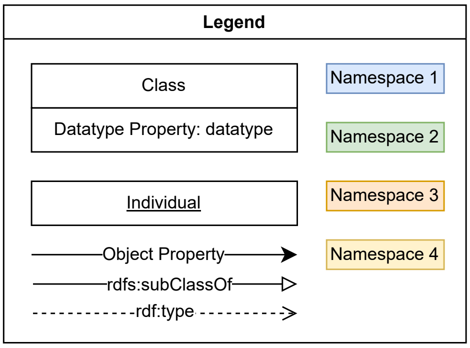
	`ns:datatypeProperty a`
	`owl:DatatypeProperty,`
	`owl:DeprcatedProperty .`

3. Tips

3.1. Legend Container

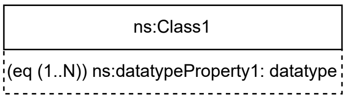
Definition of a legend container. All elements inside a container are going to be ignored by Chowlk when generating the ontology code. ou can also use different colours in the class boxes to identify different ontologies or modules (i.e. terms defined in different namespaces). Note that an element **is inside of a container when the element dissapears when container button is clicked**.

Diagram Block	OWL Element
 Legend Class Datatype Property: datatype Individual Object Property rdfs:subClassOf rdf:type Namespace 1 Namespace 2 Namespace 3 Namespace 4	-

3.2. Class axioms with cardinalities

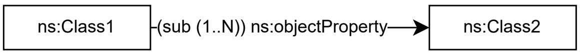
How to declare a class axiom when creating a cardinality restriction which involves a datatype property

Diagram Block	OWL Element

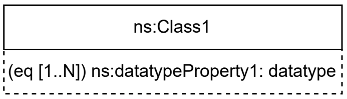
	`ns:Class1 a owl:Class ;`
	`owl:equivalentClass [ a owl:Restriction ;`
	`owl:minCardinality "1"^^xsd:nonNegativeInteger ;`
	`owl:onProperty ns:datatypeProperty1 ] .`
	
	`ns:datatypeProperty1 a owl:DatatypeProperty ;`
	`rdfs:label "datatype property1" ;`
	`dfs:range xsd:datatype .`

How to declare a class axiom when creating a cardinality restriction which involves an object property

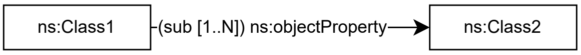
Diagram Block	OWL Element

	`ns:Class1 a owl:Class ;`
	`rdfs:subClassOf [ a owl:Restriction ;`
	`owl:minCardinality "1"^^xsd:nonNegativeInteger ;`
	`owl:onProperty ns:objectProperty ] .`
	
	`ns:Class2 a owl:Class .`
	
	`ns:objectProperty a owl:ObjectProperty ;`
	`rdfs:domain ns:Class1 ;`
	`rdfs:range ns:Class2 .`

How to declare a class axiom when creating a qualified cardinality restriction which involves a datatype property

Diagram Block	OWL Element
	ns:Class1 a owl:Class ; owl:equivalentClass [a owl:Restriction ; owl:minQualifiedCardinality "1"^^xsd:nonNegativeInteger ; owl:onDataRange xsd:datatype ; owl:onProperty ns:datatypeProperty1] . ns:datatypeProperty1 a owl:DatatypeProperty ; rdfs:range xsd:datatype .

How to declare a class axiom when creating a qualified cardinality restriction which involves an object property

Diagram Block	OWL Element
	ns:Class1 a owl:Class ; rdfs:subClassOf [a owl:Restriction ; owl:minQualifiedCardinality "1"^^xsd:nonNegativeInteger ; owl:onClass ns:Class2 ; owl:onProperty ns:objectProperty] . ns:objectProperty a owl:ObjectProperty ; rdfs:domain ns:Class1 ; rdfs:range ns:Class2 . ns:Class2 a owl:Class .

3.3. Declare multiple class restrictions

How to declare multiple restrictions which involves a datatype property

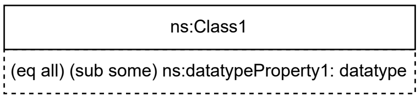
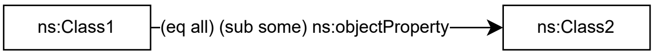
Diagram Block	OWL Element
 ns:Class1 (eq all) (sub some) ns:datatypeProperty1: datatype	ns:Class1 a owl:Class ; rdfs:subClassOf [a owl:Restriction ; owl:onProperty ns:datatypeProperty1 ; owl:someValuesFrom xsd:datatype] ; owl:equivalentClass [a owl:Restriction ; owl:allValuesFrom xsd:datatype ;

Diagram Block	OWL Element

	`owl:onProperty ns:datatypeProperty1 ] .`
	
	`ns:datatypeProperty1 a owl:DatatypeProperty ; rdfs:range xsd:datatype .`

How to declare multiple restrictions which involves an object property

Diagram Block	OWL Element

 ns:Class1
(eq all) (sub some) ns:objectProperty
ns:Class2
	`ns:Class1 a owl:Class ; rdfs:subClassOf [ a owl:Restriction ; owl:onProperty ns:objectProperty ; owl:someValuesFrom ns:Class2 ] ; owl:equivalentClass [ a owl:Restriction ; owl:allValuesFrom ns:Class2 ; owl:onProperty ns:objectProperty ] .`
	
	`ns:objectProperty a owl:ObjectProperty ; rdfs:domain ns:Class1 ; rdfs:range ns:Class2 .`
	
	`ns:Class2 a owl:Class .`

4. Combining Class Descriptions

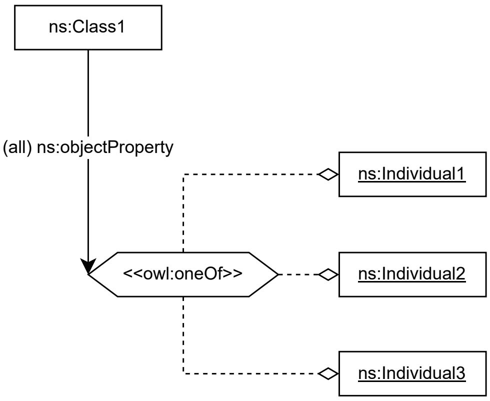
An anonymous Class acts semantically as a Class. Therefore, inside a Class Description could be another Class Description. In this section the notation used in order to concatenate class descriptions are explained. Note that **different class axioms** can be used, `owl:equivalentClass` is just used in order to not overload this section.

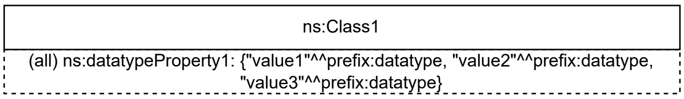
4.1. Restriction of Classes

Note that in case of property restrictions just `owl:allValuesFrom` and `owl:someValuesFrom` can be concatenated with others class descriptions.

Definition of an anonymous class which represents an **enumeration** inside of an anonymous class which represents a property restriction.

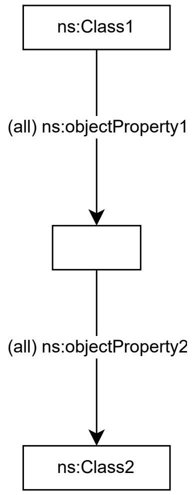
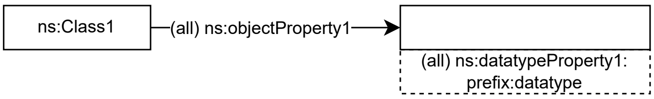
Notation Example	OWL Code

 ns:Class1
(all) ns:objectProperty
<<owl:oneOf>>
ns:Individual1
ns:Individual2
ns:Individual3
	`ns:Class1 a owl:Class ; rdfs:subClassOf [ a owl:Restriction ; owl:onProperty ns:objectProperty ; owl:allValuesFrom [ a owl:Class ; owl:oneOf ( ns:Individual1 ns:Individual2 ns:Individual3 ) ] ] .`
	
	`ns:Individual1 a owl:Class .`
	
	`ns:Individual2 a owl:Class .`
	
	`ns:Individual3 a owl:Class .`
	
	`ns:objectProperty a owl:ObjectProperty ; rdfs:domain ns:Class1 .`

 ns:Class1
(all) ns:datatypeProperty1: {"value1"^^prefix:datatype, "value2"^^prefix:datatype, "value3"^^prefix:datatype}
	`ns:Class1 a owl:Class ; rdfs:subClassOf [a owl:Restriction ; owl:onProperty ns:datatypeProperty ; owl:allValuesFrom [a rdfs:Datatype ; owl:oneOf [a rdf:List ; rdf:first "value1"^^prefix:datatype ; rdf:rest [a rdf:List ; rdf:first "value2"^^prefix:datatype ; rdf:rest [a rdf:List ; rdf:first "value3"^^prefix:datatype ; rdf:rest ()]]]] .`
	
	`ns:datatypeProperty a owl:DatatypeProperty ; rdfs:range [ a rdfs:Datatype ; owl:oneOf [ a rdf:List ; rdf:first "value1"^^prefix:datatype ;`

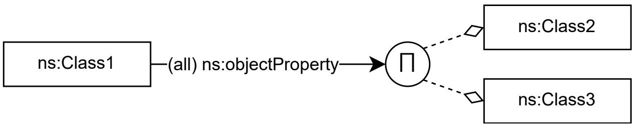
Notation Example	OWL Code
	rdf:rest [a rdf:List ; rdf:first "value2"^^prefix:datatype ; rdf:rest [a rdf:List ; rdf:first "value3"^^prefix:datatype ; rdf:rest ()]]] .

Definition of an anonymous class which represents a **property restriction** inside of an anonymous class which represents a property restriction.

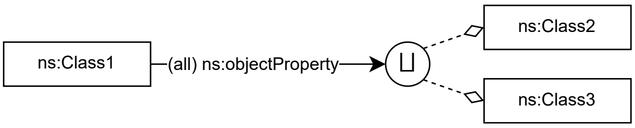
Notation Example	OWL Code
 ns:Class1 (all) ns:objectProperty1 (all) ns:objectProperty2 ns:Class2	ns:Class1 a owl:Class ; rdfs:subClassOf [a owl:Restriction ; owl:onProperty ns:objectProperty1 ; owl:allValuesFrom [a owl:Restriction ; owl:allValuesFrom ns:Class2 ; owl:onProperty ns:objectProperty2] . ns:Class2 a owl:Class . ns:objectProperty1 a owl:ObjectProperty ; rdfs:domain ns:Class1 . ns:objectProperty2 a owl:ObjectProperty ; rdfs:range ns:Class2 .
 ns:Class1 (all) ns:objectProperty1 (all) ns:datatypeProperty1: prefix:datatype	ns:Class1 a owl:Class ; rdfs:subClassOf [a owl:Restriction ; owl:onProperty ns:objectProperty1 ; owl:allValuesFrom [a owl:Restriction ; owl:allValuesFrom prefix:datatype ; owl:onProperty ns:dataypeProperty1] .

Notation Example	OWL Code
	ns:objectProperty1 a owl:ObjectProperty ; rdfs:domain ns:Class1 . ns:datatypeProperty1 a owl:DatatypeProperty ; rdfs:range prefix:datatype .

Definition of an anonymous class which represents an **intersection** inside of an anonymous class which represents a property restriction.

Notation Example	OWL Code
 ns:Class1 (all) ns:objectProperty ⊓ ns:Class2 ns:Class3	ns:Class1 a owl:Class ; rdfs:subClassOf [a owl:Restriction ; owl:onProperty ns:objectProperty ; owl:allValuesFrom [a owl:Class ; owl:intersectionOf (ns:Class2 ns:Class3)]] . ns:Class2 a owl:Class . ns:Class3 a owl:Class . ns:objectProperty1 a owl:ObjectProperty ; rdfs:domain ns:Class1 .

Definition of an anonymous class which represents an **union** inside of an anonymous class which represents a property restriction.

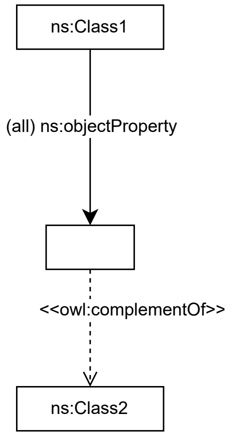
Notation Example	OWL Code
 ns:Class1 (all) ns:objectProperty ⊔ ns:Class2 ns:Class3	ns:Class1 a owl:Class ; rdfs:subClassOf [a owl:Restriction ; owl:onProperty ns:objectProperty

Notation Example	OWL Code

	`;`
	`owl:allValuesFrom`
	`[ a owl:Class ;`
	`owl:unionOf (`
	`ns:Class2`
	`ns:Class3 ) ] ] .`
	
	`ns:Class2 a`
	`owl:Class .`
	
	`ns:Class3 a`
	`owl:Class .`
	
	`ns:objectProperty1`
	`a`
	`owl:ObjectProperty`
	`;`
	`rdfs:domain`
	`ns:Class1 .`

Definition of an anonymous class which represents a **complement** inside of an anonymous class which represents a property restriction.

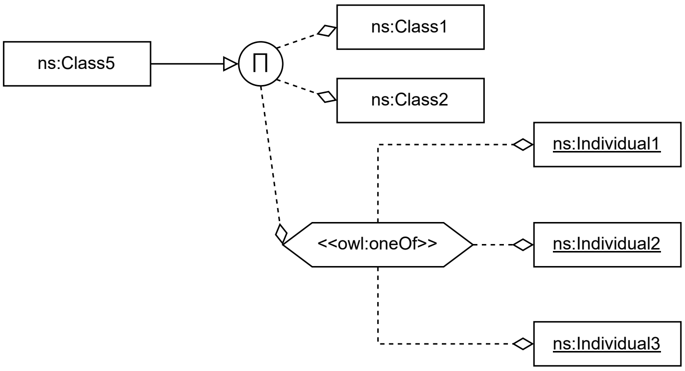
Notation Example	OWL Code

 ns:Class1
(all) ns:objectProperty
<<owl:complementOf>>
ns:Class2
	`ns:Class1 a owl:Class ;`
	`rdfs:subClassOf [ a`
	`owl:Restriction ;`
	`owl:onProperty`
	`ns:objectProperty ;`
	`owl:allValuesFrom [ a`
	`owl:Class ;`
	`owl:complementOf ns:Class2`
	`] ] .`
	
	`ns:Class2 a owl:Class .`
	
	`ns:objectProperty1 a`
	`owl:ObjectProperty ;`
	`rdfs:domain ns:Class1 .`

4.2. Intersection of Classes

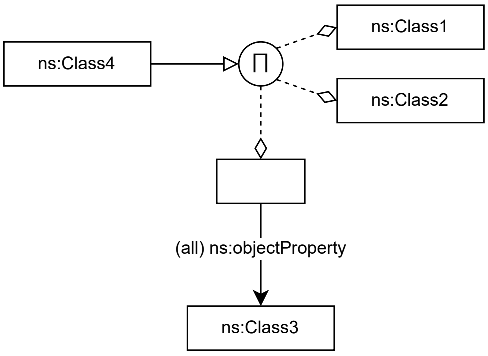
Definition of an anonymous class which represents an **enumeration** inside of an anonymous class which represents an intersection.

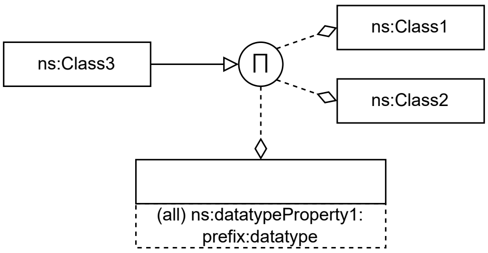
Notation Example	OWL Code

 ns:Class5
⊓
ns:Class1
ns:Class2
ns:Individual1
<<owl:oneOf>>
ns:Individual2
ns:Individual3
	`ns:Class5 a owl:Class ; rdfs:subClassOf [ a owl:Class ; owl:intersectionOf ( ns:Class1 ns:Class2 [ a owl:Class ; owl:oneOf ( ns:Individual1 ns:Individual2 ns:Individual3 ) ] ) ] .`
	`ns:Class1 a owl:Class .`
	`ns:Class2 a owl:Class .`
	`ns:Individual1 a owl:NamedIndividual .`
	`ns:Individual2 a owl:NamedIndividual .`
	`ns:Individual3 a owl:NamedIndividual .`

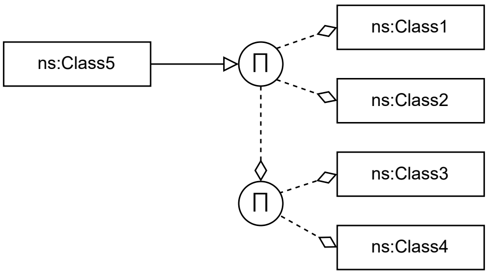
Definition of an anonymous class which represents a **property restriction** inside of an anonymous class which represents an intersection.

Notation Example	OWL Code

 ns:Class4
⊓
ns:Class1
ns:Class2
(all) ns:objectProperty
ns:Class3
	`ns:Class4 a owl:Class ; rdfs:subClassOf [ a owl:Class ; owl:intersectionOf ( ns:Class1 ns:Class2 [ a owl:Restriction ; owl:allValuesFrom ns:Class3 ; owl:onProperty ns:objectProperty ] ) .`
	`ns:Class1 a owl:Class .`
	`ns:Class2 a`

Notation Example	OWL Code
	owl:Class . ns:Class3 a owl:Class . ns:objectProperty a owl:ObjectProperty ; rdfs:range ns:Class3 .
 ns:Class1 ns:Class3 ⊓ ns:Class2 (all) ns:datatypeProperty1: prefix:datatype	ns:Class3 a owl:Class ; rdfs:subClassOf [a owl:Class ; owl:intersectionOf (ns:Class1 ns:Class2 [a owl:Restriction ; owl:allValuesFrom prefix:datatype ; owl:onProperty ns:datatypeProperty1]) . ns:Class1 a owl:Class . ns:Class2 a owl:Class . ns:datatypeProperty1 a owl:DatatypeProperty ; rdfs:range prefix:datatype .

Definition of an anonymous class which represents an **intersection** inside of an anonymous class which represents an intersection.

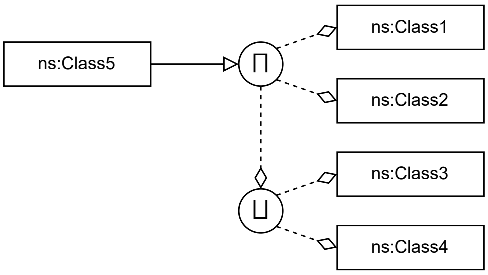
Notation Example	OWL Code
 ns:Class1 ns:Class5 ⊓ ns:Class2 ns:Class3 ⊓ ns:Class4	ns:Class5 a owl:Class ; rdfs:subClassOf [a owl:Class ; owl:intersectionOf (ns:Class1 ns:Class2 [a owl:Class ; owl:intersectionOf (ns:Class3 ns:Class4)])] . ns:Class1 a owl:Class .

Notation Example	OWL Code

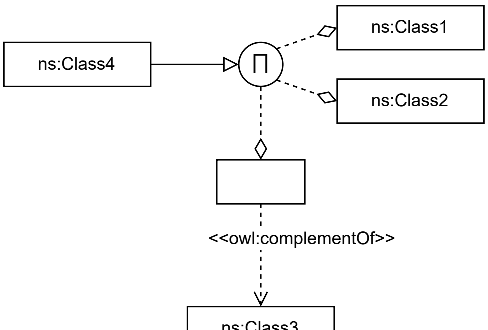
	`ns:Class2 a owl:Class .`
	`ns:Class3 a owl:Class .`
	`ns:Class4 a owl:Class .`

Definition of an anonymous class which represents an **union** inside of an anonymous class which represents an intersection.

Notation Example	OWL Code

 ns:Class5
⊓
ns:Class1
ns:Class2
⊔
ns:Class3
ns:Class4
	`ns:Class5 a owl:Class ; rdfs:subClassOf [ a owl:Class ; owl:intersectionOf ( ns:Class1 ns:Class2 [ a owl:Class ; owl:unionOf ( ns:Class3 ns:Class4 ) ] ) ] .`
	`ns:Class1 a owl:Class .`
	`ns:Class2 a owl:Class .`
	`ns:Class3 a owl:Class .`
	`ns:Class4 a owl:Class .`

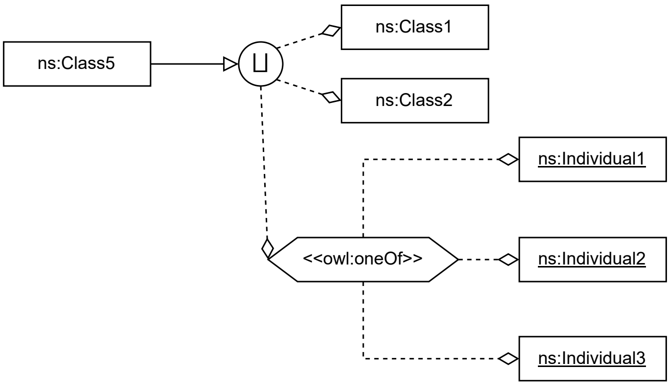
Definition of an anonymous class which represents a **complement** inside of an anonymous class which represents an intersection.

Notation Example	OWL Code
 ns:Class4 ⊓ ns:Class1 ns:Class2 <<owl:complementOf>> ns:Class3	`ns:Class4 a owl:Class ; rdfs:subClassOf [ a owl:Class ; owl:intersectionOf ( ns:Class1 ns:Class2 [ a owl:Class ; owl:complementOf ns:Class3 ] ) ] .`

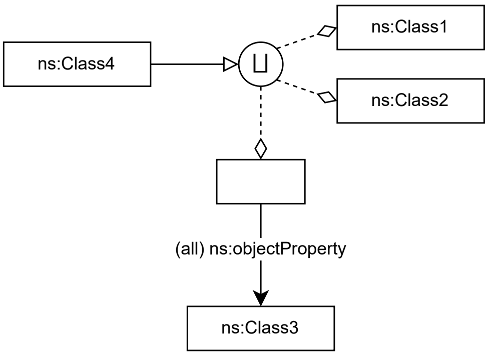
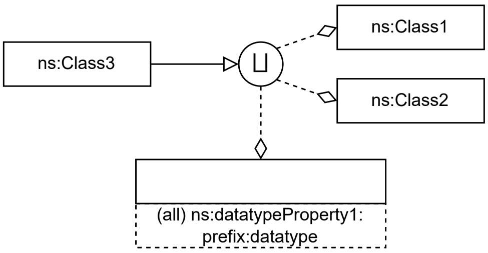
Notation Example	OWL Code
	ns:Class1 a owl:Class . ns:Class2 a owl:Class . ns:Class3 a owl:Class .

4.3. Union of Classes

Definition of an anonymous class which represents an **enumeration** inside of an anonymous class which represents an union.

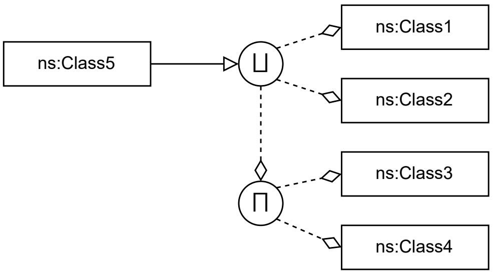
Notation Example	OWL Code
 ns:Class5 ⊔ ns:Class1 ns:Class2 ns:Individual1 <<owl:oneOf>> ns:Individual2 ns:Individual3	ns:Class5 a owl:Class ; rdfs:subClassOf [a owl:Class ; owl:unionOf (ns:Class1 ns:Class2 [a owl:Class ; owl:oneOf (ns:Individual1 ns:Individual2 ns:Individual3)])] . ns:Class1 a owl:Class . ns:Class2 a owl:Class . ns:Individual1 a owl:NamedIndividual . ns:Individual2 a owl:NamedIndividual . ns:Individual3 a owl:NamedIndividual .

Definition of an anonymous class which represents a **property restriction** inside of an anonymous class which represents an union.

Notation Example	OWL Code
 ns:Class4 ⊔ ns:Class1 ns:Class2 (all) ns:objectProperty ns:Class3	ns:Class4 a owl:Class ; rdfs:subClassOf [a owl:Class ; owl:unionOf (ns:Class1 ns:Class2 [a owl:Restriction ; owl:allValuesFrom ns:Class3 ; owl:onProperty ns:objectProperty]) . ns:Class1 a owl:Class . ns:Class2 a owl:Class . ns:Class3 a owl:Class . ns:objectProperty a owl:ObjectProperty ; rdfs:range ns:Class3 .
 ns:Class3 ⊔ ns:Class1 ns:Class2 (all) ns:datatypeProperty1: prefix:datatype	ns:Class3 a owl:Class ; rdfs:subClassOf [a owl:Class ; owl:unionOf (ns:Class1 ns:Class2 [a owl:Restriction ; owl:allValuesFrom prefix:datatype ; owl:onProperty ns:datatypeProperty1]) . ns:Class1 a owl:Class . ns:Class2 a owl:Class . ns:datatypeProperty1 a owl:DatatypeProperty ; rdfs:range prefix:datatype .

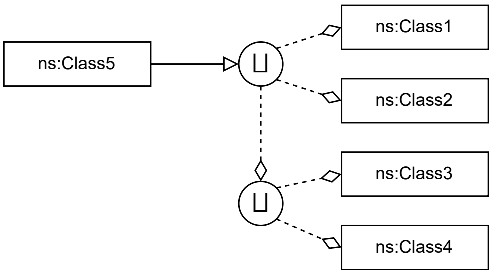
Definition of an anonymous class which represents an **intersection** inside of an anonymous class which represents an union.

Notation Example	OWL Code

 ns:Class5
⊔
ns:Class1
ns:Class2
⊓
ns:Class3
ns:Class4
	`ns:Class5 a owl:Class ; rdfs:subClassOf [ a owl:Class ; owl:unionOf ( ns:Class1 ns:Class2 [ a owl:Class ; owl:intersectionOf ( ns:Class3 ns:Class4 ) ] ) ] .`
	`ns:Class1 a owl:Class .`
	`ns:Class2 a owl:Class .`
	`ns:Class3 a owl:Class .`
	`ns:Class4 a owl:Class .`

Definition of an anonymous class which represents an **union** inside of an anonymous class which represents an union.

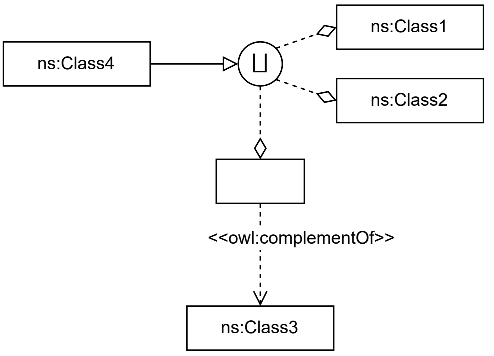
Notation Example	OWL Code

 ns:Class5
⊔
ns:Class1
ns:Class2
⊔
ns:Class3
ns:Class4
	`ns:Class5 a owl:Class ; rdfs:subClassOf [ a owl:Class ; owl:unionOf ( ns:Class1 ns:Class2 [ a owl:Class ; owl:unionOf ( ns:Class3 ns:Class4 ) ] ) ] .`
	`ns:Class1 a owl:Class .`
	`ns:Class2 a owl:Class .`
	`ns:Class3 a owl:Class .`

Notation Example	OWL Code
	`ns:Class4 a owl:Class .`

Definition of an anonymous class which represents a **complement** inside of an anonymous class which represents an union.

Notation Example	OWL Code

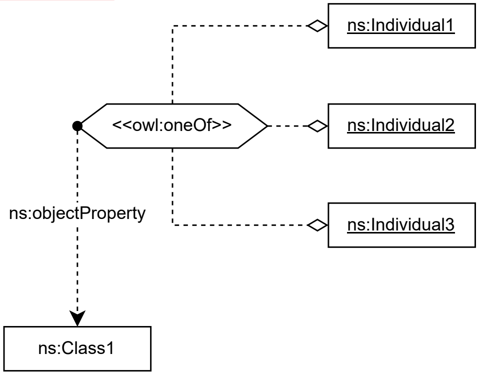
 ns:Class4
⊔
ns:Class1
ns:Class2
<<owl:complementOf>>
ns:Class3
	`ns:Class4 a owl:Class ; rdfs:subClassOf [ a owl:Class ; owl:unionOf ( ns:Class1 ns:Class2 [ a owl:Class ; owl:complementOf ns:Class3 ] ) ] .`
	`ns:Class1 a owl:Class .`
	`ns:Class2 a owl:Class .`
	`ns:Class3 a owl:Class .`

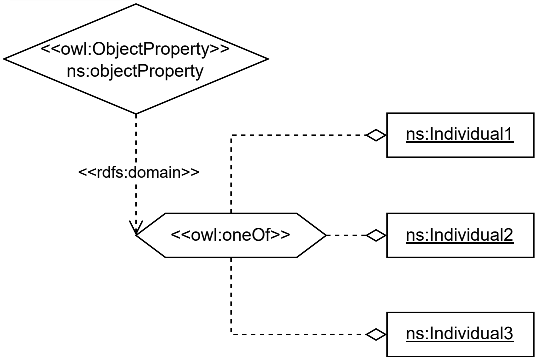
5. Anonymous classes as domain and range in properties

5.1. Object Properties Domain

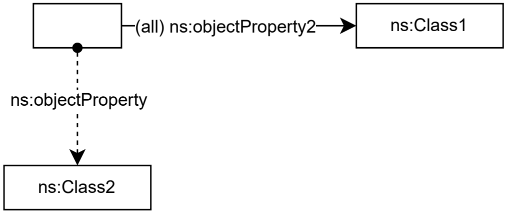
Enumeration as domain of an object property.

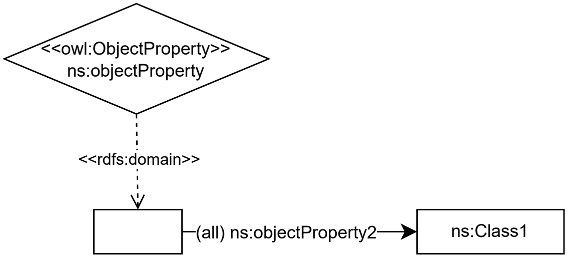
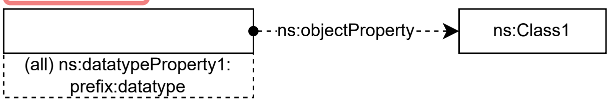
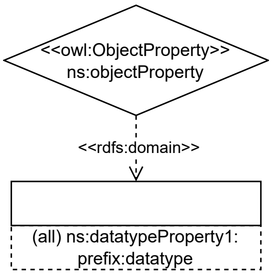
Notation Example	OWL Code

Preferred
 ns:Individual1
<<owl:oneOf>>
ns:Individual2
ns:objectProperty
ns:Individual3
ns:Class1
	`ns:objectProperty a owl:ObjectProperty ; rdfs:domain [ a owl:Class ; owl:oneOf ( ns:Individual1 ns:Individual2 ns:Individual3 ) ] .`
	`ns:Class1 a owl:Class .`
	`ns:Individual1 a owl:NamedIndividual .`

Notation Example	OWL Code
	ns:Individual2 a owl:NamedIndividual . ns:Individual3 a owl:NamedIndividual .
Alternative  <<owl:ObjectProperty>> ns:objectProperty <<rdfs:domain>> <<owl:oneOf>> ns:Individual1 ns:Individual2 ns:Individual3	ns:objectProperty a owl:ObjectProperty ; rdfs:domain [a owl:Class ; owl:oneOf (ns:Individual1 ns:Individual2 ns:Individual3)] . ns:Individual1 a owl:NamedIndividual . ns:Individual2 a owl:NamedIndividual . ns:Individual3 a owl:NamedIndividual .

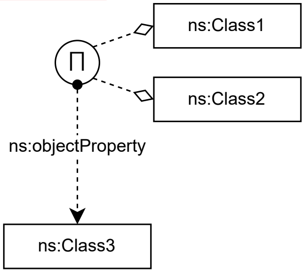
Restriction as domain of an object property.

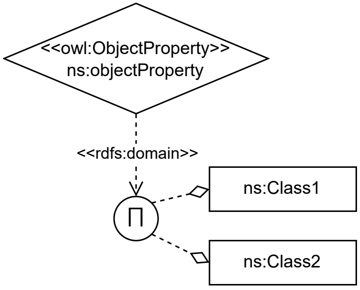
Notation Example	OWL Code
Preferred  (all) ns:objectProperty2 ns:Class1 ns:objectProperty ns:Class2	ns:objectProperty a owl:ObjectProperty ; rdfs:domain [a owl:Restriction ; owl:allValuesFrom ns:Class1 ; owl:onProperty ns:objectProperty2] . ns:objectProperty2 a owl:ObjectProperty ; rdfs:range ns:Class1 . ns:Class1 a owl:Class . ns:Class2 a owl:Class .

Notation Example	OWL Code
Alternative  <<owl:ObjectProperty>> ns:objectProperty <<rdfs:domain>> (all) ns:objectProperty2 ns:Class1	ns:objectProperty a owl:ObjectProperty ; rdfs:domain [a owl:Restriction ; owl:allValuesFrom ns:Class1 ; owl:onProperty ns:objectProperty2] . ns:objectProperty2 a owl:ObjectProperty ; rdfs:range ns:Class1 . ns:Class1 a owl:Class .
Preferred  ns:objectProperty ns:Class1 (all) ns:datatypeProperty1: prefix:datatype	ns:objectProperty a owl:ObjectProperty ; rdfs:domain [a owl:Restriction ; owl:allValuesFrom prefix:datatype ; owl:onProperty ns:datatypeProperty1] . ns:datatypeProperty1 a owl:DatatypeProperty ; rdfs:range prefix:datatype . ns:Class1 a owl:Class .
Alternative  <<owl:ObjectProperty>> ns:objectProperty <<rdfs:domain>> (all) ns:datatypeProperty1: prefix:datatype	ns:objectProperty a owl:ObjectProperty ; rdfs:domain [a owl:Restriction ; owl:allValuesFrom prefix:datatype ; owl:onProperty ns:datatypeProperty1] . ns:datatypeProperty1 a owl:DatatypeProperty ; rdfs:range prefix:datatype .

Intersection as domain of an object property.

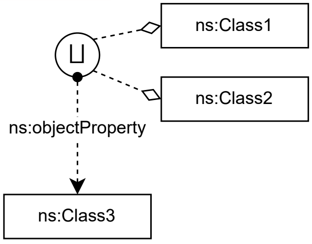
Notation Example	OWL Code

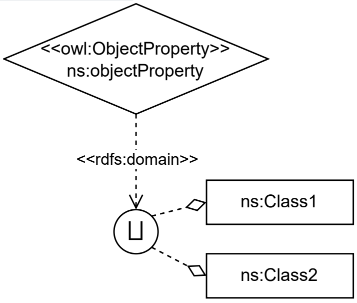
Preferred

 ns:Class1
⊓
ns:Class2
ns:objectProperty
ns:Class3
	`ns:objectProperty a`
	`owl:ObjectProperty ;`
	`rdfs:domain [ a owl:Class`
	`;`
	`owl:intersectionOf (`
	`ns:Class1 ns:Class2 ) ] .`
	
	`ns:Class1 a owl:Class .`
	
	`ns:Class2 a owl:Class .`
	
	`ns:Class3 a owl:Class .`

Alternative

 <<owl:ObjectProperty>>
ns:objectProperty
<<rdfs:domain>>
ns:Class1
⊓
ns:Class2
	`ns:objectProperty a`
	`owl:ObjectProperty ;`
	`rdfs:domain [ a owl:Class`
	`;`
	`owl:intersectionOf (`
	`ns:Class1 ns:Class2 ) ] .`
	
	`ns:Class1 a owl:Class .`
	
	`ns:Class2 a owl:Class .`

Union as domain of an object property.

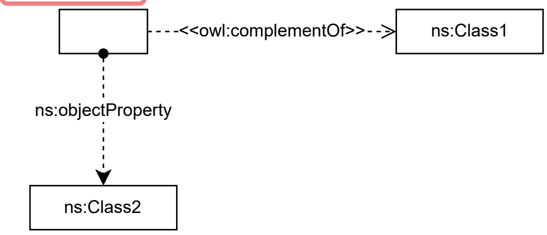
Notation Example	OWL Code

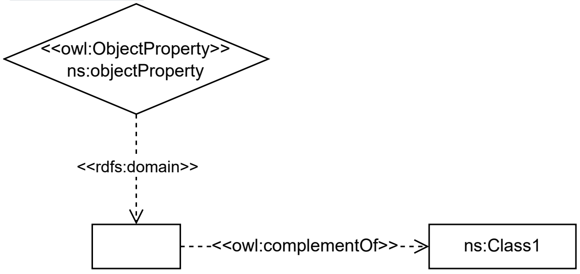
Preferred

 ns:Class1
⊔
ns:Class2
ns:objectProperty
ns:Class3
	`ns:objectProperty a`
	`owl:ObjectProperty ;`
	`rdfs:domain [ a owl:Class`
	`;`
	`owl:unionOf ( ns:Class1`
	`ns:Class2 ) ] .`
	
	`ns:Class1 a owl:Class .`
	
	`ns:Class2 a owl:Class .`
	
	`ns:Class3 a owl:Class .`

Alternative

 <<owl:ObjectProperty>>
ns:objectProperty
<<rdfs:domain>>
ns:Class1
⊔
ns:Class2
	`ns:objectProperty a`
	`owl:ObjectProperty ;`
	`rdfs:domain [ a owl:Class`
	`;`
	`owl:unionOf ( ns:Class1`
	`ns:Class2 ) ] .`
	
	`ns:Class1 a owl:Class .`
	
	`ns:Class2 a owl:Class .`

Complement as domain of an object property.

Notation Example	OWL Code

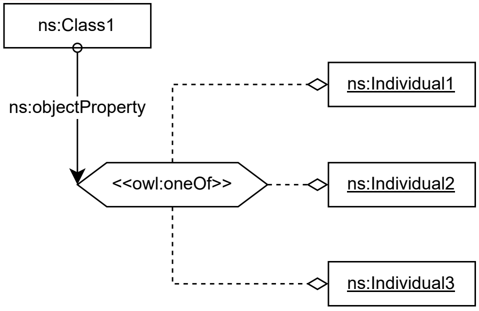
Preferred
 <<owl:complementOf>>
ns:Class1
ns:objectProperty
ns:Class2
	`ns:objectProperty a owl:ObjectProperty ; rdfs:domain [ a owl:Class ; owl:complementOf ns:Class1 ] .`
	`ns:Class1 a owl:Class .`
	`ns:Class2 a owl:Class .`

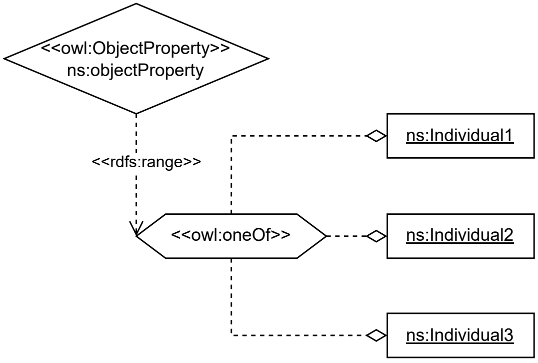
Alternative
 <<owl:ObjectProperty>>
ns:objectProperty
<<rdfs:domain>>
<<owl:complementOf>>
ns:Class1
	`ns:objectProperty a owl:ObjectProperty ; rdfs:domain [ a owl:Class ; owl:complementOf ns:Class1 ] .`
	`ns:Class1 a owl:Class .`

5.2. Object Properties Range

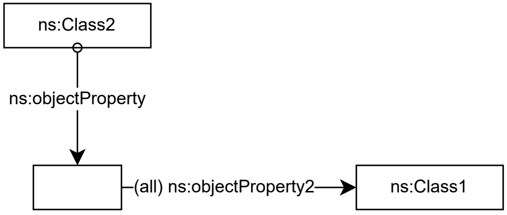
Enumeration as range of an object property.

Notation Example	OWL Code

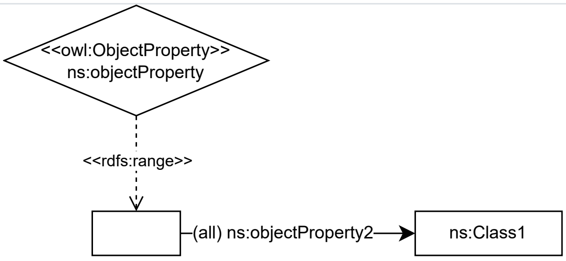
Preferred
 ns:Class1
ns:objectProperty
<<owl:oneOf>>
ns:Individual1
ns:Individual2
ns:Individual3
	`ns:objectProperty a owl:ObjectProperty ; rdfs:range [ a owl:Class ; owl:oneOf ( ns:Individual1 ns:Individual2 ns:Individual3 ) ] .`
	`ns:Class1 a owl:Class .`
	`ns:Individual1 a owl:NamedIndividual .`
	`ns:Individual2 a owl:NamedIndividual .`
	`ns:Individual3 a`

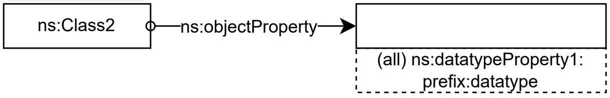
Notation Example	OWL Code
	owl:NamedIndividual .
Alternative  <<owl:ObjectProperty>> ns:objectProperty <<rdfs:range>> <<owl:oneOf>> ns:Individual1 ns:Individual2 ns:Individual3	ns:objectProperty a owl:ObjectProperty ; rdfs:range [a owl:Class ; owl:oneOf (ns:Individual1 ns:Individual2 ns:Individual3)] . ns:Individual1 a owl:NamedIndividual . ns:Individual2 a owl:NamedIndividual . ns:Individual3 a owl:NamedIndividual .

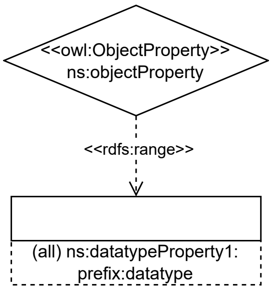
Restriction as range of an object property.

Notation Example	OWL Code
Preferred  ns:Class2 ns:objectProperty (all) ns:objectProperty2 ns:Class1	ns:objectProperty a owl:ObjectProperty ; rdfs:range [a owl:Restriction ; owl:allValuesFrom ns:Class1 ; owl:onProperty ns:objectProperty2] . ns:objectProperty2 a owl:ObjectProperty ; rdfs:range ns:Class1 . ns:Class1 a owl:Class . ns:Class2 a owl:Class .
Alternative	ns:objectProperty a owl:ObjectProperty ; rdfs:range [a owl:Restriction ; owl:allValuesFrom ns:Class1 ;

Notation Example	OWL Code

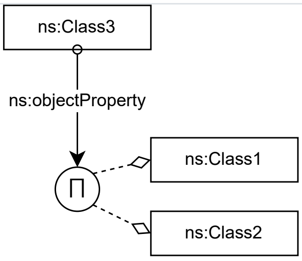
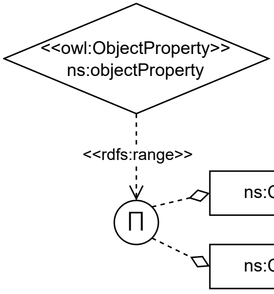
 <<owl:ObjectProperty>>
ns:objectProperty
<<rdfs:range>>
(all) ns:objectProperty2
ns:Class1
	`owl:onProperty ns:objectProperty2 ] .`
	
	`ns:objectProperty2 a owl:ObjectProperty ; rdfs:range ns:Class1 .`
	
	`ns:Class1 a owl:Class .`

Preferred
 ns:Class2
ns:objectProperty
(all) ns:datatypeProperty1: prefix:datatype
	`ns:objectProperty a owl:ObjectProperty ; rdfs:range [ a owl:Restriction ; owl:allValuesFrom prefix:datatype ; owl:onProperty ns:datatypeProperty1 ] .`
	
	`ns:datatypeProperty1 a owl:DatatypeProperty ; rdfs:range prefix:datatype .`
	
	`ns:Class1 a owl:Class .`

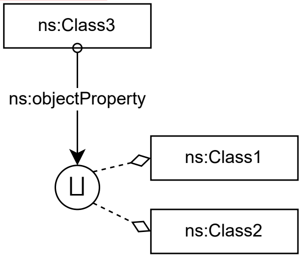
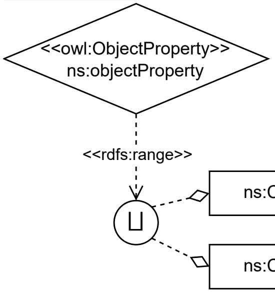
Alternative
 <<owl:ObjectProperty>>
ns:objectProperty
<<rdfs:range>>
(all) ns:datatypeProperty1: prefix:datatype
	`ns:objectProperty a owl:ObjectProperty ; rdfs:range [ a owl:Restriction ; owl:allValuesFrom prefix:datatype ; owl:onProperty ns:datatypeProperty1 ] .`
	
	`ns:datatypeProperty1 a owl:DatatypeProperty ; rdfs:range prefix:datatype .`

Intersection as range of an object property.

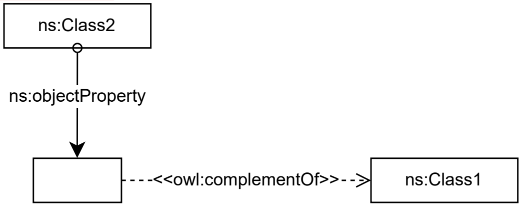
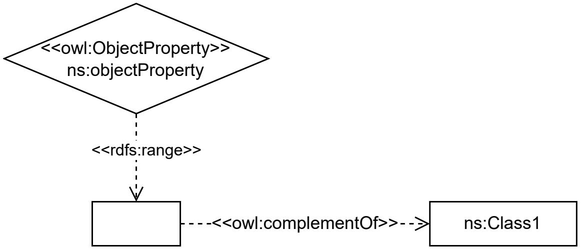
Notation Example	OWL Code
Preferred	`ns:objectProperty a owl:ObjectProperty ; rdfs:range [ a owl:Class ; owl:intersectionOf (`

Notation Example	OWL Code
 ns:Class3 ns:objectProperty ⊓ ns:Class1 ns:Class2	ns:Class1 ns:Class2)] . ns:Class1 a owl:Class . ns:Class2 a owl:Class . ns:Class3 a owl:Class .
Alternative  <<owl:ObjectProperty>> ns:objectProperty <<rdfs:range>> ⊓ ns:Class1 ns:Class2	ns:objectProperty a owl:ObjectProperty ; rdfs:range [a owl:Class ; owl:intersectionOf (ns:Class1 ns:Class2)] . ns:Class1 a owl:Class . ns:Class2 a owl:Class .

Union as range of an object property.

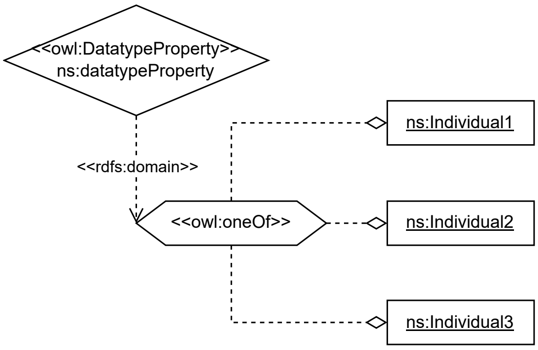
Notation Example	OWL Code
Preferred  ns:Class3 ns:objectProperty ⊔ ns:Class1 ns:Class2	ns:objectProperty a owl:ObjectProperty ; rdfs:range [a owl:Class ; owl:unionOf (ns:Class1 ns:Class2)] . ns:Class1 a owl:Class . ns:Class2 a owl:Class . ns:Class3 a owl:Class .
Alternative  <<owl:ObjectProperty>> ns:objectProperty <<rdfs:range>> ⊔ ns:Class1 ns:Class2	ns:objectProperty a owl:ObjectProperty ; rdfs:range [a owl:Class ; owl:unionOf (ns:Class1 ns:Class2)] . ns:Class1 a owl:Class . ns:Class2 a owl:Class .

Complement as range of an object property.

Notation Example	OWL Code
Preferred  ns:Class2 ns:objectProperty <<owl:complementOf>> ns:Class1	ns:objectProperty a owl:ObjectProperty ; rdfs:range [a owl:Class ; owl:complementOf ns:Class1] . ns:Class1 a owl:Class . ns:Class2 a owl:Class .
Alternative  <<owl:ObjectProperty>> ns:objectProperty <<rdfs:range>> <<owl:complementOf>> ns:Class1	ns:objectProperty a owl:ObjectProperty ; rdfs:range [a owl:Class ; owl:complementOf ns:Class1] . ns:Class1 a owl:Class .

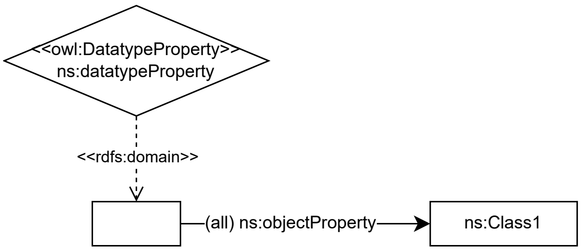
5.3. Datatype Properties Domain

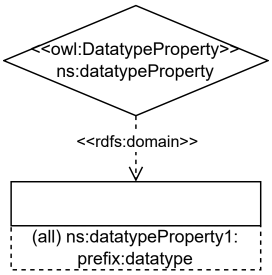
Enumeration as domain of an datatype property.

Notation Example	OWL Code
 <<owl:DatatypeProperty>> ns:datatypeProperty <<rdfs:domain>> <<owl:oneOf>> ns:Individual1 ns:Individual2 ns:Individual3	ns:datatypeProperty a owl:DatatypeProperty ; rdfs:domain [a owl:Class ; owl:oneOf (ns:Individual1 ns:Individual2 ns:Individual3)] . ns:Individual1 a owl:NamedIndividual . ns:Individual2 a owl:NamedIndividual . ns:Individual3 a owl:NamedIndividual .

Restriction as domain of an datatype property.

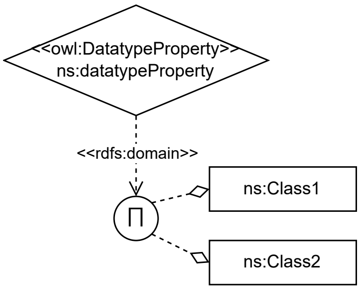
Notation Example	OWL Code

 <<owl:DatatypeProperty>>
ns:datatypeProperty
<<rdfs:domain>>
(all) ns:objectProperty
ns:Class1
	`ns:datatypeProperty a owl:DatatypeProperty ; rdfs:domain [ a owl:Restriction ; owl:allValuesFrom ns:Class1 ; owl:onProperty ns:objectProperty ] .`
	`ns:objectProperty a owl:ObjectProperty ; rdfs:range ns:Class1 .`
	`ns:Class1 a owl:Class .`

 <<owl:DatatypeProperty>>
ns:datatypeProperty
<<rdfs:domain>>
(all) ns:datatypeProperty1:
prefix:datatype
	`ns:datatypeProperty a owl:DatatypeProperty ; rdfs:domain [ a owl:Restriction ; owl:allValuesFrom prefix:datatype ; owl:onProperty ns:datatypeProperty1 ] .`
	`ns:datatypeProperty1 a owl:DatatypeProperty ; rdfs:range prefix:datatype .`

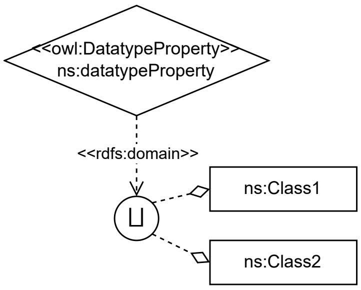
Intersection as domain of an datatype property.

Notation Example	OWL Code

 <<owl:DatatypeProperty>>
ns:datatypeProperty
<<rdfs:domain>>
⊓
ns:Class1
ns:Class2
	`ns:datatypeProperty a owl:DatatypeProperty ; rdfs:domain [ a owl:Restriction ; owl:intersectionOf ( ns:Class1 ns:Class2 ) ] .`
	`ns:Class1 a owl:Class .`
	`ns:Class2 a owl:Class .`

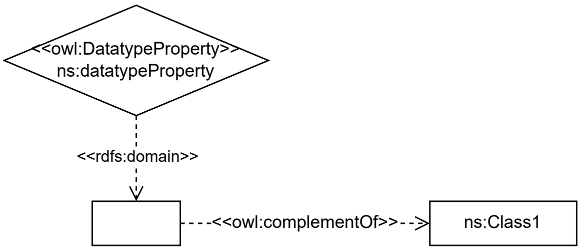
Union as domain of an datatype property.

Notation Example	OWL Code

 <<owl:DatatypeProperty>>
ns:datatypeProperty
<<rdfs:domain>>
⊔
ns:Class1
ns:Class2
	`ns:datatypeProperty a owl:DatatypeProperty ; rdfs:domain [ a owl:Restriction ; owl:unionOf ( ns:Class1 ns:Class2 ) ] .`
	
	`ns:Class1 a owl:Class .`
	
	`ns:Class2 a owl:Class .`

Complement as domain of a datatype property.

Notation Example	OWL Code

 <<owl:DatatypeProperty>>
ns:datatypeProperty
<<rdfs:domain>>
<<owl:complementOf>>
ns:Class1
	`ns:datatypeProperty a owl:DatatypeProperty ; rdfs:domain [ a owl:Restriction ; owl:complementOf ns:Class1 ] .`
	
	`ns:Class1 a owl:Class .`

María Poveda-Villalón
Contact email: chowlk@delicias.dia.fi.upm.es
Latest revision: March, 2025