Knowledge Graphs
Web Ontology Language (OWL)
Previously on “Knowledge Graphs”

• Back to the lecture on RDF Schema
• We have the following statement:
  – "Madrid is the capital of Spain."

• We can get the following information:
  – "Madrid is the capital of Spain." ✔
  – "Spain is a state." ✔
  – "Madrid is a city." ✔
  – "Madrid is located in Spain." ✔
  – "Barcelona is not the capital of Spain." ✗
  – "Madrid is not the capital of France." ✗
  – "Madrid is not a state." ✗
  – ...
Previously on “Knowledge Graphs”

• What we cannot express (up to now):
  – "Every state has exactly one capital"
    • Property cardinalities
  – "Every city can only be the capital of one state."
    • Functional properties
  – "A city cannot be a state at the same time."
    • Class disjointness
  – ...

• For those, we need more expressive languages than RDFS!
Previously on “Knowledge Graphs”

• We have learned about ontologies
  – and RDF Schema as a language for building simple ontologies

• With RDF Schema, we can express some knowledge about a domain
  – but not everything, e.g., cardinalities
  – we cannot produce contradictions
  – we cannot circumvent the Non Unique Naming Assumption
  – we cannot circumvent the Open World Assumption
  – ...

Semantic Web Technology Stack

here be dragons...

Semantic Web Technologies (This lecture)

Technical Foundations

Berners-Lee (2009): Semantic Web and Linked Data
Web Ontology Language (OWL)

• Hey, wait...
Web Ontology Language (OWL)

- More powerful than RDF Schema

- Trade-off:
  - Expressive power
  - Complexity of reasoning
  - Decidability

- Solution: different variants of OWL, e.g.,
  - OWL Lite, OWL DL, OWL Full
  - Profiles in OWL2
Web Ontology Language (OWL)

- Three variants
  - increasing expressive power
  - backwards compatible
    - each OWL Lite ontology is valid in OWL DL and OWL Full
    - each OWL DL ontology is valid in OWL Full
OWL and RDF Schema

• both are based on RDF
  – OWL ontologies can also be expressed in RDF
  – as triples or in XML notation

• Compatibility
  – OWL Lite and OWL DL are not fully compatible to RDF Schema
    • but reuse some parts of RDF Schema
  – OWL Full and RDF Schema are fully compatible
OWL: Classes

• Basic concept (owl:Class)

• Subclasses as we know them from RDFS: rdfs:subClassOf
  – In particular, the following holds:
    owl:Class rdfs:subClassOf rdfs:Class .

• Two predefined classes:
  – owl:Thing
  – owl:Nothing

• For each class $c$, the following axioms hold:
  – $c$ rdfs:subClassOf owl:Thing .
  – owl:Nothing rdfs:subClassOf $c$ .
OWL: Classes

• Classes can be intersections of others:

\[ \text{:SwimmingMammals} \text{ owl:intersectionOf} (\text{:SwimmingAnimals} \text{ :Mammals}) \] .

• There are also set unions and set differences
  – but not in OWL Lite
OWL: Properties

• RDF Schema does not distinguish literal and object valued properties:

\[
\text{:name a rdf:Property .} \\
\text{:name rdfs:range xsd:string .} \\
\text{:knows a rdf:Property .} \\
\text{:knows rdfs:range foaf:Person .}
\]

• Without specifying the range, “dual use” of an RDF property is not forbidden:

\[
\text{:peter :knows :john .} \\
\text{:peter :knows “mary” .}
\]
OWL: Properties

• RDF Schema does not distinguish literal and object valued properties:

  :name a rdf:Property .
  :name rdfs:range xsd:string .
  :knows a rdf:Property .
  :knows rdfs:range foaf:Person .

• In contrast, OWL distinguishes
  – owl:DatatypeProperty
  – owl:ObjectProperty

• The following axioms hold:
  – owl:DatatypeProperty rdfs:subClassOf rdf:Property .
  – owl:ObjectProperty rdfs:subClassOf rdf:Property .
OWL: Properties

- As in RDF Schema, there can be hierarchies and domains/ranges:
  
  :capitalOf rdfs:subPropertyOf :locatedIn .

- Domain
  - only classes for OWL Lite, classes or restrictions* for OWL DL/Full
    
    :name rdfs:domain foaf:Person .

- Range
  - XML Datatypes for owl:DatatypeProperty
    
    :name rdfs:range xsd:string .
  - Classes or restrictions* for owl:ObjectProperty
    
    :knows rdfs:range foaf:Person .

* we'll get there soon
Equality and Inequality (1)

• Equality between individuals
  – Allows using multiple definitions/descriptions of an entity
  – in other datasets as well
  – solves some problems of the Non unique naming assumption

: Muenchen owl:sameAs :Munich .

• We have seen this used for Linked Open Data
  – as a means to establish links between datasets

 myDataset: Mannheim owl:sameAs dbpedia: Mannheim .
Equality and Inequality (2)

• Equality between classes and properties
  – allows for relations between datasets on the schema level
  – gives way to more complex constructs

:UniversityTeachers owl:equivalentClass :Lecturers .
:teaches owl:equivalentProperty :lecturerFor .

• Also useful for Linked Open Data:
  dc:creator owl:equivalentProperty foaf:maker .
Equality and Inequality (3)

- Inequality between individuals
  - Allows some useful reasoning
  - as we will see soon

\[
\text{:Muenchen} \quad \text{owl:differentFrom} \quad \text{:Hamburg}.
\]

- Shorthand notation for multiple entities:

\[
\text{owl:AllDifferent} \quad \text{owl:distinctMembers}
\]
\[
(\text{:Munich} \quad \text{:Hamburg} \quad \text{:Berlin} \quad \text{:Darmstadt} \quad \text{:Mannheim})
\]
Why owl:sameAs, owl:equivalentClass, etc.?  

- In OWL (Lite+DL), we must not mix classes, properties, and instances

- owl:sameAs has owl:Thing as domain/range
- owl:equivalentClass has rdfs:Class as domain/range
  - recap: owl:Class rdfs:subClassOf rdfs:Class
- owl:equivalentProperty has rdf:Property as domain/range
  - owl:ObjectProperty rdfs:subClassOf rdf:Property
  - owl:DatatypeProperty rdfs:subClassOf rdf:Property
Special Properties in OWL

• Symmetric Properties
  
  :sitsOppositeOf a owl:SymmetricProperty .
  :Tom :sitsOppositeOf :Sarah .
  
• Inverse Properties
  
  :supervises owl:inverseOf :supervisedBy .
  :Tom :supervises :Julia .
  → :Julia :supervisedBy :Tom .
  
• Transitive Properties
  
  :hasOfficeMate a owl:TransitiveProperty .
  :Tom :hasOfficeMate :Jon . :Jon :hasOfficeMate :Kim .
  → :Tom :hasOfficeMate :Kim .
Special Properties introduced with OWL2

• Reflexive, irreflexive, and asymmetric properties
• Everybody is a relative of him/herself
  :relativeOf a owl:ReflexiveProperty.

• Nobody can be his/her own parent
  :parentOf a owl:IrreflexiveProperty.

• If I am taller than you, you cannot be taller than me
  :tallerThan a owl:AsymmetricProperty.
Restrictions on Property Types

• Only ObjectProperties may be transitive, symmetric, inverse, and reflexive
  – DataProperties may not be

• Why?

• Previously on RDF:
  – "Literals can only be objects, never subjects or predicates."
Restrictions on Property Types

- Assuming that

\[ \text{samePerson} \text{ a } \text{owl:DatatypeProperty} . \]
\[ \text{samePerson} \text{ rdfs:range xsd:string} . \]
\[ \text{samePerson} \text{ a } \text{owl:SymmetricProperty} . \]

\[ \text{Peter} : \text{samePerson} \text{ "Peter"} . \]

\[ \rightarrow \text{"Peter"} : \text{samePerson} : \text{Peter} . \]
Restrictions on Property Types

- Assuming that

```
:hasName a owl:DatatypeProperty .
:hasName rdfs:range xsd:string .
:hasName owl:inverseOf :nameOf .

:Peter :hasName "Peter" .

→"Peter" :nameOf :Peter .
```
Restrictions on Property Types

• **owl:TransitiveProperty** is also restricted to ObjectProperties

```r
:hasPseudonym a owl:DatatypeProperty .
:hasPseudonym rdfs:range xsd:string .
:hasPseudonym a owl:TransitiveProperty .

:Thomas :hasPseudonym "Dr. Evil" .

+ "Dr. Evil" :hasPseudonym "Skullhead" .

→:Thomas :hasPseudonym "Skullhead" .
```

• Which statement would we need here to make the conclusion via the **owl:TransitiveProperty**?
Functional Properties

• Usage

:hasCapital a owl:FunctionalProperty.
:Finland :hasCapital :Helsinki .
:Finland :hasCapital :Helsingfors .
→ :Helsinki owl:sameAs :Helsingfors .

• Interpretation
  – if A and B are related via fp
  – and A and C are related via fp
  – then, B and C are equal

• simply speaking: fp(x) is unique for each x

• “there can only be one”

http://www.allmystery.de/dateien/uh60808,1274716100,highlander-christopher-lambert.jpg
Inverse Functional Properties

• Usage

:capitalOf a owl:InverseFunctionalProperty .
:Helsinki :capitalOf :Finland .
:Helsingfors :capitalOf :Finland .
→:Helsinki owl:sameAs :Helsingfors .

• Interpretation
  – if A and C are in relation ifp
  – and B and C are in relation ifp
  – then, A and B are the same

• Simply speaking: ifp(x) is a unique identifier for x
  – like a primary key in a database
Pooh!

- OWL is, in fact, more powerful
- ...but we can achieve lots with what we already learned
- Let's get back to the example...
Previously on “Knowledge Graphs”

• Let's look at that sentence:
  – “Madrid is the capital of Spain.”

• We can get the following information:
  – “Madrid is the capital of Spain.” ✔
  – “Spain is a state.” ✔
  – “Madrid is a city.” ✔
  – “Madrid is located in Spain.” ✔
  – “Barcelona is not the capital of Spain.” ✗
  – “Madrid is not the capital of France.” ✗
  – “Madrid is not a state.” ✗
  – ...
Expressive Ontologies using OWL

• "Barcelona is not the capital of Spain." ✗

• Why not?
  – Countries have exactly one capital
  – Barcelona and Madrid are not the same

• In OWL:

```java
:capitalOf a owl:InverseFunctionalProperty .
:Madrid owl:differentFrom :Barcelona .

ASK { :Barcelona :capitalOf :Spain . } → false
```
"Madrid is not the capital of France." ✗

Why not?
- A city can only be the capital of one country
- Spain and France are not the same

Also:

```owl
:capitalOf a owl:FunctionalProperty .
:Spain owl:differentFrom :France .

ASK { :Madrid :capitalOf :France . } → false
```
Restrictions

• Define characteristics of a class
  – A powerful and important concept in OWL
  – Example: Vegan recipes only contain vegetables as ingredients

```owl
:VeganRecipe rdfs:subClassOf :Recipe .
:VeganRecipe rdfs:subClassOf [ 
  a owl:Restriction ;
  owl:onProperty :hasIngredient ;
  owl:allValuesFrom :Vegetable .
] .
```
Further Examples for Restrictions

- Every human as exactly one mother

  :Human rdfs:subClassOf [
      a owl:Restriction ;
      owl:onProperty :hasMother ;
      owl:cardinality 1^^xsd:integer .
  ] .

- Bicycles are vehicles without a motor

  :Bicycle rdfs:subClassOf :Vehicle .

  :Bicycle rdfs:subClassOf [ 
    a owl:Restriction ;
    owl:onProperty :hasMotor ;
    owl:cardinality 0^^xsd:integer .
  ] .
Restrictions vs. Ranges

• Restrictions are local to a class
  
  :VeganRecipe rdfs:subClassOf [ 
    a owl:Restriction ;
    owl:onProperty :hasIngredient ;
    owl:allValuesFrom :Vegetable . 
  ] .
  
  – other classes may use hasIngredient with meat or fish

• Range: a global restriction
  
  :hasIngredient rdfs:range :Food .
  
  – this holds once and for all whenever hasIngredient is used
The Anatomy of a Restriction

- **onProperty**
  - defines the property on which the restriction should hold

- **Restriction of values**
  - owl:allValuesFrom – all values must be in this class
  - owl:someValuesFrom – at least one value must be in this class

- **Restriction of cardinalities**
  - owl:minCardinality – at least n values
  - owl:maxCardinality – at most n values
  - owl:cardinality – exactly n values

- Both cannot be combined

OWL Lite: only n=0 or n=1
Further Examples for Restrictions

- All ball sports require a ball
  
  `:BallSports rdfs:subClassOf [ a owl:Restriction ;
  owl:onProperty :requires ;
  owl:someValuesFrom :Ball .
  ] .`

- All sports for which a ball is required are ball sports
  
  `:BallSports owl:equivalentClass [ a owl:Restriction ;
  owl:onProperty :requires ;
  owl:someValuesFrom :Ball .
  ] .`

- Where is the difference?
Further Examples for Restrictions

• Given:

```plaintext
:BallSports owl:equivalentClass [ 
a owl:Restriction ; 
owl:onProperty :requires ; 
owl:someValuesFrom :Ball . 
] .

:Soccer :requires :soccerBall . 
:soccerBall a :Ball.
```

• A reasoner may conclude that soccer is a ball sports
• This would not work with subClassOf
• Caveat: gymnastics with a ball are also recognized as ball sports...
Qualified Restrictions in OWL2

- In OWL, cardinalities and value restrictions may not be combined
- i.e., use either all/someValuesFrom or min/maxCardinality
- OWL 2 introduces *qualified restrictions*

- Example: a literate person has to have read at least 1,000 books (newspapers and magazines do not count!)

```
:LiteratePerson rdfs:subClassOf [a owl:Restriction;
            owl:onProperty :hasRead;
            owl:minQualifiedCardinality "1000"^^xsd:integer;
            owl:onClass :Book . ] .
```

Analogously, there are also
```
owl:maxQualifiedCardinality and owl:qualifiedCardinality
```
Using Restriction Classes as Ranges

- Restrictions can also be used in other contexts
- Example: books, newspapers, and posters can be read
  - essentially: everything that contains letters

- Range of the predicate `reads`:

  ```
  :reads rdfs:range [ 
  a owl:Restriction ;
  owl:onProperty :containsLetter ;
  owl:minCardinality 1^^xsd:integer .
  ] .
  ```
Using Restrictions as Domains

• If it works for ranges, it also works for domains
• e.g.: to think about something, a brain is required

• Domain of the \textit{thinksAbout} property:
  
  \begin{verbatim}
  :thinksAbout rdfs:domain [  
    a owl:Restriction ;  
    owl:onProperty :hasBodyPart ;  
    owl:someValuesFrom :Brain .  
  ] .
  \end{verbatim}

• Note: only in OWL DL/Full
Nesting Restrictions

• It is always possible to make things more complex
• e.g.: grandparents have children who themselves have at least one child

:GrandParent owl:equivalentClass [  
a owl:Restriction ;  
owl:onProperty :hasChild ;  
owl:someValuesFrom [  
    a owl:Restriction ;  
    owl:onProperty :hasChild ;  
    owl:minCardinality 1^^xsd:integer .
  ] .
] .
Web Ontology Language (OWL)

• What we have seen up to now
  – the vocabulary of OWL Lite
  – useful in many cases
  – "A little semantics goes a long way."
• OWL DL and OWL Full are more powerful
  – but also harder to handle
OWL DL

• DL stands for "Description Logics"
  – a subset of first order logics
  – we will get back to that next week

• OWL DL introduces
  – the full set of cardinality restrictions (OWL Lite allows only 0 and 1)
  – more set operators
  – closed classes
  – value based restrictions
  – restrictions on datatypes
  – ...

Complex Set Definitions

- **Set union**
  
  ```
  :FacultyMembers owl:unionOf (:Students, :Professors) .
  ```

- **Complement set**
  
  ```
  ```

- **Disjoint sets**
  
  ```
  ```
Previously on “Semantic Web Technologies”

• Let's look at that sentence:
  – "Madrid is the capital of Spain."

• We can get the following information:
  – "Madrid is the capital of Spain." ✔
  – "Spain is a state." ✔
  – "Madrid is a city." ✔
  – "Madrid is located in Spain." ✔
  – "Barcelona is not the capital of Spain." ✔
  – "Madrid is not the capital of France." ✔
  – "Madrid is not a state." ❌
  – ...
Previously on “Semantic Web Technologies”

- "Madrid is not a state." ✗
- Why not?
  - Madrid is a city
  - Nothing can be a city and a state at the same time.

- In OWL:

```owl
:Madrid a :City .
:City owl:disjointWith :State .

ASK { :Madrid a :State . } → false
```
Complex Set Definitions

- We can combine class definitions and restrictions:

```prolog
:VegetarianRecipe rdfs:subClassOf [ 
a owl:Restriction ;
owl:onProperty :hasIngredient ;
owl:allValuesFrom [ 
a owl:Class .
owl:complementOf [ 
owl:unionOf (:Meat :Fish) 
] 
]
]
].
```
A Tale from the Road

- ALIS: EU funded research project (2006-2009)
- Automated Legal Intelligent System
  - automatic search for relevant European laws
  - given a legal case at hand
  - using ontologies, reasoning, etc.
  - use case: copyright law
A Tale from the Road

• One important differentiation (among others):
  – Single Author Work
  – Multi Author Work

http://geekandpoke.typepad.com/geekandpoke/2006/10/copyright_and_a.html
A Tale from the Road

- Naive Solution in OWL DL:

  ```owl
  :hasAuthor a owl:ObjectProperty;
  rdfs:domain :Work ;
  rdfs:range :Author .

  :SingleAuthorWork rdfs:subClassOf
   :Work,
   [ a owl:Restriction;
     owl:onProperty :hasAuthor ;
     owl:cardinality 1^^xsd:integer ] .

  :MultiAuthorWork rdfs:subClassOf
   :Work,
   [ a owl:Restriction;
     owl:onProperty :hasAuthor ;
     owl:minCardinality 2^^xsd:integer ] .
  ```
A Tale from the Road

• Result:
  – not such a good idea
  – why not?

http://geekandpoke.typepad.com/geekandpoke/2006/10/copyright_and_a.html
A Tale from the Road

• Given
  
  :DataMining :hasAuthor :IanWitten, :EibeFrank .

• what can we derive from that?

• OK, so we need

  :DataMining :hasAuthor :IanWitten, :EibeFrank .
  :IanWitten owl:differentFrom :EibeFrank .

  → :DataMining a :MultiAuthorWork .
A Tale from the Road

• Given:
  
  :Faust :hasAuthor :Goethe .

• what can we derive from that?

• Since it worked for Multi Author Work, how about
  
  :Work owl:disjointUnionOf
  (:SingleAuthorWork, :MultiAuthorWork) .

  ?

• Note: we can classify :Faust neither as Single nor as Multi Author Work
Recap: Principles of RDF

• Basic semantic principles of RDF:
  – AAA: Anybody can say Anything about Anything

• Non-unique name assumption
  – we can control it with owl:sameAs and owl:differentFrom

• Open World Assumption
  – so far, we have to live with it
Closed Classes

• The Open World Assumption says:
  – everything we do not know *could* be true

• Example:
  
  :Tim a :PeopleInOfficeD219 .
  :John a :PeopleInOfficeD219 .
  :Mary a :PeopleInOfficeD219 .

• This does not mean that there cannot be more people in D219
  
  :Mike a :PeopleInD219 .

• Sometimes, this is exactly what we want to say
Closed Classes

• Works with `owl:oneOf` in OWL DL

• Example:

  `:PeopleInOfficeD219 owl:oneOf (:Tim :John :Mary) .`

• Now, what is the meaning of

  `:Mike a :PeopleInD219 .`

  ?
Solution:

```
:Faust a [ a owl:Restriction ;
  owl:onProperty :hasAuthor ;
  owl:allValuesFrom [
    a owl:Class ;
    owl:oneOf (:Goethe)
  ]
].
```
OWL DL: Restrictions with Single Values

• For ObjectProperties:

   :AfricanStates owl:subClassOf [ 
     a owl:Restriction ;
     owl:onProperty :locatedOnContinent
     owl:hasValue :Africa ] .

• For DatatypeProperties:

   :AlbumsFromTheEarly80s owl:subClassOf [ 
     a owl:Restriction ;
     owl:onProperty :year
     owl:dataRange
     (1980^^xsd:integer
     1981^^xsd:integer
OWL Lite/DL vs. OWL Full

- OWL Lite/DL: a resource is either an instance or a class or a property
- OWL Full does not have such restrictions:
  
  ```
  :Elephant a owl:Class .
  :Elephant a :Species .
  :Elephant :livesIn :Africa .
  :Species a owl:Class .
  ```

- OWL Lite/DL: classes are only instances of `owl:Class`
- OWL Lite/DL: classes and properties can only have a predefined set of relations (e.g., `rdfs:subClassOf`).
And Now for Something Completely Different

• Can we use OWL to solve a Sudoku?
Sudoku Solving in OWL

• What is our domain about?

• First of all, a closed class of numbers

  :Number a owl:Class ;
  ...

• ...and a lot of fields
  – that we want to fill with numbers
  – simplification: numbers are fields as well
  – we want to know which field equals which number
Sudoku Solving in OWL

- **81 Fields:**

  
  \[
  \begin{align*}
  c_{1\_11} & \text{ a :Number .} \\
  c_{1\_21} & \text{ a :Number .} \\
  \vdots & \\
  c_{1\_33} & \text{ a :Number .} \\
  c_{2\_11} & \text{ a :Number .} \\
  \vdots & \\
  c_{9\_33} & \text{ a :Number .}
  \end{align*}
  \]

<table>
<thead>
<tr>
<th></th>
<th>c1_11</th>
<th>c1_12</th>
<th>c2_11</th>
<th>c2_12</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1_21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c4_11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sudoku Solving in OWL

• Fields in a quadrant are different

\[
\begin{align*}
c_{1\_11} & \text{ owl:differentFrom } c_{1\_12}, c_{1\_13}, \ldots, c_{1\_33}. \\
c_{1\_12} & \text{ owl:differentFrom } c_{1\_13}, c_{1\_21}, \ldots, c_{1\_33} \\
& \quad \ldots \\
c_{1\_32} & \text{ owl:differentFrom } c_{1\_33} . \\
c_{2\_11} & \text{ owl:differentFrom } c_{2\_12}, c_{2\_13}, \ldots, c_{1\_33} \\
& \quad \ldots \\
c_{9\_32} & \text{ owl:differentFrom } c_{9\_33} .
\end{align*}
\]
Sudoku Solving in OWL

- Fields in a row are different
  
  \[ \text{c1}_{11} \ \text{owl:differentFrom} \]
  
  \[ \text{c1}_{12}, \ \text{c1}_{13}, \ldots, \ \text{c3}_{13} \ . \]

...
Sudoku Solving in OWL

- Fields in a column are different

```
c1_11 owl:differentFrom
  c1_21, c1_31, ..., c3_31.
```

...
Sudoku Solving in OWL

• Last step: enter known numbers

\[
c1_{11} \text{ owl:sameAs } 5. \\
c1_{12} \text{ owl:sameAs } 3. \\
c1_{21} \text{ owl:sameAs } 6. \\
\ldots
\]
Running this Example in Protégé

• We use a reasoner to infer implicit facts
• Here: number c_11 (top left)

Inferred: this is a 3

Defined conditions (horizontal, vertical, square)
Summary

• OWL allows defining more complex ontologies
• Flavors: OWL Lite, DL, Full
• Definitions of sets, restrictions, property characteristics
• In our example, we can now use the full set of conclusions:
  – "Barcelona is not the capital of Spain." ✔
  – "Madrid is not the capital of France." ✔
  – "Madrid is not a state." ✔
Coming Up Next

• Changes in OWL 2
• How does reasoning with OWL actually work?
Questions?