Ingeniería de Aplicaciones para la Web Semántica

Clase 08

OWL y las Ontologías

Mg. A. G. Stankevicius

Segundo Cuatrimestre

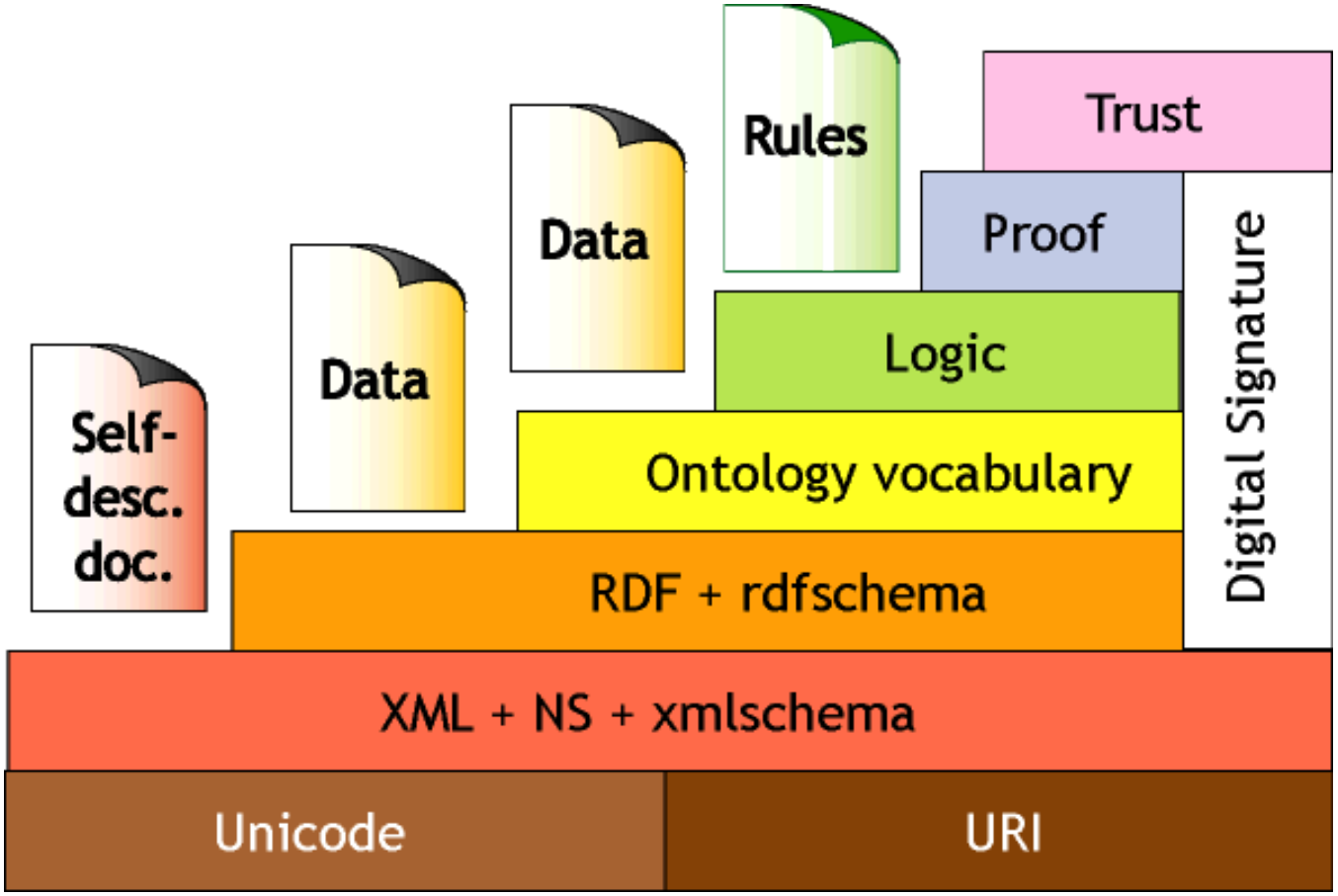
2005

Copyright

Contenidos

- Lenguajes de codificación de ontologías.
- Propósitos de estos lenguajes.
- Limitaciones de RDF y RDFS.
- Tres sabores para OWL.
- Sintaxis XML para OWL.
- Ejemplos prácticos.
- El espacio de nombres OWL
- Futuras extensiones.
- Resumen.

Implementación basada en capas de la web semántica



Lenguajes de codificación de ontologías para la web

- KIF (Knowledge Interchange Format).
- RDF (Resource Description Framework) y RDFS (RDF Schema).
- DAML (DARPA Agent Markup Language).
- OIL (Ontology Inference Layer).
- DAML+OIL, esencialmente la combinación de los anteriores.
- OWL (Web Ontology Language).

Requerimientos para estos lenguajes

- Todo lenguaje de codificación de ontologías debe permitir a los usuarios escribir conceptualizaciones explícitas y fomales acerca de un dominio:
  - Tiene que tener una sintaxis bien definida.
  - Debe contar con una semántica formal.
  - Conviene que brinde acceso a algoritmos de razonamiento eficientes.
  - Tiene que disponer de suficiente poder expresivo.

Poder expresivo vs. eficiencia

- Cuanto más rico en poder expresivo se torno un determinado lenguaje, tanto más ineficiente se torna el proceso de obtención de inferencias en el mismo.
  - Peor aún, bajo ciertas condiciones es posible alcanzar la no computabilidad.
- La solución radica en encontrar un adecuado compromiso entre estos aspectos.

Inferencias basadas en la información ontológica

- Pertenencia a una clase:
  - Si **X** es una instancia de la clase **A** y **A** es una subclase de **B**, entonces se puede inferir que **X** también es instancia de **B**.
- Equivalencia de clases:
  - Si **A** es una clase equivalente a otra clase **B**, y a su vez la clase **B** es equivalente a una tercera clase **C**, entonces se puede inferir que **A** es una clase equivalente a la clase **C**.

Inferencias basadas en la información ontológica

- Consistencia:
  - Si **X** es una instancia de la clase **A** y también lo es de la clase **B**, pero **A** y **B** son clases disjuntas, podemos concluir que estamos en presencia de un error.
- Clasificación:
  - Sabiendo que cuando ciertos atributos toman determinados valores estamos en presencia de instancias de una dada clase, es posible inferir la pertenencia a esa clase.

Posibles usos del conocimiento inferido

- En este contexto, la posibilidad de obtener inferencias permite:
  - Verificar la consistencia entre la ontología y el conocimiento representado por la misma.
  - Comprobar la existencia de relaciones inusitadas entre las clases modeladas.
  - Clasificar automáticamente las instancias dentro de las clases contempladas.
- Todos son aspectos preciados al desarrollar ontologías de gran porte.

Obtención de inferencias en OWL

- Para obtener una semántica formal y disponer de una maquinaria de inferencia asociada debemos:
  - Establecer un mapeo a un formalismo lógico ya conocido.
  - Hacer uso de los razonadores automáticos preexistentes para ese formalismo.
- OWL se mapea (parcialmente) en una lógica de descripción la cual cuenta con eficientes razonadores automáticos.

Limitaciones de RDFS

- Las propiedades tienen un alcance local:
  - El elemento `rdfs:range` explicita el rango de un propiedad para todas las clases.
  - No es posible declara que una restricción de rango se aplica sólo a ciertas clases.
  - Ejemplo: los veganos sólo se alimentan de cereales y vegetales, mientras que los vegetarianos también ingieren huevos y lácteos.

Limitaciones de RDFS

- Exclusión mutua de clases:
  - En ocasiones es útil poder expresar que dos clases se excluyen mutuamente (por ejemplo, masculino y femenino).
- Combinación de clases:
  - También es útil poder crear nuevas clases combinando las existentes, aplicando los operadores de unión, intersección y complemento (por ejemplo, las personas son la unions de masculino y femenino).

Limitaciones de RDFS

- Restricciones de cardinalidad:
  - Por ejemplo, que una persona tiene exáctamente dos progenitores.
- Características particulares de las propiedades:
  - Transitividad (*e.g.*, “mayor-que”).
  - Unicidad (*e.g.*, “madre-de”).
  - Complemento (*e.g.*, “dicta” y “dictado-por”).

OWL como extensión de RDFS

- En un mundo ideal, OWL debería ser una extensión de los esquemas RDF.
  - Recordemos la visión estructurada por capas para la web semántica.
- Se debe tener cuidado de no sacrificar el objetivo de obtener razonadores eficientes en el proceso.
- La combinación irrestricta de RDFS con la lógica conduce rápidamente a la no computabilidad.

Tres sabores para OWL

- El Web Ontology Working Group de la W3C define a OWL como tres sublenguajes diferentes:
  - **OWL Full**
  - **OWL DL**
  - **OWL Lite**
- Cada sublenguaje esta orientado a satisfacer los diferentes objetivos recién reseñados.

OWL Full

- Tiene acceso a todas las primitivas de modelado OWL.
- Permite la combinación arbitraria de estas primitivas con RDF y RDFS.
- **OWL Full** es totalmente compatible con RDF, tanto en su sintaxis como en su semántica.
- **OWL Full** es tiene tanto poder expresivo que es no decidible.

OWL DL

- Es un subconjunto de **OWL Full**:
  - No permite la aplicación de primitivas OWL a otras primitivas OWL.
  - Se corresponde a una lógica de descripción ampliamente estudiada.
- **OWL DL** cuenta con algoritmos de razonamiento altamente eficaces.
- No obstante, perdemos compatibilidad:
  - No todo documento RDF legal es un documento **OWL DL** legal.

OWL Lite

- **OWL Lite** es una restricción adicional por sobre **OWL DL**:
  - **OWL Lite** no permite hacer referencia a clases enumeradas, las declaraciones de exclusión mutua ni las de cardinalidad.
- Ventajas de **OWL Lite**:
  - Fácil de comprender por los usuarios.
  - Fácil de implementar por los desarrolladores.
- La desventaja es evidente: disponemos de un menor poder expresivo.

Compatibilidad entre los sabores de OWL

- Todas las ontologías formuladas en **OWL Lite** son ontologías válidas en **OWL DL**.
- Todas las ontologías formuladas en **OWL DL** son ontologías válidas en **OWL Full**.
- Todas las conclusiones válidas formadas en **OWL Lite** son a su vez conclusiones que puede ser obtenidas en **OWL DL**.
- Todas las conclusiones válidas formadas en **OWL DL** son a su vez conclusiones que puede ser obtenidas en **OWL Full**.

Compatibilidad entre OWL y los esquemas RDF

- Todos los sabores de OWL utilizan a RDF como su sintáxis nativa.
- Todas las instancias son definidas como en RDF, haciendo uso de declaraciones.
- La web semántica propone como objetivo de diseño la compatibilidad hacia abajo:
  - Sólo **OWL Full** goza de los beneficios de esta compatibilidad total, al precio de tornarse computacionalmente intratable.

Sintaxis XML para OWL

- OWL al estar basado en RDF puede hacer uso de su sintaxis basada en XML.
- No obtante, existen otras formulaciones alternativas para OWL:
  - Una basada en XML, un tanto más legible.
  - Una más abtracta que las basadas en XML, mucho más compacta y legible que las anteriores.
  - Una representación gráfica, centrada en torno a las herramientas provistas por UML.

Sintaxis XML para OWL

- Encabezamiento OWL XML/RDF:

```
<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#"
>
```

- Las ontologías OWL suelen contar también con un conjunto de aserciones auxiliares en torno al elemento `owl:Ontology`.

Sintaxis XML para OWL

- Uso de owl:Ontology:

```
<owl:Ontology rdf:about="">
    <rdfs:comment>
        Ejemplo de ontología OWL
    </rdfs:comment>
    <owl:priorVersion
      rdf:resource="http://cs.uns.edu.ar/uni-ns-old"/>
    <owl:imports
      rdf:resource="http://cs.uns.edu.ar/persons"/>
    <rdfs:label>Ontología DCIC</rdfs:label>
</owl:Ontology>
```

- OBS: owl:imports es transitivo.

Definición de Clases

- Las clases se definen apelando a tag `owl:Class`.
  - `owl:Class` es una subclase de `rdfs:Class`.
- La exclusión mutua se explicita apelando al tag `owl:disjointWith`.

```
<owl:Class
  rdf:about="#profesorTitular">
    <owl:disjointWith rdf:resource="#profesorAsociado"/>
    <owl:disjointWith rdf:resource="#profesorAdjunto"/>
</owl:Class>
```

Definición de Clases

- La equivalencia entre clases se define mediante el tag `owl:EquivalentClass`.

```
<owl:Class rdf:ID="docentes">
   <owl:equivalentClass
     rdf:resource="#catedráticos"/>
</owl:Class>
```

- La clase `owl:Thing` es la clase más general, abarca a todas las instancias.
- Su dual es la clase `owl:Nothing`, la clase vacía, la cual carece de instancias.

Propiedades

- OWL distingue las propiedades en dos categorías.
- Por un lado, las propiedades de los objetos, que relacionan objetos entre sí.
  - Por caso, “taught-by”.
- Por otra parte, las propiedades de los tipos de datos, que relacionan objetos con tipos de datos.
  - Por caso, “interno”, “edad”, etc.

Propiedades de los tipos de datos

- OWL aprovecha la implementación basada en capas para la web semántica, haciendo uso de los tipos de datos propios de los esquemas XML:

```
<owl:DatatypeProperty rdf:ID="edad">
   <rdfs:range
     rdf:resource="&xsd;nonNegativeInteger"/>
</owl:DatatypeProperty>
```

Propiedades de los objetos

- Naturalmente, también se pueden definir tipos de datos propios:

```
<owl:ObjectProperty
  rdf:ID="dictado-por">
    <owl:domain rdf:resource="#cursos"/>
    <owl:range rdf:resource="#docentes"/>
    <rdfs:subPropertyOf
      rdf:resource="#cátedra"/>
</owl:ObjectProperty>
```

Propiedades inversas

- OWL permite especificar que una propiedad es la inversa de otra:

```
<owl:ObjectProperty rdf:ID="dicta">
   <rdfs:range rdf:resource="#cursos"/>
   <rdfs:domain rdf:resource="#docentes"/>
   <owl:inverseOf rdf:resource="#dictado-por"/>
</owl:ObjectProperty>
```

Propiedades equivalentes

- De manera análoga, también permite especificar que dos propiedades son equivalentes:

```
<owl:ObjectProperty
  rdf:ID="a-cargo-de">
    <owl:equivalentProperty
      rdf:resource="#dicta"/>
</owl:ObjectProperty>
```

Restricciones sobre las propiedades

- OWL permite declarar que una cierta clase **A** satisface una propiedad dada.
  - Todas sus instancias han de satisfacer la propiedad en cuestión.
- Esto es lo mismo que decir que **A** es una subclase de **B**, donde **B** es la clase de todos los objetos satisfaciendo la propiedad en cuestión.
  - La clase **B** puede permanecer anonima.

Restricciones sobre las propiedades

- La especificación de clases mediante restricciones se formula mediante el elemento `owl:Restriction`.
- Este elemento debe contener anidado al elemento `owl:onProperty`, y una o más declaraciones de restricciones.
- Por caso, un tipo de restricción usual son las de cardinalidad (*e.g.*, al menos uno, exáctamente tres, etc.).

Restricciones sobre las propiedades

- También es factible especificar restricciones sobre los valores que las propiedades pueden tomar:
  - `owl:allValuesFrom` permite indicar una cuantificación universal.
  - `owl:hasValue` permite especificar un determinado valor.
  - `owl:someValuesFrom` permite expresar una cuantificación existencial.

Cuantificación universal

- Todos los docentes están en condiciones de dictar el curso CS101:

```
<owl:Class rdf:about="#CS101">
<rdfs:subClassOf>
   <owl:Restriction>
      <owl:onProperty rdf:resource="#dictado-por"/>
         <owl:allValuesFrom
           rdf:resource="#docentes"/>
   </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

Especificación de un valor determinado

- GRS y nadie más dicta CS5704:

```
<owl:Class rdf:about="#CS5704">
<rdfs:subClassOf>
   <owl:Restriction>
      <owl:onProperty
        rdf:resource="#dictado-por"/>
      <owl:hasValue
        rdf:resource="#1234"/>
   </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

Cuantificación existencial

- Todo docente está a cargo de al menos un curso:

```
<owl:Class rdf:about="#docentes">
<rdfs:subClassOf>
   <owl:Restriction>
      <owl:onProperty rdf:resource="#dicta"/>
      <owl:someValuesFrom rdf:resource="#cursos"/>
   </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

Restricciones de cardinalidad

- A su vez, es posible explicitar tanto el tope mínimo como el máximo a la cardinalidad de determinada propiedad mediante los tags `owl:minCardinality` y `owl:maxCardinality`.
- Notar que se puede fijar un determinado valor para la cardinalidad indicando el mismo valor para ambos.
- Por conveniencia, OWL también permite utilizar el tag `owl:cardinality` a tal fin.

Restricciones de cardinalidad

- Todo cursos tienen que tener al menos un responsable a cargo:

```
<owl:Class rdf:about="#cursos">
<rdfs:subClassOf><owl:Restriction>
   <owl:onProperty rdf:resource="#dictado-por"/>
   <owl:minCardinality
     rdf:datatype=”&xsd;nonNegativeInteger">
      1
   </owl:minCardinality>
</owl:Restriction></rdfs:subClassOf>
</owl:Class>
```

Propiedades especiales

- OWL incluye elementos que permiten especificar propiedades especiales:
  - `owl:TransitiveProperty`, que permite indicar que la propiedad en cuestión es de carácter transitivo (*e.g.*, “ancestro-de”).
  - `owl:SymmetricProperty`, que permite indicar que la propiedad en cuestión es simétrica (*e.g.*, “hermano-de”).
  - `owl:FunctionalProperty`, que explicita que la propiedad en cuestión sólo puede tomar un valor para cada objeto (*e.g.*, “edad”).

Propiedades especiales

- Uso de propiedades especiales para capturar al relación “hace-grupo-con”:

```
<owl:ObjectProperty
  rdf:ID="hace-grupo-con">
    <rdf:type rdf:resource="&owl;TransitiveProperty"/>
    <rdf:type rdf:resource="&owl;SymmetricProperty"/>
    <rdfs:domain rdf:resource="#estudiantes"/>
    <rdfs:range rdf:resource="#estudiantes"/>
</owl:ObjectProperty>
```

Operadores booleanos

- OWL posibilia la definición de clases a partir de las existentes mediante la aplicación de operadores booleanos:

```
<owl:Class rdf:ID="personas">
   <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#docentes"/>
      <owl:Class rdf:about="#estudiantes"/>
   </owl:unionOf>
</owl:Class>
```

Anidamiento de los Operadores booleanos

```
<owl:Class rdf:ID="administrativos">
   <owl:intersectionOf rdf:parseType="Collection">
   <owl:Class rdf:about="#personal"/>
   <owl:complementOf>
      <owl:unionOf rdf:parseType="Collection">
         <owl:Class rdf:about="#docentes"/>
         <owl:Class rdf:about="#sysops"/>
      </owl:unionOf>
   </owl:complementOf>
   </owl:intersectionOf>
</owl:Class>
```

Declaración de clases por extensión

- OWL permite declarar clases de forma explícita, indicando las instancias que la componen:

```
<owl:Class rdf:ID="finde">
   <owl:oneOf rdf:parseType="Collection">
      <owl:Thing rdf:about="#sábado"/>
      <owl:Thing rdf:about="#domingo"/>
   </owl:oneOf>
</owl:Class>
```

Declaración de instancias

- En OWL, las instancias de las clases igual que en RDF:

```
<rdf:Description rdf:ID="1234">
   <rdf:type rdf:resource="#profesorTitular"/>
</rdf:Description>

<profesorTitular rdf:ID="4321">
   <uni:age rdf:datatype="&xsd;integer">
      39
   </uni:age>
</profesorTitular>
```

Unicidad de los nombres

- OWL no asume la UNA (unique names assupmtion).
- El hecho que dos instancias posean distintos nombre no implica que se trate de dos individuos diferentes.
  - Por ejemplo, si los cursos son dictados por a lo sumo un docente, pero declaramos que dos docentes dictan un dado curso, un razonador basado en OWL no consideraría esta situación como errónea; simplemente considera que se trata del mismo individuo.

Explicitación de las diferencias

- Para asegurarnos que dos individuos sean reconocidos como tales, es posible explicitar esta situación mediante el tag `owl:differentFrom`.

```
<docentes rdf:about="1234">
   <owl:differentFrom rdf:resource="4321"/>
</docentes>
```

Explicitación de las diferencias

- Esta explicitación dos a dos es poco práctica en grandes dominios. OWL provee otro contructor que acepta una lista de individuos:

```
<owl:allDifferent>
   <owl:distinctMembers rdf:parseType="Collection">
      <Docentes rdf:about="1234"/>
      <Docentes rdf:about="4321"/>
      <Docentes rdf:about="2666"/>
      <Docentes rdf:about="2699"/>
   </owl:distinctMembers>
</owl:allDifferent>
```

Tipado de datos en OWL

- Si bien XMLS permite hacer uso de tipos de datos definidos por el usuario, los mismo carecen de semántica.
- La especificación de OWL no permite hacer uso de los mismos.
- No obstante, se permite utilizar los tipos de datos sin estructura, presentes en la especificación de XMLS:
  - Por caso, “string”, “integer”, etc.

Gestión de versiones

- OWL permite incluir información para la gestión de versiones.
- El tag `owl:priorVersion` hace referencia a versiones anteriores de la ontología.
  - Este tag no posee significado formal alguno.
  - El hacer uso del mismo queda en manos de las herramientas que hagan uso de la ontología.

Gestión de versiones

- El tag `owl:backwardCompatibleWith` permite explicitar que se preserva la compatibilidad con otras ontologías.
  - Todos los identificadores de la versión anterior conservan la misma interpretación bajo la nueva versión.
- El tag `owl:incompatibleWith` indica justamente lo opuesto, que la nueva ontología mejora a la anterior, pero se ha hecho incompatible con la misma.

Combinación de primitivas de modelado

- Los distintos sabores de OWL imponen diferentes restricciones en lo que a la combinación de primitivas de modelado respecta.
- En **OWL Full** todas las combinaciones posibles son aceptadas, siempre y cuando el resultado siga constituyendo un documento RDF válido.

Restricciones impuestas por OWL DL

- Particionado de conceptos:
  - Los recursos descriptos deben ser o bien clases, tipos de datos, propiedades de los objetos, propiedades de los tipos de datos, una instancia, un valor literal o una parte predefinida del lenguaje, y no más de uno de estos.
- Tipado explícito:
  - El particionado de conceptos debe ser explícito (*e.g.*, no basta con aparecer como superclase para inferir que es una clase).

Restricciones impuestas por OWL DL

- Exclusión mutua de propiedades:
  - El conjunto de las propiedades de los objetos y de los tipos de datos deben ser disjuntos.
  - En consecuencia, los siguientes modificadores no puede ser aplicados a propiedades de los tipos de datos:
    - `owl:inverseOf`
    - `owl:FunctionalProperty`
    - `owl:InverseFuctionalProperty`
    - `owl:SymmetricProperty`

Restricciones impuestas por OWL DL

- Restricciones de cardinalidad transitivas:
  - **OWL DL** no permite especificar restricciones de cardinalidad sobre propiedades que hayan sido declaradas transitivas.
- Restricciones en el uso de clases anónimas:
  - Las clases anónimas sólo pueden aparecer en las siguientes situaciones:
    - Como dominio o rango de `owl:equivalentClass` o bien `owl:disjointWith`.
    - Como rango de `rdfs:subclassOf`.

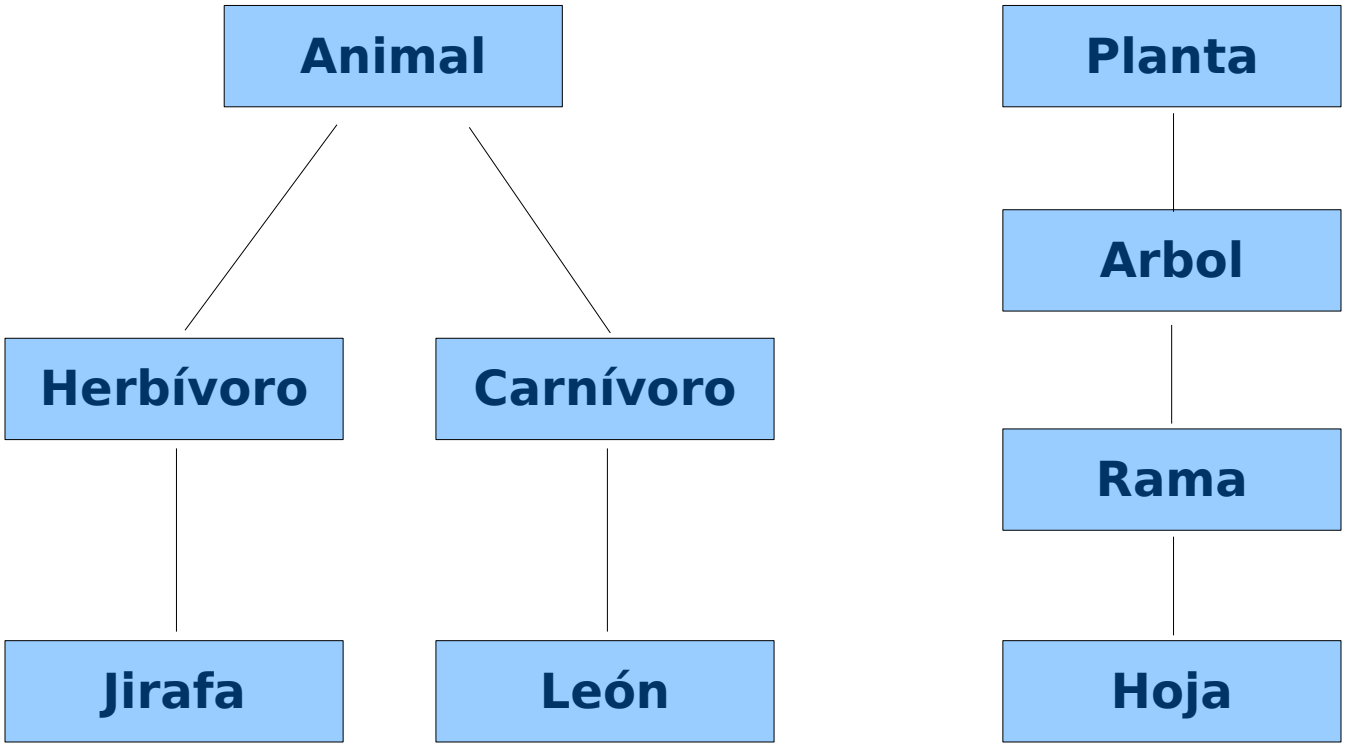
Restricciones impuestas por OWL Lite

- **OWL Lite** mantiene todas las restricciones impuestas por **OWL DL**.
- No se permite utilizar las siguientes primitivas de modelado:
  - `owl:oneOf`
  - `owl:disjointWith`
  - `owl:unionOf`
  - `owl:complementOf`
  - `owl:hasValue`

Restricciones impuestas por OWL Lite

- Las restricciones de cardinalidad permitidas en **OWL DL** pueden ser utilizadas en **OWL Lite**, pero sólo si hacen mención de los cardinales 0 y 1.
- La primitiva `owl:equivalentClass` solo puede mencionar clases definidas de manera explícita. Es decir, no puede ser utilizado en declaraciones que involucren clases anónimas.

Un ejemplo del mundo real



Propiedades consideradas

```
<owl:TransitiveProperty
  rdf:ID="parte-de"/>
<owl:ObjectProperty
  rdf:ID="come">
   <rdfs:domain rdf:resource="#animal"/>
</owl:ObjectProperty>
<owl:ObjectProperty
  rdf:ID="comido-por">
   <owl:inverseOf rdf:resource="#come"/>
</owl:ObjectProperty>
```

Plantas y árboles

```
<owl:Class rdf:ID="planta">
   <rdfs:comment>Plantas y animales no
     comparten instancias en común.
     </rdfs:comment>
   <owl:disjointWith="#animal"/>
</owl:Class>
<owl:Class rdf:ID="árbol">
   <rdfs:comment>Los árboles son un tipo de
     planta.</rdfs:comment>
   <rdfs:subClassOf rdf:resource="#planta"/>
</owl:Class>
```

Restricciones sobre las ramas

```
<owl:Class rdf:ID="rama">
   <rdfs:comment>Las ramas son parte de los
     árboles.</rdfs:comment>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#parte-de"/>
         <owl:allValuesFrom rdf:resource="#árbol"/>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Restricciones sobre las hojas

```
<owl:Class rdf:ID="hoja">
   <rdfs:comment>Las hojas forman parte de las
     ramas.</rdfs:comment>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#parte-de"/>
         <owl:allValuesFrom rdf:resource="#rama"/>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Caracterización de los carnívoros

```
<owl:Class rdf:ID="carnívoro">
   <rdfs:comment>Los carnívoros son aquellos
     animales que comen a otros animales.
     </rdfs:comment>
   <owl:intersectionOf rdf:parsetype="Collection">
      <owl:Class rdf:about="#animal"/>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#come"/>
         <owl:someValuesFrom
           rdf:resource="#animal"/>
      </owl:Restriction>
   </owl:intersectionOf>
</owl:Class>
```

Caracterización de los herbívoros

```
<owl:Class rdf:ID="herbívoro">
   <rdfs:comment>
      Los herbívoros son aquellos animales que comen
        plantas o bien partes de plantas.
   </rdfs:comment>

   <rdfs:comment>
      ¿Qué estructura debería tener esta definición?
   </rdfs:comment>
</owl:Class>
```

Restricciones acerca de las jirafas

```
<owl:Class rdf:ID="jirafa">
   <rdfs:comment>Las jirafas son herbívoros,
     pero sólo comen hojas.</rdfs:comment>
   <rdfs:subClassOf rdf:type="#herbívoro"/>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#come"/>
         <owl:allValuesFrom rdf:resource="#hoja"/>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Restricciones acerca de las jirafas

```
<owl:Class rdf:ID="león">
   <rdfs:comment>Los leones son animales que
     sólo comen herbívoros.</rdfs:comment>
   <rdfs:subClassOf rdf:type="#carnívoro"/>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#come"/>
         <owl:allValuesFrom
           rdf:resource="#herbívoro"/>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Plantas digestivas

```
<owl:Class rdf:ID="plantasDigestivas">
   <rdfs:comment>
      Las plantas digestivas son comidas tanto por
       los carnívoros como por los herbívoros.
   </rdfs:comment>

   <rdfs:comment>
      ¿Qué estructura debería tener esta definición?
   </rdfs:comment>
</owl:Class>
```

Definición de OWL en términos de sí mismo

- La especificación formal de OWL está expresada en ese mismo lenguaje:
  - http://www.w3.org/2002/07/owl
- A continuación repasaremos un fragmento de esta especificación.
- Notemos que la misma no captura la totalidad de la semántica:
  - Sigue haciendo falta una caracterización externa al igual que en RDF y RDFS.

Clases OWL

- La clase de todas las clases OWL es a su vez una subclase de la clase de todas las clases RDFS:

```
<rdfs:Class rdf:ID="Class">
   <rdfs:label>Class</rdfs:label>
   <rdfs:subClassOf
     rdf:resource="&rdfs;Class"/>
</rdfs:Class>
```

Las clases Thing y Nothing

- `Thing` es la clase más general en OWL.
- `Nothing` es la clase más específica, la clase vacía.
- Las siguientes relaciones se verifican entre estos conceptos:
  - $Thing = Nothing \cup (Nothing)^c$
  - $Nothing = (Thing)^c = (Nothing \cup (Nothing)^c)^c$
  - $Nothing = (Nothing)^c \cap ((Nothing)^c)^c = \varnothing$

Especificación de Thing y de Nothing

```
<Class rdf:ID="Thing">
    <rdfs:label>Thing</rdfs:label>
    <unionOf rdf:parseType="Collection">
        <Class rdf:about="#Nothing"/>
        <Class>
            <complementOf rdf:resource="#Nothing"/>
        </Class>
    </unionOf>
</Class>
<Class rdf:ID="Nothing">
    <rdfs:label>Nothing</rdfs:label>
    <complementOf rdf:resource="#Thing"/>
</Class>
```

Equivalencia de clases y de propiedades

```
<rdf:Property rdf:ID="EquivalentClass">
   <rdfs:label>EquivalentClass</rdfs:label>
   <rdfs:subPropertyOf
     rdf:resource="&rdfs;subClassOf"/>
   <rdfs:domain rdf:resource="#Class"/>
   <rdfs:range rdf:resource="#Class"/>
</rdf:Property>
<rdf:Property rdf:ID="EquivalentProperty">
   <rdfs:label>EquivalentProperty</rdfs:label>
   <rdfs:subPropertyOf
     rdf:resource="&rdfs;subPropertyOf"/>
</rdf:Property>
```

Exclusión mutua de clases

- La exclusión mutua capturada por el tag `owl:disjointWith` se caracteriza de la siguiente manera:

```
<rdf:Property rdf:ID="disjointWith">
   <rdfs:label>disjointWith</rdfs:label>
   <rdfs:domain rdf:resource="#Class"/>
   <rdfs:range rdf:resource="#Class"/>
</rdf:Property>
```

Igualdad y desigualdad

- OWL permite especificar la igualdad o la desigualdad entre cosas arbitrarias:
  - `owl:sameIndividualsAs`
  - `owl:sameAs`
  - `owl:differentFrom`
- OWL Full permite incluso aplicar estos modificadores a las propias clases.

Unión e intersección de clases

- En OWL, la unión e intersección de clases permite definir nuevas clases a partir de un conjunto de clases especificado mediante una lista.

```
<rdf:Property rdf:ID="unionOf">
   <rdfs:domain rdf:resource="#Class"/>
   <rdfs:range rdf:resource="&rdf;List"/>
</rdf:Property>
```

Restricciones sobre las clases

- En OWL, la restricción de clases permite definir la clase de todas las instancias que satisfacen las restricciones impuestas.

```
<rdfs:Class rdf:ID="Restriction">
   <rdfs:label>Restriction</rdfs:label>
   <rdfs:subClassOf rdf:resource="#Class"/>
</rdfs:Class>
```

Definición de las propiedades de igualdad y desigualdad

```
<rdf:Property
  rdf:ID="sameIndividualAs">
   <rdfs:domain rdf:resource="#Thing"/>
   <rdfs:range rdf:resource="#Thing"/>
</rdf:Property>

<rdf:Property rdf:ID="sameAs">
   <EquivalentProperty
     rdf:resource="#sameIndividualAs"/>
</rdf:Property>
```

Especificación de restricciones

- La especificación de restricciones sólo pueden aparecer en el marco de la definición de una restricción.
- Las especificaciones de restricciones (`owl:onProperty`, `owl:allValuesFrom`, `owl:minCardinality`, etc.), siempre tienen a `owl:Restriction` como dominio, si bien su rango puede diferir.

Especificación de restricciones

```
<rdf:Property rdf:ID="onProperty">
   <rdfs:label>onProperty</rdfs:label>
   <rdfs:domain rdf:resource="#Restriction"/>
   <rdfs:range rdf:resource="&rdf;Property"/>
</rdf:Property>

<rdf:Property rdf:ID="allValuesFrom">
   <rdfs:label>allValuesFrom</rdfs:label>
   <rdfs:domain rdf:resource="#Restriction"/>
   <rdfs:range rdf:resource="&rdfs;Class"/>
</rdf:Property>
```

Especificación de restricciones

```
<rdf:Property rdf:ID="hasValue">
   <rdfs:label>hasValue</rdfs:label>
   <rdfs:domain rdf:resource="#Restriction"/>
</rdf:Property>

<rdf:Property rdf:ID="minCardinality">
   <rdfs:label>minCardinality</rdfs:label>
   <rdfs:domain rdf:resource="#Restriction"/>
   <rdfs:range
     rdf:resource="&xsd;nonNegativeInteger"/>
</rdf:Property>
```

Propiedades

- Tanto owl:objectProperty como owl:dataTypeProperty son casos especiales de rdf:Property:

```
<rdfs:Class rdf:ID="ObjectProperty">
   <rdfs:label>ObjectProperty</rdfs:label>
   <rdfs:subClassOf
     rdf:resource="&rdf;Property"/>
</rdfs:Class>
```

Propiedades

- Los modificadores *“simétrico”*, *“funcional”* y *“funcional inverso”* sólo se aplican a las propiedades de los objetos:

```
<rdfs:Class
  rdf:ID="TransitiveProperty">
   <rdfs:label>TransitiveProperty</rdfs:label>
   <rdfs:subClassOf
     rdf:resource="#ObjectProperty"/>
</rdfs:Class>
```

Propiedades

- El modificadores *“inverso de”*, debe relacionar a propiedades de los objetos:

```
<rdfs:Class
  rdf:ID="TransitiveProperty">
   <rdfs:label>TransitiveProperty</rdfs:label>
   <rdfs:subClassOf
     rdf:resource="#ObjectProperty"/>
</rdfs:Class>
```

Posibles extensiones de OWL

- Mejorar la modularización.
- Manejo de información por defecto.
- Suposición de mundo cerrado.
- Unicidad de los indentificadores.
- Definiciones procedurales.
- Reglas para explicitar el encadenamientos de propiedades.

Mejorar la modularización

- El esquema de importación actual es muy básico.
  - Sólo se puede importar la totalidad de la ontología.
- El concepto de módulo se centra en la noción de ocultación de información: declará la funcionalidad pero evitame los detalles de implementación.
  - ¿Como se podría aplicar este concepto al desarrollo de ontologías en OWL?

Manejo de información por defecto

- Los sistemas de representación de conocimiento usualmente permiten que los atributos heredados sean refinados por la información propia de las clases más específicas.
  - Solución: trata a los valores heredados como si fueran reglas por defecto.
  - No obstante, todavía no se ha alcanzado un adecuado consenso a la hora de capturar el comportamiento no monotono de las reglas por defecto.

Suposición de mundo cerrado

- OWL en la actualidad opera en el marco de un mundo abierto:
  - Una declaración no se supone inválida simplemente por no poder demostrarla.
  - Decisión acertada en el contexto de internet.
- No obstante, sería interesante también poder contar con la CWA:
  - Una declaración se supone válida si su negación no puede ser probada.
  - Conduce a un comportamiento no monótono.

Unicidad de los indentificadores

- En el marco de las bases de datos es usual asumir la UNA:
  - Los individuos con nombres diferentes son de hecho diferentes.
- OWL sigue la convención usual de la lógica, donde este no es el caso.
  - Una vez más, tiene sentido en el marco de internet.
  - Sería útil poder elegir en qué porciones de una ontología queremos que valga o no.

Definiciones procedurales

- En el area de representación de conocimiento es usual definir un concepto mediante una porción de código que al ser ejecutado computa el significado de ese concepto.
  - Observemos que la definición de este concepto se torna implícita.
- Si bien es una práctica difundida, parece un tanto difícil de integrar en el marco de la web semántica.

Encadenamiento de propiedades

- OWL no permite el encadenamiento de propiedades para no caer en la no decidibilidad.
- No obstante, sería muy útil poder definir propiedades de esta forma.
  - Por ejemplo, empleando reglas genéricas acerca de otras propiedades.
- La integración del conocimiento basado en reglas y el conocimiento estilo DL es un area de investigación muy activa.

En síntesis...

- OWL es el estandar actual para representar ontologías en la red.
- OWL elabora sobre la base establecida por RDF y RDFS:
  - Se adopta la misma sintaxis basada en XML introducida para RDF.
  - Las instancias se definen igual que en RDF.
  - La mayor parte de las primitivas de modelado introducidas para RDFS siguen siendo usadas en el marco de OWL.

En síntesis...

- La semántica forma y la maquinaría de razonamiento surge al mapear OWL en una lógica convencional.
  - Se han ensayado con la lógica de predicados y con una lógica de descripción.
- Si bien OWL es lo suficientemente poderoso por sí mismo, diversas extensiones están en desarrollo.
  - Estas extensiones intentan abarcar nuevos aspectos lógicos.