Open Information Systems

<u>Lecture 4: Web Ontology Language (OWL)</u>

Christophe Debruyne

Preamble

- In the previous lecture, we covered Description Logics. Description Logics provide the foundation of the Web Ontology Language.
- This lecture will focus on the syntax and capabilities of the Web Ontology Language.

OWL

- RDF(S) too weak to describe resources in sufficient detail
  - No localized range and domain constraints.
    - E.g., cannot state that the range of hasChild is person when applied to persons and elephant when applied to elephants
  - No combination of classes with union, intersection, and complement.
  - No existence/cardinality constraints.
    - E.g., cannot state that all persons have a mother that is also a person, or that persons have exactly 2 parents
  - No transitive, inverse or symmetrical properties.
    - E.g., cannot state that isPartOf is a transitive property, that hasPart is the inverse of isPartOf or that touches is symmetrical
  - …

- Difficult to provide reasoning support for predicates beyond the RDF and RDFS namespace as there is not formal foundation on which you can build those.

Example

Dogs and cats can have names. Is the following correct?

```
:name a rdf:Property ;
  rdfs:domain :Dog ;
  rdfs:domain :Cat ;
  rdfs:range xsd:string ;
  .
```

No. Why not?

  If something plays the role of having a name, then it is an instance of both :Dog and :Cat. In other words: multiple domain/range declarations act as a conjunction.

Can we model something that represents Dogs UNION Cats?

  In RDFS only by using superclasses, which may be not elegant.

Remember …

- Why do we need reasoning?
  - Ensure that a knowledge base (KB) is
    - Meaningful - all named classes can have instances
    - Correct - captured intuitions of domain experts
    - Minimally redundant - no unintended synonyms

- Answer queries over ontology classes and instances, e.g.:
  - Find more general and specific classes
  - Retrieve individuals or tuples matching a given query

Web Ontology Language (OWL)

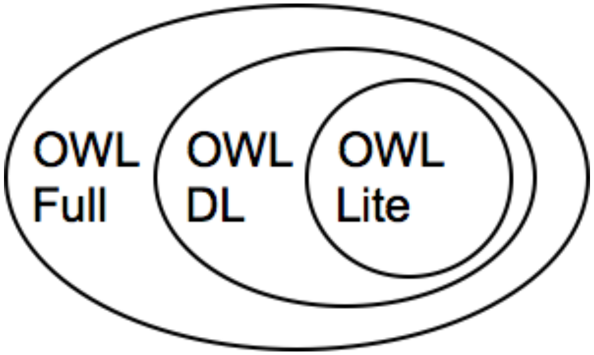
- Ontology languages SHOULD allow users to provide an explicit, formal conceptualization of a domain of discourse

- OWL is a W3C Recommendation
  - OWL was published in 2004
  - OWL 2 was published in 2012

- Motivations:
  - A well-defined syntax
  - A formal semantics
  - Efficient reasoning support

OWL 1 “flavors”

- W3C’s Web Ontology Working Group defined OWL as three different sublanguages:
  - OWL Full
  - OWL DL
  - OWL Lite



- Each sublanguage geared toward fulfilling different aspects of requirements

OWL Full

- It uses all the OWL languages primitives
- It allows the combination of these primitives in arbitrary ways with RDF
  - OWL Full is fully upward-compatible with RDF both syntactically and semantically
- OWL Full is so powerful that it is **undecidable**
  - No complete reasoning support

OWL DL

- OWL DL (Description Logics)
  - Based on FOL semantics
  - Disallow certain way of applying constructors
  - Corresponds to SHOIN(D) – see previous lecture

- OWL DL permits efficient reasoning support
  - The most expressive decidable OWL sub-language

- But we lose full compatibility with RDF:
  - Every legal OWL DL document is a legal RDF document.
  - Not every RDF document is a legal OWL DL document.

OWL Lite

- An even further restriction limits OWL DL to a subset of the language constructors in a syntactic way (e.g., SHIF(D))
  - E.g., OWL Lite excludes enumerated classes, disjointness statements, and arbitrary cardinality.
  - DL-Lite are families of DL dialects that are tractable

- However, the complexity is not far away from OWL DL

OWL 2

- W3C Recommendation since December 2012
- Profiles
  - OWL 2 EL: Polynomial time algorithms for standard reasoning tasks. Based on the description logic $\mathcal{EL}$ with support for existential restrictions and conjunction. Suitable for ontologies with a large number of properties and classes.
  - OWL 2 QL: Conjunctive queries answered in LogSpace. Suitable for querying knowledgebases with a large number of instances. Name comes from the relation between the profile and ER.
  - OWL 2 RL: Rule-based reasoning over large large numbers of individuals (polynomial time algorithms). Name comes from the fact that all reasoning tasks within this profile can be implemented using standard rule languages.
  - **Each profile prescribes the constructs you are allowed to use.**

OWL DL Syntax

- Lecturer ≡ Person ⊓ ¬MScStudent
- RDF/XML

```
<owl:Class rdf:ID="Lecturer">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Person"/>
        <owl:Class>
          <owl:complementOf rdf:resource="#MScStudent"/>
        </owl:Class>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

Abstract syntax

```
EquivalentClasses(ex:Lecturer
  ObjectIntersectionOf(ex:Person ObjectComplementOf(ex:MScStudent)))
```

Namespaces

```
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY star "http://starlab.vub.ac.be/example/#" >
]>

 <rdf:RDF xml:base="http://example.com/owl/families/"
   xmlns="http://starlab.vub.ac.be/starexamples/"
   xmlns:star="http://starlab.vub.ac.be/example/#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
...
```

OWL DL Declarations and Typings

- **Class axioms**
  - Declaration (Class(:Student))
  - SubClassOf(:Professor :Staff)
  - EquivalentClasses(:Staff :Employee)
  - DisjointClasses(:Professor :Student)

- **Property Axioms**
  - Declaration (ObjectProperty(:teaches))
  - ObjectPropertyDomain(:teaches :Lecturer)
  - ObjectPropertyRange(:teaches :Course)
  - Declaration (DatatypeProperty(:age))
  - DatatypePropertyRange(:age xsd:string)

- **Individual Axioms**
  - Declaration (NamedIndividual(:Debruyne))
  - ClassAssertion(:Professor :Debruyne)
  - ObjectPropertyAssertion(:teaches :Debruyne :OIS)

Class Axioms

```
<owl:Class rdf:ID="Person">

<owl:Class rdf:ID="Muggle">
  <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>

<Muggle rdf:ID="Dudley"/>
```

```
<owl:Class rdf:ID="Human">

<owl:Class rdf:ID="Person">
  <owl:equivalentClass rdf:resource="#Human"/>
</owl:Class>
```

Class Axioms

```
<owl:Class rdf:ID="Muggle">
  <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>

<owl:Class rdf:ID="Wizard">
  <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
```

```
<owl:AllDisjointClasses>
  <owl:members rdf:parseType="Collection">
    <owl:Class rdf:about="#Muggle"/>
    <owl:Class rdf:about="#Wizard"/>
  </owl:members>
</owl:AllDisjointClasses>
```

OWL DL Class Descriptions

- **ObjectIntersectionOf(classexpression+)** #CE
  - ObjectIntersectionOf(:Professor :Lecturer)
- **ObjectUnionOf(CE +)**
  - ObjectUnionOf(:Male :Female :X)
- **ObjectComplementOf(CE)**
  - ObjectComplementOf(:Student)
- **ObjectOneOf(namedindividual+)**
  - ObjectOneOf(:DatabaseTheory :OIS :IS)
- **DataOneOf(literal+)**
  - DataOneOf("Victor" "Bettina" "Gaston" "1"^^xsd:integer)

OWL DL Class Descriptions

```
<owl:Class rdf:ID="Parent">
    <owl:equivalentClass>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Mother"/>
          <owl:Class rdf:about="#Father"/>
        </owl:unionOf>
      </owl:Class>
    </owl:equivalentClass>
    <owl:equivalentClass>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasChild"/>
        <owl:someValuesFrom rdf:resource="#Person"/>
      </owl:Restriction>
    </owl:equivalentClass>
 </owl:Class>
```

What is being declared?

- $Parent \equiv Mother \sqcup Father$
- $Parent \equiv \exists hasChild.Person$

How would it look like in the functional syntax?

Property Axioms

```
<owl:ObjectProperty rdf:ID="hasWife">
  <rdfs:subPropertyOf rdf:resource="#hasSpouse"/>
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Woman"/>
</owl:ObjectProperty>
<owl:SymmetricProperty rdf:about="#hasSpouse"/>
```

```
<owl:ObjectProperty rdf:ID="hasSon">
  <owl:propertyDisjointWith rdf:resource="#hasDaughter"/>
</owl:ObjectProperty>
```

```
<owl:DatatypeProperty rdf:ID="hasAge">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdfs:Datatype="&xsd;nonNegativeInteger"/>
  <owl:equivalentProperty rdf:resource="&star;age"/>
</owl:DatatypeProperty>
```

OWL DL Restrictions

- **ObjectSomeValuesFrom(objectpropertyexp CE)** #OPE
  - ObjectSomeValuesFrom(:teaches :Course)
- **ObjectAllValuesFrom(OPE CE)**
  - ObjectAllValuesFrom(:eat :Plant)
- **ObjectMinCardinality(OPE non-negative-integer)**
  - ObjectMinCardinality(:hasFinger 5)
- **ObjectMaxCardinality(OPE non-negative-integer)**
  - ObjectMaxCardinality(:hasFinger 5)
- **ObjectExactCardinality(OPE non-negative-integer)**
  - ObjectExactCardinality(:hasFinger 5)

OWL DL Restrictions

```
<owl:Class rdf:ID="HappyFriend">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasFriend"/>
          <owl:allValuesFrom rdf:resource="#HappyFriend"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasFriend"/>
          <owl:someValuesFrom rdf:resource="#HappyFriend"/>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

$$HappyFriend \equiv \forall hasFriend.HappyFriend \sqcap \exists hasFriend.HappyFriend$$

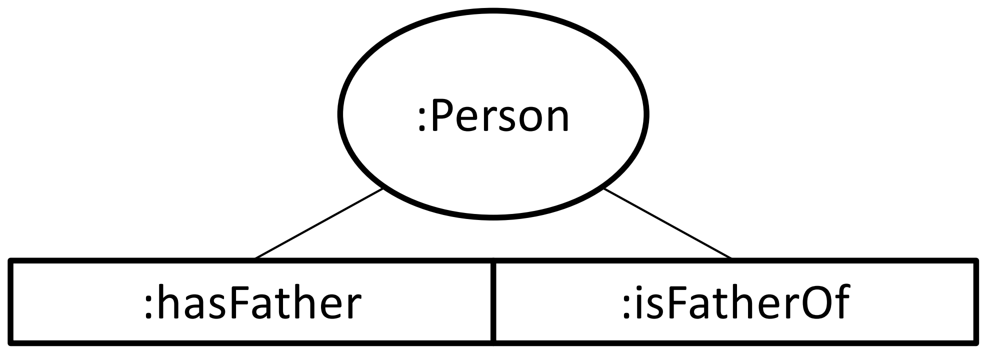
OWL DL Object Property Axioms

- **ObjectProperty(OPE)**
  - ObjectProperty(:primarilyTeach)
  - SubObjectPropertyOf(:primarilyTeach :Teach)
- **InverseObjectProperties(OPE OPE)**
  - InverseObjectProperties(:partOf :hasPart)
- **ObjectPropertyDomain(OPE CE)**
  - ObjectPropertyDomain(:teach :Lecturer)
- **ObjectPropertyRange(OPE CE)**
  - ObjectPropertyRange(:teach :Course)

OWL DL Object Property Axioms

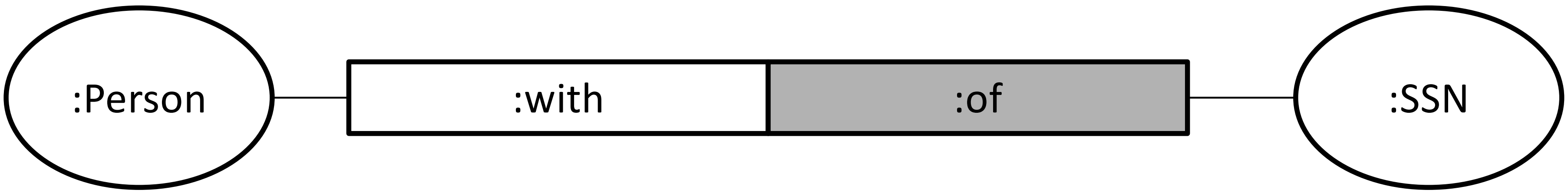
- **TransitiveObjectProperty(OPE)**
  - TransitiveObjectProperty(:locatedIn)
- **SymmetricObjectProperty(OPE)**
  - SymmetricObjectProperty(:adjacentRegion)
- **FunctionalObjectProperty(OPE)**
  - FunctionalObjectProperty(:hasFather)
- **InverseFunctionalObjectProperty(OPE)**
  - InverseFunctionalObjectProperty(:isFatherOf)

OWL DL Object Property Axioms



- FunctionalObjectProperty(:hasFather) → Each person can have at most one father. We state something about the role :hasFather.
- InverseFunctionalObjectProperty(:isFatherOf) → Each person can have at most one father, but we stated it by using the inverse of :isFatherOf. In other words: “the inverse of :isFatherOf is functional”

OWL DL Object Property Axioms



- Assume that People have SSN, and that an SSN is a class (not a literal). Also assume we only declare the property from `:Person` to `:SSN`.

- We do not state that `:of` is a property *and* therefore cannot state that `:of` is the the inverse of `:with`.

- People have at most one SSN, so we can declare:
  FunctionalObjectProperty(:with)

- An SSN is also related with at most one Person, but we do not have declared that property. How do we make that role functional?
  Via `:with` by means of InverseFunctionalObjectProperty(:with)
  → The inverse of `:with`, for which we did not provide a name, is functional.

OWL DL Object Property Axioms

```
<owl:SymmetricProperty rdf:about="#hasSpouse"/>

<owl:AsymmetricProperty rdf:about="#hasChild"/>

<owl:ReflexiveProperty rdf:about="#hasRelative"/>

<owl:IrreflexiveProperty rdf:about="#parentOf"/>

<owl:FunctionalProperty rdf:about="#hasHusband"/>

<owl:InverseFunctionalProperty rdf:about="#hasHusband"/>

<owl:TransitiveProperty rdf:about="#hasAncestor"/>
```

OWL DL Semantics and Reasoning

- See previous lecture … ☺
- Class Satisfiability Checking
  - Given the following ontology O – with some abuse of functional notation:
    - Class (:Cat complete intersectionOf(:Animal complementOf(:Dog))
    - Class(:Dog partial :Person)
    - Class(:CatDog partial :Cat)
    - Class(:CatDog partial :Dog)
  - CatDog is unsatisfiable as
    - CatDog $\sqsubseteq$ Animal $\sqcap$ $\neg$Dog $\sqcap$ Dog

OWL 1 vs. OWL 2

- Syntactic Sugar
  - **DisjointUnion(Class CE+)**
    - DisjointUnion(:CarDoor :FrontDoor :ReadDoor :TrunkDoor)
  - **DisjointClasses(C1 … Cn)**
    - DisjointClasses(:Chinese :Belgian :Scottish)
  - **NegativeObjectPropertyAssertion(Property a1 a2)**
    - NegativeObjectPropertyAssertion(:livesIn :Christophe :Russia)
  - **NegativeDataPropertyAssertion(Property a l)**
    - NegativeDataPropertyAssertion(:hasAge :ThisPatient 5^^xsd:integer)

OWL 1 vs. OWL 2

- New constructs
  - **ObjectHasSelf(OPE)**
    - SubClassOf(:AutoRegulatingProcess ObjectHasSelf(:regulate))
    - *Auto-regulating processes are things that regulate themselves*
  - **Object[Min|Max|Exact]Cardinality(n Property C)**
    - ObjectExactCardinality(3 :hasDoor :Car)
    - *Similar for data properties exist*
  - **DisjointObjectProperties(OPE1 … OPEn)**
    - DisjointObjectProperties(:connectedTo :contiguousWith)
    - *There exist similar constructs for data properties*
  - **ObjectPropertyChain(OPE1 … OPEn)**
    - SubPropertyOf(ObjectPropertyChain(:locatedIn :partOf) :locatedIn)
  - **HasKey(C (OPE1 … OPEn) (DPE1 … DPEn))**
    - HasKey(:Transplantation :donorId :recipientId :ofOrgan)

Small Exercise

- EACH Person has EXACTLY 1 First Name .
- EACH Person has EXACTLY 1 Last Name .
- EACH Person x IS IDENTIFIED BY First Name of x AND Last Name of x .
- EACH Person IS born in AT MOST ONE Country .
- Also model the relation from Country to Person (e.g., :birthplaceOf)

```
Prefix(:=<http://www.example.org/foo/bar#>)
Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
Prefix(rdf:=<http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
Prefix(xml:=<http://www.w3.org/XML/1998/namespace>)
Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
Prefix(rdfs:=<http://www.w3.org/2000/01/rdf-schema#>)

Ontology(<http://www.example.org/foo/bar>
    Declaration(Class(:Country))
    Declaration(Class(:Person))
    Declaration(ObjectProperty(:birthplaceOf))
    Declaration(ObjectProperty(:bornIn))
    Declaration(DataProperty(:firstName))
    Declaration(DataProperty(:lastName))

    FunctionalDataProperty(:firstName)
    DataPropertyDomain(:firstName :Person)
    DataPropertyRange(:firstName xsd:string)

    FunctionalDataProperty(:lastName)
    DataPropertyDomain(:lastName :Person)
    DataPropertyRange(:lastName xsd:string)

    FunctionalObjectProperty(:bornIn)
    ObjectPropertyDomain(:bornIn :Person)
    ObjectPropertyRange(:bornIn :Country)
    InverseObjectProperties(:birthplaceOf :bornIn)

    SubClassOf(:Person DataExactCardinality(1 :firstName owl:rational))
    SubClassOf(:Person DataExactCardinality(1 :lastName owl:rational))

    HasKey(:Person () (:firstName :lastName))
)
```

If you're a person, you have at most one first/last name. Other UoDs can be considered. Remember the tension field between a conceptual model for an IS and an ontology.

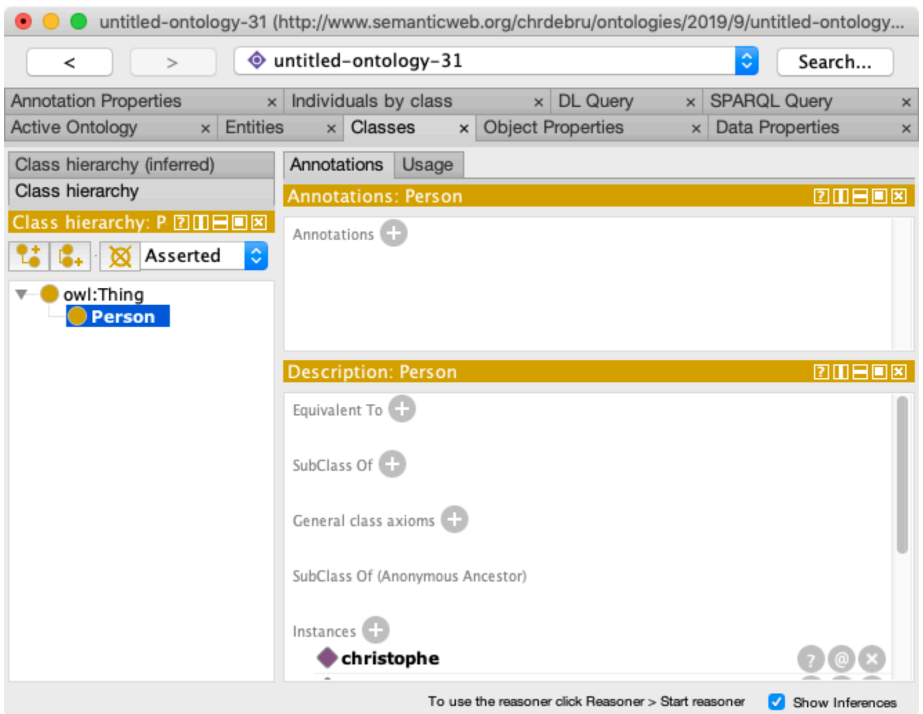
References

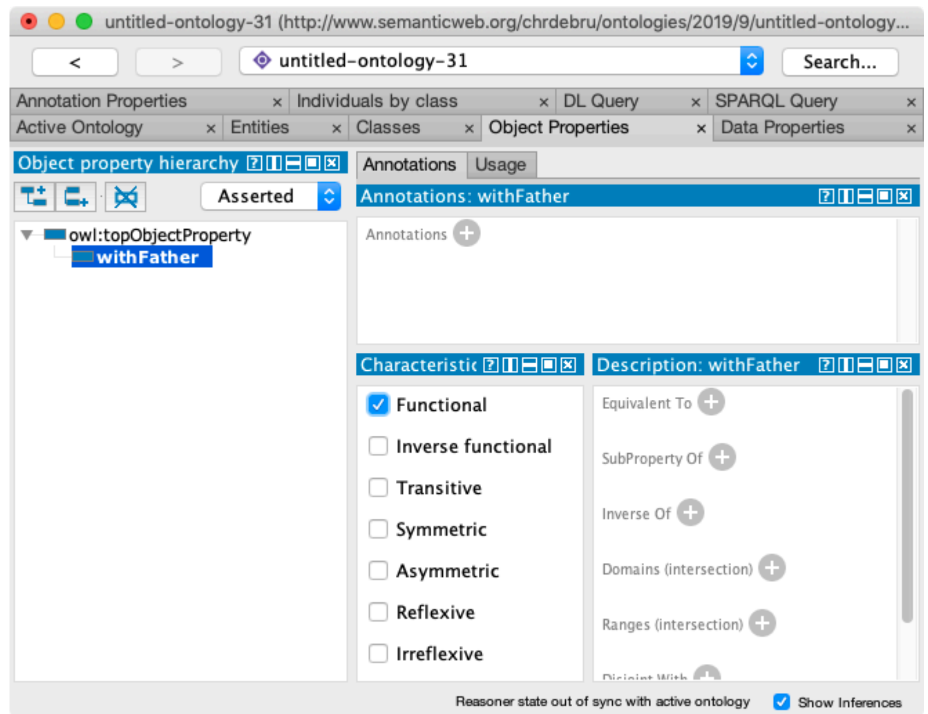
- Grigoris Antoniou, Frank van Harmelen: Web Ontology Language: OWL. Handbook on Ontologies 2009: 91-110
- Web Ontology Language (OWL) 1.0
  - http://www.w3.org/2001/sw/WebOnt/
- OWL 2.0
  - http://www.w3.org/TR/2009/WD-owl2-new-features-20090421/
  - https://www.w3.org/TR/owl2-syntax/

Demonstrating FunctionalObjectProperty with Protégé

1) Create a class Person

2) Create an ObjectProperty withFather and make it functional

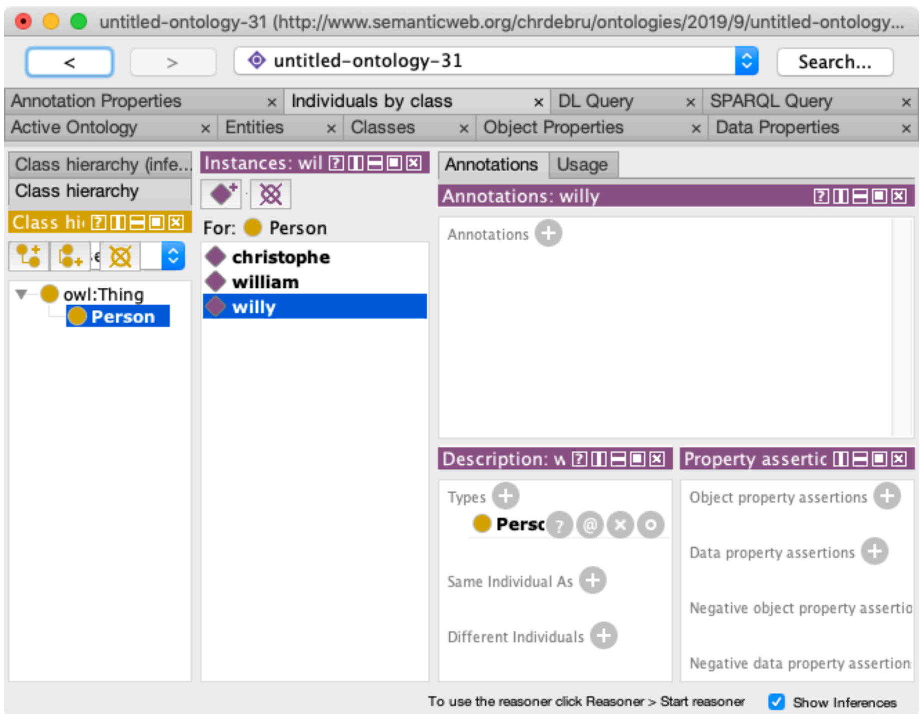


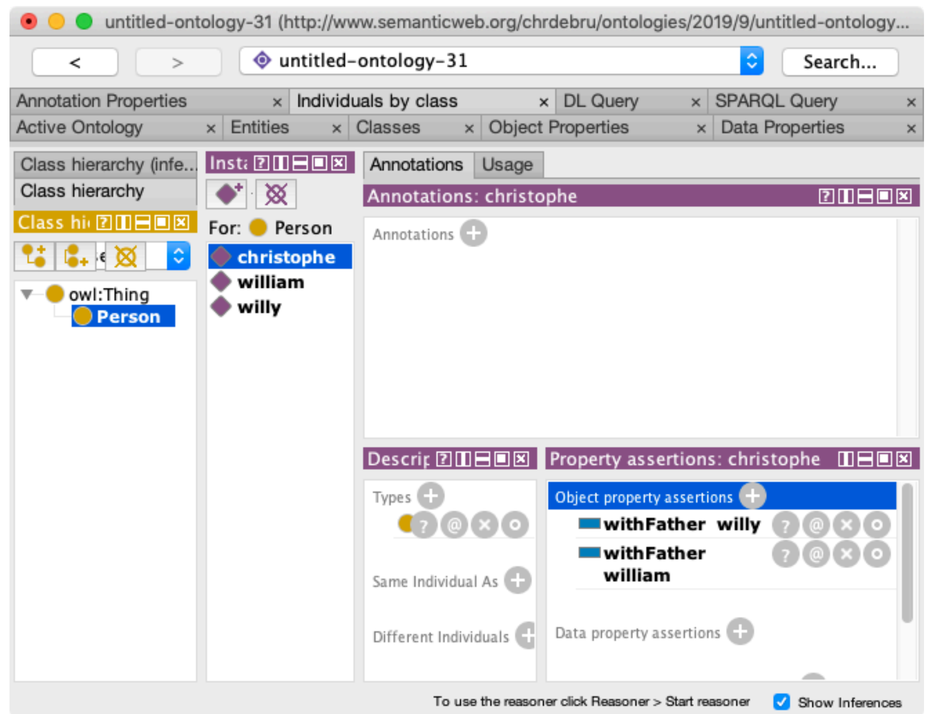


Demonstrating FunctionalObjectProperty with Protégé

3) Create three individuals of the class Person: christophe, willy, and william

4) state that willy and william are the fathers of christophe

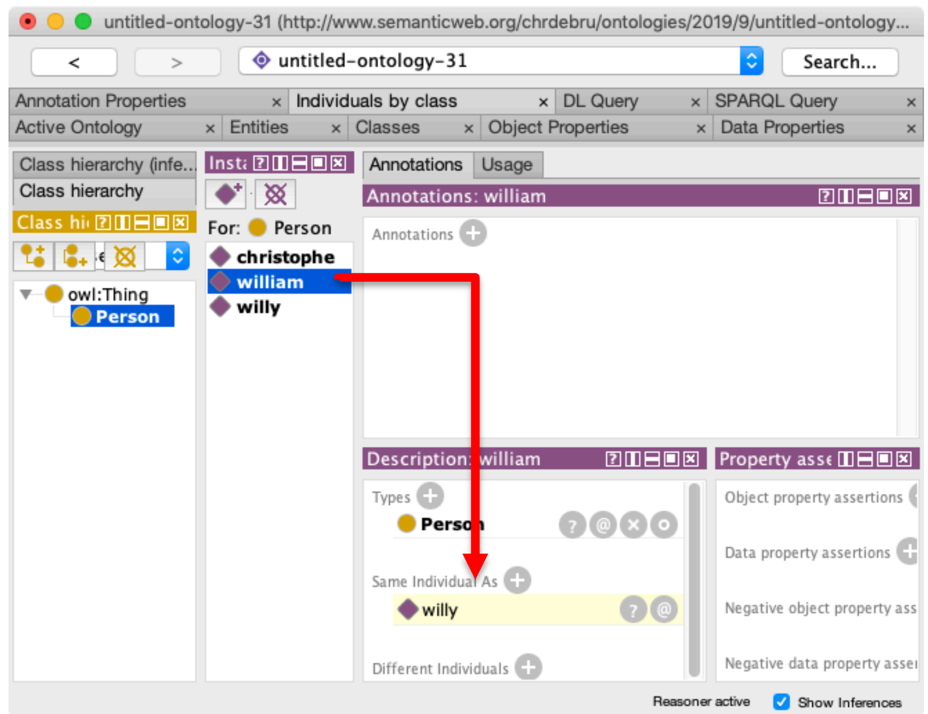




Demonstrating FunctionalObjectProperty with Protégé

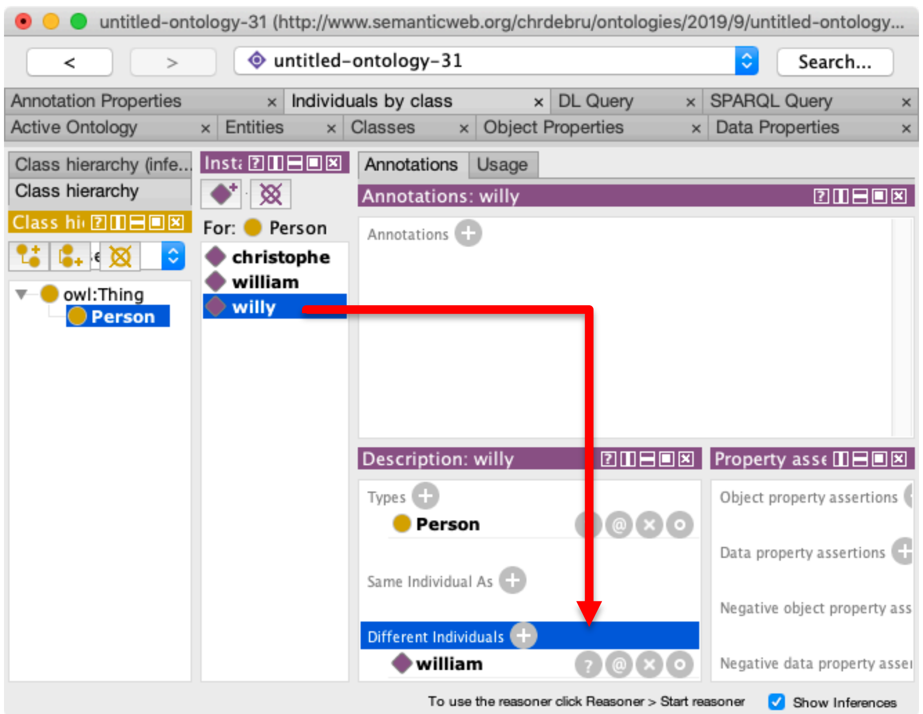
5) Run the reasoner

- Menu "Reasoner" → "Start Reasoner"
- Now click on the individuals willy and william, and you will notice that they are inferred to be referring to the same "thing"



Demonstrating FunctionalObjectProperty with Protégé

6) Now state that willy and william are different (first stop the reasoner)



7) Re-run the reasoner

"An error occurred during reasoning: Cannot do reasoning with inconsistent ontologies! Reason for inconsistency: Individual http://...#christophe has more than one value for the functional property http://...#withFather."