

# INFO216: Advanced Modelling

Theme, spring 2017:  
**Modelling and Programming  
the Web of Data**

Andreas L. Opdahl  
<Andreas.Opdahl@uib.no>



# Session S03: RDF Schema (RDFS)

- Themes:
  - *why RDFS?*
  - *utility properties*
  - *classes and subclasses*
  - *properties and subproperties*
  - *entailments and axioms*
  - *overview*
  - *motivation for OWL*



# Readings

- Allemang & Hendler (2011):  
Semantic Web for the Working Ontologist.
  - chapters 6 and 7
- Electronic materials in the wiki:
  - [wiki.uib.no/info216](http://wiki.uib.no/info216)



# RDF Schema (RDFS) (RDF Vocabulary Description Language)



# From RDF to RDFS

- RDF is a good start
  - excellent normal form for basic facts
  - less suitable for complex concept systems
    - e.g., vocabularies, ontologies
- RDF Schema (RDFS):
  - small RDF vocabulary for more expressive graphs
  - conventional prefix:
    - `rdfs:` <http://www.w3.org/2000/01/rdf-schema#>
  - many vocabularies defined in plain RDFS
    - also the foundation for SKOS, OWL and OWL2



# Why RDF Schema (RDFS)

- *More expressive RDF graphs*, for example:
  - resources can have different types
  - predicates can have different types
  - which types of resources
    - are the subject of which properties?
    - are the object of which properties?
  - additional triples that are *entailed by* our data set?
  - predefined *axioms*
- *RDFS is expressed in RDF*
  - a small extension of the basic RDF vocabulary
  - RDF tools can be used on RDFS
    - ...but the semantics may get lost



# Old friends...

- We have covered some RDFS-related stuff already:
  - instances: `rdf:type`, `rdf:Property`
    - ...belongs to “plain RDF” (*why?*)
  - collections: `rdf:List` `rdf:first` `rdf:rest` `rdf:nil`
    - ...belongs to “plain RDF”
  - containers: `rdf:Alt` `rdf:Bag` `rdf:Seq` `rdfs:Container`  
`rdf:_1` `rdf:_2` ... `rdfs:member`  
`rdfs:ContainerMembershipProperty`
  - reified triples
  - `rdf:value`
  - utilities: `rdfs:label`, `rdfs:comment`, `rdfs:seeAlso`...



# Utility properties

- *Straightforward and much used:*
    - rdfs:label:
      - a human-readable label
    - rdfs:comment:
      - a human-readable comment
    - rdfs:seeAlso:
      - reference to further information
    - rdfs:isDefinedBy:
      - a human-readable definition
      - is a rdfs:subPropertyOf rdfs:seeAlso
- ...often take “language-tagged”@en strings as objects*





# Resource classes (categories, types...)

- Classes are resources that represent a type of similar resources, which are the individuals in the class
  - e.g., `dbpedia:Person`, `schema:Person`, `foaf:Person`
  - belonging to a class is expressed by `rdf:type` (**a**):  
`<RDF individual> rdf:type <RDFS class> .`  
`<RDF individual> a <RDFS class> .`
  - an individual can belong to several classes
  - an RDFS class is defined as a resource that is the object of an `rdf:type` predicate ( $\rightarrow$  *entailment*)
  - every RDFS class has `rdf:type rdfs:Class`
- *By convention, resources are named with an upper-case initial letter, properties with lower-case initial letters...*



# Resource classes (categories, types...)

- Why resource classes?
  - the type of a resource is an important part of its semantics
  - knowing the type of a resource often means we can infer additional information about it (**entailment**)
  - classes are important for defining and using other RDFS concepts



# Not (quite) like object-oriented programming

- *Typical* OOP:
  - classes are templates for instantiating objects
  - objects with fixed class
  - adding a property restricts a class
  - property encapsulation
  - property overriding
  - local value ranges:  
*“Tigers only have parents that are Tigers and Humans only have parents that are Humans”*
- RDFS *is different*:
  - the properties of a resource determines its class
  - resources change classes
  - adding a property enriches a resource
  - properties are global
  - no property overriding
  - no local value ranges  
(but you can extend RDFS with a rule language)



# Subclasses (rdfs:subClassOf)

- Whenever an individual resource belongs to some class, it necessarily belongs to another class too, e.g.,
  - `dbpedia:Physician rdfs:subClassOf dbpedia:Person` .
- Why subclasses?
  - arranging classes in subclass hierarchies makes their semantics more precise!
  - subclasses facilitate more complete query answering
  - knowing the type of a resource often means we can infer additional information about it (entailment)
  - subclasses are important because different vocabularies may define overlapping, but not identical, classes
    - introduce a new class in the merged data set
    - make the old classes subclasses of the new class



# RDFS entailment

- The meaning of `rdfs:subClassOf` and the other RDFS concepts is defined by *entailment rules*...
- Example: classical *syllogism*:
  - “*All men are mortal.*” (Major Premise)
  - “*Socrates is a man.*” (Minor Premise)

---

  - “*Socrates is a mortal.*” (Valid conclusion)



# RDFS entailment

- The meaning of `rdfs:subClassOf` and the other RDFS concepts is defined by *entailment rules*...
- Example: classical *sylllogism* in RDFS:
  - *ex:Man rdfs:SubclassOf ex:Mortal .* (Major Premise)
  - *ex:Socrates rdf:type ex:Man .* (Minor Premise)

---

  - *ex:Socrates rdf:type ex:Mortal .* (Valid conclusion)

Entailment means that *some triples are there in our RDFS models even when we have not asserted them*



# RDFS entailment

- The meaning of `rdfs:subClassOf` and the other RDFS concepts is defined by *entailment rules*...
- Example: *pattern* for classical syllogism in RDFS:
  - *?c1 rdfs:SubclassOf ?c2 . (Major Premise)*
  - *?s rdf:type ?c1 . (Minor Premise)*

---

  - *?s rdf:type ?c2 . (Valid conclusion)*

Entailment means that *some triples are there in our RDFS models even when we have not asserted them*



# RDFS entailment [rdfs9]

- The meaning of `rdfs:subClassOf` and the other RDFS concepts is defined by *entailment rules* [rdfs9]:
- “The triples `?s rdf:type ?c1 .`  
`?c1 rdfs:subClassOf ?c2 .`  
entail that `?s rdf:type ?c2 .`”

```
PREFIX rdf: <...>  
PREFIX rdfs: <...>
```

```
INSERT {  
    ?s rdf:type ?c2 .  
} WHERE {  
    ?s rdf:type ?c1 .  
    ?c1 rdfs:subClassOf ?c2 .  
}
```

Here, we express the rule using SPARQL.

Writing rules in RDF and TURTLE is called SPIN (SPARQL Inferencing Notation).

(RDFS does not *really* specify the rules in SPARQL/SPIN).





# What does entailment mean?

- Entailment means that *some triples are there in our RDFS models even when we have not asserted them*
  - full list at <http://www.w3.org/TR/rdf-mt/>
  - around 30 entailment rules in RDFS
- Different RDFS tools may support entailment rules in different ways, e.g.:
  - strategy 1: always add entailed triples when possible
  - strategy 2: only extract entailed triples when needed
  - in Jena, this work is done by inference engines:
    - an RDFS model (InfModel) is an RDF model with an RDFS inference engine (Reasoner) attached
    - *ModelFactory.createRDFSModel(baseModel);*



# Transitive properties

- `rdfs:subClassOf` is *transitive*:
  - `ex:Dolphins rdfs:subClassOf ex:Whales .`
  - `ex:Whales rdfs:subClassOf ex:Mammals .`

---

  - `ex:Dolphins rdfs:subClassOf ex:Mammals .`
- Entails new `rdf:type` triples about which classes an individual belongs to



# Transitive properties

- `rdfs:subClassOf` is *transitive*:
  - `?c1 rdfs:subClassOf ?c2 .`
  - `?c2 rdfs:subClassOf ?c3 .`

---

  - `?c1 rdfs:subClassOf ?c3 .`
- Entails new `rdf:type` triples about which classes an individual belongs to



# RDFS entailment [rdfs11]

- `rdfs:subClassOf` is *transitive*:
- “The triples  
    ?`c1` `rdfs:subClassOf` ?`c2` .  
    ?`c2` `rdfs:subClassOf` ?`c3` .  
entail that  
    ?`c1` `rdfs:subClassOf` ?`c3` .”

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
INSERT {  
    ?c1 rdfs:subClassOf ?c3 .  
} WHERE {  
    ?c1 rdfs:subClassOf ?c2 .  
    ?c2 rdfs:subClassOf ?c3 .  
}
```



# RDFS entailment [rdfs10]

- `rdfs:subClassOf` is also *reflexive*:
- “The triple `?c rdf:type rdfs:Class .`  
entails that `?c rdfs:subClassOf ?c .`”

```
PREFIX rdf: <...>
```

```
PREFIX rdfs: <...>
```

```
INSERT {  
    ?c rdfs:subClassOf ?c .  
} WHERE {  
    ?c rdf:type rdfs:Class .  
}
```



# Properties

- All predicates have *rdf:type rdf:Property*
  - this is expressed by an *entailment* (next slide!)
  - properties have *domains* and *ranges*, i.e., their subjects and objects belong to particular classes
  - properties can be *transitive*
- Why properties?
  - needed in RDF (along with *rdf:type*) to express that only certain resources act as predicates in triples
  - as for classes/subclasses:
    - clearer semantics, entailment, complete answers to queries and defining other concepts, *e.g.*,
    - *many classes are defined in terms of properties...*



# RDF entailment [rdf1]

- “The triple `?s ?p ?o .`  
entails that `?p rdf:type rdf:Property .`”

PREFIX rdf: <...>

```
INSERT {  
    ?p rdf:type rdf:Property .  
} WHERE {  
    ?s ?p ?o .  
}
```

- Resources *become properties* by being used as predicates in triples!



# Domain and range of properties

- The subjects and objects that occur in triples along with some property belong to certain classes
- Example:
  - *<subject>* ex:hasPassportNumber *<object>* .
  - when we see this triple, we know that:
    - the *<subject>* has rdf:type ex:Person
    - the *<object>* has rdf:type ex:PassportNumber
  - this is part of the semantics of ex:hasPassportNumber
  - ...can be expressed as follows:
    - ex:hasPassportNumber rdfs:domain ex:Person .
    - ex:hasPassportNumber rdfs:range ex:PassportNumber .





# RDFS entailment [rdfs2]

- “The triples `?s ?p ?o .`  
`?p rdfs:domain ?t .`  
entail that `?s rdf:type ?t .`”

PREFIX rdf: <...>

PREFIX rdfs: <...>

```
INSERT {  
    ?s rdf:type ?t .  
} WHERE {  
    ?s ?p ?o .  
    ?p rdfs:domain ?t .  
}
```



# RDFS entailment [rdfs3]

- “The triples `?s ?p ?o .`  
`?p rdfs:range ?t .`  
entail that `?o rdf:type ?t .`”

```
PREFIX rdf: <...>
```

```
PREFIX rdfs: <...>
```

```
INSERT {  
    ?o rdf:type ?t .  
} WHERE {  
    ?s ?p ?o .  
    ?p rdfs:range ?t .  
}
```



# RDFS axioms

- RDFS axioms:
  - triples that are “built into” RDFS
  - predefined in any RDFS graph
  - essential part of the semantics of RDFS
  - full list at <http://www.w3.org/TR/rdf-mt/>
  - 40 axioms and 3 axiom schemas
- Example axioms for *rdf:type*:
  - `rdf:type rdfs:range rdfs:Class .`
  - `rdf:type rdfs:domain rdfs:Resource .`



# RDFS entailment [rdfs3]

- “The triples `?s ?p ?o .`  
`?p rdfs:range ?t .`  
entail that `?o rdf:type ?t .`”

PREFIX rdf: <...>

PREFIX rdfs: <...>

```
INSERT {  
    ?o rdf:type ?t .  
} WHERE {  
    ?s ?p ?o .  
    ?p rdfs:range ?t .  
}
```

Remember:

`rdf:type rdfs:range rdfs:Class .`  
is an axiom in RDFS. This axiom  
fits straight into the rule:

`?p = rdf:type`

`?t = rdfs:Class`



# RDFS entailment [rdfs3 + axiom]

- “The triples `?s rdf:type ?o .`  
`rdf:type rdfs:range rdfs:Class .`  
entail that `?o rdf:type rdfs:Class .`”

```
PREFIX rdf: <...>  
PREFIX rdfs: <...>
```

```
INSERT {  
    ?o rdf:type rdfs:Class .  
} WHERE {  
    ?s rdf:type ?o .  
    rdf:type rdfs:range rdfs:Class .  
}
```

**Remember:**

`rdf:type rdfs:range rdfs:Class .`  
is an axiom in RDFS. This axiom  
fits straight into the rule!

?p = `rdf:type`  
?t = `rdfs:Class`

*This is an axiom in RDFS!*



# RDFS entailment

- “The triples `?s rdf:type ?o .`  
~~`rdf:type rdfs:range rdfs:Class .`~~  
entail that `?o rdf:type rdfs:Class .`”

PREFIX rdf: <...>

PREFIX rdfs: <...>

```
INSERT {  
    ?o rdf:type rdfs:Class .  
} WHERE {  
    ?s rdf:type ?o .  
    rdf:type rdfs:range rdfs:Class .  
}
```

Because

`rdf:type rdfs:range rdfs:Class .`  
is an axiom in RDFS, this rule  
entails that every object in an  
`rdf:type`-triple is an RDFS  
class.



# RDFS entailment

- “The triples `?s rdf:type ?o .`  
entail that `?o rdf:type rdfs:Class .`”

```
PREFIX rdf: <...>  
PREFIX rdfs: <...>
```

```
INSERT {  
    ?o rdf:type rdfs:Class .  
} WHERE {  
    ?s rdf:type ?o .  
}
```

This rule entails that every object in an `rdf:type`-triple is an RDFS class.



# Subordinate properties (rdfs:subPropertyOf)

- Expresses that: whenever a subject resource and an object resource are related by a particular property, they are necessarily also related by another property, e.g.,
  - whenever this is a fact:  
dbpedia:Alex\_Horwath ex:goalkeeperFor dbpedia:SK\_Brann .
  - then this is necessarily also a fact:  
dbpedia:Alex\_Horwath ex:playsFor dbpedia:SK\_Brann .
  - *because* we have defined a *subproperty relationship*:  
ex:goalkeeperFor rdfs:subPropertyOf ex:playsFor .
- Is useful for connecting overlapping properties from distinct data sets, e.g.:
  - move:actor\_name, movie:film\_name dc:name, rdfs:label
  - ...just like rdfs:subClassOf





# RDFS entailment [rdfs7]

- “The triples `?s ?p1 ?o .`  
`?p1 rdfs:subPropertyOf ?p2 .`  
entail that `?s ?p2 ?o .`”

PREFIX rdfs: <...>

```
INSERT {  
    ?s ?p2 ?o .  
} WHERE {  
    ?s ?p1 ?o .  
    ?p1 rdfs:subPropertyOf ?p2 .  
}
```



# RDFS entailment [rdfs5]

- `rdfs:subPropertyOf` is *transitive*:
- “The triples  
    ?`p1` `rdfs:subPropertyOf` ?`p2` .  
    ?`p2` `rdfs:subPropertyOf` ?`p3` .  
entail that   ?`p1` `rdfs:subPropertyOf` ?`p3` .”

PREFIX `rdfs:` <...>

```
INSERT {  
    ?p1 rdfs:subPropertyOf ?p3 .  
} WHERE {  
    ?p1 rdfs:subPropertyOf ?p2 .  
    ?p2 rdfs:subPropertyOf ?p3 .  
}
```



# RDFS entailment [rdfs6]

- `rdfs:subPropertyOf` is *reflexive*:
- “The triple `?p rdf:type rdf:Property .`  
entails that `?p rdfs:subPropertyOf ?p .`”

```
PREFIX rdf: <...>
```

```
PREFIX rdfs: <...>
```

```
INSERT {  
    ?p rdfs:subPropertyOf ?p .  
} WHERE {  
    ?p rdf:type rdf:Property .  
}
```



# Additional classes

- RDFS also defines classes for:
  - resources: `rdfs:Resource`
    - the class of all resources
  - literals: `rdfs:Literal`
    - the class of all literals
    - `rdfs:subClassOf rdfs:Resource`
  - datatypes: `rdfs:Datatype`
    - the class of all datatypes
    - `rdfs:subClassOf rdfs:Class`
  - all of them have `rdf:type rdfs:Class`
  - all of them have entailment rules and axioms



# RDFS entailment [rdfs4a]

- *Every subject in a triple is a resource...*
- “The triple `?s ?p ?o .`  
entails that `?s rdf:type rdfs:Resource .`”

PREFIX rdf: <...>

PREFIX rdfs: <...>

```
INSERT {  
    ?s rdf:type rdfs:Resource .  
} WHERE {  
    ?s ?p ?o .  
}
```



# RDFS entailment [rdfs4b]

- *...and every object too*
- “The triple `?s ?p ?o .`  
entails that `?o rdf:type rdfs:Resource .`”

PREFIX rdf: <...>

PREFIX rdfs: <...>

```
INSERT {  
    ?o rdf:type rdfs:Resource .  
} WHERE {  
    ?s ?p ?o .  
}
```



# RDFS entailment [rdfs8]

- Every class corresponds to a subset of all resources.
- “The triple `?c rdf:type rdfs:Class .`  
entails that `?c rdfs:subClassOf rdfs:Resource .`”

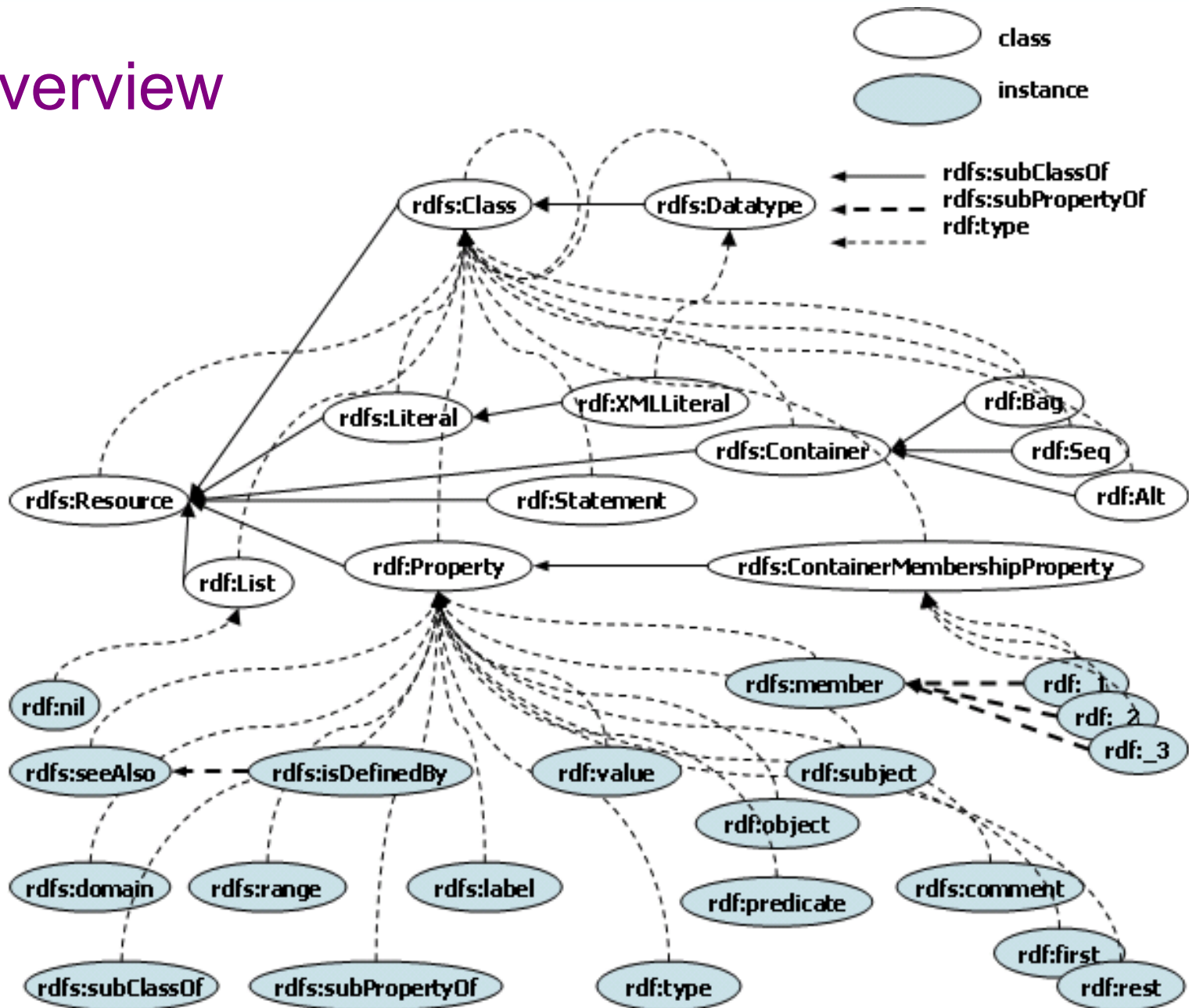
PREFIX rdf: <...>

PREFIX rdfs: <...>

```
INSERT {  
    ?c rdfs:subClassOf rdfs:Resource .  
} WHERE {  
    ?c rdf:type rdfs:Class .  
}
```



# Overview





# Summary of resources and properties

- Resources:

- rdfs:Class rdfs:Resource rdfs:Datatype rdfs:Literal
- rdf:Property, rdf:XMLLiteral, rdf:HTML
- rdfs:Container rdfs:ContainerMembershipProperty
- rdf:Alt rdf:Bag rdf:Seq rdf:List rdf:Statement

- Properties:

- rdfs:subClassOf rdfs:subPropertyOf
- rdfs:domain rdfs:range
- rdfs:comment rdfs:seeAlso rdfs:isDefinedBy rdfs:label
- rdfs:member rdf:\_1 rdf:\_2 ...
- rdf:first rdf:rest rdf:nil
- rdf:subject rdf:predicate rdf:object
- rdf:value



# What we cannot express...

- RDFS has many limitations, e.g., it cannot say:
  - *“my ancestors’ ancestors are also my ancestors”*
  - *“a Person has a unique birth number”*
  - *“a Person has exactly one father”*
  - *“a SoccerTeam has 11 players, but a BasketballTeam has 5”*
  - *“classes with different IRIs actually represent the same class”*
  - *“resources with different IRIs represent the same resource”*
  - *“properties with different IRIs are actually the same”*
  - *“two individuals with different IRIs are actually different”*
  - *“two classes cannot share individuals (they are disjoint)”*
  - *“a class is a combination (union or intersection) of other classes”*
  - *“a class is a negation of another class”*
- ***Web Ontology Language (OWL) does all this and more!***

