



# Web Semântica

OWL – WEB ONTOLOGY LANGUAGE



- As linguagens para a criação de ontologias permitem a escrita de conceptualizações explícitas e formais de modelos de domínio.
- Os principais requisitos, são:
  - uma sintaxe bem definida
  - uma semântica formal
  - poder expressivo suficiente
  - conveniência de expressão
  - suporte de raciocínio eficiente



- Quanto mais rica for a linguagem de ontologia, mais ineficiente é o suporte ao raciocínio
  - Às vezes, atravessa a fronteira da **não-computabilidade**
- É, por isso, necessário um compromisso:
  - Uma linguagem suportada por raciocínios razoavelmente eficientes;
  - Uma linguagem que pode expressar grandes classes de ontologias e conhecimentos.



- O suporte ao raciocínio é importante para:
  - A verificação da consistência da ontologia e do conhecimento;
  - A verificação de relacionamentos não intencionais entre classes;
  - A classificação automática de instâncias com classes.
- Estas verificações são valiosas para:
  - O projeto de ontologias grandes, onde vários autores estão envolvidos;
  - A integração e a partilha de ontologias de várias fontes.



- **Classes e Subclasses:**

se  $x$  é instância da classe  $C$   
e  $C$  é uma subclasse de  $D$   
então,  $x$  é instância de  $D$

- **Equivalência de Classes:**

se a classe  $A$  é equivalente à classe  $B$   
e  $B$  é equivalente à classe  $C$   
então,  $A$  também é equivalente a  $C$



- **Consistência:**

se  $x$  é instância das classes  $A$  e  $B$ ;  
mas  $A$  e  $B$  são disjuntas;  
então, existe um erro na ontologia.

- **Classificação:**

certos pares de propriedade-valor são condição  
suficiente para pertencer a uma classe  $A$ ;  
se um indivíduo  $x$  satisfaz essas condições,  
então conclui-se que  $x$  é uma instância de  $A$ .



- O RDFS possui limitações que se traduzem na impossibilidade de fazer certas declarações, como:
- Âmbito de uma Propriedade:
  - Declarar restrições a uma gama de valores, que se aplicam apenas a algumas classes
  - `rdfs:range` define a gama de valores de uma propriedade para todas as classes a que esta se aplica
  - Por exemplo: para a propriedade “**come**”, não é possível definir que certos animais só comem plantas, enquanto outros também comem carne



- Classes Disjuntas:
  - Declarar duas classes como disjuntas, por exemplo, as classes “Masculino” e “Feminino”
- Combinações booleanas de classes:
  - A criação de novas classes combinando outras classes usando a união, a interseção e o complemento
  - Por exemplo: a classe “Pessoa” é a união desunida das classes “Masculino” e “Feminino”





- Restrições de cardinalidade
  - Por exemplo:
    - uma pessoa tem exatamente dois pais
    - um curso é ministrado por pelo menos um professor
- Características especiais de propriedades:
  - Propriedade transitiva (como "maior que")
  - Propriedade exclusiva (como "é mãe de")
  - Propriedade inversa (como "come" e "é comido por")



W3C Recommendation 11 December 2012

<http://www.w3.org/TR/owl-features/>

# **OWL 2 WEB ONTOLOGY LANGUAGE**



- OWL - Web Ontology Language
  - Presentemente é a linguagem mais popular para a criação de ontologias.
  - O propósito da OWL é exactamente o mesmo que o RDFS:
    - Definir ontologias que incluem classes, propriedades e suas relações num dado domínio de conhecimento.
  - Comparando com o RDFS, a OWL tem a capacidade de expressar relações muito mais complexas e ricas.
  - Pode-se dizer que:
    - OWL = RDFS + estruturas de maior expressão



- Com vista à adaptação da utilização da OWL aos diferentes cenários, que podem ser mais ou menos complexos, o W3C divide a OWL em 3 níveis de linguagem ou sub-linguagens:
  - OWL Full
    - Nível atribuído a toda a linguagem e no qual são utilizadas todas as primitivas definidas na mesma.
    - Extremamente flexível, contudo torna difícil a inferência por máquinas.
  - OWL DL
    - Utiliza um subconjunto da OWL Full e já permite a inferência eficiente por parte das máquinas.
  - OWL Lite
    - Subconjunto da OWL DL
    - Restringe a expressividade da OWL, mas tem a vantagem de ser mais facilmente compreensível pelos utilizadores e manipulável pelas ferramentas de construção de ontologias.



- Os documentos OWL são chamados ontologias OWL e também são documentos RDF, por isso o seu elemento raiz, em XML, é o `<rdf:RDF>`
- Exemplo

```
<rdf:RDF
  xmlns:owl = "http://www.w3.org/2002/07/owl#"
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd = "http://www.w3.org/2001/XMLSchema#">
  ...
```



- Os documentos OWL podem começar por conter um cabeçalho contendo comentários, informação para controlo de versões, inclusão de outras ontologias, etc.
- Exemplo

```
<rdf:RDF>
...
<owl:Ontology rdf:about="">
  <rdfs:comment>Exemplo de uma ontologia OWL</rdfs:comment>
  <owl:priorVersion rdf:resource="http://www.mydomain.org/uni-ns-old"/>
  <owl:imports rdf:resource="http://www.mydomain.org/persons"/>
  <rdfs:label>University Ontology</rdfs:label>
</owl:Ontology> ...
```



- Elementos que podem ser utilizados no cabeçalho

**Cabeçalho:**  
Ontology  
imports

**Versões:**  
versionInfo  
priorVersion  
backwardCompatibleWith  
incompatibleWith  
DeprecatedClass  
DeprecatedProperty

**Anotações:**  
rdfs:label  
rdfs:comment  
rdfs:seeAlso  
rdfs:isDefinedBy  
AnnotationProperty  
OntologyProperty



- As classes são definidas usando **owl:Class**
  - owl:Class é uma subclasse de rdfs:Class

- Definindo Subclasses

```
<owl:Class rdf:ID="Carro">  
  <rdfs:subClassOf rdf:resource="#MeioTransporte"/>  
</owl:Class>
```

- Definindo classes Equivalentes

```
<owl:Class rdf:ID="carro">  
  <owl:equivalentClass rdf:resource="#automovel"/>  
</owl:Class>
```





- Definindo classes Disjuntas

```
<owl:Class rdf:about="#ProfessorAssociado">  
  <owl:disjointWith rdf:resource="#ProfessorAuxiliar"/>  
  <owl:disjointWith rdf:resource="#ProfessorCatedratico"/>  
</owl:Class>
```

- Classes Genéricas

- **owl:Thing** – classe base de todas as classes OWL
- **owl:Nothing** – subclasse de todas as classes OWL
- **owl:Individual** – instâncias de classes



- Em OWL existem dois tipos de propriedades:
  - de objecto – que relacionam objetos com outros objetos
  - de tipo de dados – que relacionam objetos com tipos de dados
- Exemplo de propriedade de tipo de dados
  - A OWL faz uso dos tipos de dados definidos no XML Schema

```
<owl:DatatypeProperty rdf:ID="idade">  
  <rdfs:range  
    rdf:resource="http://www.w3.org/2001/XMLSchema  
    #nonNegativeInteger"/>  
</owl:DatatypeProperty>
```



- Exemplo de propriedade de objecto

```
<owl:ObjectProperty rdf:ID="conduz">  
  <rdfs:domain rdf:resource="#Pessoa"/>  
  <rdfs:range rdf:resource="#MeioTransporte"/>  
  <rdfs:subPropertyOf rdf:resource="#manipula"/>  
</owl:ObjectProperty>
```



- Exemplo de propriedade inversa

```
<owl:ObjectProperty rdf:ID="conduz">  
  <rdfs:domain rdf:resource="#pessoa"/>  
  <rdfs:range rdf:resource="#meioTransporte"/>  
  <owl:inverseOf rdf:resource="#eConduzidoPor">  
</owl:ObjectProperty>
```



- Exemplo de propriedade equivalente

```
<owl:ObjectProperty rdf:ID="leciona">  
  <owl:equivalentProperty rdf:resource="#ensina">  
</owl:ObjectProperty>
```



- São usadas para inferir os membros de uma classe, impondo uma restrição aos valores das suas propriedades
  - o elemento **owl:Restriction** é usado para criar a restrição
  - o elemento **owl:onProperty** é usado para designar a propriedade sobre a qual é imposta a restrição
  - São usados os seguintes elementos para indicar o tipo de restrição: **owl:allValuesFrom**, **owl:hasValue**, **owl:someValuesFrom**



- Restrição de valores do tipo **owl:allValuesFrom**
- Exemplo:
  - É considerado individuo da Classe “CarroCorrida”, se todos os valores da propriedade “eConduzidoPor” pertencem à classe “PilotoCorrida”.

```
<owl:Class rdf:about="#CarroCorrida">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eConduzidoPor"/>
      <owl:allValuesFrom rdf:resource="#PilotoCorrida"/>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```



- Restrição de valores do tipo **owl:hasValue**
- Exemplo:
  - É considerado individuo da Classe “CursoMatematica”, se a propriedade “lecionado” possui o valor específico “#456123”.

```
<owl:Class rdf:about="#CursoMatematica">  
  <owl:equivalentClass>  
    <owl:Restriction>  
      <owl:onProperty rdf:resource="#lecionado"/>  
      <owl:hasValue rdf:resource="#456123"/>  
    </owl:Restriction>  
  </owl:equivalentClass>  
</owl:Class>
```





- Restrição de valores do tipo **owl:someValuesFrom**
- Exemplo:
  - Para cada membro da classe “Piloto”, existe um valor da propriedade “conduz” que pertence à classe “Carro”.

```
<owl:Class rdf:about="#Piloto">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#conduz"/>
      <owl:someValuesFrom rdf:resource="#Carro"/>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```



- Restrição do número mínimo (cardinalidade)
- Exemplo:
  - É considerado individuo da Classe “Piloto”, se no mínimo um valor da propriedade “conduz” pertence à classe “Carro”.

```
<owl:Class rdf:about="#Piloto">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#conduz"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger"> 1
    </owl:minCardinality>
      <owl:onClass rdf:resource="#Carro"/>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```



- Restrição do número máximo (cardinalidade)
- Exemplo:
  - É considerado individuo da Classe “Piloto”, se no máximo 2 valores da propriedade “conduz” pertencem à classe “Carro”.

```
<owl:Class rdf:about="#Piloto">  
  <owl:equivalentClass>  
    <owl:Restriction>  
      <owl:onProperty rdf:resource="#conduz"/>  
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger"> 2  
    </owl:maxCardinality>  
      <owl:onClass rdf:resource="#Carro"/>  
    </owl:Restriction>  
  </owl:equivalentClass>  
</owl:Class>
```



- Restrição ao número exato (cardinalidade)
- Exemplo:
  - É considerado individuo da Classe “Piloto”, se existe exatamente um valor da propriedade “conduz” que pertence à classe “Carro”.

```
<owl:Class rdf:about="#Piloto">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#conduz"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1
    </owl:cardinality>
      <owl:onClass rdf:resource="#Carro"/>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```



- É possível fazer combinação de classes através de operações booleanas – união, intersecção e complemento
- Exemplo de um Complemento

```
<owl:Class rdf:about="#Viatura">  
  <rdfs:subClassOf>  
    <owl:Restriction>  
      <owl:complementOf rdf:resource="#Condutor"/>  
    </owl:Restriction>  
  </rdfs:subClassOf>  
</owl:Class>
```



- Exemplo de uma União

```
<owl:Class rdf:about="#Viatura">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#CarroLigeiro"/>
    <owl:Class rdf:about="#CarroPesado"/>
  </owl:unionOf>
</owl:Class>
```



- Exemplo de uma Intersecção

```
<owl:Class rdf:about="#Pai">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Homem"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#tem"/>
      <owl:someValue rdf:resource="#Filho"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```



- owl:TransitiveProperty (propriedade transitiva)
  - Ex: “é maior que”, “é pai de”, etc.
- owl:SymmetricProperty (propriedade simétrica)
  - Ex: “tem o mesmo grau que”, “é irmão de”, etc.
- owl:FunctionalProperty
  - Tem no máximo um valor para cada sujeito
  - Ex: “idade”, “altura”, “supervisor”, etc.
- owl:InverseFunctionalProperty
  - Para dois sujeitos diferentes, não pode ter o mesmo valor
  - Ex: “esposa”, “marido”, “pai”, etc.





- Exemplo de propriedade transitiva e simétrica:
  - A propriedade “temMesmaNotaQue” é transitiva porque estabelece uma relação de grandeza e é simétrica porque pode ser feita nos dois sentidos.

```
<owl:ObjectProperty rdf:about="#temMesmaNotaQue">  
  <rdf:Type rdf:resource="&owl:TransitiveProperty"/>  
  <rdf:Type rdf:resource="&owl:SimmetricProperty"/>  
  <rdfs:domain rdf:resource="#Estudante">  
  <rdfs:range rdf:resource="#Estudante">  
</owl:ObjectProperty>
```



- owl:sameAs (o mesmo que)
  - Permite declarar que duas entidades são a mesma

```
<Aluno rdf:about="98456">
```

```
<owl:sameAs rdf:resource="jjj@ua.pt"/>
```

```
</Aluno>
```



- owl:differentFrom (diferente de)
  - Permite declarar que uma entidade é diferente de outra

```
<Professor rdf:about="949318">  
  <owl:differentFrom rdf:resource="949352"/>  
</Professor>
```



- **owl:allDifferent** (todos diferentes)
  - Permite declarar que um conjunto de entidades são todas diferentes, entre si
  - é usado em conjunto com o elemento **owl:distinctMembers**

```
<owl:allDifferent>
```

```
  <owl:distinctMembers rdf:parseType="Collection">
```

```
    <Aluno rdf:about="949318"/>
```

```
    <Aluno rdf:about="949352"/>
```

```
    <Aluno rdf:about="949111"/>
```

```
  </owl:distinctMembers>
```

```
</owl:allDifferent>
```



## Capítulo 4

Grigoris Antoniou, Frank van Harmelen ,  
“A Semantic Web Primer”, MIT Press, 2<sup>a</sup> ed, 2012.