OWL Web Ontology Language Overview

The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics.

OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full.

Why OWL?

The Semantic Web is a vision for the future of the Web in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web. The Semantic Web will build on XML's ability to define customized tagging schemes and RDF's flexible approach to representing data.

The first level above RDF required for the Semantic Web is an ontology language what can formally describe the meaning of terminology used in Web documents. If machines are expected to perform useful reasoning tasks on these documents, the language must go beyond the basic semantics of RDF Schema.

OWL has been designed to meet this need for a Web Ontology Language.

OWL is part of the growing stack of W3C recommendations related to the Semantic Web.

- XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents.
- XML Schema is a language for restricting the structure of XML documents and also extends XML with datatypes.

- RDF is a datamodel for objects ("resources") and relations between them, provides a simple semantics for this datamodel, and these datamodels can be represented in an XML syntax.
- RDF Schema is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalization hierarchies of such properties and classes.

- OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.

The three sublanguages of OWL

OWL provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users.

- *OWL Lite* supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. OWL Lite also has a lower formal complexity than OWL DL.

- *OWL DL* supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with *description logics*, a field of research that has studied the logics that form the formal foundation of OWL.

- *OWL Full* is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

Ontology developers adopting OWL should consider which sublanguage best suits their needs. The choice between OWL Lite and OWL DL depends on the extent to which users require the more-expressive constructs provided by OWL DL. The choice between OWL DL and OWL Full mainly depends on the extent to which users require the meta-modeling facilities of RDF Schema (e.g. defining classes of classes, or attaching properties to classes). When using OWL Full as compared to OWL DL, reasoning support is less predictable since complete OWL Full implementations do not currently exist.

Language Description of OWL Lite

OWL Lite uses only some of the OWL language features and has more limitations on the use of the features than OWL DL or OWL Full. For example, in OWL Lite classes can only be defined in terms of named superclasses (superclasses cannot be arbitrary expressions), and only certain kinds of class restrictions can be used.

Equivalence between classes and subclass relationships between classes are also only allowed between named classes, and not between arbitrary class expressions. Similarly, restrictions in OWL Lite use only named classes. OWL Lite also has a limited notion of cardinality - the only cardinalities allowed to be explicitly stated are 0 or 1.

OWL Lite RDF Schema Features

- ***Class***: A class defines a group of individuals that belong together because they share some properties. For example, Deborah and Frank are both members of the class Person. Classes can be organized in a specialization hierarchy using *subClassOf*. There is a built-in most general class named Thing that is the class of all individuals and is a superclass of all OWL classes. There is also a built-in most specific class named Nothing that is the class that has no instances and a subclass of all OWL classes.

- ***rdfs:subClassOf***: Class hierarchies may be created by making one or more statements that a class is a subclass of another class. For example, the class Person could be stated to be a subclass of the class Mammal. From this a reasoner can deduce that if an individual is a Person, then it is also a Mammal.

- ***rdf:Property***: Properties can be used to state relationships between individuals or from individuals to data values. Examples of properties include hasChild, hasRelative, hasSibling, and hasAge. The first three can be used to relate an instance of a class Person to another instance of the class Person (and are thus occurences of ObjectProperty), and the last (hasAge) can be used to relate an instance of the class Person to an instance of the datatype Integer (and is thus an occurence of DatatypeProperty). Both owl:ObjectProperty and owl:DatatypeProperty are subclasses of the RDF class rdf:Property.

- ***rdfs:subPropertyOf***: Property hierarchies may be created by making one or more statements that a property is a subproperty of one or more other properties. For example, hasSibling may be stated to be a subproperty of hasRelative. From this a reasoner can deduce that if an individual is related to another by the hasSibling property, then it is also related to the other by the hasRelative property.

- ***rdfs:domain***: A domain of a property limits the individuals to which the property can be applied. If a property relates an individual to another individual, and the property has a class as one of its domains, then the individual must belong to the class. For example, the property hasChild may be stated to have the domain of Mammal. From this a reasoner can deduce that if Frank hasChild Anna, then Frank must be a Mammal. Note that *rdfs:domain* is called a global restriction since the restriction is stated on the property and not just on the property when it is associated with a particular class.

- ***rdfs:range***: The range of a property limits the individuals that the property may have as its value. If a property relates an individual to another individual, and the property has a class as its range, then the other individual must belong to the range class. For example, the property hasChild may be stated to have the range of Mammal. From this a reasoner can deduce that if Louise is related to Deborah by the hasChild property, (i.e., Deborah is the child of Louise), then Deborah is a Mammal. Range is also a global restriction as is domain above.

- ***Individual***: Individuals are instances of classes, and properties may be used to relate one individual to another. For example, an individual named Deborah may be described as an instance of the class Person and the property hasEmployer may be used to relate the individual Deborah to the individual StanfordUniversity.

OWL Lite Equality and Inequality

- ***equivalentClass***: Two classes may be stated to be equivalent. Equivalent classes have the same instances. Equality can be used to create synonymous classes. For example, Car can be stated to be *equivalentClass* to Automobile. From this a reasoner can deduce that any individual that is an instance of Car is also an instance of Automobile and vice versa.

- ***equivalentProperty***: Two properties may be stated to be equivalent. Equivalent properties relate one individual to the same set of other individuals. Equality may be used to create synonymous properties. For example, hasLeader may be stated to be the *equivalentProperty* to hasHead. From this a reasoner can deduce that if X is related to Y by the property hasLeader, X is also related to Y by the property hasHead and vice versa. A reasoner can also deduce that hasLeader is a subproperty of hasHead and hasHead is a subProperty of hasLeader.

- ***sameAs***: Two individuals may be stated to be the same. These constructs may be used to create a number of different names that refer to the same individual. For example, the individual Deborah may be stated to be the same individual as DeborahMcGuinness.

- ***differentFrom***: An individual may be stated to be different from other individuals. For example, the individual Frank may be stated to be different from the individuals Deborah and Jim. Thus, if the individuals Frank and Deborah are both values for a property that is stated to be functional (thus the property has at most one value), then there is a contradiction.

Explicitly stating that individuals are different can be important in when using languages such as OWL (and RDF) that do not assume that individuals have one and only one name. For example, with no additional information, a reasoner will not deduce that Frank and Deborah refer to distinct individuals.

- ***AllDifferent***: A number of individuals may be stated to be mutually distinct in one AllDifferent statement. For example, Frank, Deborah, and Jim could be stated to be mutually distinct using the AllDifferent construct. Unlike the differentFrom statement above, this would also enforce that Jim and Deborah are distinct (not just that Frank is distinct from Deborah and Frank is distinct from Jim).

The AllDifferent construct is particularly useful when there are sets of distinct objects and when modelers are interested in enforcing the unique names assumption within those sets of objects. It is used in conjunction with distinctMembers to state that all members of a list are distinct and pairwise disjoint.

OWL Lite Property Characteristics

- ***inverseOf***: One property may be stated to be the inverse of another property. If the property P1 is stated to be the inverse of the property P2, then if X is related to Y by the P2 property, then Y is related to X by the P1 property. For example, if hasChild is the inverse of hasParent and Deborah hasParent Louise, then a reasoner can deduce that Louise hasChild Deborah.

- ***TransitiveProperty***: Properties may be stated to be transitive. If a property is transitive, then if the pair (x,y) is an instance of the transitive property P, and the pair (y,z) is an instance of P, then the pair (x,z) is also an instance of P. For example, if ancestor is stated to be transitive, and if Sara is an ancestor of Louise (i.e., (Sara,Louise) is an instance of the property ancestor) and Louise is an ancestor of Deborah (i.e., (Louise,Deborah) is an instance of the property ancestor), then a reasoner can deduce that Sara is an ancestor of Deborah (i.e., (Sara,Deborah) is an instance of the property ancestor).

OWL Lite (and OWL DL) impose the side condition that transitive properties (and their superproperties) cannot have a maxCardinality 1 restriction. Without this side-condition, OWL Lite and OWL DL would become undecidable languages.

- ***SymmetricProperty***: Properties may be stated to be symmetric. If a property is symmetric, then if the pair (x,y) is an instance of the symmetric property P, then the pair (y,x) is also an instance of P. For example, friend may be stated to be a symmetric property. Then a reasoner that is given that Frank is a friend of Deborah can deduce that Deborah is a friend of Frank.

- ***FunctionalProperty***: Properties may be stated to have a unique value. If a property is a FunctionalProperty, then it has no more than one value for each individual (it may have no values for an individual). This characteristic has been referred to as having a unique property. FunctionalProperty is shorthand for stating that the property's minimum cardinality is zero and its maximum cardinality is 1.

For example, hasPrimaryEmployer may be stated to be a FunctionalProperty. From this a reasoner may deduce that no individual may have more than one primary employer. This does not imply that every Person must have at least one primary employer however.

- ***InverseFunctionalProperty***: Properties may be stated to be inverse functional. If a property is inverse functional then the inverse of the property is functional. Thus the inverse of the property has at most one value for each individual. This characteristic has also been referred to as an unambiguous property.

For example, hasUSSocialSecurityNumber (a unique identifier for United States residents) may be stated to be inverse functional (or unambiguous). The inverse of this property (which may be referred to as isTheSocialSecurityNumberFor) has at most one value for any individual in the class of social security numbers. Thus any one person's social security number is the only value for their isTheSocialSecurityNumberFor property.

From this a reasoner can deduce that no two different individual instances of Person have the identical US Social Security Number. Also, a reasoner can deduce that if two instances of Person have the same social security number, then those two instances refer to the same individual.

OWL Lite Property Restrictions

OWL Lite allows restrictions to be placed on how properties can be used by instances of a class. These type restrictions (and the cardinality restrictions in the next subsection) are used within the context of an owl:Restriction. The owl:onProperty element indicates the restricted property. The following two restrictions limit which values can be used while the next section's restrictions limit how many values can be used.

- ***allValuesFrom***: The restriction *allValuesFrom* is stated on a property with respect to a class. It means that this property on this particular class has a local range restriction associated with it. Thus if an instance of the class is related by the property to a second individual, then the second individual can be inferred to be an instance of the local range restriction class. For example, the class Person may have a property called hasDaughter restricted to have *allValuesFrom* the class Woman.

This means that if an individual person Louise is related by the property hasDaughter to the individual Deborah, then from this a reasoner can deduce that Deborah is an instance of the class Woman. This restriction allows the property hasDaughter to be used with other classes, such as the class Cat, and have an appropriate value restriction associated with the use of the property on that class.

In this case, hasDaughter would have the local range restriction of Cat when associated with the class Cat and would have the local range restriction Person when associated with the class Person. Note that a reasoner can not deduce from an *allValuesFrom* restriction alone that there actually is at least one value for the property.

- ***someValuesFrom***: The restriction *someValuesFrom* is stated on a property with respect to a class. A particular class may have a restriction on a property that at least one value for that property is of a certain type. For example, the class SemanticWebPaper may have a *someValuesFrom* restriction on the hasKeyword property that states that some value for the hasKeyword property should be an instance of the class SemanticWebTopic.

This allows for the option of having multiple keywords and as long as one or more is an instance of the class SemanticWebTopic, then the paper would be consistent with the *someValuesFrom* restriction. Unlike *allValuesFrom*, *someValuesFrom* does not restrict all the values of the property to be instances of the same class.

If myPaper is an instance of the SemanticWebPaper class, then myPaper is related by the hasKeyword property to at least one instance of the SemanticWebTopic class. Note that a reasoner can not deduce (as it could with *allValuesFrom* restrictions) that all values of hasKeyword are instances of the SemanticWebTopic class.

OWL Lite Restricted Cardinality

OWL Lite includes a limited form of cardinality restrictions. OWL (and OWL Lite) cardinality restrictions are referred to as local restrictions, since they are stated on properties with respect to a particular class. That is, the restrictions constrain the cardinality of that property on instances of that class. OWL Lite cardinality restrictions are limited because they only allow statements concerning cardinalities of value 0 or 1 (they do not allow arbitrary values for cardinality, as is the case in OWL DL and OWL Full).

- ***minCardinality***: Cardinality is stated on a property with respect to a particular class. If a *minCardinality* of 1 is stated on a property with respect to a class, then any instance of that class will be related to at least one individual by that property. This restriction is another way of saying that the property is required to have a value for all instances of the class.

For example, the class Person would not have any minimum cardinality restrictions stated on a hasOffspring property since not all persons have offspring. The class Parent, however would have a minimum cardinality of 1 on the hasOffspring property. If a reasoner knows that Louise is a Person, then nothing can be deduced about a minimum cardinality for her hasOffspring property.

Once it is discovered that Louise is an instance of Parent, then a reasoner can deduce that Louise is related to at least one individual by the hasOffspring property. From this information alone, a reasoner can not deduce any maximum number of offspring for individual instances of the class parent. In OWL Lite the only minimum cardinalities allowed are 0 or 1.

- ***maxCardinality***: Cardinality is stated on a property with respect to a particular class. If a *maxCardinality* of 1 is stated on a property with respect to a class, then any instance of that class will be related to at most one individual by that property. A maxCardinality 1 restriction is sometimes called a functional or unique property. For example, the property hasRegisteredVotingState on the class UnitedStatesCitizens may have a maximum cardinality of one (because people are only allowed to vote in only one state).

From this a reasoner can deduce that individual instances of the class USCitizens may not be related to two or more distinct individuals through the hasRegisteredVotingState property. From a maximum cardinality one restriction alone, a reasoner can not deduce a minimum cardinality of 1. It may be useful to state that certain classes have no values for a particular property.

For example, instances of the class UnmarriedPerson should not be related to any individuals by the property hasSpouse. This situation is represented by a maximum cardinality of zero on the hasSpouse property on the class UnmarriedPerson.

- ***cardinality***: Cardinality is provided as a convenience when it is useful to state that a property on a class has both *minCardinality* 0 and *maxCardinality* 0 or both *minCardinality* 1 and *maxCardinality* 1. For example, the class Person has exactly one value for the property hasBirthMother. From this a reasoner can deduce that no two distinct individual instances of the class Mother may be values for the hasBirthMother property of the same person.

OWL Lite Class Intersection

- ***intersectionOf***: OWL Lite allows intersections of named classes and restrictions. For example, the class EmployedPerson can be described as the *intersectionOf* Person and EmployedThings (which could be defined as things that have a minimum cardinality of 1 on the hasEmployer property). From this a reasoner may deduce that any particular EmployedPerson has at least one employer.

Incremental Language Description of OWL DL and OWL Full

Both OWL DL and OWL Full use the same vocabulary although OWL DL is subject to some restrictions. Roughly, OWL DL requires type separation (a class can not also be an individual or property, a property can not also be an individual or class). This implies that restrictions cannot be applied to the language elements of OWL itself (something that is allowed in OWL Full).

Furthermore, OWL DL requires that properties are either ObjectProperties or DatatypeProperties: DatatypeProperties are relations between instances of classes and RDF literals and XML Schema datatypes, while ObjectProperties are relations between instances of two classes.

We describe the OWL DL and OWL Full vocabulary that extends the constructions of OWL Lite below.

- ***oneOf:*** (enumerated classes): Classes can be described by enumeration of the individuals that make up the class. The members of the class are exactly the set of enumerated individuals; no more, no less. For example, the class of daysOfTheWeek can be described by simply enumerating the individuals Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday. From this a reasoner can deduce the maximum cardinality (7) of any property that has daysOfTheWeek as its allValuesFrom restriction.

- ***hasValue:*** (property values): A property can be required to have a certain individual as a value (also sometimes referred to as property values). For example, instances of the class of dutchCitizens can be characterized as those people that have theNetherlands as a value of their nationality. (The nationality value, theNetherlands, is an instance of the class of Nationalities).

- ***disjointWith:*** Classes may be stated to be disjoint from each other. For example, Man and Woman can be stated to be disjoint classes. From this disjointWith statement, a reasoner can deduce an inconsistency when an individual is stated to be an instance of both and similarly a reasoner can deduce that if A is an instance of Man, then A is *not* an instance of Woman.

- ***unionOf, complementOf, intersectionOf*** (Boolean combinations): OWL DL and OWL Full allow arbitrary Boolean combinations of classes and restrictions: unionOf, complementOf, and intersectionOf. For example, using unionOf, we can state that a class contains things that are either USCitizens or DutchCitizens. Using complementOf, we could state that children are *not* SeniorCitizens. (i.e. the class Children is a subclass of the complement of SeniorCitizens). Citizenship of the European Union could be described as the union of the citizenship of all member states.

- ***minCardinality, maxCardinality, cardinality*** (full cardinality): While in OWL Lite, cardinalities are restricted to at least, at most or exactly 1 or 0, full OWL allows cardinality statements for arbitrary non-negative integers. For example the class of DINKs ("Dual Income, No Kids") would restrict the cardinality of the property hasIncome to a minimum cardinality of two (while the property hasChild would have to be restricted to cardinality 0).

- ***complex classes***: In many constructs, OWL Lite restricts the syntax to single class names (e.g. in subClassOf or equivalentClass statements). OWL Full extends this restriction to allow arbitrarily complex class descriptions, consisting of enumerated classes, property restrictions, and Boolean combinations. Also, OWL Full allows classes to be used as instances (and OWL DL and OWL Lite do not).