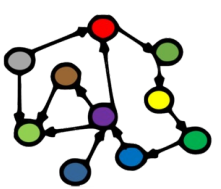
Welcome to INFO216:
Knowledge Graphs
Spring 2023

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

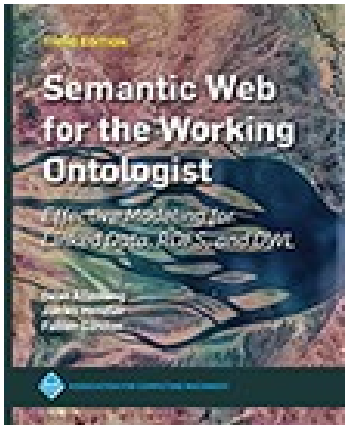
Session 10: Formal ontologies (OWL-DL)

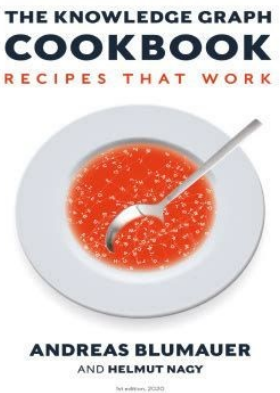
- Themes:
  - OWL-DL
    - core OWL concepts
    - restriction classes
  - description logic
  - decision problems



Readings

- Sources:
  - Allemang, Hendler & Gandon (2020): Semantic Web for the Working Ontologist, 3rd edition: chapters 12-13, but chapters 11-12 in the 2nd edition
  - 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>, e.g.:
  - OWL 2 Overview (http://www.w3.org/TR/owl-overview/)
  - OWL 2 Primer (http://www.w3.org/TR/owl-primer/):
    - show: Turtle and Manchester syntax
    - hide: other syntaxes

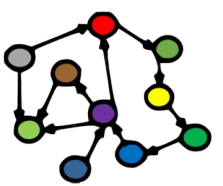




The Core OWL Concepts

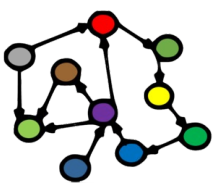
Web Ontology Language versions

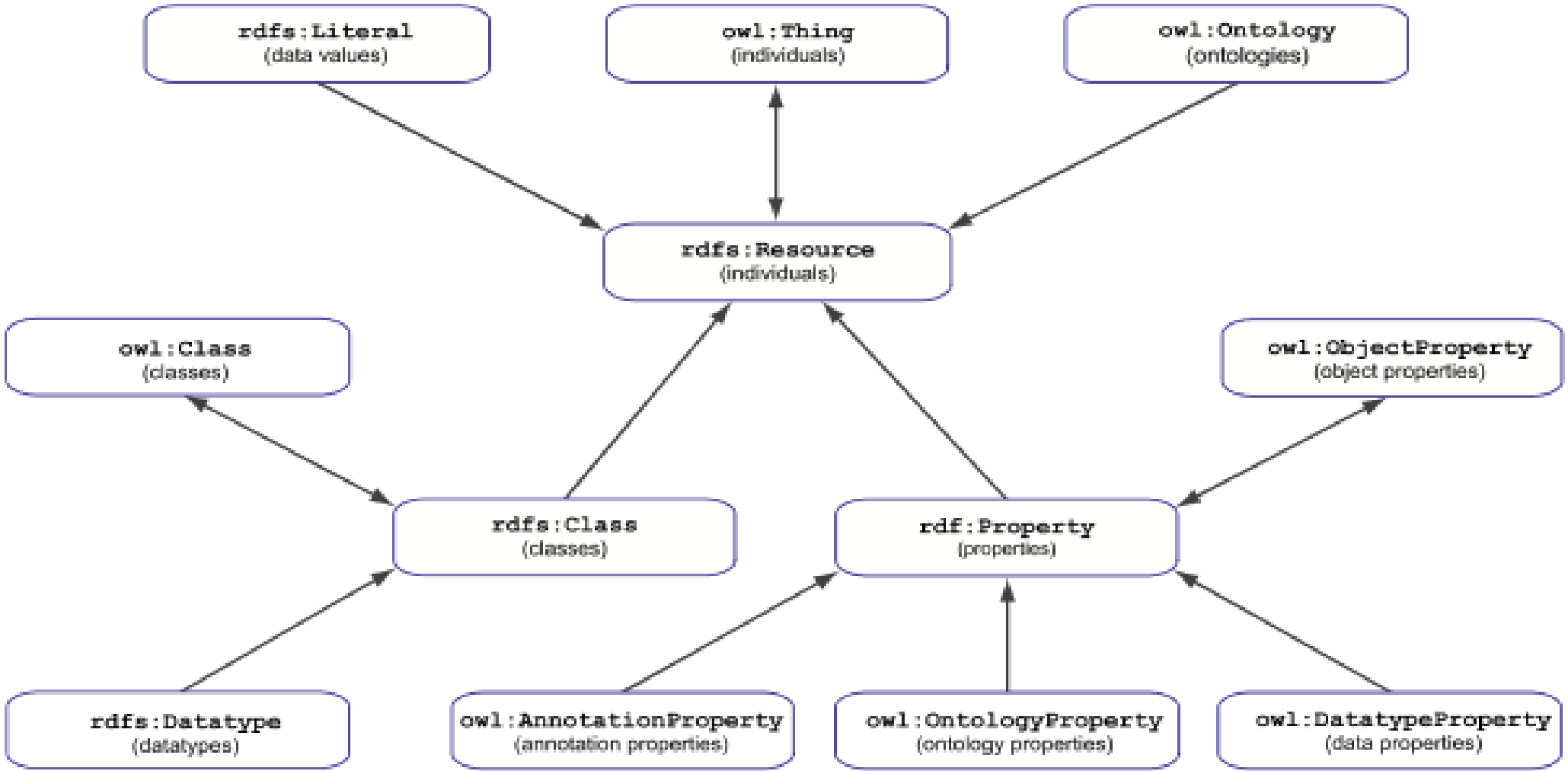
- OWL “1” (2002):
  - OWL Full – anything goes
  - OWL DL – fragment of OWL Full, formal semantics through *description logic*
  - OWL Lite – simple fragment of OWL DL, not much used
- OWL 2 (2008):
  - *backwards compatible* with OWL “1”!
  - OWL2 DL – fragment of OWL2 full, extension of OWL DL
    - 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



Classes, properties, and individuals (←S08)

- Web Ontology Language (OWL):
  - builds on RDF and RDFS
  - uses classes and properties from RDF and 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*
  - *in OWL DL, this is mandatory...*



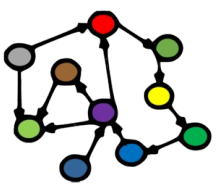


http://www.w3.org/TR/owl2-rdf-based-semantics/

Building blocks

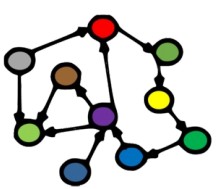
OWL2 can be seen as an extension of RDF and RDFS, but can also stand on its own feet.

- 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



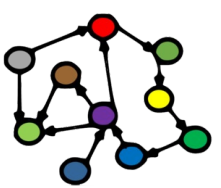
More building blocks

- 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

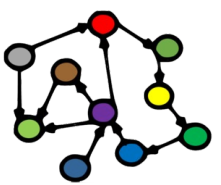
- owl:Class-es that have an URI:
  - semantics are given by:
    - URI-s, labels and other annotations
    - domain, range, subClassOf and other relationships
- *Constructed* (or *complex*) owl:Class:
  - built from existing classes, properties, individuals
    - which can be named *or anonymous*
  - 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



Object and datatype properties

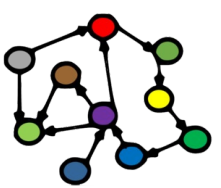
- In RDF triples, the object is either a resource or a literal
  - OWL has two corresponding types of properties
  - owl:ObjectProperty:
    - rdfs:range (“verdiområde”) is usually an OWL-class of individuals
    - used in axioms (e.g., RDF triples) with a *resource* object
  - owl:DatatypeProperty:
    - rdfs:range is an RDFS-datatype
    - used in axioms (e.g., RDF triples) with a *literal* object
  - the rdfs:domain (“definisjonsmengden”) is always an OWL-class of individuals

Formally, owl:DatatypeProperty is rdfs:subPropertyOf owl:ObjectProperty .



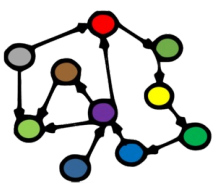
Summary: core OWL concepts

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



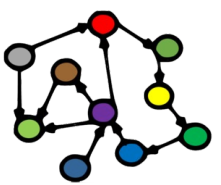
Summary: more precise properties (←S08)

- 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:hasGrandparent
    owl:propertyChainAxiom (:hasParent :hasParent) .



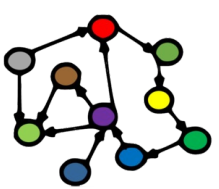
Summary: sameness and difference (←S08)

- 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



Summary: complex classes (←S08)

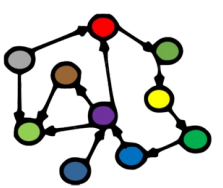
- owl:oneOf
- owl:unionOf
- owl:intersectionOf
- owl:complementOf (and the *CWA*)
- owl:NegativePropertyAssertion, owl:sourceIndividual, owl:assertionProperty, owl:targetIndividual



OWL restriction classes

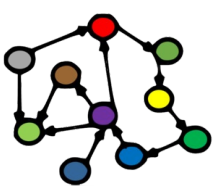
Property value restrictions

- 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) .



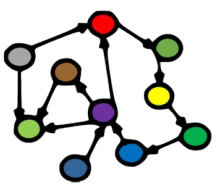
Property value restrictions

- 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)
    ]
) .



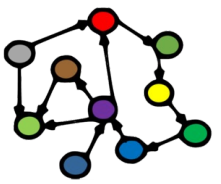
Existential property restrictions

- 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)



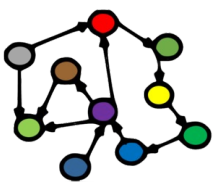
Existential property restrictions

- 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)



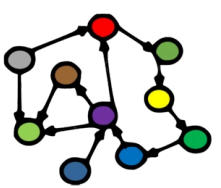
Existential property restrictions

- 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

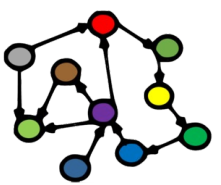
- Defining a class by the necessity of a relation (object property) only to individuals in (another or the same) class, e.g.:
  - ex:RepublicanCommittee owl:intersectionOf (
    foaf:Group
    [a owl:Restriction ;
    owl:onProperty foaf:member ;
    owl:allValuesFrom ex:Republican_(United_States)
 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



Universal property restrictions

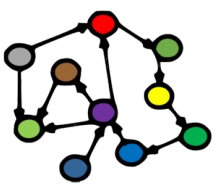
- Defining a class by the necessity of a relation (object property) only to individuals in (another or the same) class, e.g.:
  - ex:RepublicanCommittee owl:intersectionOf (
    foaf:Group
    [a owl:Restriction ;
    owl:onProperty foaf:member ;
    owl:allValuesFrom ex:Republican_(United_States)
 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

What is wrong here?



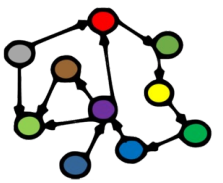
Universal property restrictions

- 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

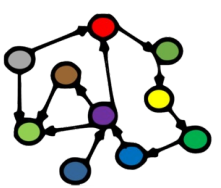
- Defining a class by the necessity of a relation (object property) only to individuals in (another or the same) class, e.g.:
  - ex:RepublicanCommittee owl:intersectionOf (
    foaf:Group
    [a owl:Restriction ;
    owl:onProperty foaf:member ;
    owl:allValuesFrom [
    a owl:Restriction ;
    owl:onProperty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



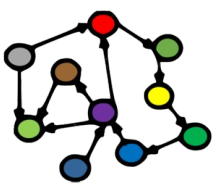
Datatype property restriction

- 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 ]  )
] .
```

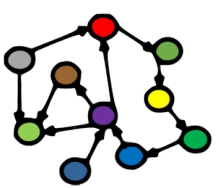
  - :toddlerAge rdfs:range

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



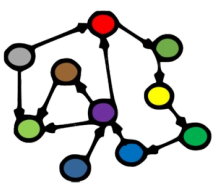
Cardinality restriction

- 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*



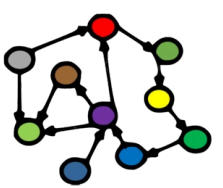
Qualified cardinality restriction

- 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*



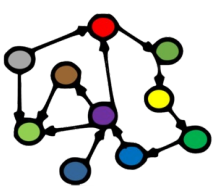
Qualified cardinality restriction

- 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]) .

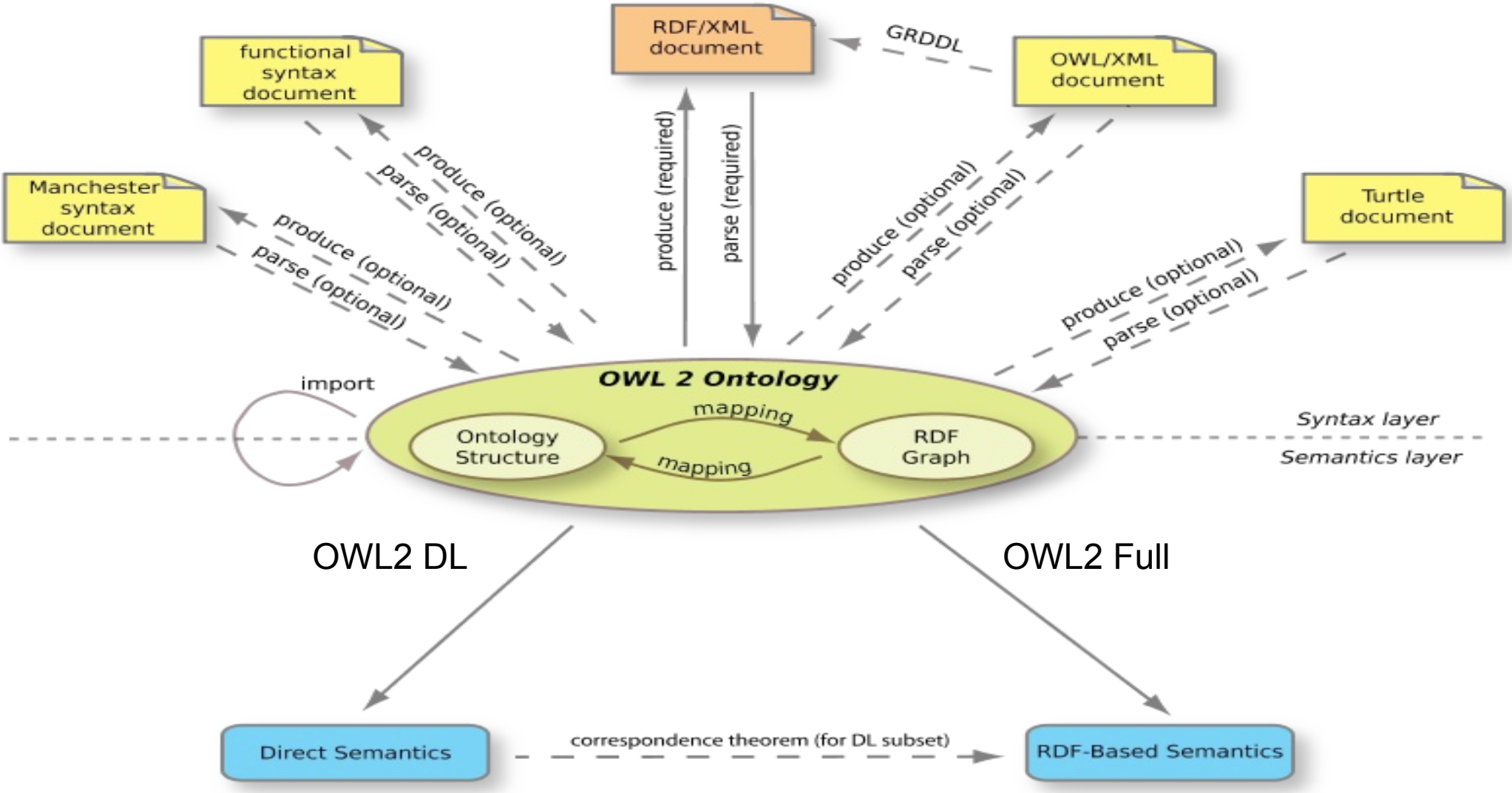



Summary: property restrictions

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

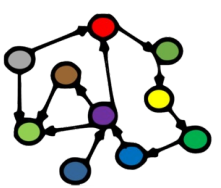


Description logic


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

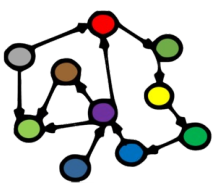
Relation to OWL

- OWL DL and description logic are closely matched
  - everything in OWL DL has a DL-counterpart
  - almost everything in DL can be expressed in OWL DL
- DL is a family of logic systems:
  - some of them correspond to particular OWL profiles (more later)
  - OWL1 DL: $\mathcal{SHOIN}^{(\mathcal{D})}$
  - OWL2 DL: $\mathcal{SROIQ}^{(\mathcal{D})}$



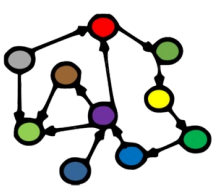
Description logic and other logics

- *Proposition logics* are about *statements* (*propositions*):
  - **“Robin is a StudentAssistant” ⇐ “Robin is a Student” ∧ “Robin is a Teacher”**
- (First order) *predicate logics* are about *predicates* and *objects*:
  - **∀x.(StudentAssistant(x) ⇔ Student(x) ∧ Teacher(x))**
- *Description logics* are about *concepts*:
  - **StudentAssistant ≐ Student ⊓ Teacher**
    - ...and also about *roles* and *individuals*



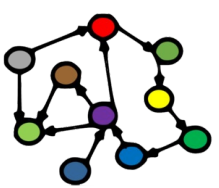
Description logics

- Description Logic (DL)
  - a simple *fragment* of predicate logic
    - ...or, rather, a *family of such fragments*
  - not very *expressive* (“uttrykkskraftig”)
  - but can answers many *decision problems* (rather) quickly
- Suitable for describing *concepts* (“begreper”)
  - formal basis for *OWL DL*
  - can be used to:
    - describe *concepts* (“Tbox”) and their *roles* (“Rbox”)
    - describe *individuals* and their relations (“ABox”)



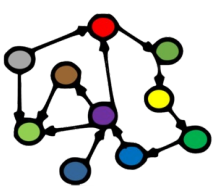
Definition of concepts (“begreper”)

- **InternalCensor ≐ Censor ⊓ Employee**
- **ExternalCensor ≐ Censor ⊓ ¬ Employee**
- **Agent ≐ Person ⊔ Organisation ⊔ Group**
  - concepts: **InternalCensor, Censor, Employee…**
  - definition: ≐
  - conjunction (and): ⊓
  - disjunction (or): ⊔
  - negation (not): ¬
  - nested expressions: **( )**
- **Childless** ≐ ..using Human and Parent..



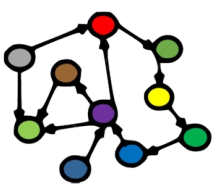
Definition of concepts (“begreper”)

- **InternalCensor ≐ Censor ⊓ Employee**
- **ExternalCensor ≐ Censor ⊓ ¬ Employee**
- **Agent ≐ Person ⊔ Organisation ⊔ Group**
  - concepts: **InternalCensor, Censor, Employee…**
  - definition: ≐
  - conjunction (and): ⊓
  - disjunction (or): ⊔
  - negation (not): ¬
  - nested expressions: **( )**
- **Childless ≐ Human ⊓ ¬ Parent**



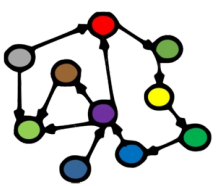
Types of concepts (“begreper”)

- **InternalCensor ≐ Censor ⊓ Employee**
- **ExternalCensor ≐ Censor ⊓ ¬ Employee**
- **Agent ≐ Person ⊔ Organisation ⊔ Group**
  - atomic (or basic, primitive) concepts:
    Censor, Employee, Person…
    - only used on the r.h.s. of definitions
  - concept expressions (complex concepts):
    Censor ⊓ Employee, ¬ Employee…
    - only used on the r.h.s. of definitions
  - defined (and named) concepts:
    InternalCensor, ExternalCensor, Agent…
    - defined on the *l.h.s. (left-hand side)* of definitions



Atomic, complex and defined concepts

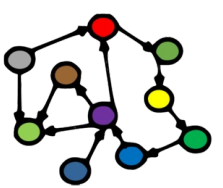
- Atomic (or basic) concepts
  - given, always named
  - can only be used on the r.h.s. (right-hand side) of a $\doteq$ definition
  - correspond to simple OWL classes
- Concept expressions
  - expressed using other concepts (and roles)
  - can only be used on the r.h.s. (right-hand side) of a $\doteq$ definition
  - correspond to complex OWL classes
- Defined concepts can also be named
  - must appear on the l.h.s. (right-hand side) of a $\doteq$ definition
  - concept_name $\doteq$ concept_expression
- ...similar distinction between atomic and defined roles



Roles

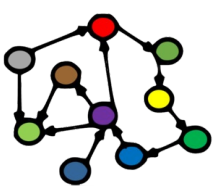
An atomic (or basic) role

- **President** ≐ **Person** ⊓ **∃presidentOf.**⊤
- **Independent** ≐ **Person** ⊓ **¬∃hasParty.**⊤
- **PresidentOfGovernment** ≐ **Person** ⊓ **∃presidentOf.Nation**
  - roles: **presidentOf, hasParty…**
  - universal concept (“top”): ⊤
  - existential restriction: **∃**
- **Grandparent** ≐ ..using Human, hasChild, Parent..
- **Grandparent** ≐ ..using only Human, hasChild..
- **Uncle** ≐ ..using Male, hasSibling, hasChild..



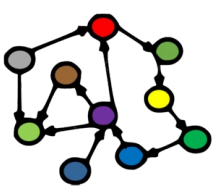
Roles

- **President** ≐ **Person** ⊓ **∃presidentOf.**⊤
- **Independent** ≐ **Person** ⊓ **¬∃hasParty.**⊤
- **PresidentOfGovernment** ≐ **Person** ⊓ **∃presidentOf.Nation**
  - roles: **presidentOf, hasParty…**
  - universal concept (“top”): ⊤
  - existential restriction: **∃**
- **Grandparent** ≐ **Human** ⊓ **∃hasChild.Parent**
- **Grandparent** ≐ ..using only Human, hasChild..
- **Uncle** ≐ ..using Male, hasSibling, hasChild..



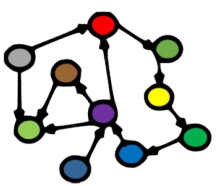
Roles

- **President** ≐ **Person** ⊓ **∃presidentOf.**⊤
- **Independent** ≐ **Person** ⊓ **¬∃hasParty.**⊤
- **PresidentOfGovernment** ≐ **Person** ⊓ **∃presidentOf.Nation**
  - roles: **presidentOf, hasParty…**
  - universal concept (“top”): ⊤
  - existential restriction: **∃**
- **Grandparent** ≐ **Human** ⊓ **∃hasChild.Parent**
- **Grandparent** ≐ **Human** ⊓ **∃ hasChild.∃ hasChild.**⊤
- **Uncle** ≐ ....using Male, hasSibling, hasChild....



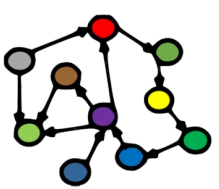
Roles

- **President** ≐ **Person** ⊓ **∃presidentOf.**⊤
- **Independent** ≐ **Person** ⊓ **¬∃hasParty.**⊤
- **PresidentOfGovernment** ≐ **Person** ⊓ **∃presidentOf.Nation**
  - roles: **presidentOf, hasParty…**
  - universal concept (“top”): ⊤
  - existential restriction: **∃**
- **Grandparent** ≐ **Human** ⊓ **∃hasChild.Parent**
- **Grandparent** ≐ **Human** ⊓ **∃ hasChild.∃ hasChild.**⊤
- **Uncle** ≐ **Male** ⊓ **∃ hasSibling.∃ hasChild.**⊤



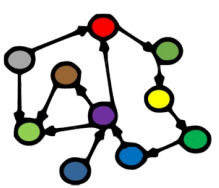
Null concept

- **Person** ⊓ **Group** ⊑ ⊥
  - null concept (“bottom”): ⊥
  - subsumption (sub concept): ⊑
- ⊑ is used for *subsumption axioms*
  - or: containment / specialisation axioms
- ≐ is used for *definitions* (or just ≡)
  - ≡ is also used for *equivalence axioms*
- Note the use of . . . ⊑ ⊥ (“subsumption of bottom”) to say that *something is not the case*



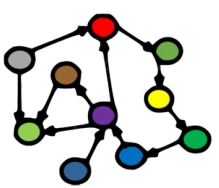
Null concept

- **Person** ⊓ **Group** ⊑ ⊥
  - null concept (“bottom”): ⊥
  - subsumption (sub concept): ⊑
- ⊑ is used for *subsumption axioms*
  - or: containment / specialisation axioms
- ≐ is used for *definitions* (or just ≡)
  - ≡ is also used for *equivalence axioms*
- Note the use of . . . ⊑ ⊥ (“subsumption of bottom”) to say that something is not the case
- *This is a DL axiom*
  - so far we have just *defined* concepts



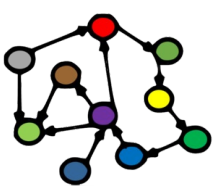
Axioms

- $\doteq$ is used for *definitions* of new concepts (and thus not for axioms)
- $\equiv$ is used for *equivalence axioms* about concepts
  - ...but some authors used it for *definitions* too :-/
- Axioms are equivalences or subsumptions:
  - *subsumption* axioms ($\sqsubseteq$):
    - composite concept (role) expressions on both sides
  - *equivalence axioms* ($\equiv$):
    - composite concept (role) expressions on both sides
    - corresponds to: $\mathtt{C} \sqsubseteq \mathtt{D}, \mathtt{D} \sqsubseteq \mathtt{C}$



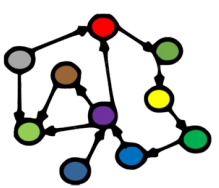
More role definitions

- **RepublicanCommittee** ≐ **Group** ⊓ **∀member.Republican**
  - universal restriction: **∀**
- **Monotheist** ≐ **=1 believesInDeity.**⊤
- **Polygamist** ≐ **≥3 hasSpouse.**⊤
  - number restrictions: **=, ≥, ≤**
- **Narcissist** ≐ **∃hasLoveFor.Self**
  - **self references**: **Self**
- **MassMurderer** ≐ ...using hasKilled, Human...



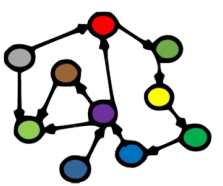
More role definitions

- **RepublicanCommittee ≐ Group ⊓ ∀member.Republican**
  - universal restriction: **∀**
- **Monotheist ≐ =1 believesInDeity.⊤**
- **Polygamist ≐ ≥3 hasSpouse.⊤**
  - number restrictions: **=, ≥, ≤**
- **Narsissist ≐ ∃hasLoveFor.Self**
  - **self references**: **Self**
- **MassMurderer ≐ ≥4 hasKilled.Human**



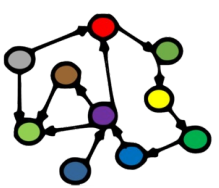
Inverse and transitive roles

- **StrayDog** ≐ **Dog** ⊓ ¬∃**ownerOf**$^{-}$.⊤
- **hasParent** ≐ **hasChild**$^{-}$
- **PureBred** ≐ ∀**hasParent***.**PureBred**
  - inverse role: **hasChild**$^{-}$
  - transitive role: **hasParent***
- **Niece** ≐ ..Woman, hasChild, hasSibling..



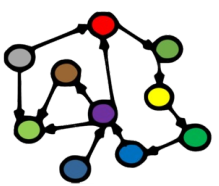
Inverse and transitive roles

- $\mathbf{StrayDog} \doteq \mathbf{Dog} \sqcap \neg\exists \mathbf{ownerOf}^{-}.\top$
- $\mathbf{hasParent} \doteq \mathbf{hasChild}^{-}$
- $\mathbf{PureBred} \doteq \forall \mathbf{hasParent}^{*}.\mathbf{PureBred}$
  - inverse role: $\mathbf{hasChild}^{-}$
  - transitive role: $\mathbf{hasParent}^{*}$
- $\mathbf{Niece} \doteq \mathbf{Woman} \sqcap \exists \mathbf{hasChild}^{-}.\mathbf{hasSibling}.\top$
- *We just defined a role!*
  - until now, we have only defined *concepts*



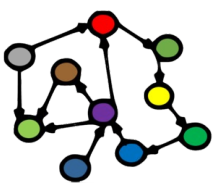
Composite roles

- Similar to composite concepts, e.g.:
  - **holdsPresidency ≐ hasParty⁻ o presidentOf**
  - **hasLovedChild ≐ hasChild ⊓ hasLoveFor**
  - **hasBrother ≐ (hasSibling | Male)**
- Not all supported by OWL-DL and the reasoning engines
  - they can have “bad decision problems”
    - i.e., they compute slowly or intractably
  - ...with some exceptions
- **hasDaughter** ≐ ..using hasChild, Female..



Composite roles

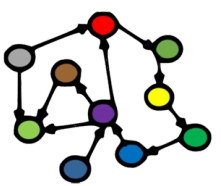
- Similar to composite concepts, e.g.:
  - **holdsPresidency ≐ hasParty⁻ o presidentOf**
  - **hasLovedChild ≐ hasChild ⊓ hasLoveFor**
  - **hasBrother ≐ (hasSibling | Male)**
- Not all supported by OWL-DL and the reasoning engines
  - they can have “bad decision problems”
    - i.e., they compute slowly or intractably
  - ...with some exceptions
- **hasDaughter ≐ (hasChild | Female)**



Putting it together

- $\mathtt{Source} \doteq \exists\ \mathtt{hasSource}^{-}.\mathtt{Content}$
- $\mathtt{TrustedContent} \doteq \exists\ \mathtt{hasSource}.\mathtt{TrustedSource}$
- $\mathtt{VerifiedContent} \doteq \exists\ \mathtt{verifiedBy}.\mathtt{FactChecker}$
- $\mathtt{DebunkedContent} \doteq \exists\ \mathtt{debunkedBy}.\mathtt{FactChecker}$
- $\mathtt{UnreliableSource} \doteq \exists\ \mathtt{hasSource}^{-}.\mathtt{DebunkedContent}$
- $\mathtt{VerifyingSource} \doteq \exists\ \mathtt{hasSource}^{-}.\mathtt{VerifiedContent} \sqcap \forall\ \mathtt{hasSource}^{-}.\mathtt{VerifiedContent}$

An *acyclic*, *definitional* TBox



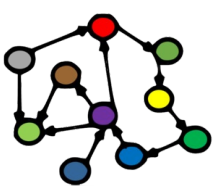
Putting it together

- $\mathtt{Source} \doteq \exists\, \mathtt{hasSource}^{-}.\mathtt{Content}$
- $\mathtt{TrustedContent} \doteq \exists\, \mathtt{hasSource}.\mathtt{TrustedSource}$
- $\mathtt{VerifiedContent} \doteq \exists\, \mathtt{verifiedBy}.\mathtt{FactChecker}$
- $\mathtt{DebunkedContent} \doteq \exists\, \mathtt{debunkedBy}.\mathtt{FactChecker}$
- $\mathtt{UnreliableSource} \doteq \exists\, \mathtt{hasSource}^{-}.\mathtt{DebunkedContent}$
- $\mathtt{VerifyingSource} \doteq \exists\, \mathtt{hasSource}^{-}.\mathtt{VerifiedContent} \sqcap \forall\, \mathtt{hasSource}^{-}.\mathtt{VerifiedContent}$

Concept expressions of atomic concepts

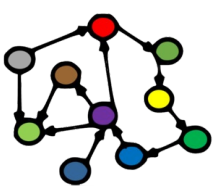
Defined concepts

Acyclic and unequivocal!



Expanded definitional TBox

- Acyclicity: no cyclic definitions in the TBox (“Terminology Box”)
- Unequivocality: each named defined term is only used on the l.h.s. of a single definition
- Concept expansion:
  - every concept can be written as an expression of only atomic concepts
  - algorithm:
    - start with the expression that defines the concept
    - recursively replace all the defined concepts used in the expression with their definitions
    - halt when only atomic concepts remain



Expanded definitional TBox

- $\mathbf{Source} \doteq \exists\, \mathbf{hasSource^{-}.Content}$
- $\mathbf{TrustedContent} \doteq \exists\, \mathbf{hasSource.TrustedSource}$
- $\mathbf{VerifiedContent} \doteq \exists\, \mathbf{verifiedBy.FactChecker}$
- $\mathbf{DebunkedContent} \doteq \exists\, \mathbf{debunkedBy.FactChecker}$
- $\mathbf{UnreliableSource} \doteq \exists\, \mathbf{hasSource^{-}}.\ \exists\, \mathbf{debunkedBy.FactChecker}$
- $\mathbf{VerifyingSource} \doteq \exists\, \mathbf{hasSource^{-}}.\ \exists\, \mathbf{verifiedBy.FactChecker} \sqcap \forall\, \mathbf{hasSource^{-}}.\ \exists\, \mathbf{verifiedBy.FactChecker}$

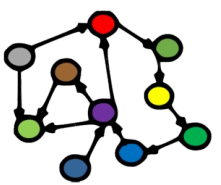
Only basic concepts on the right hand sides!

Types of axioms

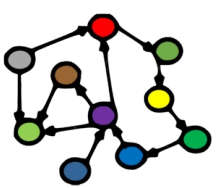
- Terminology axioms (TBox)
  - subsumptions: $C \sqsubseteq D$
  - equivalences: $C \equiv D$
    corresponds to: $C \sqsubseteq D,\ D \sqsubseteq C$
- Role axioms (RBox)
  - e.g., subsumptions: $R \sqsubseteq S$
- Individual assertion axioms (ABox)
  - class assertions: $a:C$
  - role assertions: $\langle a,b \rangle : R$
- Knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ or $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$

C and D are *expressions*!

a and b are *individuals*.
R and S are *roles*!

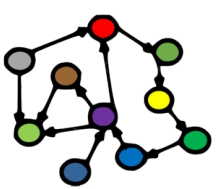


Decision Problems



Reasoning over knowledge bases

- *What more can we do with ontologies?*
- For example:
  - given a *source ontology* that describes media content along with its sources and trustworthiness
  - we can *answer questions* like, e.g.:
    - is trusted content a type of content?
    - can content be both verified and debunked?
    - is all verified content trusted?
  - *competency questions* are what we want an ontology to answer
  - *DL offers a clear and compact way or representing and reasoning about questions such as these!*

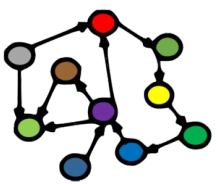


Decision problems

- A computational problem with a yes/no answer, e.g.
  - is C *subsumed* by D? $\mathcal{K} \models \mathtt{C} \sqsubseteq \mathtt{D}$
  - are C and D *consistent*? $\mathcal{K} \models \mathtt{a:(C} \sqcap \mathtt{D)}$
  - are C and D *equivalent*? $\mathcal{K} \models \mathtt{C} \equiv \mathtt{D}$
  - are C and D *disjoint*? $\mathcal{K} \models \mathtt{C} \sqcap \mathtt{D} \sqsubseteq \bot$
  - does *a belong* to C: $\mathcal{K} \models \mathtt{a:C}$?
  - is *a R-related* to *b:* $\mathcal{K} \models \mathtt{<a,b>:R}$?

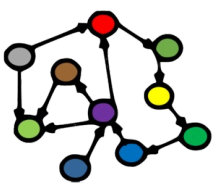
C and D are classes, a and b are individuals. R is a role!

- Given a knowledge base $\mathcal{K}$, reasoning engines can give yes / no answers
  - ...but not all decision problems are *decidable*
  - ...or have tractable *complexity*
  - *depends on the expressions used!*



Decision problems in practice

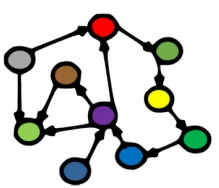
- Description logic is implemented by *reasoning engines/OWL reasoner*
  - *e.g., HermiT and Pellet*
  - distinct from *inference engines*, such as OWL-RL
- Protegé-OWL
  - comes with HermiT, more plugins can be installed
- Solves decision problems, e.g.,
  - classifiy individuals
  - find subclass relationships (subsumptions)
  - find unsatisfiable classes (concepts)
  - detect inconsistencies



Manchester OWL syntax

Manchester OWL-syntax

- A simple DL notation without special symbols
  - used by Protege-OWL to construct classes
  - similar to DL syntax
- **Class: InternalCensor**
  EquivalentTo: Censor and Employee
- **Class: ExternalCensor**
  EquivalentTo: Censor and not Employee
- **Class: Agent**
  EquivalentTo: Person or Organisation or Group
- Can be used to *serialise* complete ontologies
  - ...we will look mostly at TBox expressions
- http://www.w3.org/TR/owl2-manchester-syntax/



Comparison

- DL:

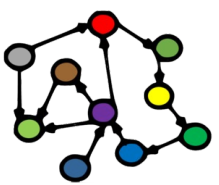
  ExternalCensor ≐ Censor ⊓ ¬Employee

- Machester OWL:

```
Class: ExternalCensor
        EquivalentTo: Censor and not Employee
```

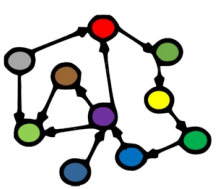
- Turtle:

```
uib:ExternalCensor owl:equivalentClass
        owl:intersectionOf (
                uib:Censor
                [    a owl:Class ;
                     owl:complementOf uib:Employee
                ]
        ) .
```



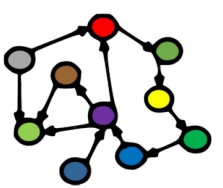
Roles in Manchester OWL syntax

- **Class: President**
  EquivalentTo:
  Person and presidentOf some owl:Thing
- **Class: PresidentOfGovernment**
  EquivalentTo:
  Person and presidentOf some Nation
- **Class: Independent**
  EquivalentTo:
  Person and not hasParty some owl:Thing
  - universal concept (top): **owl:Thing**
  - existential restriction: **some**



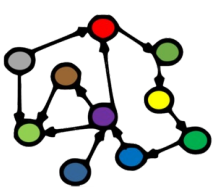
Null concept in Manchester OWL syntax

- **Class:** <class-name>
  EquivalentTo: Person **and** Group
  SubClassOf: **owl:Nothing**
  - null concept (bottom): **owl:Nothing**
  - subsumption (subconcept): **SubClassOf:**
  - equivalence: **EquivalentTo:**
    - ...used both for *definitions* and for *axioms*



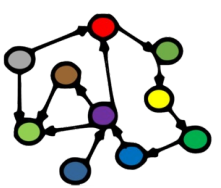
More roles in Manchester OWL syntax

- **Class:** RepublicanCommittee
  EquivalentTo:
  Group **and** member **only** Republican
  - value restriction: **only**
- **Class:** Monotheist
  EquivalentTo: Person **and**
  believesInDeity **exactly** 1
- **Class:** Polygamist
  EquivalentTo: hasSpouse **min** 3
  - number restriction: **exactly**, **min**, **max**
- **Class:** Narcissist
  EquivalentTo: loves **some** Self



Inverse, symmetric and transitive roles

- **Class:** StrayDog
  EquivalentTo: Dog **and** not **inverse** hasOwner **some** Person
- **Class:** hasParent
  EquivalentTo: **inverse** hasChild
- **ObjectProperty:** hasSibling
  Characteristics: **Symmetric**
- **ObjectProperty:** hasAncestor
  Characteristics: **Transitive**
- inverse role: **inverse**
- symmetric role:
  Characteristics: **SymmetricProperty**
- transitive role:
  Characteristics: **TransitiveProperty**



After Easter:
Graph Embeddings