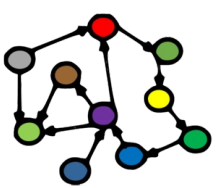
Welcome to INFO216:
Knowledge Graphs
Spring 2024

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

Session 9: Ontologies (OWL)

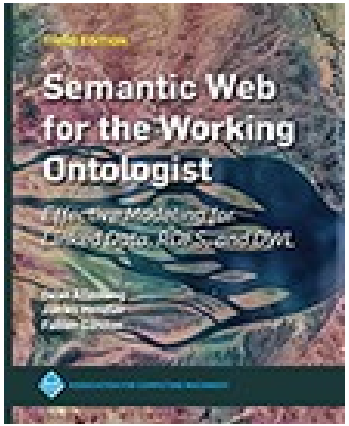
- Themes:
  - what and why?
  - basic OWL constructs (“RDFS-Plus”):
    - more precise properties
    - sameness and difference
    - complex classes
  - more advanced OWL
    - restriction classes
  - Programming in RDFLib

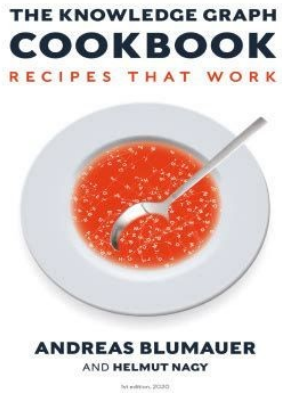


Readings

- Sources:
  - Allemang, Hendler, Gandon (2020): Semantic Web for the Working Ontologist, 3rd edition: chapter 9-10 (“RDFS Plus”, chapters 8-9 in the 2nd ed.)

    advanced: chapters 12-13 (chapters 11-12 in the 2nd ed.)
  - Blumauer & Nagy (2020): Knowledge Graph Cookbook – Recipes that Work: e.g., pages 105-109, 123-124, *(supplementary)*
- Resources in the wiki <http://wiki.uib.no/info216>:
  - OWL 2 Primer, sections 2-6 *(advanced: 9-10)*: *http://www.w3.org/TR/owl-primer/*
    - show: Turtle
  - VOWL: Visual Notation for OWL Ontologies

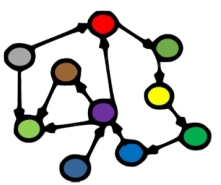




Web Ontology Language (OWL)

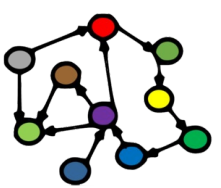
Why do we need vocabularies?

- Shared, well-defined terms (dereferencable URIs) for types, properties and some individuals that can be used to represent a domain
- Domains can be:
  - people, their friends and workplaces (FOAF, BIO)
  - electronic and other documents (DC, BIBO)
  - commerce (schema.org)
  - classification in libraries etc. (SKOS)
  - general encyclopedic information (DBpedia, Wikidata)
  - general time and place (OWL-Time, geo)
  - ...and *lots* of others (→S10)



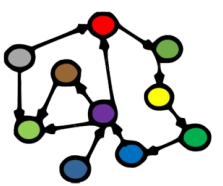
Why do we need vocabularies?

- To make knowledge graphs more precisely defined
- To make semantic data sets easier to use
  - encourage reuse
  - avoid misunderstandings and errors
  - easier to understand, recombine, enrich...
- To support computer processing
  - more powerful
  - more general



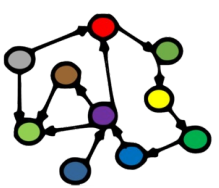
RDFS is a useful starting point...

- We can say:
  - "a president is a politician" so that
    - saying "Trump is a president" entails saying "Trump is a politician"
  - "a politician is a human" so that
    - saying "Trump is a president" *also* entails saying "Trump is a human"
  - "the president of something is a politician" so that
    - saying "Trump is a president of U.S.A." entails "saying Trump is a politician"
  - "something having a president is a country" so that
    - saying "Trump is a president of U.S.A." entails saying "U.S.A. is a country"
  - "being president also means being citizen" so that
    - saying "Trump is a president of U.S.A." entails saying "Trump is a citizen of U.S.A."
- *RDFS expresses this but not (so much) more...*



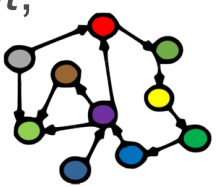
RDFS is a useful starting point...

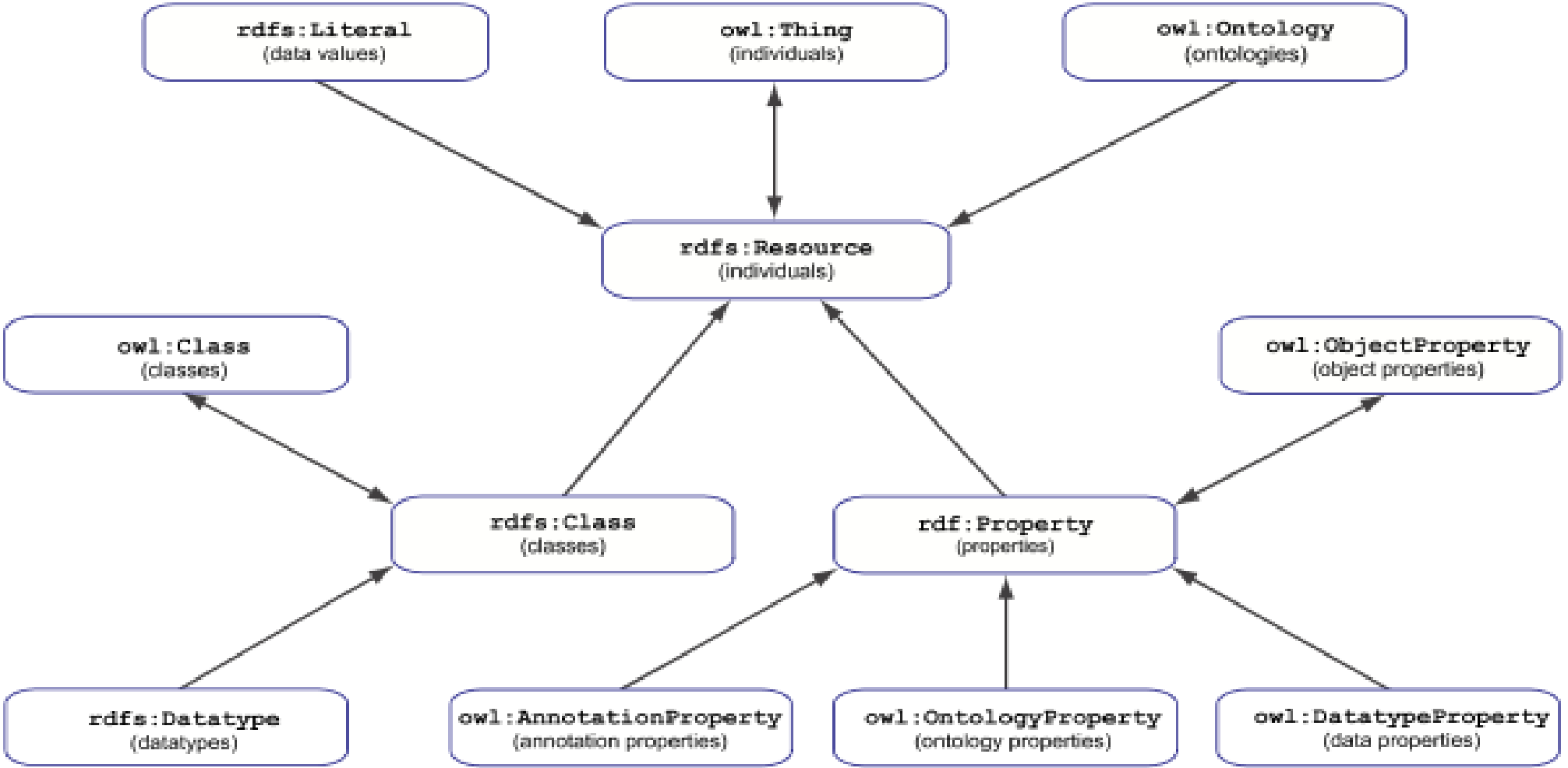
- But lots of simple stuff it cannot express, e.g.:
  - “every ancestor of an ancestor is an ancestor too”
  - “the BirthNumber of a Person is unique”
  - “a Republic has exactly one President”
  - “a FootballTeam has 11 activePlayers, a VolleyballTeam 6”
  - “a StringQuartet has two violins but only one viola and one cello”
  - “classes with different URIs actually represent the same class”
  - “resources with different URIs represent the same resource”
  - “properties with different URIs are actually the same”
  - “two individuals are different”, “two classes are disjoint”
  - “a class is a union (or intersection) of other classes”
  - “a class is a negation of another class”
- *OWL expresses all this and more!*



Basic idea

- Web Ontology Language (OWL):
  - *builds* on RDF and RDFS
  - *reuses*, *renames* and *specialises* classes and properties from RDFS
  - *adds* precision and formality
- Basic OWL-concepts:
  - owl:Thing (equivalent to rdfs:Resource)
  - owl:Class (equivalent to rdfs:Class)
  - owl:ObjectProperty (equivalent to rdf:Property)
  - owl:NamedIndividual (things with URIs and that are not classes)
- Good practice: keep *Classes, Individuals, and Properties disjoint*, i.e., no resource has more than one of them as *rdf:type*

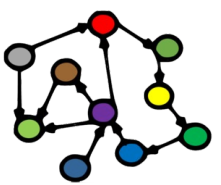




http://www.w3.org/TR/owl2-rdf-based-semantics/

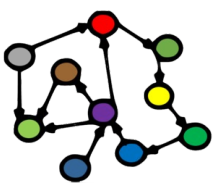
What does OWL offer?

- Extensions of RDFS, e.g.:
  - more *specific types* of properties
  - *identical and different* classes, properties, individuals
  - *defining new classes*:
    - complex classes (union, intersection, complement)
    - property restrictions, enumeration of individuals
  - *defining new properties* based on existing ones
  - *mathematical formality* (for large parts of OWL)
    - (more on this later)



Reuses or specialises RDFS

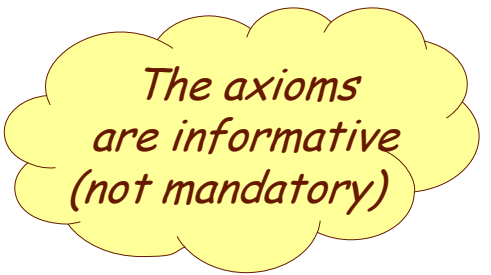
- *Reused* in OWL:
  - rdf:type, rdf:Property,
    rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, rdfs:range
  - ...and lots of other stuff...
- *Renamed* by OWL:
  - owl:Thing (equivalent to rdfs:Resource)
  - owl:Class (equivalent to rdfs:Class)
  - owl:ObjectProperty (equivalent to rdf:Property)
- *Specialised* by OWL:
  - everything else in OWL *specialises* something in RDF / RDFS
  - but also introduces its own, and more powerful,
    formal underpinning

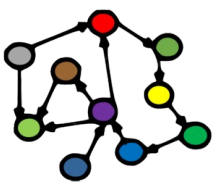


Basic OWL
("RDFS Plus")

Inverse properties

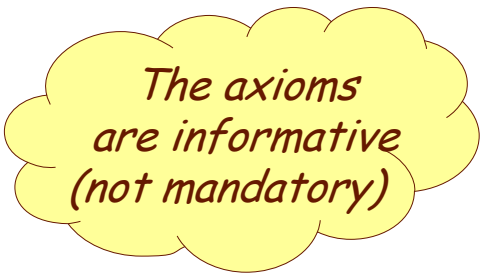
- Properties can be inverses (or reverses) of one another, e.g.,
  - ex:DonaldTrump ex:presidentOf ex:USA .
  - ex:USA ex:hasPresident ex:DonaldTrump .
- P1 owl:inverseOf P2:
  - ex:presidentOf owl:inverseOf ex:hasPresident .
  - owl:inverseOf owl:inverseOf owl:inverseOf .
  - owl:inverseOf a owl:ObjectProperty .
- Entailment rules:
  - if *P1 owl:inverseOf P2* then
    - *P2 owl:inverseOf P1 .*
  - if *S P1 O . P1 owl:inverseOf P2* then
    - O P2 S .

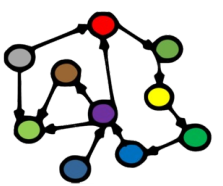




Symmetric properties

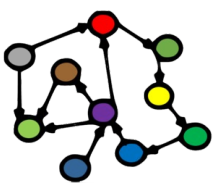
- Some properties are their own inverse, e.g.,
  - ex:DonaldTrump ex:marriedTo ex:MelianiaTrump .
  - ex:MelianiaTrump ex:marriedTo ex:DonaldTrump .
- P rdf:type owl:SymmetricProperty:
  - ex:marriedTo a owl:SymmetricProperty .
  - owl:inverseOf a owl:SymmetricProperty .
  - owl:SymmetricProperty rdfs:subClassOf owl:ObjectProperty .
- Entailment rules:
  - if *P a owl:SymmetricProperty* then
    - P owl:inverseOf P .
  - if *S P O . P a owl:SymmetricProperty* then
    - O P S .





Asymmetric, reflexive, irreflexive properties

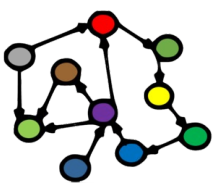
- New in OWL2:
  - both *symmetric* and *asymmetric* properties:
    - ex:marriedTo a owl:SymmetricProperty .
      - "marriage is always mutual (two-way)"
    - ex:hasChild a owl:AsymmetricProperty .
      - "two resources cannot be the child of each other"
    - *many properties are neither – leave it open!*
  - both *reflexive* and *irreflexive* properties:
    - owl:sameAs a owl:ReflexiveProperty .
      - "every resource is the same as itself"
    - ex:hasChild a owl:IrreflexiveProperty .
      - "no resource can be its own child"
    - *many properties are neither – leave it open!*



Transitive properties

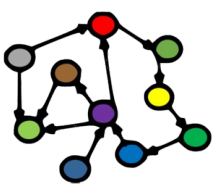
- Some properties can form chains so that the result is the property itself, e.g.:
  - ex:DonaldTrump ex:hasPredecessor ex:BarackObama .
  - ex:BarackObama ex:hasPredecessor ex:GeorgeWBush .
  - ex:DonaldTrump ex:hasPredecessor ex:GeorgeWBush .
- P a owl:TransitiveProperty:
  - ex:hasPredecessor a owl:TransitiveProperty .
  - rdfs:subClassOf a owl:TransitiveProperty .
  - rdfs:subPropertyOf a owl:TransitiveProperty .
- Entailment rules:
  - "if *S P X . X P O . P a owl:TransitiveProperty* then
    - S P O ."

The axioms are informative (not mandatory)



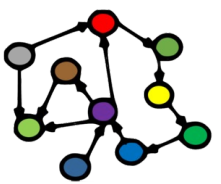
Functional properties

- Each subject *can only have one* object value for the functional property, e,g.,
  - ex:hasPresident a owl:FunctionalProperty .
  - ex:dateOfBirth a owl:FunctionalProperty .
  - owl:FunctionalProperty rdfs:subClassOf owl:ObjectProperty .
- Entailment rule:
  - if *S P O1 . S P O2 . P a owl:FunctionalProperty* then
    - *O1 owl:sameAs O2 .*
  - The rule also holds for *owl:DatatypeProperties*, but:
    - if two different literals become asserted as *owl:sameAs* one another, *the ontology is inconsistent*



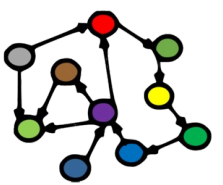
Functional properties

- Each subject *can only have one* object value for the functional property, e,g.,
  - ex:hasPresident a owl:FunctionalProperty .
  - ex:dateOfBirth a owl:FunctionalProperty .
  - owl:FunctionalProperty rdfs:subClassOf owl:ObjectProperty .
- Entailment rule:
  - if *S P O1 . S P O2 . P a owl:FunctionalProperty* then
    - *O1 owl:sameAs O2 .*
  - The rule also holds for *owl:DatatypeProperties*, but:
    - if two different literals become asserted as *owl:sameAs* one another, *the ontology is inconsistent*



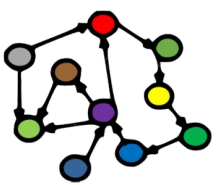
Inverse functional properties

- Two different subjects cannot have the same object for an inverse functional property, i.e.,
  - ex:presidentOf a owl:InverseFunctionalProperty .
- Entailment rule:
  - if *S1 P O . S2 P O . P a owl:InverseFunctionalProperty* then
    - *S1 owl:sameAs S2 .*
- Inverse functional properties are *unique* for each individual
  - used for *identifiers*
  - OWL 2 also has a built-in *owl:hasKey* property for identifiers



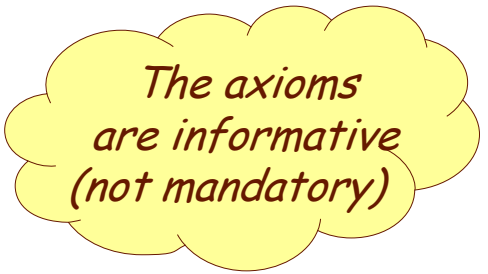
Summary: more precise properties

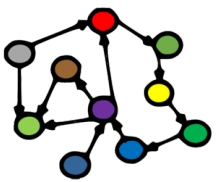
- owl:inverseOf
- owl:SymmetricProperty, owl:AsymmetricProperty
- owl:ReflexiveProperty, owl:IrreflexiveProperty
- owl:TransitiveProperty
- owl:FunctionalProperty, owl:InverseFunctionalProperty
- owl:hasKey
- Also:
  - negated properties
  - chained properties, e.g.:
    fam:hasGrandmother
    owl:propertyChainAxiom
    (:hasParent :hasMother) .



Individual equivalence

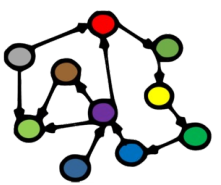
- Sometimes, two individuals (with different URI-s) represent the same thing:
  - http://dbpedia.org/resource/Donald_Trump
  - http://wikidata.org/entity/Q22686
- I1 owl:sameAs I2:
  - owl:sameAs a owl:ReflexiveProperty .
  - owl:sameAs a owl:SymmetricProperty .
  - owl:sameAs a owl:TransitiveProperty .
- owl:sameAs is an *equivalence relation*:
  - because it is *reflexive*, *symmetric* and *transitive*





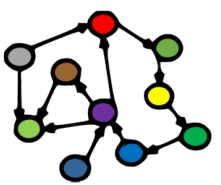
Unique Name Assumption (UNA)

- If two resources have different names, do they necessarily represent different things?
- RDF and OWL does *not* assume this!
  - *in RDF and OWL, we do not know whether resources with different names represent different things or not*
- We can use
  - owl:sameAs – two resources represent the same thing
  - owl:differentFrom – they represent different things
  - ...or we can leave it open
- Some ICT-languages and technologies use UNA, others do not!



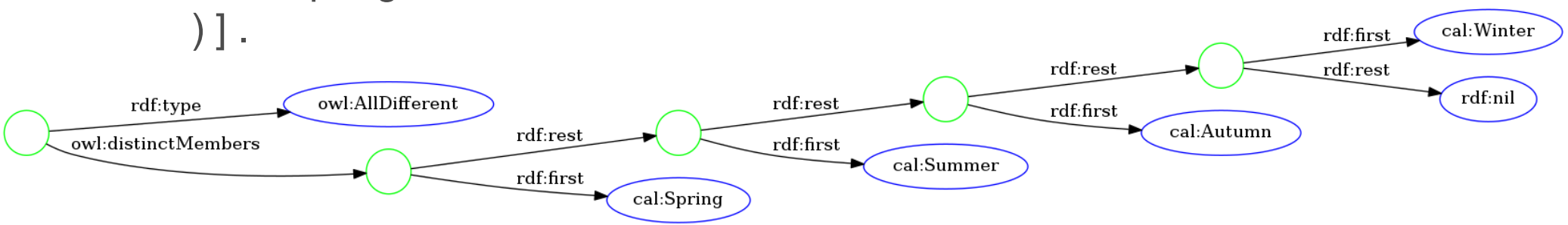
Individual difference

- Sometimes, a *pair* of individuals with different names (URI-s) represent *different* things, e.g.,
  - cal:Spring owl:differentFrom cal:Summer .
- owl:differentFrom
  - *not* transitive
  - *not* reflexive



Individual difference

- Sometimes, a pair of individuals with different names (URI-s) represent different things, e.g.,
  - cal:Spring owl:differentFrom cal:Summer .
- Sometimes, a *group* of individuals with different names (URI-s) *all* represent *different* things, e.g.,
  - [a owl:AllDifferent ;
    owl:distinctMembers (
    cal:Spring cal:Summer cal:Autumn cal:Winter
)] .

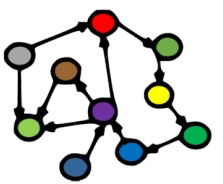


Namespaces:
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
cal: http://ex.org/cal/
owl: http://www.w3.org/2002/07/owl#

https://www.ldf.fi/service/rdf-grapher

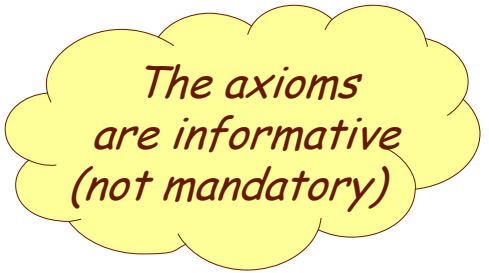
Individual difference

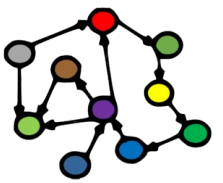
- Sometimes, a pair of individuals with different names (URI-s) represent different things, e.g.,
  - cal:Spring owl:differentFrom cal:Summer .
- Sometimes, a group of individuals with different names (URI-s) all represent different things, e.g.,
  - [a owl:AllDifferent ;
    owl:distinctMembers (
    cal:Spring cal:Summer cal:Autumn cal:Winter
)] .
  - *owl:AllDifferent* and *owl:distinctMembers* are special constructs in OWL
    - they must always be used together
  - ...corresponds to pairwise *owl:differentFrom* between *all* individuals in the *owl:distinctMembers*-list



Equivalent classes

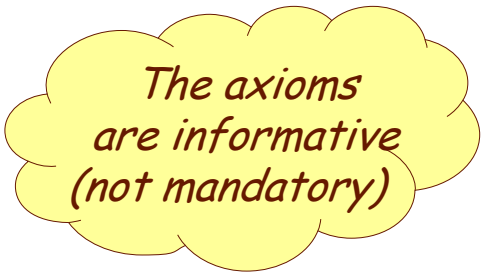
- Sometimes, two classes (with different URI-s) represent the *same* class:
- C1 owl:equivalentClass C2:
  - owl:equivalentClass a owl:ReflexiveProperty .
  - owl:equivalentClass a owl:SymmetricProperty .
  - owl:equivalentClass a owl:TransitiveProperty .
- owl:equivalentClass is another *equivalence relation*:
    - it is *reflexive*, *symmetric* and *transitive*
- C1 owl:equivalentClass C2 means the same as
  - *C1 rdfs:subClassOf C2* and *C2 rdfs:subClassOf C1*
- *Also disjoint classes:*
    - *uib:InternalCensor*
      owl:disjointWith skos:ExternalCensor .

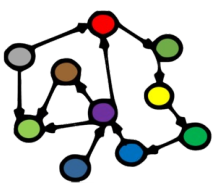




Equivalent properties

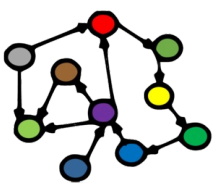
- Two properties (with different URI-s) can represent the same property:
- P1 owl:equivalentProperty P2:
  - owl:equivalentProperty a owl:ReflexiveProperty .
  - owl:equivalentProperty a owl:SymmetricProperty .
  - owl:equivalentProperty a owl:TransitiveProperty .
- owl:equivalentProperty is another *equivalence relation*:
    - it is *reflexive*, *symmetric* and *transitive*
- *Also disjoint properties:*
    - skos:prefLabel owl:propertyDisjointWith skos:altLabel .





Summary: sameness and difference

- Individuals:
  - pairwise: owl:sameAs, owl:differentFrom
  - groupwise difference: owl:AllDifferent
- Classes:
  - pairwise: owl:equivalentClass, owl:disjointWith
  - groupwise difference: owl:AllDisjointClasses
- Properties:
  - pairwise: equivalentProperty, propertyDisjointWith
  - groupwise difference: owl:AllDisjointProperties
- Membership in the groups:
  - owl:distinctMembers *(preferred)* or owl:members



Basic OWL reasoning in Python and rdflib

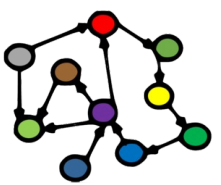
Basic OWL inference in RDFLib

- import owlrl
  ...

  DeductiveClosure(RDFS_Semantics).expand(graph) # RDFS reasoning
  ...

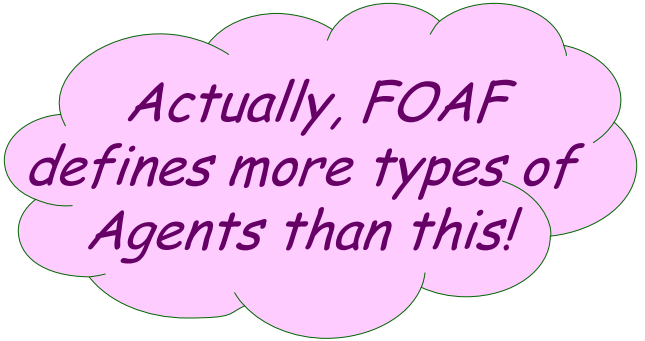
  DeductiveClosure(OWLRL_Semantics).expand(graph) # OWL-RL reasoning
  ...

  DeductiveClosure(OWLRL_Extension, # Maximum reasoning
  rdfs_closure = True, axiomatic_triples = True,
  datatype_axioms = True).expand(graph)
  ...



Complex OWL classes

Union classes



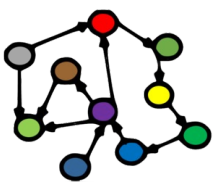
- A union class contains all the individuals in *either of* two or more other classes, e.g.,
  - foaf:Agent
    a owl:Class;
    owl:unionOf (foaf:Person foaf:Organization) .
- Entailment rule:
  - if *C owl:equivalentClass [owl:unionOf (C1... Cn)]* then
    - C1 rdfs:subClassOf C . ... Cn rdfs:subClassOf C .
- why not say just, e.g.,:
  - foaf:Person rdfs:subClassOf foaf:Agent .
  - foaf:Organization rdfs:subClassOf foaf:Agent .

?

Intersection classes

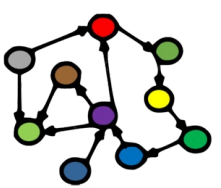
- An intersection class contains all the individuals in *all of* two or more other classes, e.g.
  - uib:StudentAssistant
    a owl:Class;
    owl:intersectionOf (uib:Student uib:Teacher) .
- Entailment rule:
  - if *C owl:equivalentClass [owl:intersectionOf (C1... Cn)]* then
    - C rdfs:subClassOf C1 . ... C rdfs:subClassOf Cn .
- why not say, e.g.:
  - uib:StudentAssistant rdfs:subClassOf uib:Student .
  - uib:StudentAssistant rdfs:subClassOf uib:Teacher .

?



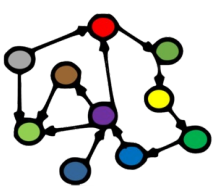
Complement classes

- A complement class contains all the individuals *that are not* in another class:
  - uib:ExternalCensor owl:complementOf uib:InternalCensor .



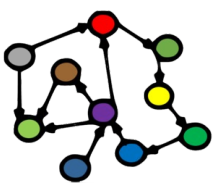
Complement classes

- A complement class contains all the individuals *that are not* in another class:
  - uib:ExternalCensor owl:complementOf uib:InternalCensor .
  - *...but is this correct?!*



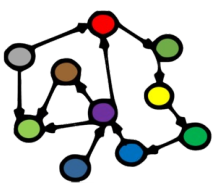
Complement classes

- A complement class contains all the individuals *that are not* in another class:
  - uib:ExternalCensor
    a owl:Class;
    owl:complementOf uib:InternalCensor .



Complement classes

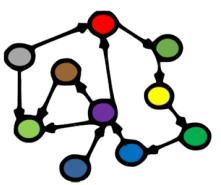
- A complement class contains all the individuals *that are not* in another class:
  - uib:ExternalCensor
    owl:intersectionOf (
      uib:Censor
      owl:complementOf uib:InternalCensor
) .



Complement classes

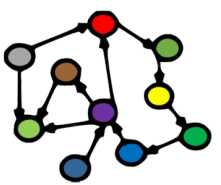
- A complement class contains all the individuals *that are not* in another class:
  - uib:ExternalCensor

```
owl:intersectionOf (
        uib:Censor
        [   a owl:Class ;
            owl:complementOf uib:InternalCensor
        ]
) .
```



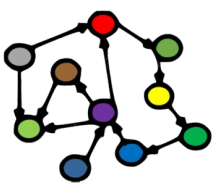
Complement classes

- A complement class contains all the individuals *that are not* in another class:
  - uib:ExternalCensor
    owl:intersectionOf (
      uib:Censor
      [owl:complementOf uib:InternalCensor]
) .



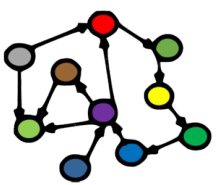
Closed World Assumption (CWA)

- Whenever something is not explicitly stated in the ontology, can we assume that the opposite is the case?
  - DBpedia only lists three James Dean movies –
    can we thus assume that he only played in three?
- Classical logic and many ICT languages assume so:
  - this is the *"Closed World Assumption" (CWA)*
- *In RDF and OWL, we do not assume that something is false just because it is not stated*
  - this is the *"Open World Assumption" (OWA)*



Enumeration classes

- An *enumeration class* is defined by exhaustively listing all its member individuals, e.g.:
  - [a owl:Class ;
    owl:oneOf (cal:Spring ... cal:Winter)] .
- An enumeration class is *closed*
  - there are no other member individuals
  - ensured by using *RDF Collections:*
    - rdf:List, rdf:first, rdf:rest, rdf:nil
- Does *not* imply that the individuals are distinct
  - this must be stated explicitly



Other ways to write complex classes

- Why can also write:

  cal:Season
  owl:oneOf (cal:Spring ... cal:Winter) .

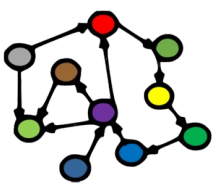
  or

  cal:Season owl:equivalentClass [
  owl:oneOf (cal:Spring ... cal:Winter)] .

- or (a weaker claim):

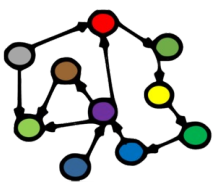
  cal:Season owl:subClassOf [
  owl:oneOf (cal:Spring ... cal:Winter)] .

- Reason:
  - *rdfs:subClassOf* can be computationally more efficient
  - *owl:equivalentClass* is sometimes implemented as a costly two-way *rdfs:subClassOf*



Summary: complex classes

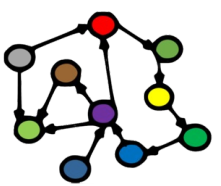
- owl:oneOf
- owl:unionOf
- owl:intersectionOf
- owl:complementOf (and the *CWA*)
- owl:NegativePropertyAssertion, owl:sourceIndividual, owl:assertionProperty, owl:targetIndividual



OWL restriction classes

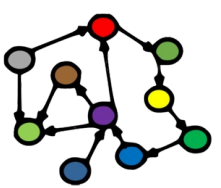
Property value restrictions

- Defining a class by a particular value on one of its properties, e.g.:
  - ex:Republican
    a owl:Restriction ;
    owl:onProperty dbo:hasParty ;
    owl:hasValue dbr:Republican_Party_(United_States) .



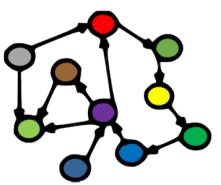
Property value restrictions

- Defining a class by a particular value on one of its properties, e.g.:
  - ex:Republican owl:intersectionOf (
    dbr:Person
    [a owl:Restriction ;
    owl:onProperty dbo:hasParty ;
    owl:hasValue dbr:Republican_Party_(United_States)
    ]
) .



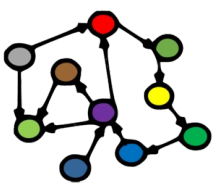
Existential property restrictions

- Defining a class by the existence of a relation (object property) to an individual in (another or the same) class, e.g.:
  - ex:President owl:intersectionOf (
    dbr:Person
    [a owl:Restriction ;
    owl:onProperty ex:presidentOf ;
    owl:someValuesFrom owl:Thing
    ]
) .
- *owl:someValuesFrom:* each individual in the defined class has *at least one* object property (given by owl:onProperty) to an individual in the other class (given by owl:someValuesFrom)



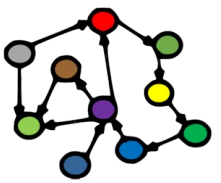
Existential property restrictions

- Defining a class by the existence of a relation (object property) to an individual in (another or the same) class, e.g.:
  - dbr:President_(government_title) owl:intersectionOf (
    dbr:Person
    [a owl:Restriction ;
    owl:onProperty ex:presidentOf ;
    owl:someValuesFrom dbr:Nation
    ]
) .
- *owl:someValuesFrom:* each individual in the defined class has *at least one* object property (given by owl:onProperty) to an individual in the other class (given by owl:someValuesFrom)



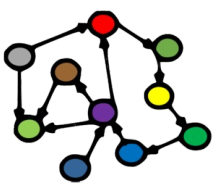
Existential property restrictions

- Defining a class by the existence of a relation (object property) to an individual in (another or the same) class, e.g.:
  - ex:BipartisanCommittee owl:intersectionOf (
    foaf:Group
    [a owl:Restriction ;
    owl:onProperty foaf:member ;
    owl:someValuesFrom ex:Republican_(United_States)
    ]
    [a owl:Restriction ;
    owl:onProperty foaf:member ;
    owl:someValuesFrom ex:Democrat_(United_States)
    ]
) .



Universal property restrictions

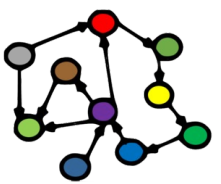
- Defining a class by the necessity of a relation (object property) only to individuals in (another or the same) class, e.g.:
  - ex:RepublicanCommittee owl:intersectionOf (
    foaf:Group
    [a owl:Restriction ;
    owl:onProperty foaf:member ;
    owl:allValuesFrom ex:Republican_(United_States)
    ]
) .



Universal property restrictions

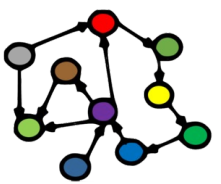
- Defining a class by the necessity of a relation (object property) only to individuals in (another or the same) class, e.g.:
  - ex:RepublicanCommittee owl:intersectionOf (
    foaf:Group
    [a owl:Restriction ;
    owl:onProperty foaf:member ;
    owl:allValuesFrom ex:Republican_(United_States)
    ]
) .

What is wrong here?



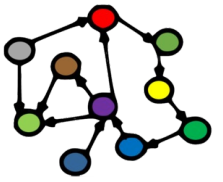
Universal property restrictions

- Defining a class by the necessity of a relation (object property) only to individuals in (another or the same) class, e.g.:
  - ex:RepublicanCommittee owl:intersectionOf (
    foaf:Group
    [a owl:Restriction ;
    owl:onProperty foaf:member ;
    owl:allValuesFrom ex:Republican_(United_States)
    ]
    [a owl:Restriction ;
    owl:onProperty foaf:member ;
    owl:someValuesFrom owl:Thing
    ]
) .



Universal property restrictions

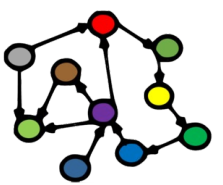
- Defining a class by the necessity of a relation (object property) only to individuals in (another or the same) class, e.g.:
  - ex:RepublicanCommittee owl:intersectionOf (
    foaf:Group
    [a owl:Restriction ;
    owl:onProperty foaf:member ;
    owl:allValuesFrom [
    a owl:Restriction ;
    owl:onProperty ex:hasParty ;
    owl:hasValue ex:Republican_Party_(United_States)
    ]]
    [a owl:Restriction ;
    owl:onProperty foaf:member ;
    owl:someValuesFrom owl:Thing
    ]) .



Property self-reflexion

- Defining a class by a *Self* value on one of its properties, e.g.:
  - ex:Narcissist

```
a                 owl:Restriction ;
owl:onProperty    ex:loves ;
owl:hasSelf       "true"^^xsd:boolean .
```



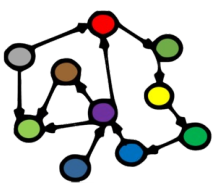
Datatype property restriction

- Restrictions on data range, e.g.:
  - fam:personAge rdfs:range

```
[     a rdfs:Datatype;
      owl:onDatatype  xsd:integer;
      owl:withRestrictions (
            [ xsd:minInclusive  "0"^^xsd:integer ]
            [ xsd:maxInclusive  "130"^^xsd:integer ]  )
] .
```

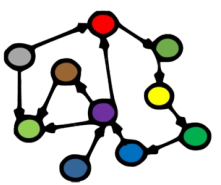
  - :toddlerAge rdfs:range

```
[     a rdfs:Datatype;
      owl:oneOf ( "1"^^xsd:integer  "2"^^xsd:integer)
] .
```



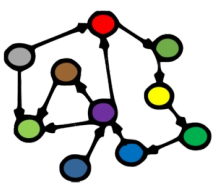
Cardinality restriction

- Defining a class by the number of object values its individuals have for some property, e.g.:
  - music:Quartet owl:intersectionOf (
    music:Ensemble
    [a owl:Restriction ;
    owl:onProperty music:hasMusician ;
    owl:cardinality 4]
) .
- owl:cardinality gives the *exact cardinality*
  owl:minCardinality gives the *least cardinality*
  owl:maxCardinality gives the *greatest cardinality*



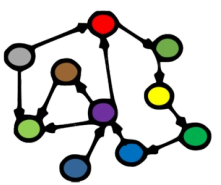
Qualified cardinality restriction

- Defining a class by the number of object values its individuals have *of a given class* for some property, e.g.:
  - pol:Triumvirate owl:intersectionOf (
    pol:PoliticalLeadership
    [a owl:Restriction ;
    owl:onProperty pol:hasMember ;
    owl:qualifiedCardinality 3 ;
    owl:onClass pol:PoliticalLeader]
) .
- owl:qualifiedCardinality gives the *exact cardinality*
  owl:minQualifiedCardinality gives the *least cardinality*
  owl:maxQualifiedCardinality gives the *greatest cardinality*



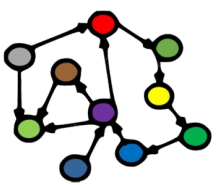
Qualified cardinality restriction

- music:StringQuartet owl:intersectionOf (
  music:MusicalQuartet
  [a owl:Class ;
  owl:onProperty music:hasMusician ;
  owl:qualifiedCardinality “2” ;
  owl:onClass music:Violinist]
  [a owl:Class ;
  owl:onProperty music:hasMusician ;
  owl:qualifiedCardinality “1” ;
  owl:onClass music:Violist]
  [a owl:Class ;
  owl:onProperty music:hasMusician ;
  owl:qualifiedCardinality “1” ;
  owl:onClass music:Cellist]) .



Summary: property restrictions

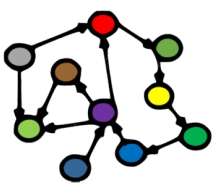
- owl:Restriction owl:onProperty
- owl:someValuesFrom, owl:allValuesFrom, owl:hasValue
- owl:cardinality, owl:minCardinality, owl:maxCardinality
- owl:qualifiedCardinality, owl:minQualifiedCardinality, owl:maxQualifiedCardinality, owl:onClass



OWL as (a) Formal System(s)

Web Ontology Language versions

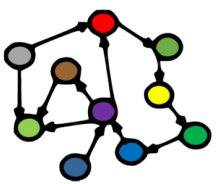
- OWL "1" (2002):
  - OWL Full – anything goes
  - OWL DL – fragment of OWL Full, formal semantics through *description logic*
  - OWL Lite – simple fragment of OWL DL, not much used
- OWL 2 (2008):
  - *backwards compatible* with OWL "1"!
  - OWL2 DL – fragment of OWL2 full, extension of OWL DL
    - formal and powerful, but *reasoning can get prohibitively slow*
  - OWL2 DL – defines three faster fragments of OWL2 DL:
    - OWL2 RL – rule-based semantics, also OWL LD – for Linked Data
    - OWL2 EL – quick DL reasoning
    - OWL2 QL – suitable for query rewriting



Building blocks

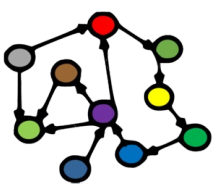
OWL2 can be seen as an extension of RDF and RDFS, but can also stand on its own feet.

- OWL 2 has three building blocks:
  - *entities:*
    - refer to real-world entities using URIs
    - owl:Class, owl:NamedIndividual
    - owl:ObjectProperty, owl:DatatypeProperty, owl:AnnotationProperty, owl:ObjectProperty
  - *axioms:* ← *can be true or false!*
    - basic statements expressed by the OWL ontology
    - every triple in the RDF graph is an axiom
  - *expressions:*
    - use *constructors to*
    - *define more complex entities*
    - by *combining* simpler ones



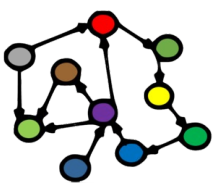
More building blocks

- owl:Thing:
  - is equivalent to *rdfs:Resource*
  - logic interpretation: *True*
    - called the *top concept* in description logic (DL)
- owl:Nothing
  - is the empty set
  - no resource has it as its *rdf:type*
  - logic interpretation: *False*
    - called the *bottom concept* in DL



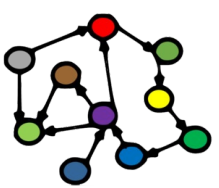
Named and constructed classes

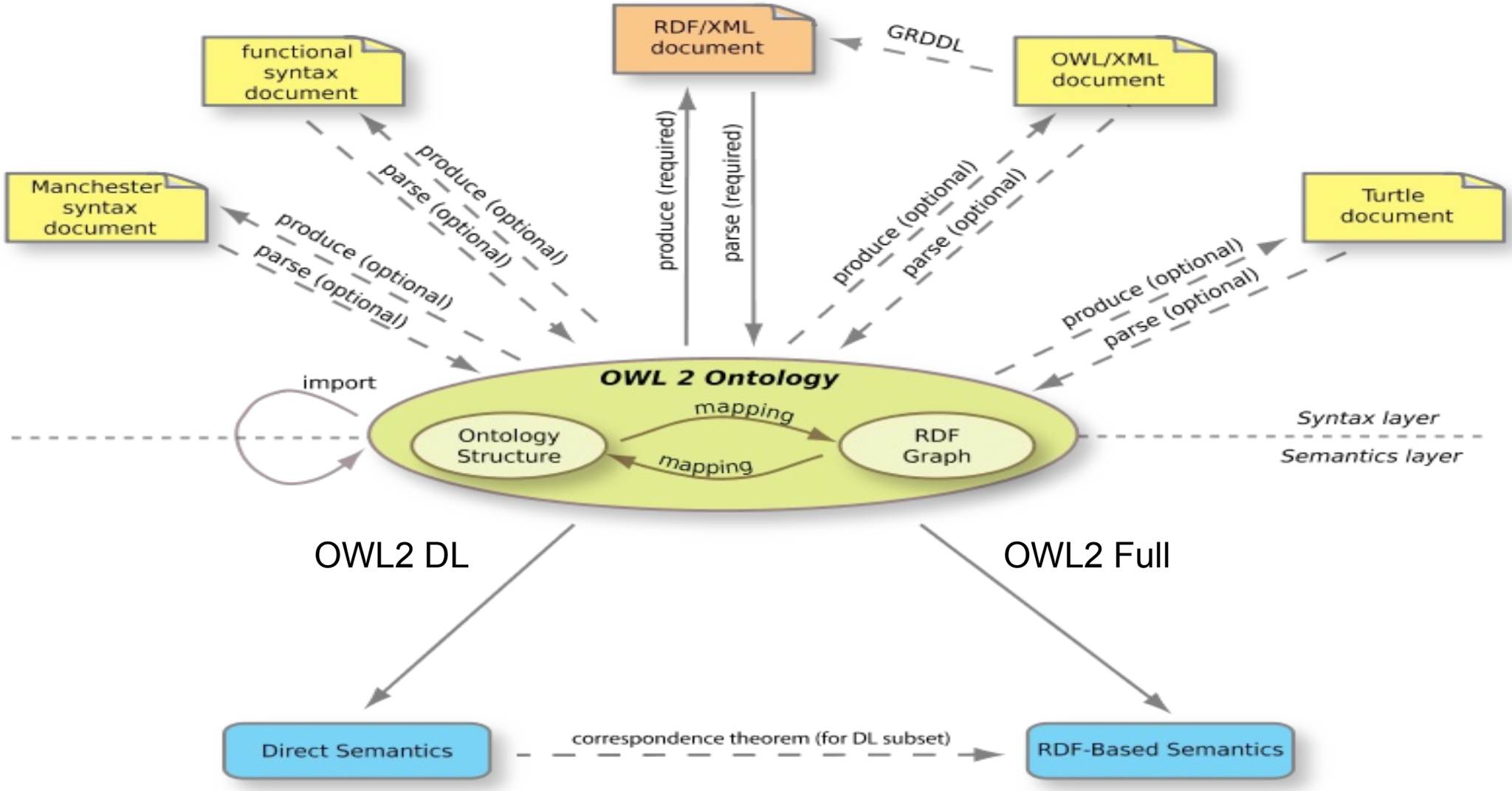
- owl:Class-es that have an URI:
  - semantics are given by:
    - URI-s, labels and other annotations
    - domain, range, subClassOf and other relationships
- *Constructed* (or *complex*) owl:Class:
  - built from existing classes, properties, individuals
    - which can be named *or anonymous*
  - constructed classes are *anonymous upon declaration,*
    - but can be *named* later
  - *unions, intersections* and *negations* of existing classes (←S08)
  - *enumeration* of existing individuals (←S08)
  - *restrictions* on existing properties



Summary: core OWL concepts

- owl:Thing, owl:Nothing
  owl:NamedIndividual
- owl:Class
- owl:ObjectProperty, owl:DatatypeProperty
- owl:AnnotationProperty, owl:OntologyProperty




RDF/XML document
GRDDL
OWL/XML document
functional syntax document
Manchester syntax document
Turtle document
produce (required)
parse (required)
produce (optional)
parse (optional)
produce (optional)
parse (optional)
produce (optional)
parse (optional)
produce (optional)
parse (optional)
import
OWL 2 Ontology
mapping
mapping
Ontology Structure
RDF Graph
Syntax layer
Semantics layer
OWL2 DL
OWL2 Full
Direct Semantics
correspondence theorem (for DL subset)
RDF-Based Semantics

Next week:
Vocabularies