University of Rome "Tor Vergata"

Short notes about OWL 2

Manuel Fiorelli

fiorelli@info.uniroma2.it

Most examples are borrowed or adapted from the references cited in the slides

OWL 2

- OWL 2 superseded OWL 1 at the end of December 2012
- Relationship to OWL 1 [1]
  - Backward compatibility with OWL 1 is, to all intents and purposes, complete all OWL 1 Ontologies remain valid OWL 2 Ontologies, with identical inferences in all practical cases
  - New features [2]: some are syntactic sugar, while others increase the expressiveness of the language

[1] http://www.w3.org/TR/owl2-overview/#Relationship_to_OWL_1
[2] http://www.w3.org/TR/owl2-new-features/

SYNTACTIC SUGAR

Syntactic sugar: DisjointUnion (1/2)

A `:CarDoor` door is exclusively either a `:FrontDoor`, a `:RearDoor` or a `:TrunkDoor`

In **OWL 1**:

```
:CarDoor rdf:type owl:Class ;
    owl:equivalentClass [ a owl:Class ;
        owl:unionOf ( :FrontDoor :RearDoor :TrunkDoor )
    ] .
:FrontDoor a owl:Class ; owl:disjointWith :RearDoor, :TrunkDoor .
:RearDoor a owl:Class ; owl:disjointWith :FrontDoor, :TrunkDoor .
:TrunkDoor a owl:Class ; owl:disjointWith :FrontDoor, :RearDoor .
```


Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

Syntactic sugar: DisjointUnion (2/2)

A `:CarDoor` door is exclusively either a `:FrontDoor`, a `:RearDoor` or a `:TrunkDoor`

In **OWL 2**:

```
:CarDoor rdf:type owl:Class ;

    owl:disjointUnionOf (:FrontDoor :RearDoor :TrunkDoor)

.
```

Syntactic sugar: DisjointClasses

OWL 1 had `owl:AllDifferent` as a shorthand for asserting that more than two individuals were pairwise different.

OWL 2 introduces `owl:AllDisjointClasses` as a shorthand for asserting that more than two classes are pairwise disjoint.

```
[] rdf:type owl:AllDisjointClasses ;
     owl:members (:UpperLobeOfLung
                  :MiddleLobeOfLung
                  :LowerLobeOfLung ) .
```


Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

Syntactic sugar: NegativePropertyAssertion (1/2)

OWL 2 allows to assert explicitly that a given individual (w) is not connected to a given individual (z) by a certain object property (y). [*negative object property assertion*]

```
_:x rdf:type owl:NegativePropertyAssertion .
_:x owl:sourceIndividual w .
_:x owl:assertionProperty y .
_:x owl:targetIndividual z .
```

Syntactic sugar: NegativePropertyAssertion (2/2)

OWL 2 allows to assert explicitly that a given individual (w) does not have a given value (z) for a given data property (y). [*negative data property assertion*]

```
_:x rdf:type owl:NegativePropertyAssertion .
_:x owl:sourceIndividual w .
_:x owl:assertionProperty y .
_:x owl:targetValue z .
```


Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

EXPRESSIVENESS IMPROVEMENTS

Additions for Properties: Self-restriction

OWL 2 allows to describe the (anonymous) class of things that are related to themselves through a certain property.

Such classes can then be used in class axioms for named classes such as the following:

```
:AutoRegulatingProcess rdfs:subClassOf [
                           a owl:Restriction ;
                           owl:onProperty :regulate ;
                           owl:hasSelf "true"^^xsd:boolean
                       ] .
```

Additions for Properties: Property Qualified Cardinality Restrictions

In OWL 2, a qualified cardinality restriction allows to express the class of things having for a given property at least, at most or exactly a certain number of values belonging to the given class (or data range).

```
[a owl:Restriction ;
 owl:onProperty some object property expression ;
 owl:{min|max|}QualifiedCardinality "some n"^^xsd:nonNegativeInteger  ;
 owl:onClass a class ]

[a owl:Restriction ;
 owl:onProperty some data property expression ;
 owl:{min|max|}QualifiedCardinality "some n"^^xsd:nonNegativeInteger  ;
 owl:onDataRange a data range] .
```

The 'q' is lowercase if neither "min" nor "max" are used

Additions for Properties: Disjoint Properties

In OWL 1 it is possible to assert the disjointness of classes.

OWL 2 also allows to assert the disjointness of properties.

Start time must be different from end time

```
:startTime owl:propertyDisjointWith :endTime .
```

If there are more than two properties, it is possible to use the following syntax:

_:x rdf:type owl:AllDisjointProperties .

_:x owl:members (*(data or object) property* $expr_1$---$expr_N$).

Additions for Properties: Reflexive, Irreflexive, and Asymmetric Object Properties

Università di Roma Tor Vergata

Every x is a part of itself

```
:part_of rdf:type owl:ReflexiveProperty  .
```

No x can be a proper part of itself

```
:proper_part_of rdf:type owl:IrreflexiveProperty .
```

If x is a proper part of y, then y cannot be a proper part of x

```
:proper_part_of rdf:type owl:AsymmetricProperty   .
```

Additions for Properties: (Object) Property Chain Inclusion (1/3)

In OWL 2 it is possible to define a property as the composition of other properties.

If x has parent y and y has parent z then x has grandparent z

```
:hasGrandparent owl:propertyChainAxiom (
:hasParent :hasParent ) .
```

If x is located in y and y is part of z then x is located in z

```
:locatedIn owl:propertyChainAxiom (
:locatedIn :partOf  ) .
```


Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

Additions for Properties: (Object) Property Chain Inclusion (2/3)

Allowed forms of property chain inclusion axioms:

$S^{-} \sqsubseteq R$

Allows us to express symmetry

$R^{-} \sqsubseteq R$

$R \circ R \sqsubseteq R$ (transitivity)

$S_1 \circ \ldots \circ S_n \sqsubseteq R$

$R \circ S_1 \circ \ldots \circ S_n \sqsubseteq R$

$S_1 \circ \ldots \circ S_n \circ R \sqsubseteq R$

For all i, it should be that $S_i < R$, where '<' is a strict partial order between properties (irreflexive, transitive and asymmetric).

The rationale is to avoid recursion with some exceptions in order to avoid undecidability.

Additions for Properties: (Object) Property Chain Inclusion (3/3)

A *simple role* cannot be inferred (directly or indirectly) from a property chain.

R is <u>*non simple*</u> if:

$S_1 \circ \ldots \circ S_n \sqsubseteq R$ with $n > 1$

$S \sqsubseteq R$ where S is non simple

the inverse of R is non simple

In the following cases *only simple roles are allowed:*

- cardinality restrictions (both qualified and unqualified) and self-restrictions
- functional, inverse functional, irreflexive, asymmetric and disjoint object properties

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

Additions for Properties: Keys

In OWL 2 it is possible to associate a class with a set of (either object or data) properties that uniquely identifies its instances.

Students are uniquely identified by their student identification number

```
:Student a owl:Class ;
      owl:hasKey ( :studentIdentificationNumber ).
```


Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

Extended datatype support

- Additional supported datatypes
- Possibility to define new datatypes by constraining an existing datatype through various facets (e.g. integers lower than 1024)
- A new construct to define datatypes
- Combine data ranges via intersection, union and negation

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

Simple metamodeling

Punning: relaxing the strict separation of names that was mandated by OWL 1 DL

Extended annotations

- Annotations on axioms, ontologies as well as ontology elements
- Domains and ranges of annotation properties
- Hierarchies of annotation properties

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

Other innovations

- owl:topObjectProperty, owl:topDataProperty
- owl:bottomObjectProperty, owl:bottomDataProperty
- owl:Nothing
- Versioning and imports. An ontology has:
  - an ontology IRI (common to different versions of the ontology)
  - version IRI (identifying a particular version of the ontology)
- In OWL 2 property expressions can be in class expressions, thus it is possible to use the inverse of a property without giving it a name:

```
:KnownByMark owl:equivalentClass [ a owl:Restriction ;
                                   owl:onProperty [
                                     owl:inverseOf :knows
                                  ] ;
                                   owl:hasValue :mark ] .
```

Useful References

- OWL 2 Web Ontology Language
  - New Features and Rationale (https://www.w3.org/TR/owl2-new-features/)
  - Document Overview (https://www.w3.org/TR/owl2-overview/)
  - Structural Specification and Functional-Style Syntax (https://www.w3.org/TR/owl2-syntax/) *[just below the table of contents, it is possible to clic on a button to show RDF in the examples]*
  - Primer (https://www.w3.org/TR/owl2-primer/) *[at the end of section 1.2, it is possible to clic on a button to show RDF in the examples]*
- Knowledge Representation for the Semantic Web Part I: OWL 2 (http://semantic-web-book.org/w/images/b/b0/KI09-OWL-Rules-1.pdf)