Engenharia de Ontologias (Ontology Engineering)

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Prof. Fabiano Azevedo Dorça

OWL2
Web Ontology Language

OWL2
Web Ontology Language

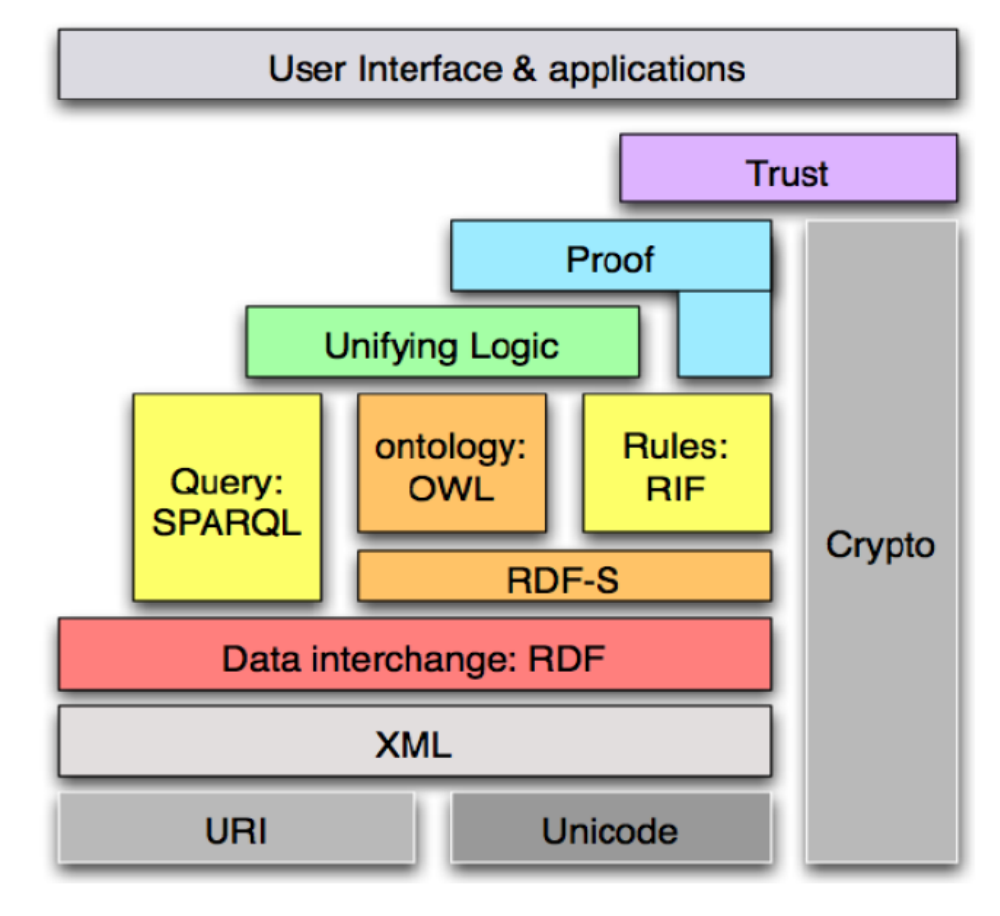
- Uma ontologia é um conjunto de **declarações descritivas** precisas sobre alguma **parte do mundo**
    - geralmente referida como o **domínio** de interesse ou o assunto da ontologia.
- **Descrições precisas** satisfazem vários propósitos:
    - evitam **mal-entendidos** na comunicação humana
    - possibilitam que um sistema possa se comunicar adequadamente com seus usuários e outros sistemas.

OWL2
Web Ontology Language

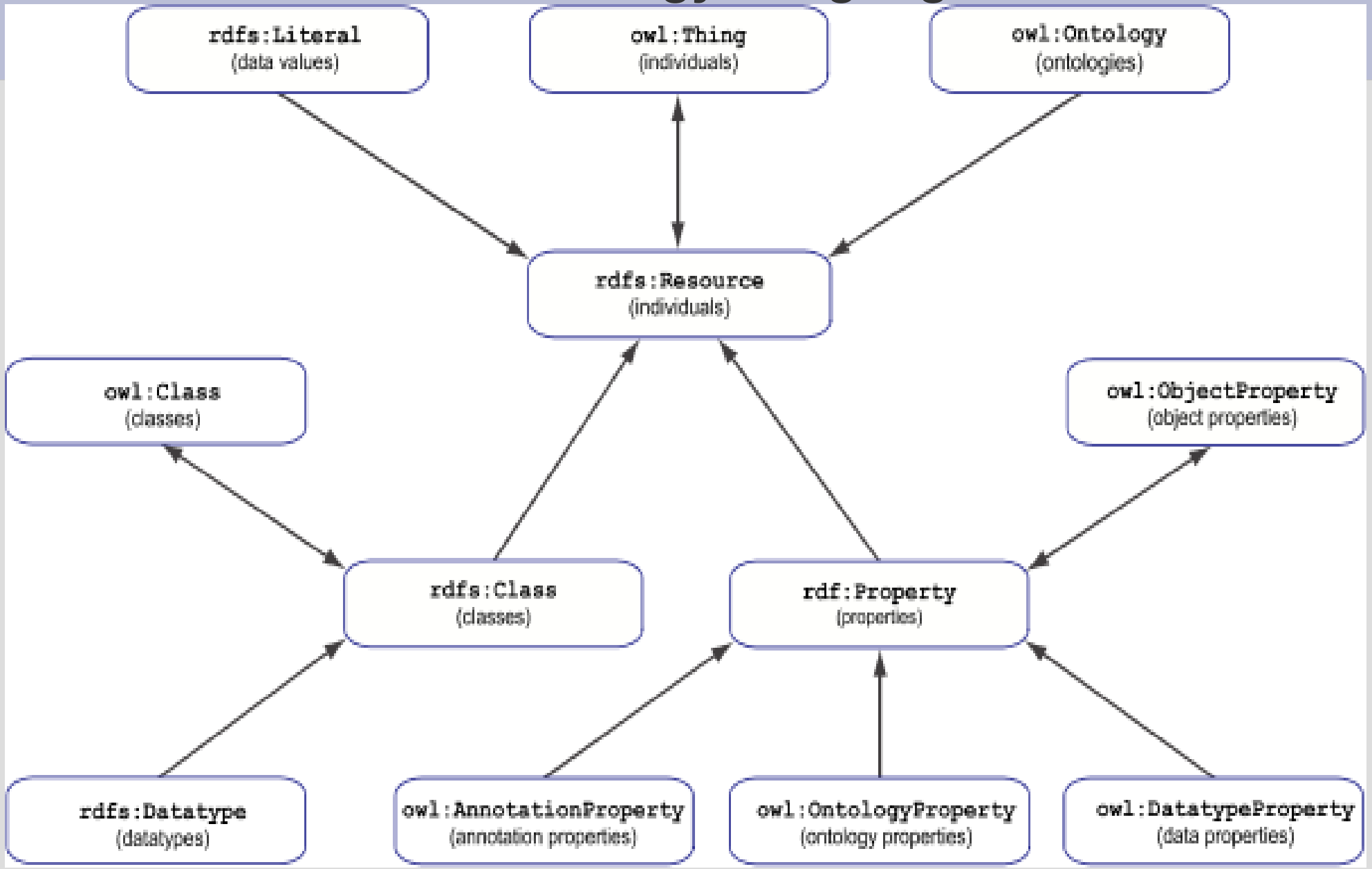
- O OWL2 Web Ontology Language
  - é uma linguagem de ontologia para a Web Semântica com significado formalmente definido.
  - fornece classes, propriedades, indivíduos e valores de dados e são armazenadas como documentos da Web Semântica.
  - pode ser usadas juntamente com informações escritas em RDF.
  - **é compartilhada principalmente como documentos RDF.**
  - utiliza tipos de dados definidos no XML Schema Definition Language (XSD).

OWL2
Web Ontology Language

- OWL não existe isoladamente, mas é parte da pilha da Web Semântica.



OWL2
Web Ontology Language



OWL2
Web Ontology Language

- Semântica de mundo aberto:
  - Se algum fato não está presente em um banco de dados, geralmente é considerado falso (mundo fechado),
  - no caso de uma ontologia OWL 2 ele pode estar simplesmente faltando (mas possivelmente verdadeiro) – semância de mundo aberto.

- **RDF/XML Syntax**

Exemplo:

Tutorial: https://www.w3.org/TR/owl-guide/
Ontologias: www.w3.org/TR/owl-guide/wine.rdf importa: http://www.w3.org/TR/2004/REC-owl-guide-20040210/food.rdf

```
<!DOCTYPE rdf:RDF [
    <!ENTITY vin  "http://www.w3.org/TR/2003/PR-owl-guide-
20031209/wine#" >
    <!ENTITY food "http://www.w3.org/TR/2003/PR-owl-guide-
20031209/food#" >
    <!ENTITY owl  "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd  "http://www.w3.org/2001/XMLSchema#" >
  ]>
<rdf:RDF
  xmlns     = "http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#"
  xmlns:vin = "http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#"
  xml:base  = "http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#"
  xmlns:food= "http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#"
  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#">
```

- **Metadados OWL para o documento e imports**

```
<owl:Ontology rdf:about="">
  <rdfs:comment>An example OWL ontology</rdfs:comment>
  <owl:priorVersion>
    <owl:Ontology rdf:about="http://www.w3.org/TR/2003/CR-owl-guide-
20030818/wine"/>
  </owl:priorVersion>
  <owl:imports rdf:resource="http://www.w3.org/TR/2003/PR-owl-guide-
20031209/food"/>
  <rdfs:comment>Derived from the DAML Wine ontology at
    http://ontolingua.stanford.edu/doc/chimaera/ontologies/wines.daml
    Substantially changed, in particular the Region based relations.
  </rdfs:comment>
  <rdfs:label>Wine Ontology</rdfs:label>
</owl:Ontology>
```

- O atributo rdf:about fornece um nome ou referência para a ontologia. Onde o valor do atributo for "", o caso padrão, o nome da ontologia é o URI base do elemento owl:Ontology.

- Considerando ontologias baseadas na Web, essas duas classes podem ser definidas ontologias separadas (food) que é importada para a ontologia do vinho...

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
  <rdfs:label xml:lang="en">wine</rdfs:label>
  <rdfs:label xml:lang="fr">vin</rdfs:label>
  ...
</owl:Class>


<owl:Class rdf:ID="Pasta">
  <rdfs:subClassOf rdf:resource="#EdibleThing" />
  ...
</owl:Class>
```

- Indivíduos
- <Region rdf:ID="CentralCoastRegion" />
  - Observe que o seguinte é idêntico em significado ao exemplo acima.

```
<owl:Thing rdf:ID="CentralCoastRegion" />

<owl:Thing rdf:about="#CentralCoastRegion">
   <rdf:type rdf:resource="#Region"/>
</owl:Thing>
```

- **Propriedades**
- ObjectProperty, DatatypeProperty, rdfs:subPropertyOf, rdfs:domain, rdfs:range

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>


<owl:ObjectProperty rdf:ID="course">
  <rdfs:domain rdf:resource="#Meal" />
  <rdfs:range rdf:resource="#MealCourse" />
</owl:ObjectProperty>

<owl:Thing rdf:ID="LindemansBin65Chardonnay">
  <madeFromGrape rdf:resource="#ChardonnayGrape" />
</owl:Thing>
```

- Podemos inferir que o LindemansBin65Chardonnay é um vinho porque o domínio do madeFromGrape é o vinho.

- **Propriedades, como classes, podem ser organizadas em uma hierarquia. A relação rdfs:subPropertyOf nesse caso significa que qualquer coisa com uma propriedade hasColor com valor X também possui uma propriedade hasWineDescriptor com o valor X.**

```
<owl:Class rdf:ID="WineDescriptor" />


<owl:Class rdf:ID="WineColor">
  <rdfs:subClassOf rdf:resource="#WineDescriptor" />

  ...</owl:Class>


<owl:ObjectProperty rdf:ID="hasWineDescriptor">
  <rdfs:domain rdf:resource="#Wine" />
  <rdfs:range  rdf:resource="#WineDescriptor" />
</owl:ObjectProperty>


<owl:ObjectProperty rdf:ID="hasColor">
  <rdfs:subPropertyOf rdf:resource="#hasWineDescriptor" />
  <rdfs:range rdf:resource="#WineColor" />
  ...
</owl:ObjectProperty>
```

- Em seguida, introduzimos a propriedade locateIn, que relaciona as coisas com as regiões nas quais elas estão localizadas.

```
<owl:ObjectProperty rdf:ID="locatedIn">
  ...
  <rdfs:domain
rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
  <rdfs:range rdf:resource="#Region" />
</owl:ObjectProperty>
```

Observe como o domínio e o intervalo de locatedIn estão definidos. O domínio permite que qualquer coisa seja localizada em uma região, incluindo as próprias regiões.

- Agora é possível expandir a definição de vinho para incluir a noção de que um vinho é feito a partir de pelo menos um WineGrape. Assim como nas definições de propriedade, as definições de classe têm várias subpartes que são implicitamente associadas.

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#madeFromGrape"/>
      <owl:minCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

- A restrição de subclasse destacada acima

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#madeFromGrape"/>
  <owl:minCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
</owl:Restriction>
```

- define uma classe sem nome que representa o conjunto de coisas com pelo menos uma propriedade madeFromGrape.
- Classes anônimas. Incluindo esta restrição no corpo de definição da classe Wine, afirma que as coisas que são vinhos também são membros desta classe anônima.
- Ou seja, cada vinho deve participar em pelo menos uma relação madeFromGrape.

- A propriedade vintageOf amarra um vintage a um vinho, através do domain e range...

```
<owl:ObjectProperty rdf:ID="vintageOf">
  <rdfs:domain rdf:resource="#Vintage" />
  <rdfs:range  rdf:resource="#Wine" />
</owl:ObjectProperty>
```

- Propriedades relacionam indivíduos a indivíduos (propriedades de objetos) ou indivíduos a tipos de dados (propriedades de tipos de dados).

- As propriedades do tipo de dados podem variar sobre literais RDF ou tipos simples definidos de acordo com os tipos de dados do XML Schema.

```
<owl:Class rdf:ID="VintageYear" />

<owl:DatatypeProperty rdf:ID="yearValue">
  <rdfs:domain rdf:resource="#VintageYear" />
  <rdfs:range  rdf:resource="&xsd;positiveInteger"/>
</owl:DatatypeProperty>
```

- Propriedades de **indivíduos (instâncias de classes)**. Exemplo:

Classe ***Indivíduo***

```
<Region rdf:ID="SantaCruzMountainsRegion">
  <locatedIn rdf:resource="#CaliforniaRegion" />
</Region>


<Winery rdf:ID="SantaCruzMountainVineyard" />


<CabernetSauvignon
  rdf:ID="SantaCruzMountainVineyardCabernetSauvignon" >
  <locatedIn  rdf:resource="#SantaCruzMountainsRegion"/>
  <hasMaker   rdf:resource="#SantaCruzMountainVineyard" />
</CabernetSauvignon>
```

- As propriedades do tipo de dados podem ser adicionadas aos indivíduos de maneira semelhante.

- Exemplo: uma instância de VintageYear e a vinculamos a um valor específico &xsd:positiveInteger.

```
<VintageYear rdf:ID="Year1998">
  <yearValue
rdf:datatype="&xsd;positiveInteger">1998</yearValue>
</VintageYear>
```

- Características de propriedades
- É possível especificar as características da propriedade, o que fornece um mecanismo poderoso para o aprimoramento do raciocínio sobre uma propriedade.
- Propriedade transitiva
- Se uma propriedade, P, for especificada como transitiva, então, para qualquer x, y e z:
- P(x,y) and P(y,z) implies P(x,z)

- A propriedade locatedIn é transitiva.

```
<owl:ObjectProperty rdf:ID="locatedIn">
  <rdf:type rdf:resource="&owl;TransitiveProperty" />
  <rdfs:domain rdf:resource="&owl;Thing" />
  <rdfs:range rdf:resource="#Region" />
</owl:ObjectProperty>

<Region rdf:ID="SantaCruzMountainsRegion">
  <locatedIn rdf:resource="#CaliforniaRegion" />
</Region>

<Region rdf:ID="CaliforniaRegion">
  <locatedIn rdf:resource="#USRegion" />
</Region>
```

- **Propriedade simétrica**

- **Se uma propriedade, P, for marcada como simétrica, então para qualquer x e y:**

- **P(x,y) iff P(y,x)**

```
<owl:ObjectProperty rdf:ID="adjacentRegion">
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
  <rdfs:domain rdf:resource="#Region" />
  <rdfs:range rdf:resource="#Region" />
</owl:ObjectProperty>
```

```
<Region rdf:ID="MendocinoRegion">
  <locatedIn rdf:resource="#CaliforniaRegion" />
  <adjacentRegion rdf:resource="#SonomaRegion" />
</Region>
```

- Propriedade funcional

- Se uma propriedade, P, for marcada como funcional, então para todos os x, y e z:

  P(x,y) and P(x,z) implies y = z

  - Na ontologia de vinhos, hasVintageYear é funcional. Um vinho tem um único vintage year.

  - Ou seja, um determinado indivíduo Vintage só pode ser associado a um único ano usando a propriedade hasVintageYear.

```
<owl:Class rdf:ID="VintageYear" />

<owl:ObjectProperty rdf:ID="hasVintageYear">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#Vintage" />
  <rdfs:range  rdf:resource="#VintageYear" />
</owl:ObjectProperty>
```

- Se uma propriedade, P1, for marcada como owl:inverseOf P2, então para todos os x e y:
  - P1(x,y) iff P2(y,x) ----(A implica B) e (B implica A).
- Observe que a sintaxe para owl:inverseOf usa um nome de propriedade como argumento.

```
<owl:ObjectProperty rdf:ID="hasMaker">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="producesWine">
  <owl:inverseOf rdf:resource="#hasMaker" />
</owl:ObjectProperty>
```

- **Quantificadores universal e existencial**
  - allValuesFrom, someValuesFrom
  - A restrição owl: allValuesFrom requer que, para cada instância da classe que possui instâncias da propriedade especificada, os valores da propriedade sejam todos membros da classe indicada pela cláusula owl:allValuesFrom.

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid" />
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker" />
      <owl:allValuesFrom rdf:resource="#Winery" />
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

  - The maker of a Wine must be a Winery

- **owl:someValuesFrom** – pelo menos uma das propriedades hasMaker de um Wine deve apontar para um indivíduo que seja uma Winery.

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker" />
      <owl:someValuesFrom rdf:resource="#Winery" />
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>                                    ¬
```

- **hasValue**
- hasValue nos permite especificar classes com base na existência de determinados valores de propriedade.
- Portanto, um indivíduo será um membro dessa classe sempre que pelo menos um dos seus valores de propriedade for igual ao recurso hasValue.

```
<owl:Class rdf:ID="Burgundy">
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasSugar" />
      <owl:hasValue rdf:resource="#Dry" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

- **Equivalência entre Classes e Propriedades**
  - equivalentClass, equivalentProperty

```
<owl:Class rdf:ID="Wine">
  <owl:equivalentClass rdf:resource="&vin;Wine"/>
</owl:Class>
```

- A propriedade owl:equivalentClass é usada para indicar que duas classes têm precisamente as mesmas instâncias.

```
<owl:Class rdf:ID="TexasThings">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#locatedIn" />
      <owl:someValuesFrom
rdf:resource="#TexasRegion" />
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>                                    ¬
```

- Identidade entre Indivíduos

  - sameAs

  - Esse mecanismo é semelhante ao das classes, mas declara que dois indivíduos são idênticos. Um exemplo seria:

```
<Wine rdf:ID="MikesFavoriteWine">
  <owl:sameAs
   rdf:resource="#StGenevieveTexasWhite"/>
</Wine>                                                    ¬
```

- Isso traz um ponto importante.
- OWL não possui uma suposição de nome exclusivo.
- Só porque dois nomes são diferentes, não significa que eles se referem a indivíduos diferentes.

- **Different Individuals**
  differentFrom, AllDifferent

```
<WineSugar rdf:ID="Dry" />

<WineSugar rdf:ID="Sweet">
  <owl:differentFrom rdf:resource="#Dry"/>
</WineSugar>

<WineSugar rdf:ID="OffDry">
  <owl:differentFrom rdf:resource="#Dry"/>
  <owl:differentFrom rdf:resource="#Sweet"/>
</WineSugar>
```

- Essa é uma maneira de afirmar que esses três valores são mutuamente distintos.
- Haverá casos em que é importante garantir identidades distintas.
- Sem essas afirmações, poderíamos descrever um vinho que fosse seco e doce.

- Existe um mecanismo mais conveniente para definir um conjunto de indivíduos mutuamente distintos.

- O seguinte afirma que Tinto, Branco e Rose são diferentes em pares.

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <vin:WineColor rdf:about="#Red" />
    <vin:WineColor rdf:about="#White" />
    <vin:WineColor rdf:about="#Rose" />
  </owl:distinctMembers>
</owl:AllDifferent>
```

- Set Operators
  - intersectionOf, unionOf, complementOf

  - Intersection

```
<owl:Class rdf:ID="WhiteBurgundy">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Burgundy" />
    <owl:Class rdf:about="#WhiteWine" />
  </owl:intersectionOf>
</owl:Class>
```

- **Union**

```
<owl:Class rdf:ID="Fruit">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#SweetFruit" />
    <owl:Class rdf:about="#NonSweetFruit" />
  </owl:unionOf>
</owl:Class>
```

- A classe Fruit inclui tanto a extensão do SweetFruit quanto a extensão do NonSweetFruit.

- **Complement**

- O complemento indica todos os indivíduos do domínio do discurso que não pertencem a uma determinada classe.

  - <owl:Class rdf:ID="ConsumableThing" />

```
<owl:Class rdf:ID="NonConsumableThing">
  <owl:complementOf
rdf:resource="#ConsumableThing" />
</owl:Class>
```

Exemplo:

```
<owl:Class rdf:ID="NonFrenchWine">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Wine"/>
    <owl:Class>
      <owl:complementOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#locatedIn" />
          <owl:hasValue rdf:resource="#FrenchRegion" />
        </owl:Restriction>
      </owl:complementOf>
    </owl:Class>
  </owl:intersectionOf>
</owl:Class>
```

- Define a classe NonFrenchWine como a interseção de Wine com o conjunto de todas as coisas não localizadas na França.

- **Disjoint Classes**

- A disjunção de um conjunto de classes pode ser expressa usando o construtor owl:disjointWith.

- Ele garante que um indivíduo que seja membro de uma classe não possa ser simultaneamente uma instância de outras classes especificada.

```
<owl:Class rdf:ID="Pasta">
  <rdfs:subClassOf rdf:resource="#EdibleThing"/>
  <owl:disjointWith rdf:resource="#Meat"/>
  <owl:disjointWith rdf:resource="#Fowl"/>
  <owl:disjointWith rdf:resource="#Seafood"/>
  <owl:disjointWith rdf:resource="#Dessert"/>
  <owl:disjointWith rdf:resource="#Fruit"/>
</owl:Class>
```

OWL2
Web Ontology Language

- Fim!