UVa

Introducción a la Web Semántica

M. Mercedes Martínez
Dep. Informática (U. Valladolid, España)

Ontologías y OWL

Qué es una ontología

- "*Una especificación explícita de una conceptualización*" [Gruber, 1993]
  - "una visión abstracta y simplificada del mundo que queremos representar con algún propósito"

- "*Una especificación formal de una conceptualización compartida*" [Borst, 1997]

- "*Una descripción formal y explícita de los dominios del discurso*" [McGuinnes2000]

Ontologías ligeras y pesadas

- Ontologías ligeras (*lightweight ontologies*)
  - Incluyen conceptos, taxonomías de conceptos, relaciones entre conceptos y propiedades que los describen
- Ontologías pesadas (*heavyweight ontologies*)
  - Tienen en cuenta además axiomas y restricciones de las propiedades

Utilidad de las ontologías

- Compartir una comprensión común de la estructura de la información entre personas y agentes de software
- Permitir la reutilización del conocimiento propio de un dominio
- Hacer las asunciones propias de un dominio explícitas para terceras partes ajenas a él
- Separar el conocimiento del dominio del conocimiento operacional
- Analizar el conocimiento del dominio

[NoyMcGuinness2000]

Representación de ontologías

- Posibilidades:
  - Usar marcos y lógica de primer orden
  - Usar lógica descriptiva
  - Usar técnicas de ingeniería de software
  - Usar tecnologías de bases de datos

Representación de ontologías con lógica de primer orden

UVa

- Hay 5 componentes en una ontología:
  - Clases
  - Relaciones
  - Axiomas
  - Funciones
  - Lógica de primer orden

Representación de ontologías con lógica descriptiva

UVa

- Hay 3 tipos de componentes:
  - Conceptos (clases)
  - Roles (relaciones binarias / propiedades)
  - Individuos (instancias de las clases)

- Se razona sobre todo en base a la subsunción.

Requerimientos para un lenguaje de ontologías

- Una sintaxis bien definida
- Una semántica formal
- Soporte para razonamiento eficiente
- Capacidad expresiva

Representación de ontologías con OWL

UVa

- OWL (*Ontology Web Language*) es un lenguaje estándar para ontologías propuesto por el W3C.
  - http://www.w3.org/

- Surge para resolver las limitaciones de RDF y RDF Schema
  - Restricciones de cardinalidad
  - Restricciones de participación
  - …

OWL y los lenguajes de programación

UVa

- OWL es *declarativo*: describe un estado de las cosas usando lógica
  - La inferencia de nuevo conocimiento se hace usando razonadores. **Cómo** lo implementan NO es competencia de OWL. Los estándares sólo dicen **qué** debe haber en el resultado.

- ✓ Algunas nociones de la ingeniería del software son trasladables a la ingeniería de ontologías:
  - Modularización
  - Patrones
  - Aspectos colaborativos

OWL y las bases de datos

- La semántica utilizada es diferente:
  - Si algo no está en la base de datos, es FALSO
  - Si algo no está en la ontología, simplemente puede ser CIERTO, pero no estar incorporado expresamente

✓ En ambos casos se almacena información.

✓ Existe una cierta analogía entre las aserciones y el contenido de una base de datos.

Limitaciones de RDF Schema (I)

- No es posible restringir el rango de una propiedad (rdfs:range) a un conjunto de clases. Es global para todas las clases.
  - “Las vacas sólo comen plantas, mientras que otros animales comen también carne”
- No se puede decir que dos clases son disjuntas.
  - ‘Hombre’ y ‘Mujer’ son clases disjuntas
- No podemos definir nuevas clases como unión, diferencia, … de otras existentes.
  - ‘Persona’ es la clase resultante de hacer la unión de ‘Hombre’ y ‘Mujer’

Limitaciones de RDF Schema (I)

- No es posible restringir el rango de una propiedad (rdfs:range) a un conjunto de clases. Es global para todas las clases.
  - "Las vacas sólo comen plantas, mientras que otros animales comen también carne"
    - **owl:Restriction**
- No se puede decir que dos clases son disjuntas.
  - 'Hombre' y 'Mujer' son clases disjuntas
    - **owl:disjointWith**
- No podemos definir nuevas clases como unión, diferencia, … de otras existentes.
  - 'Persona' es la clase resultante de hacer la unión de 'Hombre' y 'Mujer'
    - **owl:unionOf, intersectionOf, …**

Limitaciones de RDF Schema (II)

- No podemos expresar restricciones de cardinalidad.
  - Una persona sólo puede tener dos padres

- No podemos expresar ciertas carácterísticas de las propiedades.
  - Propiedades transitivas (‘mayor que’), unicidad (‘madre de’), propiedades inversas (‘escuchar’, ‘ser escuchado por’)

Limitaciones de RDF Schema (II)

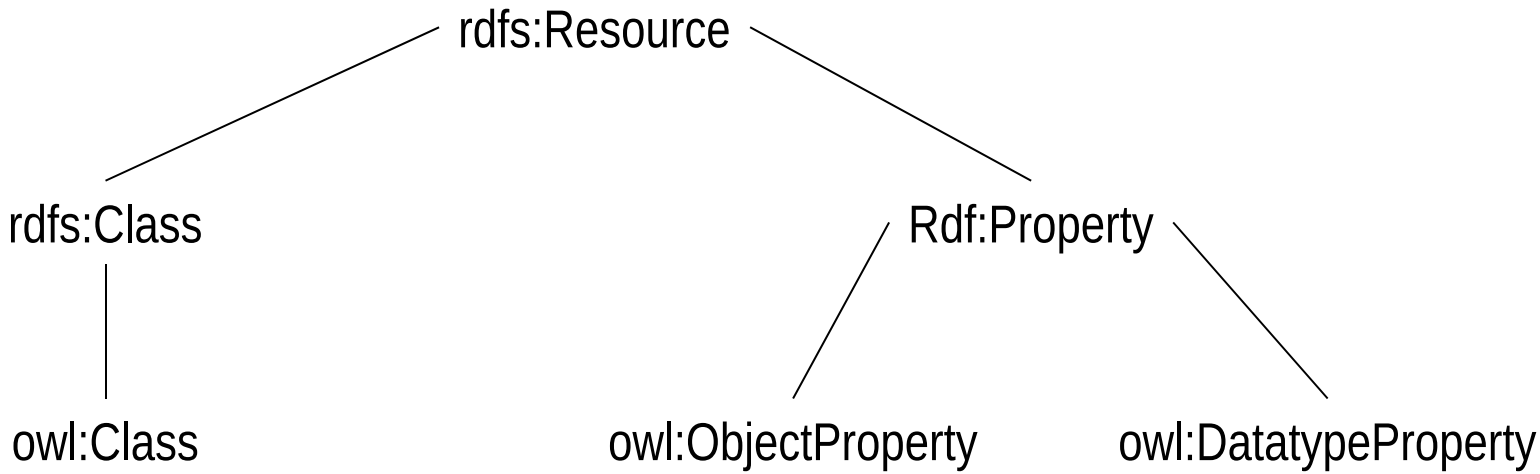
- No podemos expresar restricciones de cardinalidad.
  - Una persona sólo puede tener dos padres
    - **owl:Restriction**

- No podemos expresar ciertas carácterísticas de las propiedades.
  - Propiedades transitivas (‘mayor que’), unicidad (‘madre de’), propiedades inversas (‘escuchar’, ‘ser escuchado por’)
    - **owl:TransitiveProperty, owl:inverseOf, …**

Compatibilidad con RDF/RDFS

- OWL se basa en RDF y RDFS

Relación entre clases OWL y RDF/RDFS



Fuente: [Antoniou04]

Documentos OWL

- Son documentos RDF

Documentos OWL - Cabecera

- Cabecera:

1) El elemento raíz es un elemento rdf:RDF:

```
<rdf:RDF
    xmlns:owl: http://www.w3.org/2002/07/owl#
    ...
```

[Antoniou04; pag. 116]

2) Siguen datos sobre la ontología:

```
<owl:Ontology rdf:about="">
  <rdfs:comment>An example OWL ontology </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>
```

Lista otras ontologías cuyo contenido se importa

[Antoniou04; pag. 116]

Documentos OWL - Clases

- Se usan elementos de tipo owl:Class
  - Ejemplo: Definimos una clase *associateProfessor*

```
< owl:Class rdf:ID="associateProfessor">
      <rdfs:subClassOf rdf:resource="#academicStaffMember"/>
</owl:Class>
```

[Antoniou04; pag. 117]

  - Ejemplo: Indicamos que es disjunta de las clases *professor* y *associateProfessor*

```
<owl:Class rdf:about="#associateProfessor">
      <owl:disjointWith rdf:resource="#professor"/>
      <owl:disjointWith rdf:resource="#assistantProfessor"/>
</owl:Class>
```

[Antoniou04; pag. 117]

Documentos OWL - Clases

- Se usan elementos de tipo owl:Class
  - Ejemplo: Decimos que *faculty* y *academicStaffMember* son clases equivalentes

```
<owl:Class rdf:ID="faculty">
    <owl:equivalentClass rdf:resource="#academicStaffMember"/>
</owl:Class>
```

[Antoniou04; pag. 117]

Clases predefinidas

- *owl:Thing*
  - Clase más general (todo es una ‘cosa’)
  - Cualquier clase es una subclase de *owl:Thing*

- *owl:Nothing*
  - Clase vacía
  - Cualquier clase es una superclase de *owl:Nothing*

Propiedades

- Hay dos tipos de propiedades:
  - **De objetos**: relacionan objetos con otros objetos
    - Ejemplo: *isTaughtBy*
  - **De tipos de datos**: relacionan objetos con valores de tipos de datos
    - Ejemplo: *phone*, *title*, *age*, etc.
    - No hay tipos de datos predefinidos para OWL; se pueden usar los de XML Schema

Documentos OWL - Propiedades

- Para las propiedades de tipo de datos se usan elementos de tipo owl:DatatypeProperty
  - Ejemplo: Definimos la propiedad age

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

[Antoniou04; pag. 117]

Documentos OWL - Propiedades

- Para las propiedades de objetos se usan elementos de tipo *owl:ObjectProperty*
  - Ejemplo: Definimos la propiedad *isTaughtBy*
    - Rango: *academicStaffMember*
    - Dominio: *course*
    - Subpropiedad de *involves*

```
<owl:ObjectProperty rdf:ID="isTaughtBy">
    <rdfs:domain rdf:resource="#course"/>
    <rdfs:range rdf:resource="#academicStaffMember"/>
    <rdfs:subPropertyOf rdf:resource="#involves"/>
</owl:ObjectProperty>
```

[Antoniou04; pag. 118]

Propiedades con múltiples dominios y rangos

UVa

- Una propiedad puede tener varios dominios y varios rangos, en cuyo caso se considera la intersección

[Antoniou04; pag. 118]

Propiedades inversas

- Se pueden declarar dos propiedades como inversas usando *owl:inverseOf*
  - Ejemplo: Definimos la propiedad *isTaughtBy* como inversa de *teaches*

```
<owl:ObjectProperty rdf:ID=”teaches">
      <rdfs:domain rdf:resource="#academicStaffMember"/>
      <rdfs:range rdf:resource="#course"/>
      <owl:inverseOf rdf:resource=”isTaughtBy"/>
</owl:ObjectProperty>
```

[Antoniou04; pag. 118]

Propiedades equivalentes

- Se pueden declarar dos propiedades equivalentes usando *owl:equivalentProperty*
  - Ejemplo: Definimos la propiedad *lecturesIn* como equivalente de *teaches*

```
<owl:ObjectProperty rdf:ID="lecturesIn">
        <owl:equivalentProperty rdf:resource="#teaches"/>
</owl:ObjectProperty>
```

[Antoniou04; pag. 118]

owl:Restriction

- Clase anónima, definida mediante la adición de restricciones a una propiedad
  - Todas las instancias de la clase deben satisfacer la restricción expresada.

- Cómo se define:
  1) Indicando la propiedad a la que afecta: *owl:onProperty*
  2) Expresando la restricción:
     - Sobre el **valor**
     - Sobre la **cardinalidad**

Restricciones de propiedades sobre los valores posibles (I)

UVa

- Se puede especificar las condiciones que deben cumplir todos los miembros de una clase (cuantificador universal)
  - Ejemplo: Requerimos que todos los cursos de primer año sean impartidos por catedráticos (professor)

```
<owl:Class rdf:about="#firstYearCourse">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#isTaughtBy"/>
            <owl:allValuesFrom rdf:resource="#Professor"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

[Antoniou04; pag. 120]

Restricciones de propiedades sobre los valores posibles (II)

UVa

- Se puede especificar el valor que debe tomar una propiedad
  - Ejemplo: Requerimos que todos los cursos de Matemáticas sean impartidos por David Billington

```
<owl:Class rdf:about="#fmathCourse">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#isTaughtBy"/>
            <owl:hasValue rdf:resource="#949352"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

[Antoniou04; pag. 120]

Restricciones de propiedades sobre los valores posibles (III)

UVa

- De modo equivalente se puede expresar el cuantificador existencial
  - Ejemplo: Requerimos que todos los académicos impartan *al menos* un curso

```
<owl:Class rdf:about="#academicStaffMember">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#teaches"/>
            <owl:someValuesFrom rdf:resource="#undergraduateCourse"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

[Antoniou04; pag. 120]

Restricciones de propiedades sobre cardinalidad

UVa

- Se puede expresar cardinalidad mínima usando owl:minCardinality y máxima con owl:maxCardinality
  - Ejemplo: Requerimos que todos los cursos sean impartidos por alguien

```
<owl:Class rdf:about="#course">
   <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#isTaughtBy"/>
            <owl:minCardinality
                    rdf:datatype="&xsd;nonNegativeInteger">
                1
            </owl:minCardinality>
        </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

[Antoniou04; pag. 120]

Propiedades especiales

- *owl:TransitiveProperty* define una propiedad transitiva
  - Ejemplo: *más_alto_que*, *predecesor*
- *owl:SymmetricProperty* define una propiedad simétrica
  - Ejemplo: h*ermano_de*
- *owl:FunctionalProperty* define una propiedad que tiene como máximo un valor para cada objeto
  - Ejemplo: *edad*, *peso*
- *owl:InverseFunctionalProperty* define una propiedad que no puede tomar el mismo valor para dos objetos diferentes
  - Ejemplo: *Numero_SeguridadSocial_de*

Operadores de conjuntos (I)

- Se puede expresar combinaciones de clases (unión, intersección, complemento)
  - Ejemplo: Indicamos que los académicos y los no académicos son conjuntos complementarios

```
<owl:Class rdf:about="#academicStaffMember">
    <rdfs:subClassOf>
        <owl:Class>
            <owl:complementOf rdf:resource="#nonacademicStaffMember"/>
        </owl:Class>
    </rdfs:subClassOf>
</owl:Class>
```

[Antoniou04; pag. 123]

Operadores de conjuntos (II)

- Usamos constructores anidades:
  - Ejemplo: Definimos los académicos como miembros de plantilla, y pertenecientes al complemento de la clase de los no académicos

```
<owl:Class rdf:about="#academicStaffMember">
    <owl:intersectionOf owl:parseType="Collection">
        <owl:Class rdf:about="#StaffMember"/>
        <owl:Class>
            <owl:complementOf   rdf:resource="#nonacademicStaffMember"/>
        </owl:Class>
    </owl:intersectionOf>
</owl:Class>
```

Operadores de conjuntos

- owl:intersectionOf
- owl:unionOf
- owl:complementOf

Enumeraciones

- Una enumeración se expresa con elementos de tipo *owl:OneOf*
  - Ejemplo: Listamos los días de la semana

```
<owl:Class rdf:ID="weekdays">
   <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Monday"/>
        <owl:Thing rdf:about="#Tuesday"/>
        <owl:Thing rdf:about="#Wednesday"/>
        <owl:Thing rdf:about="#Thursday"/>
        <owl:Thing rdf:about="#Friday"/>
        <owl:Thing rdf:about="#Saturday"/>
        <owl:Thing rdf:about="#Sunday"/>
    </owl:oneOf>
</owl:Class>
```

[Antoniou04; pag. 123]

Razonamiento usando los operadores de conjuntos

UVa

- Si *A* es una clase *owl:complementOf* otra clase *B*, entonces todas las subclases de *A* son disjuntas (*owl:disjointWith*) de *B*
  - Ejemplo: La clase *Académico* es complementaria de la clase *NoAcadémico*. Luego, las subclases *Ayudante*, *Catedrático*, … son disjuntas respecto a *NoAcadémico*.

- Si *A* es la unión de dos clases B_1, …, B_N, entonces B_1, …, B_N son subclases de *A*

Instancias

- Las instancias se declaran como en RDF

```
<rdf:Description rdf:ID="949352">
   <rdf:type rdf:resource="#academicStaffMember"/>
</rdf:Description>
```

```
<academicStaffMember rdf:ID="949352" />
```

Capas de OWL

- OWL Full
  - Se basa (asigna significado) según la semántica RDF (*OWL 2 RDF-based Semantics*):
    - Es una extensión de la semántica de RDFS
    - Se tratan las ontologías como grafos RDF
- OWL DL
  - Se basa en la lógica descriptiva
  - Facilita la tarea de los desarrolladores

Variantes de OWL

- OWL Full
  - La más completa: Incluye toda las primitivas OWL
  - 100% compatible con RDF, tanto sintácticamente como semánticamente:
    - ✓ Cualquier documento RDF es también un documento OWL Full
    - ✓ Cualquier conclusión válida en RDF/RDF Schema es válida en OWL Full
    - ↓ Tan potente que es indecidible: no es posible conseguir razonamiento completo (o eficiente)
- OWL DL

Variantes de OWL

- OWL Full

- OWL DL
  - Sublenguaje de OWL Full que restringe los constructores que se pueden usar:
    - No es posible aplicar constructores a otros constructores → Se trabaja con lógica descriptiva
    - ✓ Permite razonamiento eficiente
    - ↓ *Se pierde compatibilidad con RDF:*
      - *Cualquier documento OWL DL es un documento RDF correcto*
      - *Un documento RDF correcto tendrá que modificarse para ser un documento OWL DL correcto*

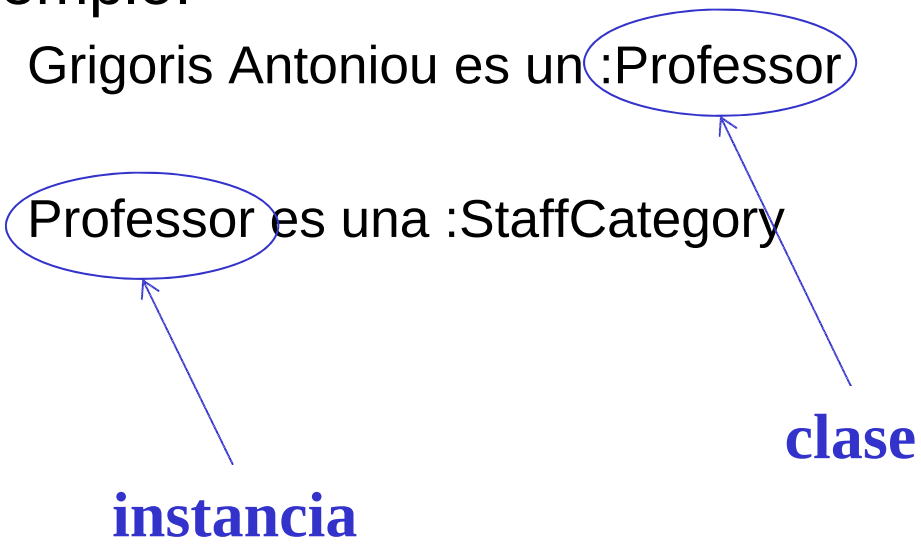
Compatibilidad

- Entre las variantes de OWL:
  - Toda ontología OWL DL válida es OWL Full válida
  - Toda conclusión válida en OWL DL es válida en OWL Full

- Con RDF Schema:
  - Se usa RDF para la sintaxis
  - Las instancias se declaran igual que en RDF, usando descripciones RDF e información sobre tipo
  - Los constructores OWL como *owl:Class*, *owl:DatatypeProperty* y *owl:ObjectProperty* son especializaciones de los equivalentes constructores RDF
  - La compatibilidad total sólo se consigue con OWL Full

¿Qué incorpora OWL 2 sobre OWL 1?

- Poder usar en OWL DL un recurso (IRI) como instancia y como clase simultáneamente:
  - Ejemplo:
    - Grigoris Antoniou es un :Professor
    - Professor es una :StaffCategory



Perfiles en OWL 2

- Son subconjuntos de OWL *(luego, más fáciles de implementar y manipular*), suficientes para una variedad de aplicaciones

- Propiedades comunes:
  - No soportan la negación ni la disyunción

- Tres perfiles:
  1) OWL 2 EL
  2) OWL 2 QL
  3) OWL 2 RL

OWL 2 EL

- Basado en la Lógica Descriptiva (proporciona muchos cuantificadores existenciales)

- Se pueden usar “expresiones de clases” en ambos lados de una aserción *A subClassOf B*, o inferir este tipo de relaciones

OWL 2 QL

- Se basa en la tecnología de bases de datos (ej., SQL)
- Se puede usar los datos “sin tocarlos”
- Permite representar esquemas de bases de datos e integrarlos reformulando consultas

OWL 2 RL

- Se puede usar con cualquier grafo RDF
- Adecuado para enriquecer datos RDF
- El razonamiento se lleva a cabo usando un lenguaje de reglas (**R**ule **L**anguage) estándar

Referencias

- Libros:

  - [Antoniou04] “A Semantic Web Primer”. Grigoris Antoniou and Frank van Harmelen. MIT Press. 2004. ISBN 0-262-01210-3.

  - [Gómez-Pérez03] “Ontological Engineering”. Asunción Gómez-Pérez, Mariano Fernández-López, Oscar Corcho. Springer Verlag. 2003. ISBN 1-85233-551-3.

Referencias

- Libros:
  - [Allemang11] “Semantic Web for the Working Ontologist”. Dean Allemang and Jim Hendler. Morgan Kaufmann. 2011. 2a ed.

Referencias

- URLs:
  - OWL 2 Web Ontology Language Primer. W3C Recommendation 27 October 2009. http://www.w3.org/TR/2009/REC-owl2-primer-20091027/

Ejercicio: Dieta vegetariana

- Ontología de partida

```
:Persona a owl:Class .
:Comida a owl:Class .
:come rds:domain :Persona .
:come rdfs:range :Comida .

:Vegetariano a owl:Class ;
            rdfs:subClassOf :Persona .
:ComidaVegetariana a owl:Class ;
            rdfs:subClassOf :Comida .
```

- Instancias

```
:Pedro :come :Pollo .

:Juan a:Vegetariano ;
            come :Pisto .
```

- Queremos inferir

```
:Pisto a:ComidaVegetariana .
```

- No queremos inferir

```
:Pollo a:ComidaVegetariana .
```

- **¿Cómo hacerlo?**