OWL DL

Third Part

Valentina Bartalesi Lenzi

Istituto di Scienza e Tecnologie dell'Informazione - Consiglio Nazionale delle Ricerche – Pisa

A.A. 2024-2025

OWL 2 DL Ontologies

OWL 2 DL ontologies consist of three different syntactic categories:

1. Entities: classes, properties, and individuals, identified by IRIs. They form the primitive terms and the basic elements of an ontology
2. Expressions: complex notions capturing the intensions of classes and properties
3. **Axioms**: statements that are asserted to be true

Axioms

Axioms are the logical content of an OWL ontology

Axioms express **statements** that are **true** in the domain that the ontology models

There are several kinds of axioms:

- **Declarations**
- **Axioms about classes**
- **Axioms about object or data properties**
- **Datatype definitions**
- **Keys**
- **Assertions, also called *facts***
- **Axioms about annotations**

Axioms

Axioms are the logical content of an OWL ontology

Axioms express **statements** that are **true** in the domain that the ontology models

There are several kinds of axioms:

- **Declarations**
- Axioms about classes
- Axioms about object or data properties
- Datatype definitions
- Keys
- Assertions, also called *facts*
- Axioms about annotations

Declarations

Each IRI used in an OWL 2 ontology can be declared

To **declare** an entity in an ontology, one can simply i**nclude the appropriate axiom in the ontology**

These axioms are non-logical in the sense that they do not affect the consequences of an OWL 2 ontology

A declaration for the IRI I in O serves two purposes:

1. A declaration says that **I exists** — that is, it says that **I is part of the vocabulary of O**
2. A declaration **associates with I an entity type** - that is, it says whether I is used in O as a class, datatype, object property, data property, annotation property, an individual, or a combination thereof

Declarations for the built-in entities of OWL 2 are implicitly present in every OWL 2 ontology

Declarations

Functional Syntax	RDF Syntax
Declaration (Class(CN))	CN rdf:type owl:Class .
Declaration (Datatype(DN))	DN rdf:type rdfs:Datatype .
Declaration (ObjectProperty(PN))	PN rdf:type owl:ObjectProperty .
Declaration (DataProperty(R))	R rdf:type owl:DatatypeProperty .
Declaration (AnnotationProperty(A))	A rdf:type owl:AnnotationProperty .
Declaration (NamedIndividual(aN))	aN rdf:type owl:NamedIndividual .

Declarations

Declarations should satisfy some **obvious constraints**

Property typing constraints:

- if an object/data/annotation property with IRI I occurs in an ontology O, I is consistently declared in O as an object/data/annotation property
- no IRI is declared to be of two different property types

Class typing constraints:

- if a class/datatype with IRI I occurs in an ontology O, I is consistently declared in O as a class/datatype
- no IRI is declared to be both a class and a datatype

All other declarations are optional

Axioms

Axioms are the logical content of an OWL ontology

Axioms express **statements** that are **true** in the domain that the ontology models

There are several kinds of axioms:

- Declarations
- **Axioms about classes**
- Axioms about object or data properties
- Datatype definitions
- Keys
- Assertions, also called *facts*
- Axioms about annotations

Axioms about Classes

Axioms about classes establish **relationships between class expressions**

There are four kinds of class axioms:

1. **subclass axioms**
2. **equivalent class axioms**
3. **disjoint class axioms**
4. **disjoint union axioms**

Note: **subclass and equivalent class axioms** are present in RDF Schema too, however in **OWL they connect *class expressions*,** while in **RDF** Schema they **connect simply *classes***
In contrast, disjoint and disjoint union axioms are **not expressible** at all in RDF Schema

Subclass Axiom

Construct Name	Subclass Axiom
Construct Type	Axiom
Functional Syntax	SubClassOf(CE_1 CE_2)
Description	true (in I) if the extension of class expression CE_1 is a subset of the extension of class expression CE_2 (in I)
Semantics	$(CE_1)^C \subseteq (CE_2)^C$
RDF Turtle Syntax	CE_1 rdfs:subClassOf CE_2 .
Example	SubClassOf(Male Person)

Subclass Axiom: Example

Let us consider an ontology O with the following 5 axioms:

1. A person who has a child has at least one boy or a girl:
SubClassOf(PersonWithChild
ObjectSomeValuesFrom(hasChild ObjectUnionOf(Boy Girl)))

2. Each boy is a child: SubClassOf(Boy Child)

3. Each girl is a child: SubClassOf(Girl Child)

4. If some resource has a child, then this resource is a parent:
SubClassOf(ObjectSomeValuesFrom(hasChild Child) Parent)

5. Then O entails:
SubClassOf(PersonWithChild Parent)

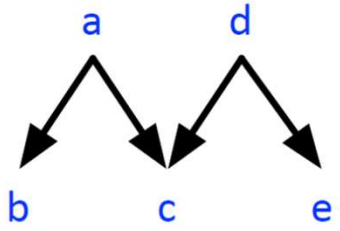
In every interpretation in which (1)-(4) are true, also (5) is true

Subclass Axiom: Example

To verify the entailment, let us consider a "reasonable" interpretation I

$\Delta_I = \{a, b, c, d, e\}$
$(\text{Boy})^C = \{b, c\}$
$(\text{Girl})^C = \{e\}$
$(\text{Child})^C = \{b, c, e\}$
$(\text{PersonWithChild})^C = \{a, d\}$
$(\text{Parent})^C = \{a, d\}$
$(\text{ObjectUnionOf(Boy Girl))}^I = \{b, c, e\}$
$(\text{ObjectSomeValuesFrom(hasChild ObjectUnionOf(Boy Girl)))}^I = \{a, d\}$
$(\text{ObjectSomeValuesFrom(hasChild Child))}^I = \{a, d\}$

$(\text{hasChild})^{OP}$:



This interpretation is a model of the ontology O because it satisfies all axioms in O. And it satisfies also (5). So the entailment is verified by I.

Equivalent Class Axiom

Construct Name	Equivalent Classes Axiom
Construct Type	Axiom
Functional Syntax	EquivalentClasses($CE_1 \ldots CE_n$) $n \geq 2$
Description	class expressions $CE_1 \ldots CE_n$ are all synonymous of each other
Semantics	$(CE_1)^C = \ldots = (CE_n)^C$
RDF Turtle Syntax	CE_1 owl:equivalentClass CE_2 . ... CE_{n-1} owl:equivalentClass CE_n .
Example	EquivalentClasses(Person Persona Personne)

Disjoint Class Axiom

Construct Name	Disjoint Classes Axiom
Construct Type	Axiom
Functional Syntax	DisjointClasses($CE_1 \ldots CE_n$) $n \geq 2$
Description	class expressions $CE_1 \ldots CE_n$ are pairwise disjoint, so no individual can be an instance of any two of them
Semantics	$(CE_j)^C \cap (CE_k)^C = \emptyset$ for each $1 \leq j, k \leq n$ such that $j \neq k$
RDF Turtle Syntax	_:x rdf:type owl:AllDisjointClasses . _:x owl:members ($CE_1 \ldots CE_n$) .
Example	DisjointClasses (Adult Young)

Disjoint Union Axiom

Construct Name	Disjoint Union Axiom
Construct Type	Axiom
Functional Syntax	DisjointUnion(C $CE_1 \ldots CE_n$) $n \geq 2$
Description	each instance of C is an instance of exactly one CE_i, and each instance of CE_i is an instance of C
Semantics	$(C)^C = (CE_1)^C \cup \ldots \cap (CE_n)^C$ and $(CE_j)^C \cap (CE_k)^C = \emptyset$ for each $1 \leq j, k \leq n$ such that $j \neq k$
RDF Turtle Syntax	C owl:disjointUnionOf ($CE_1 \ldots CE_n$) .
Example	DisjointUnion (CnrEmployee Researcher Technologist)

Axioms

Axioms are the logical content of an OWL ontology

Axioms express **statements** that are **true** in the domain that the ontology models

There are several kinds of axioms:

- Declarations
- Axioms about classes
- **Axioms about object** or data **properties**
- Datatype definitions
- Keys
- Assertions, also called *facts*
- Axioms about annotations

Object Property Axioms

OWL 2 provides axioms that can be used to characterize and **establish relationships between object property expressions**
These axioms are of the following kinds:

- **object sub-property** and **complex role inclusion axioms**
- **equivalent** object property axioms
- **disjoint** object property axioms
- **inverse** object property axioms
- **domain** object property axioms
- **range** object property axioms
- **functional** and **inverse functional** object property axioms
- **reflexive** and **irreflexive** object property axioms
- **symmetric** and **asymmetric** object property axioms
- **transitive** object property axioms

Object Sub-Property Axiom

Construct Name	Object Sub-property Axiom
Construct Type	Axiom
Functional Syntax	SubObjectPropertyOf(OPE_1 OPE_2)
Description	object property expression OPE_1 is a sub-property of object property expression OPE_2
Semantics	$(OPE_1)^{OP} \subseteq (OPE_2)^{OP}$
RDF Turtle Syntax	OPE_1 rdfs:subPropertyOf OPE_2 .
Example	SubObjectPropertyOf(isBestFriendOf isFriendOf)

Complex Role Inclusion Axiom

Construct Name	Complex Role Inclusion Axiom
Construct Type	Axiom
Functional Syntax	SubObjectPropertyOf(ObjectPropertyChain($\mathrm{OPE}_1 \ldots \mathrm{OPE}_n$) OPE)
Description	the composition of object property expressions $\mathrm{OPE}_1, \ldots, \mathrm{OPE}_n$ is a sub-property of object property expression OPE
Semantics	$\forall y_0, y_1, \ldots, y_n : (y_0, y_1) \in (OPE_1)^{OP}$ and ... and $(y_{n-1}, y_n) \in (OPE_n)^{OP}$ imply $(y_0, y_n) \in (OPE)^{OP}$
RDF Turtle Syntax	OPE owl:propertyChainAxiom ($\mathrm{OPE}_1 \ldots \mathrm{OPE}_n$) .

Complex Role Inclusion Axiom

The expression ObjectPropertyChain($OPE_1 \ldots OPE_n$) denotes the **mathematical composition** (◦) **of the extensions of its operands**, e.g:

A student who takes an exam in a session of a course edition is registered to that course edition

V_C = {:Student, :Exam, :Session, :CourseEdition}
V_{OP} = {:takes, :isExamOf, :isSessionOf, :isRegisteredTo}

SubObjectPropertyOf(ObjectPropertyChain(:takes :isExamOf :isSessionOf) :isRegisteredTo)

(takes ∘ isExamOf ∘ isSessionOf) ⊆ isRegisteredTo

(marco, exam 6/12/2022) ∘ (exam 6/12/2022, session 12/2022) ∘ (session 12/2022, course edition 2022) → (marco, course edition 2022)

Disjoint Object Properties Axiom

Construct Name	Disjoint Object Properties Axiom
Construct Type	Axiom
Functional Syntax	DisjointObjectProperties($\mathrm{OPE}_1 \ldots \mathrm{OPE}_n$) $n \geq 2$
Description	object property expressions $\mathrm{OPE}_1, \ldots, \mathrm{OPE}_n$ are pairwise disjoint, so no pair of individuals can be in the extension of any two of them
Semantics	$(\mathrm{OPE}_j)^{OP} \cap (\mathrm{OPE}_k)^{OP} = \emptyset$ for each $1 \leq j, k \leq n$ such that $j \neq k$
RDF Turtle Syntax	_:x rdf:type owl:AllDisjointProperties . _:x owl:members ($\mathrm{OPE}_1 \ldots \mathrm{OPE}_n$) .
Example	DisjointObjectProperties(hasBrother hasSister)

Inverse Object Properties Axiom

Construct Name	Inverse Object Properties Axiom
Construct Type	Axiom
Functional Syntax	InverseObjectProperties(OPE_1 OPE_2)
Description	the object property expression OPE_1 is an inverse of the object property expression OPE_2. Thus, if an individual x is connected by OPE_1 to an individual y, then y is also connected by OPE_2 to x, and vice versa.
Semantics	$(\mathrm{OPE}_1)^{OP} = \{(y, x) \mid (x, y) \in (\mathrm{OPE}_2)^{OP}\}$
RDF Turtle Syntax	OPE_1 owl:inverseOf OPE_2 .
Example	InverseObjectProperties(isChildOf isParentOf)

Object Property Domain Axiom

Construct Name	Object Property Domain Axiom
Construct Type	Axiom
Functional Syntax	ObjectPropertyDomain(OPE CE)
Description	the domain of the object property expression OPE is the class expression CE, *i.e.*, if an individual x is connected by OPE with some other individual, then x is an instance of CE
Semantics	$\forall x, y : (x, y) \in (\text{OPE})^{OP}$ implies $x \in (\text{CE})^{C}$
RDF Turtle Syntax	OPE rdfs:domain CE .
Example	ObjectPropertyDomain(hasFather Person)

Object Property Range Axiom

Construct Name	Object Property Range Axiom
Construct Type	Axiom
Functional Syntax	ObjectPropertyRange(OPE CE)
Description	the range of the object property expression OPE is the class expression CE, *i.e.*, if an individual is connected by OPE with some individual y, then y is an instance of CE
Semantics	$\forall x, y : (x, y) \in (\text{OPE})^{OP}$ implies $y \in (\text{CE})^{C}$
RDF Turtle Syntax	OPE rdfs:range CE .
Example	ObjectPropertyRange(hasSister Female)

Functional Object Property Axiom

Construct Name	Functional Object Property Axiom
Construct Type	Axiom
Functional Syntax	FunctionalObjectProperty(OPE)
Description	for each individual x, there can be at most one distinct individual y such that x is connected by OPE to y
Semantics	$\forall x, y_1, y_2 : (x, y_1) \in (\text{OPE})^{OP}$ and $(x, y_2) \in (\text{OPE})^{OP}$ imply $y_1 = y_2$
RDF Turtle Syntax	OPE rdf:type owl:FunctionalProperty .
Example	FunctionalObjectProperty(hasMother)

Inverse Functional Object Property Axiom

Construct Name	Inverse Functional Object Property Axiom
Construct Type	Axiom
Functional Syntax	InverseFunctionalObjectProperty(OPE)
Description	for each individual x, there can be at most one distinct individual y such that y is connected by OPE with x
Semantics	$\forall x_1, x_2, y : (x_1, y) \in (\text{OPE})^{OP}$ and $(x_2, y) \in (\text{OPE})^{OP}$ imply $x_1 = x_2$
RDF Turtle Syntax	OPE rdf:type owl:InverseFunctionalProperty .
Example	InverseFunctionalObjectProperty(motherOf)

(Irr)Reflexive Object Property Axiom

Construct Name	(Irr)Reflexive Object Property Axiom
Construct Type	Axiom
Functional Syntax	(Irr)ReflexiveObjectProperty(OPE)
Description	each individual x is (not) connected by OPE to itself
Semantics	$\forall x : x \in \Delta_I$ implies $(x, x) \in (\text{OPE})^{OP}$ $(\forall x : x \in \Delta_I$ implies $(x, x) \notin (\text{OPE})^{OP})$
RDF Turtle Syntax	OPE rdf:type owl:ReflexiveProperty . (OPE rdf:type owl:IrreflexiveProperty .)
Example	ReflexiveObjectProperty(knows) IrreflexiveObjectProperty(marriedTo)

(A)Symmetric Object Property Axiom

Construct Name	(As)Symmetric Object Property Axiom
Construct Type	Axiom
Functional Syntax	(As)SymmetricObjectProperty(OPE)
Description	if individual x is connected by OPE to individual y, then y is (not) connected by OPE to x
Semantics	$\forall x, y : (x, y) \in (\text{OPE})^{OP}$ implies $(y, x) \in (\text{OPE})^{OP}$ $(\forall x, y : (x, y) \in (\text{OPE})^{OP}$ implies $(y, x) \notin (\text{OPE})^{OP})$
RDF Turtle Syntax	OPE rdf:type owl:SymmetricProperty . (OPE rdf:type owl:AsymmetricProperty .)
Example	SymmetricObjectProperty(isRelativeOf) AsymmetricObjectProperty(hasFather)

Transitive Object Property Axiom

Construct Name	Transitive Object Property Axiom
Construct Type	Axiom
Functional Syntax	TransitiveObjectProperty(OPE)
Description	if OPE connects an individual x with an individual y, and y with an individual z, then OPE connects x to z
Semantics	$\forall x, y, z : (x, y) \in (\text{OPE})^{OP}$ and $(y, z) \in (\text{OPE})^{OP}$ imply $(x, z) \in (\text{OPE})^{OP}$
RDF Turtle Syntax	OPE rdf:type owl:TransitiveProperty .
Example	TransitiveObjectProperty (isDescendantOf)

Axioms

Axioms are the logical content of an OWL ontology

Axioms express **statements** that are **true** in the domain that the ontology models

There are several kinds of axioms:

- Declarations
- Axioms about classes
- **Axioms about** object or **data properties**
- Datatype definitions
- Keys
- Assertions, also called *facts*
- Axioms about annotations

Data Property Axioms

OWL 2 also provides for data property axioms, which are similar to object property axioms

These axioms are of the following kinds:

- **data sub-property** axioms
- **equivalent** data property axioms
- **disjoint** data property axioms
- **domain** data property axioms
- **range** data property axioms
- **functional** data property axioms

Data Sub-Property Axiom

Construct Name	Data Sub-property Axiom
Construct Type	Axiom
Functional Syntax	SubDataPropertyOf(DPE_1 DPE_2)
Description	data property expression DPE_1 is a sub-property of data property expression DPE_2
Semantics	$(DPE_1)^{DP} \subseteq (DPE_2)^{DP}$
RDF Turtle Syntax	DPE_1 rdfs:subPropertyOf DPE_2 .
Example	SubDataPropertyOf(hasLastName hasName)

Equivalent Data Properties Axiom

Construct Name	Equivalent Data Properties Axiom
Construct Type	Axiom
Functional Syntax	EquivalentDataProperties($\mathrm{DPE}_1 \ldots \mathrm{DPE}_n$) $n \geq 2$
Description	data property expressions $\mathrm{DPE}_1, \ldots, \mathrm{DPE}_n$ are all synonymous of each other
Semantics	$(\mathrm{DPE}_1)^{DP} = \ldots = (\mathrm{DPE}_n)^{DP}$
RDF Turtle Syntax	DPE_j owl:equivalentProperty DPE_{j+1} . $j = 1 \ldots n-1$
Example	EquivalentDataProperties (hasName siChiama)

Disjoint Data Properties Axiom

Construct Name	Disjoint Data Properties Axiom
Construct Type	Axiom
Functional Syntax	DisjointDataProperties($\text{DPE}_1 \ldots \text{DPE}_n$) $n \geq 2$
Description	Data property expressions $\text{DPE}_1, \ldots, \text{DPE}_n$ are pairwise disjoint, so no pair of individual-literal can be an instance of any two of them
Semantics	$(\text{DPE}_j)^{DP} \cap (\text{DPE}_k)^{DP} = \emptyset$ for each $1 \leq j, k \leq n$ such that $j \neq k$
RDF Turtle Syntax	_:x rdf:type owl:AllDisjointProperties . _:x owl:members ($\text{DPE}_1 \ldots \text{DPE}_n$) .
Example	DisjointDataProperties(hasName hasAddress)

Data Property Domain Axiom

Construct Name	Data Property Domain Axiom
Construct Type	Axiom
Functional Syntax	DataPropertyDomain(DPE CE)
Description	the domain of the data property expression DPE is the class expression CE, *i.e.*, if an individual x is connected by DPE with some other individual, then x is an instance of CE
Semantics	$\forall x, y : (x, y) \in (\text{DPE})^{DP}$ implies $x \in (\text{CE})^{C}$
RDF Turtle Syntax	DPE rdfs:domain CE .
Example	DataPropertyDomain(hasName Person)

Data Property Range Axiom

Construct Name	Data Property Range Axiom
Construct Type	Axiom
Functional Syntax	DataPropertyRange(DPE DR)
Description	the range of the data property expression DPE is the data range DR, *i.e.*, if an individual is connected by DPE with some literal y, then y is an instance of DR
Semantics	$\forall x, y : (x, y) \in (\text{DPE})^{DP}$ implies $y \in (\text{DR})^{DT}$
RDF Turtle Syntax	DPE rdfs:range DR .
Example	DataPropertyRange(hasName xsd:string)

Functional Data Property Axiom

Construct Name	Functional Data Property Axiom
Construct Type	Axiom
Functional Syntax	FunctionalDataProperty(DPE)
Description	for each individual x, there can be at most one distinct literal y such that x is connected by DPE to y
Semantics	$\forall x, y_1, y_2 : (x, y_1) \in (\text{DPE})^{DP}$ and $(x, y_2) \in (\text{DPE})^{DP}$ imply $y_1 = y_2$
RDF Turtle Syntax	DPE rdf:type owl:FunctionalProperty .
Example	FunctionalDataProperty(hasFiscalCode)

Axioms

Axioms are the logical content of an OWL ontology

Axioms express **statements** that are **true** in the domain that the ontology models

There are several kinds of axioms:

- Declarations
- Axioms about classes
- Axioms about object or data properties
- **Datatype definitions**
- Keys
- Assertions, also called *facts*
- Axioms about annotations

Datatypes Definitions

A datatype definition defines a new datatype DT as being semantically equivalent to (*i.e.*, a synonym of) a specified data range.

Construct Name	Datatype Definition
Construct Type	Axiom
Functional Syntax	DatatypeDefinition(DT DR)
Description	for each literal x, x is an instance of datatype DT if and only if it is an instance of data range DR
Semantics	$(\mathrm{DT})^{DT} = (\mathrm{DR})^{DT}$
RDF Turtle Syntax	DT owl:equivalentClass DR .

Datatypes Definitions

```
Declaration( Datatype( CodiceFiscale ) )
DatatypeDefinition(
    CodiceFiscale
    DatatypeRestriction(
        xsd:string
        xsd:pattern
        "[A-Z]{6}-[0-9]{2}-[A-Z]{1}-[0-9]{2}-[A-Z]{1}-[0-9]{3}-[A-Z]{1}" ) )
```

Axioms

Axioms are the logical content of an OWL ontology

Axioms express **statements** that are **true** in the domain that the ontology models

There are several kinds of axioms:

- Declarations
- Axioms about classes
- Axioms about object or data properties
- Datatype definitions
- **Keys**
- Assertions, also called *facts*
- Axioms about annotations

Keys

Similarly to keys in databases, a key in an OWL ontology is a **set of object or data property expressions whose values uniquely identify** the **instances of a class expression**

More technically, a key axiom states that each **named instance** of a given class expression CE is **uniquely identified by the value of a given object property expressions OPE_i and/or by given data property expressions DPE_j**

Construct Name	Key Axiom
Construct Type	Axiom
Functional Syntax	HasKey(CE ($OPE_1 \ldots OPE_m$) ($DPE_1 \ldots DPE_n$)) $m + n \geq 1$
Description	No two distinct named instances of CE can coincide on the values of all object property expressions OPE_i and of all data property expressions DPE_j.

Keys

Semantics	$\forall x, y, z_1, \ldots, z_m, w_1, \ldots, w_n$: if $x \in (\mathrm{CE})^C$, $x \in$NAMED, $y \in (\mathrm{CE})^C$, $y \in$NAMED, $(x, z_i) \in (\mathrm{OPE}_i)^{OP}$, $(y, z_i) \in (\mathrm{OPE}_i)^{OP}$, $z_i \in$ NAMED for $1 \leq i \leq m$, $(x, w_j) \in (\mathrm{DPE}_j)^{DP}$, $(y, w_j) \in (\mathrm{DPE}_j)^{DP}$ for $1 \leq j \leq n$, $x = y$
RDF Turtle Syntax	CE owl:hasKey ($\mathrm{OPE}_1 \ldots \mathrm{OPE}_m$ $\mathrm{DPE}_1 \ldots \mathrm{DPE}_n$) .
Example	HasKey(Person () (HasFiscalCode))

Note: if a key axiom is violated by two named individuals a and b, this does not per sé cause an inconsistency. The inconsistency arises only if the ontology states or entails that a and b are *different* individuals.

Axioms

Axioms are the logical content of an OWL ontology

Axioms express **statements** that are **true** in the domain that the ontology models

There are several kinds of axioms:

- Declarations
- Axioms about classes
- Axioms about object or data properties
- Datatype definitions
- Keys
- **Assertions, also called *facts***
- Axioms about annotations

Assertions

Assertions are **axioms about individuals**, establishing r**elationships between individuals**, or **between individuals and class or property expressions**.

There are four broad kinds of assertions:

1. **same or different individual assertions**
2. **class assertions**
3. **negative or positive object property assertions**
4. **negative or positive data property assertions**

Note: class assertions and positive property assertions are present in RDF Schema too; however, in OWL they connect individuals to class and property *expressions,* while in RDF they connect individuals to classes and properties only, respectively.

In contrast, the other kinds of assertions are not expressible in RDF Schema

Individual Equality Assertion

Construct Name	Individual Equality Assertion
Construct Type	Axiom
Functional Syntax	SameIndividual($a_1 \ldots a_n$)
Description	all the named or anonymous individuals $a_1 \ldots a_n$ are equal to each other
Semantics	$(a_j)^I = (a_k)^I$ for each $1 \leq j, k \leq n$
RDF Turtle Syntax	a_j owl:sameAs a_{j+1} . $j = 1 \ldots n-1$
Example	SameIndividual(Batman BruceWayne _:bat)

Individual Inequality Assertion

Construct Name	Individual Inquality Assertion
Construct Type	Axiom
Functional Syntax	DifferentIndividuals($a_1 \ldots a_n$)
Description	all the named or anonymous individuals $a_1 \ldots a_n$ are different from each other
Semantics	$(a_j)^I \neq (a_k)^I$ for each $1 \leq j \neq k \leq n$
RDF Turtle Syntax	_:x rdf:type owl:AllDifferent . _:x owl:members ($a_1 \ldots a_n$) .
Example	DifferentIndividuals(Superman BruceWayne _:robin)

Class Assertion

Construct Name	Class Assertion
Construct Type	Axiom
Functional Syntax	ClassAssertion(CE a)
Description	the named or anonymous individual a is an instance of class expression CE
Semantics	$(a)^I \in (CE)^C$
RDF Turtle Syntax	a rdf:type CE .
Example	ClassAssertion(Person carlo)

Positive Object Property Assertion

Construct Name	Positive Object Property Assertion
Construct Type	Axiom
Functional Syntax	ObjectPropertyAssertion(OPE a_1 a_2)
Description	the individual a_1 is connected to individual a_2 by object property expression OPE
Semantics	$((a_1)^I,(a_2)^I) \in (\mathrm{OPE})^{OP}$
RDF Turtle Syntax	a_1 OPE a_2 .
Example	ObjectPropertyAssertion(hasMother carlo giovanna)

Negative Object Property Assertion

Construct Name	Negative Object Property Assertion
Construct Type	Axiom
Functional Syntax	NegativeObjectPropertyAssertion(OPE a_1 a_2)
Description	the individual a_1 is not connected to individual a_2 by object property expression OPE
Semantics	$((a_1)^I,(a_2)^I) \notin (\mathrm{OPE})^{OP}$
RDF Turtle Syntax	_:x rdf:type owl:NegativePropertyAssertion . _:x owl:sourceIndividual a_1 . _:x owl:assertionProperty OPE . _:x owl:targetIndividual a_2 .
Example	NegativeObjectPropertyAssertion(hasMother carlo clara)

Positive Data Property Assertion

Construct Name	Positive Data Property Assertion
Construct Type	Axiom
Functional Syntax	DataPropertyAssertion(DPE a lt)
Description	the individual a is connected to literal lt by data property expression DPE
Semantics	$((a)^I,(lt)^{LT}) \in (DPE)^{DP}$
RDF Turtle Syntax	a DPE lt .
Example	DataPropertyAssertion(wasBorn Mozart "1756-01-27"^^xsd:date)

Negative Data Property Assertion

Construct Name	Negative Data Property Assertion
Construct Type	Axiom
Functional Syntax	NegativeDataPropertyAssertion(DPE a lt)
Description	the individual a is not connected to literal lt by data property expression DPE
Semantics	$((a)^I,(lt)^{LT}) \notin (DPE)^{DP}$
RDF Turtle Syntax	_:x rdf:type owl:NegativePropertyAssertion . _:x owl:sourceIndividual a . _:x owl:assertionProperty DPE . _:x owl:targetValue lt .
Example	NegativeDataPropertyAssertion(wasBorn Mozart "1956-01-27"^^xsd:date)

Axioms

Axioms are the logical content of an OWL ontology

Axioms express **statements** that are **true** in the domain that the ontology models

There are several kinds of axioms:

- Declarations
- Axioms about classes
- Axioms about object or data properties
- Datatype definitions
- Keys
- Assertions, also called *facts*
- **Axioms about annotations**

Axioms about annotations

- **Annotation Assertion**. The annotation assertion AnnotationAssertion(*AP as av*) states that the annotation subject *as*, an IRI or an anonymous individual, is annotated with the annotation property *AP* and the annotation value *av*.
  Example: AnnotationAssertion(rdfs:label :Person "Represents the set of all people.")
- **Annotation Subproperties**. The annotation subproperty axiom SubAnnotationPropertyOf(*AP1 AP2*) states that the annotation property *AP1* is a subproperty of the annotation property *AP2*
- **Annotation Property Domain.** An annotation property domain axiom AnnotationPropertyDomain(*AP U*) states that the domain of the annotation property *AP* is the IRI *U*
- **Annotation Property Range.** An annotation property range axiom AnnotationPropertyRange(AP *U*) states that the range of the annotation property *AP* is the IRI *U*

Useful Readings

- OWL 2 Web Ontology Language Document Overview (Second Edition)
- OWL 2 Web Ontology Language Primer (Second Edition)
- OWL 2 web ontology language profiles