Semantic Data

Chapter 6 : The Web Ontology Language OWL

Jean-Louis Binot

Sources and recommended readings

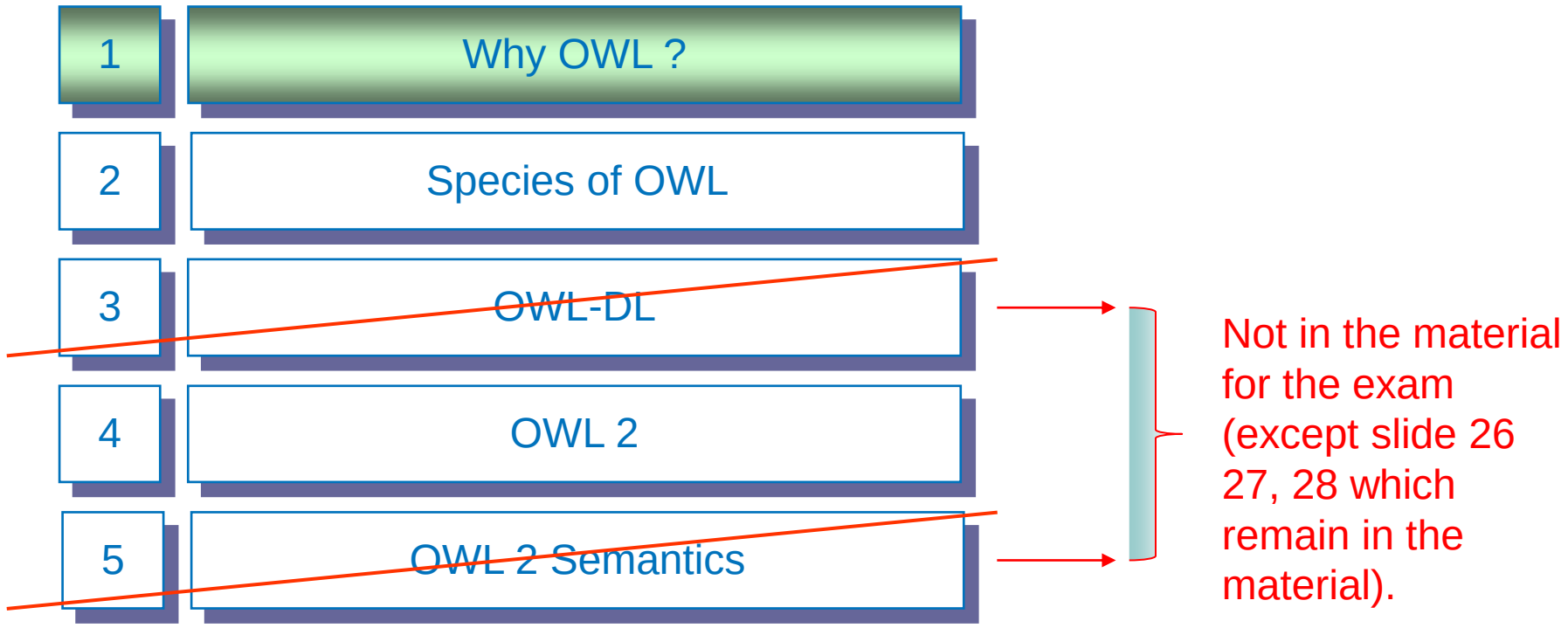
- There are no additional required references for this chapter for the theory.
  Some sections have been taken out of the material for the exam.

The Manchester syntax is not covered in theory but is useful in practice (in Protégé).
Students are expected to acquire the knowledge they need for the project.

- Sources and useful additional readings :
  - The presentation of OWL DL is based on the *OWL guide 2004*.
  - The presentation of OWL 2 is based on the OLW 2 primer and the OWL 2 new features and rationale.
  - *Web Data Management* (*Abiteboul et al. 2011)* covers succinctly the link between DL and OWL.
  - *An introduction to description logic (Baader et al. 2017), chapter 8*, covers the mappings between OWL and description logics.

- University courses having partially inspired ideas and examples for this chapter :
  - *Description Logics part 1 Languages, I. Horrocks, Oxford University.*
  - *Apprentissage symbolique et web sémantique, B. Amman, Université Pierre et Marie Curie, Paris.*
  - *OWL 2 Web Ontology Language Revised, S. Wandelt, Humboldt-Universität zu Berlin.*
  - *Semantic Web Technologies, H. Paulheim, Universität Manheim.*

Agenda

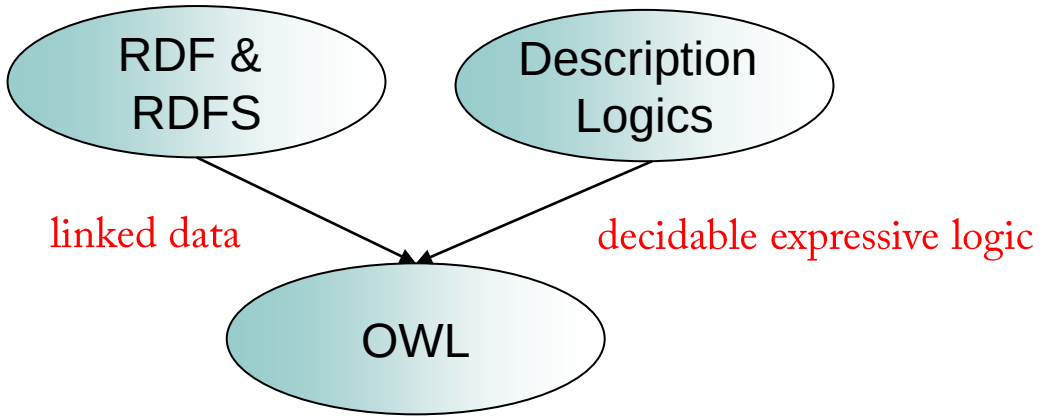

1
Why OWL ?
2
Species of OWL
3
OWL-DL
4
OWL 2
5
OWL 2 Semantics
Not in the material for the exam (except slide 26 27, 28 which remain in the material).

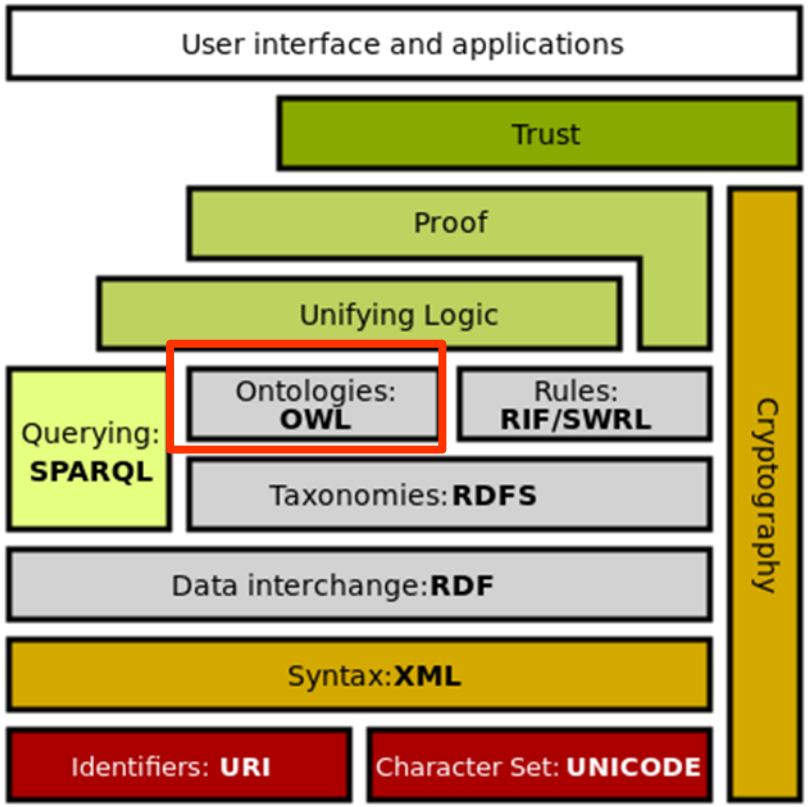
OWL in the W3C standards stack

RDFS supports only lightweight ontologies (cf. chapter 4).

OWL (Web Ontology Language) is a language for building full Web ontologies with model-theoretic semantics.

- OWL uses RDF and RDFS to manipulate linked data.
- OWL (in its main variant) uses description logics to support logical inferences.
- This "merging of two worlds" raises a few difficulties.





The semantic web stack of standards

Why not just build OWL as an extension of RDFS ?

- RDFS modeling primitives are sources of unwanted complexity.
  - Constructions such as *rdfs:Class* and *rdf:Property* are very powerful :

    A class may be an instance of itself; a property may be applied to itself.
  - Combined with OWL's language extensions, this may create undecidability. W3C has given a choice :

- OWL-Full : total freedom to use RDF, for those wanting to use that freedom.

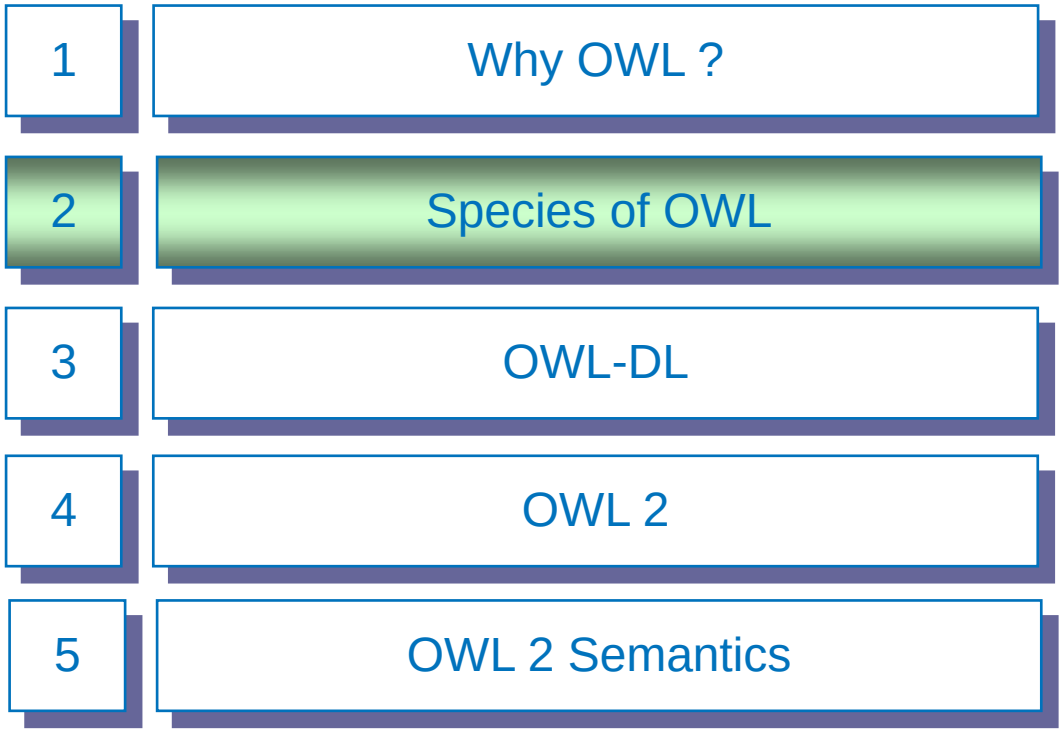
  But : not reasoner exist for all features; the entailment problem in OWL-Full is undecidable !

- OWL-DL : The model-theoretic semantics of description logics apply; reasoners exist.

  But: RDF is restricted to what is needed to express OWL DL concepts and axioms.

  owl:Class is a proper subclass of rdfs:Class. A class cannot be also an instance. Other restrictions apply.

Agenda



Species of OWL

- OWL 1 (2004): three increasingly expressive sublanguages for distinct uses:
  - OWL Full: maximum expressiveness; syntactic freedom of RDF (e.g., classes can also be instances).
    No reasoner supports all features of OWL Full.
  - OLW DL: maximum description logic expressivity while guaranteeing completeness and decidability.
    Every entailment can be computed by a reasoner.
  - OWL Lite: classification hierarchy and simple constraints. Quick migration path for thesauri and taxonomies.
- All these languages support upward compatibility :
  - Every legal *OWL Lite* ontology is a legal *OWL DL* ontology.
    Every valid *OWL Lite* conclusion is a valid *OWL DL* conclusion.
  - Every legal *OWL DL* ontology is a legal *OWL Full* ontology.
    Every valid *OWL DL* conclusion is a valid *OWL Full* conclusion.
- OLW 2 (2009), evolution of OWL 1.

Mapping with description logics

Reminder : $\mathcal{S}$ denotes logic $\mathcal{ALC}$ extended with transitive roles :

- $\mathcal{S}$ = $\mathcal{ALCR}^{+}$ (cf. chapter 4 for a description of the naming scheme).

OWL <-> description logics correspondences :

- OWL Lite : <-> $\mathcal{SHIF}$ (role hierarchies, inverse roles, functions)
- OWL – DL : <-> $\mathcal{SHOIN}(\mathcal{D})$ (role hierarchies, nominals, inverse roles, number restrictions, data types)
- OWL – Full : <-> undecidable
- OLW 2 : <-> $\mathcal{SROIQ}(\mathcal{D})$ (complex roles and role axioms, nominals, inverse roles, qualified number restrictions, data types)

OWL 2 is the most expressive OWL language where inferencing is still decidable.

However, many large ontologies are still in OWL1.

OLW Syntaxes

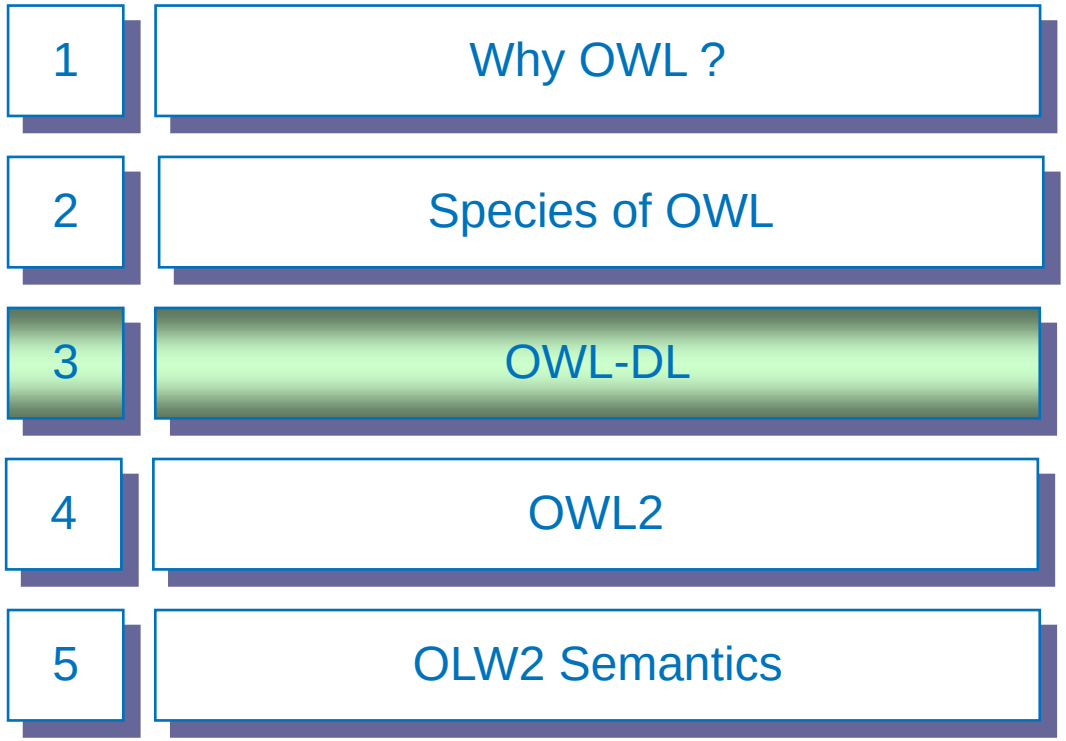
Examples of the first 5 syntaxes can be found in the OLW2 Primer

- RDF/XML syntax :
  - Still the only *normative* syntax.
- Turtle syntax : straightforward Turtle version of the RDF/XML Syntax.

We will only cover Turtle

- Functional Style syntax :
  - Prefix-syntax, given as formal grammar; clean, adjustable, modifiable, easily parsable; used in W3C Specs.
- Manchester syntax: user-friendly syntax, used in Protégé.
- OWL/XML syntax :
  - Notational variant of Functional Style Syntax. Does not use RDF triples, but XML tree structure.
- Graphical syntax based on UML conventions.
- Probably too many syntaxes : complex design choices, multiplication of conversions…

Agenda



Namespaces and prefixes

- A typical OWL ontology starts with namespace declarations (as RDF).
- The OWL namespace is defined by :

  PREFIX owl: <http://www.w3.org/2002/07/owl#> .

- Other namespaces seen in previous chapters will be reused, among which :

  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

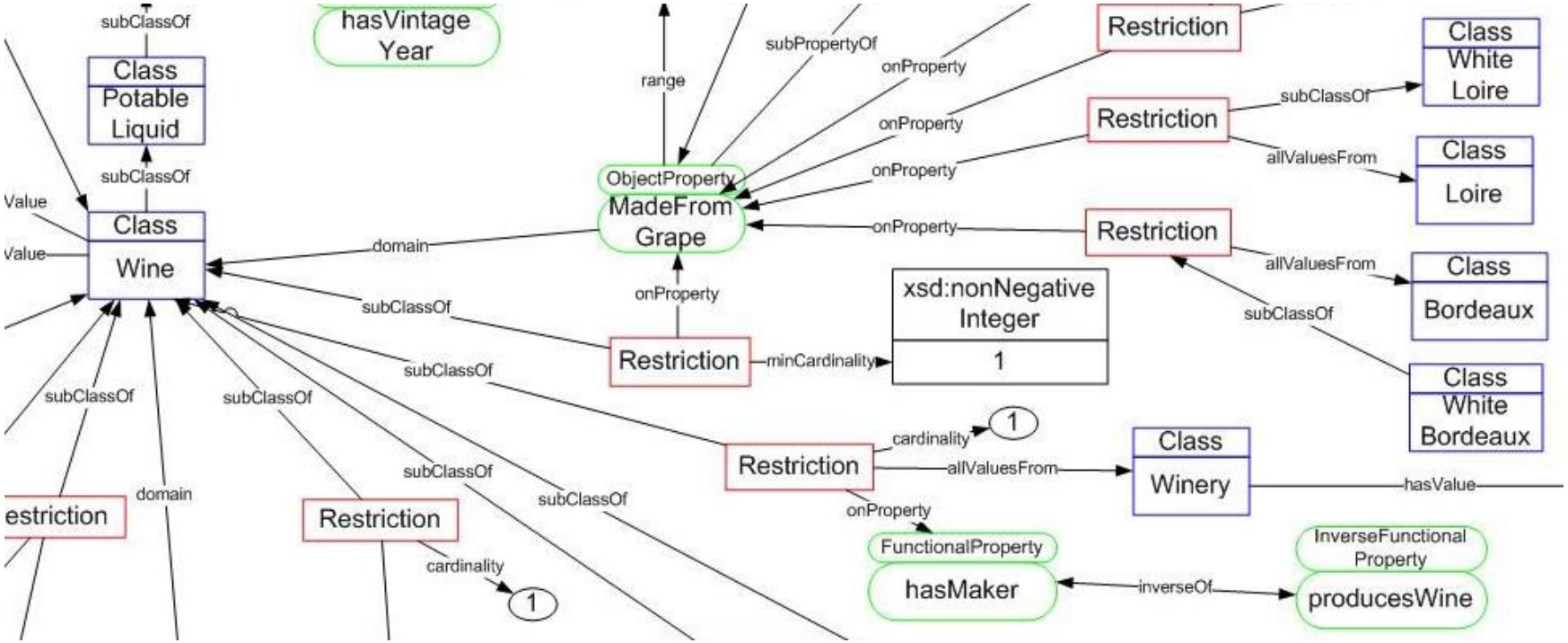
  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

- Examples from the wine ontology (downloadable) use the following prefixes :

  PREFIX vin: <http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#> .

  PREFIX food: <http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#> .

Example ontology : the Wine ontology



A wine is a potable liquid produced by one maker of type winery, and made from at least one type of grape…

(Source for the ontology: wine and food)

Source for image: Owl language guide: http://www.w3.org/TR/owl-guide.rdf, using VisioOWL)

Expressing DL in OWL, an example

- DL axioms :

$\text{Wine} \subseteq \text{PotableLiquid} \sqcap \forall \text{hasMaker.Winery}$
$\text{ChateauChevalBlanc} : \text{Winery}$

- In OWL :

```
vin:PotableLiquid a owl:Class .

vin:Winery a owl:Class .

vin:Wine a owl:Class ;
                rdfs:subClassOf vin:PotableLiquid ;
                rdfs:subClassOf  [ a owl:Restriction ;
                                   owl:onProperty vin:hasMaker ;
                                   owl:allValuesFrom vin:Winery ] .

vin:ChateauChevalBlanc a vin:Winery .
```

- A class may be :
  - Identified by name (an URI);
  - Anonymous (RDF [] blank node notation).
- Syntactic constructs :
  - rdf:type (a) : to type resources / declare instances.
  - rdfs:subClassOf : specialization hierarchy.
  - owl:Class : the type Class.

Thing and Nothing

- Owl:Thing is the universal concept, $\top$.
  - Every individual in the interpretation domain is a member of the class owl:Thing.
  - Each user-defined class is implicitly a subclass of owl:Thing.

- Owl:Nothing is the inconsistent concept, $\bot$.
  - No individual of the interpretation domain belongs to Owl:Nothing.

Types of class restrictions

An OWL class can be defined by :

1. Constraints on properties of the instances (intensional definition);
2. Enumeration of its instances (extensional definition);
3. Set relationships on the extensions of other classes.

1. Constraints on the properties of the instances

- Structure:

<class> rdfs:subClassOf <restriction>

<restriction> ::= [a owl:Restriction ;
owl:onProperty <property> ;
<restriction name> <restriction argument>]

- Example :

vin:Burgundy a owl:Class ;
rdfs:subClassOf [a owl:Restriction ;
owl:onProperty vin:hasSugar ;
owl:hasValue vin:Dry] .

  - In DL :

Burgundy ⊆ ∃hasSugar.{Dry}

Constraints on the properties of the instances ./.

Constraint	OWL	DL
Property value type restrictions		
Existential restriction	[owl:onProperty r ; owl:someValuesFrom C]	∃r.C
Universal restriction	[owl:onProperty r ; owl:allValuesFrom C]	∀r.C
Cardinality restrictions		
Minimum cardinality	[owl:onProperty r ; owl:minCardinality n]	≤n r (*)
Maximum cardinality	[owl:onProperty r ; owl:maxCardinality n]	≥n r (*)
Property value restriction	[owl:onProperty r ; owl:hasValue v]	∃r.{v}

(*): n is written in RDF format, e.g. *"1"^^xsd*

2. Enumerated classes

- A class can be specified via a direct enumeration using owl:oneOf :
  - This completely specifies the class extension : no other individuals can be declared to belong to the class.

```
vin:WineColor a owl:Class ;
                owl:oneOf ( vin:Red vin:Rose vin:White) ;
                rdfs:subClassOf vin:WineDescriptor .
```

- In DL ($\mathcal{O}$) : the set (or "one-of") constructor :

$$\text{WineColor} \equiv \{\text{Red}, \text{Rose}, \text{White}\}$$

3. Class definitions using set relationships

Class definitions may use inclusion, equivalence, disjoint relationships.

- Inclusion : rdfs:subClassOf. — In DL : a GCI axiom : $C \sqsubseteq D$.
- Equivalence : owl:equivalentClass. — In DL : an equivalence axiom : $C \equiv D$.
- Disjoint classes : owl:disjointWith. — In DL : $A \sqcap B \sqsubseteq \bot$.

Example :

```
food:Fowl a owl:Class ;
        rdfs:subClassOf food:EdibleThing ;
        owl:disjointWith food:Fruit ;
        owl:equivalentClass food:Poultry .
```

3. Class definitions using set relationships ./.

- Complex class definitions may use the Boolean set operators :
  - DL: ⊓, ⊔, ¬ .
  - OWL : owl:intersectionOf, owl:unionOf, owl:complementOf .
- Structure :

  [rdf:type owl :Class; <set operator> (<class> … <class>)]
- Example :

  vin:WhiteWine owl:equivalentClass
  [a owl:Class ;
  owl:intersectionOf (vin:Wine [a owl:Restriction ; owl:onProperty vin:hasColor ;
  owl:hasValue vin:White])] .

  DL: WhiteWine ≡ Wine ⊓ ∃hasColor.{White}

A larger example

- Let us revise the definition of *Wine* to cover the following information :

  "*A wine is a potable liquid made from at least one grape; its maker must be a winery; it must be located in one region*".

```
vin:Wine a owl:Class ;
        rdfs:subClassOf food:PotableLiquid ,
        [ a owl:Restriction ;  owl:onProperty vin:locatedIn ; owl:someValuesFrom vin:Region ] ,
        [ a owl:Restriction ;  owl:onProperty vin:hasMaker ; owl:allValuesFrom vin:Winery ] ,
        [ a owl:Restriction ;  owl:onProperty vin:madeFromGrape ; owl:minCardinality
                                                                  "1"^^xsd:nonNegativeInteger] .
```

- In DL :
  - $\text{Wine} \sqsubseteq \text{PotableLiquid} \sqcap \exists \text{locatedIn}.\text{Region} \sqcap \forall \text{hasMaker}.\text{Winery} \sqcap \geq 1\ \text{madeFromGrape}$

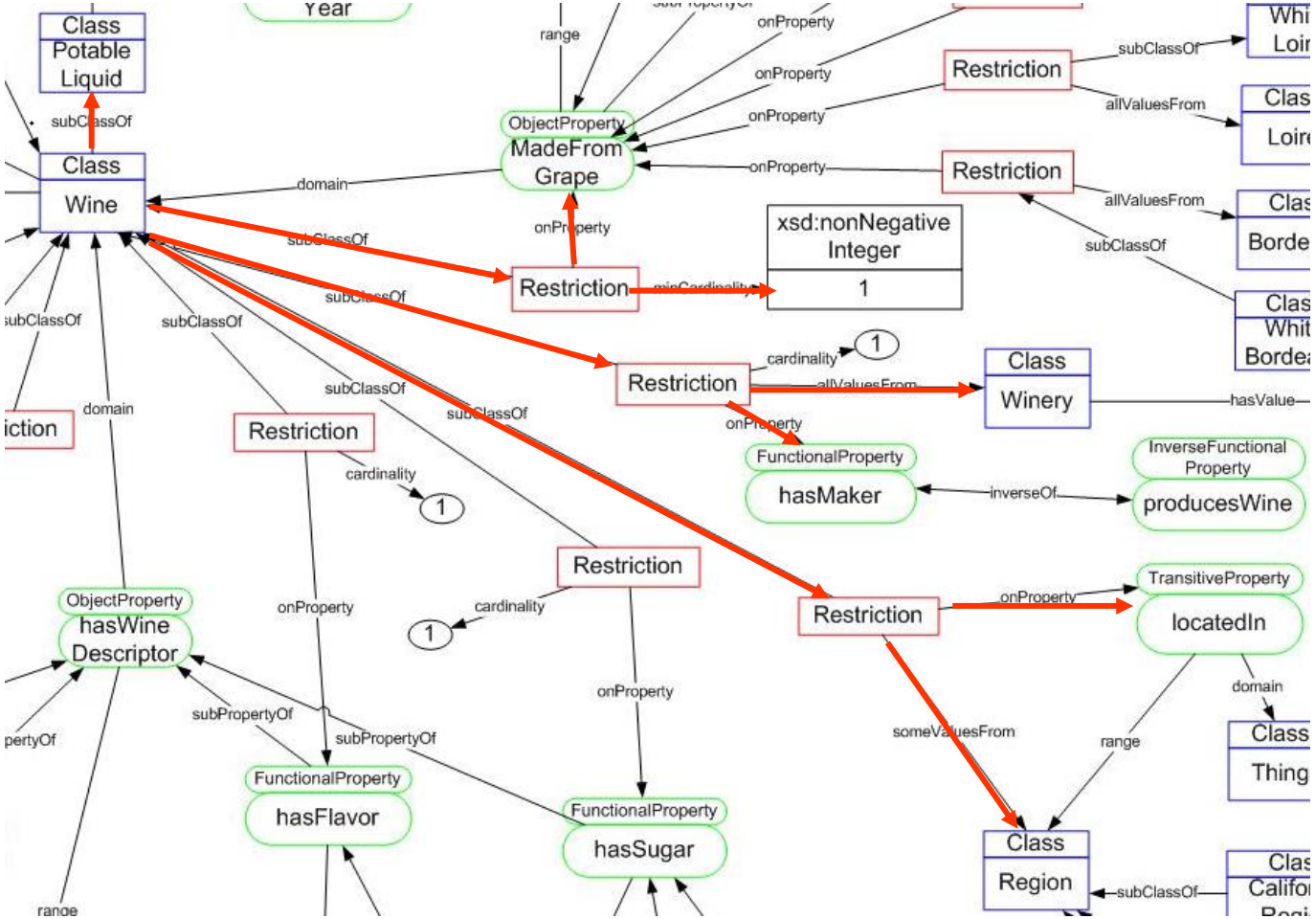
Graph-based visualisation

vin:Wine rdf:type owl:Class ;
rdfs:subClassOf food:PotableLiquid ,

[rdf:type owl:Restriction ;
owl:onProperty vin:locatedIn ;
owl:someValuesFrom vin:Region] ,

[rdf:type owl:Restriction ;
owl:onProperty vin:hasMaker ;
owl:allValuesFrom vin:Winery] ,

[rdf:type owl:Restriction ;
owl:onProperty vin:madeFromGrape
; owl:minCardinality
"1"^^xsd:nonNegativeInteger] .



Properties

Two types of properties are distinguished:

- Object properties : type owl:ObjectProperty.
  - Express relations between instances of two classes.
- Datatype or value properties : type owl:DatatypeProperty.
  - Express relations between instances of classes and datatypes (RDF or XML Schema types).

Both are subclasses of rdf:Property.

OWL property characteristics

- RDFS property hierarchies, domain and range are still usable.

```
vin:madeFromGrape a owl:ObjectProperty ;
        rdfs:subPropertyOf food:madeFromFruit ;
        rdfs:domain vin:Wine ;
        rdfs:range vin:WineGrape .

vin:FormanCabernetSauvignon vin:madeFromGrape vin:CabernetSauvignonGrape .
```

  - Property hierarchies in DL ($\mathcal{H}$) : a role inclusion axiom: $\text{madeFromGrape} \sqsubseteq \text{madeFromFruit}$
  - Domain in DL : $\exists \text{madeFromGrape}.\top \sqsubseteq \text{Wine}$
  - Range in DL : $\top \sqsubseteq \forall \text{madeFromGrape}.\text{WineGrape}$

  We can, among others, infer from the example above that FormanCabernetSauvignon is a wine.

- As OWL uses DL, the domain or range can be a complex class (in contrast with RDFS) :
  - Domain : $\exists \text{childOf}.\top \sqsubseteq \text{Father} \sqcup \text{Mother}$

Property axioms

- Property axioms (transitive, symmetric …) are defined by making the property an instance of the appropriate built-in OWL property class :
  - vin:locatedIn a owl:TransitiveProperty . In DL : *Trans(locatedIn)*

  The range of a transitive property must be subsumed by its domain.
  - vin:adjacentRegion a owl:SymmetricProperty . In *DL* : adjacentRegion ≡ adjacentRegion$^-$

  The domain and range of a symmetric property must be the same.
  - vin:hasMaker a owl:FunctionalProperty . In *DL* : $\top \subseteq (\leq 1\ \text{hasMaker})$
- Inverse properties are defined by using the built-in *owl:inverseOf* property :
  - *vin:madeFromGrape owl:inverseOf vin:madeIntoWine .* In DL : madeFromGrape ≡ madeIntoWine$^-$

Revisiting the example from chapter 1

(example after Paulheim, Semantic Web Technologies)

Status with RDFS :

✓ Madrid is a city.

✓ Spain is a country.

✓ Madrid is located in Spain.

✗ Barcelona is not the capital of Spain.

✗ Madrid is not the capital of France.

✗ Madrid is not a country.

- How can we do it with DL and OWL ?

Revisiting the example from chapter 1 ./.

DL

OWL

- Madrid is not the capital of France

City ⊆ ≤ 1 capitalOf.Country

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

:Madrid :capitalOf :France . => inconsistent ontology

- Barcelona is not the capital of Spain

Country ⊆ =1 hasCapital.City (abbreviation for ≥ 1 hasCapital.City ⊓ ≤ 1 hasCapital.City)

:Country a owl:Class ; rdfs:subClassOf [a owl:Restriction ; owl:onProperty :hasCapital ;
owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ; owl:onClass :City] .
:Spain hasCapital :Madrid.
:Madrid owl:differentFrom :Barcelona .

:Spain :hasCapital :Barcelona . => inconsistent ontology

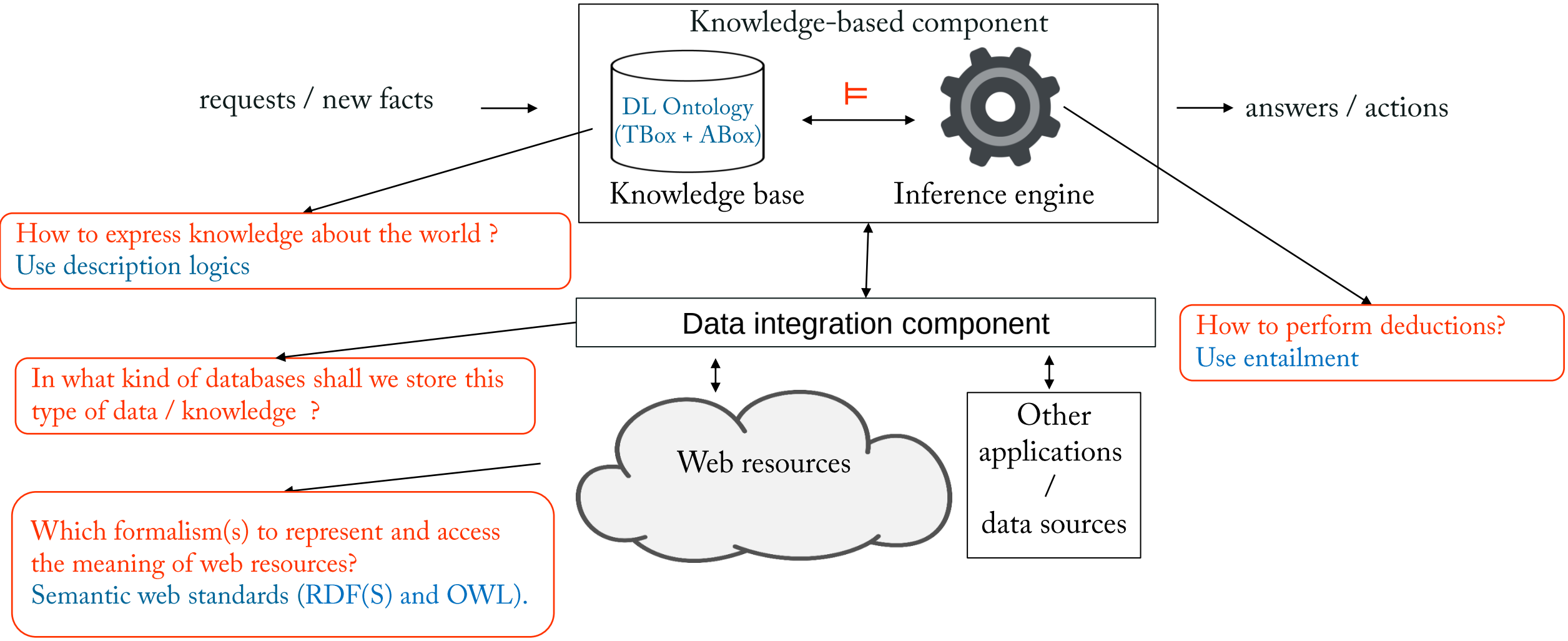
- Madrid is not a country.

City ⊓ Country ⊆ ⊥

:Madrid a :City .
:City owl:disjointWith :Country .

:Madrid a :Country => inconsistent ontology

Updated framework of reference for semantic applications



Ontology management

- Meta-information about the ontology itself can be specified:
  - rdfs:label provides a human-readable version of the ontology name.
  - rdfs:comment provides the obviously needed ability to comment.
  - owl:priorVersion provides information that may be used for version control.
  - owl:imports allows to import other ontologies, bringing their full set of assertions into the current ontology.
- Class declarations are also part of ontology management. All syntaxes except Manchester syntax offer facilities to do so :

vin:Bancroft rdf:type owl:NamedIndividual.

vin:Winery rdf:type owl:Class .

vin:adjacentRegion rdf:type owl:ObjectProperty .

vin:yearValue rdf:type owl:DatatypeProperty .

Ontology mapping

- The Semantic Web is distributed; so is ontology construction.
  - It is common for ontologies to use different names for the same concept, property, or individual.
  - It is common usage to develop ontologies in several parts, which then need to be mapped together.
- Equivalence declarations can be used for that purpose : equivalentClass for classes, equivalentProperty for properties and sameAs for individuals :

:Mary owl:sameAs otherOnt:MaryBrown .
:Adult owl:equivalentClass otherOnt:Grownup .
:hasChild owl:equivalentProperty otherOnt:child .

- To make sure individuals are different, this must be specified.
  - OWL does not have the unique name assumption.

:John owl:differentFrom :Bill .

Agenda

OLW 2

- OLW 1 has been successful; its usage has shown additional needs :
  - A suitable set of built-in datatypes (OLW 1 still relied on XML-Schema datatypes);
  - Additional features identified by users for which effective reasoning algorithms are now available.
- OLW 2 was adopted as W3C recommendation in 2009, backward compatible :
  - All OWL 1 ontologies remain valid OWL2 ontologies, with identical inferences in all practical cases.

OLW 2 relationship to OWL 1

- The overall structure of OWL 2 is very similar to OWL 1. Almost all OWL 2 building blocks were present in OWL 1, sometimes in a different name.
- OWL 2 adds new features with respect to OWL 1 :
  - Syntactic sugar : doesn't change expressiveness or complexity but makes patterns easier to write and allows for more efficient processing in reasoner : disjoint union, disjoint classes (cf. chapter 5), negative assertions.
  - New constructs for properties : cardinality restrictions, property chain inclusion, new axioms (reflexive, asymmetric …).
  - Extended datatypes : richer set of datatypes (e.g. various types of numbers) and datatype restrictions.
  - Punning : the same name can be used as class and as instance (*dog* is a class and an instance of *species*).

    Note : OWL 2 DL treats the two usages as totally different instances of the same name, to preserve decidability.
  - Extended annotations : OWL1 could only annotate the ontology. OWL 2 can annotate classes and properties.
  - Other minor features.

OWL 2 Profiles

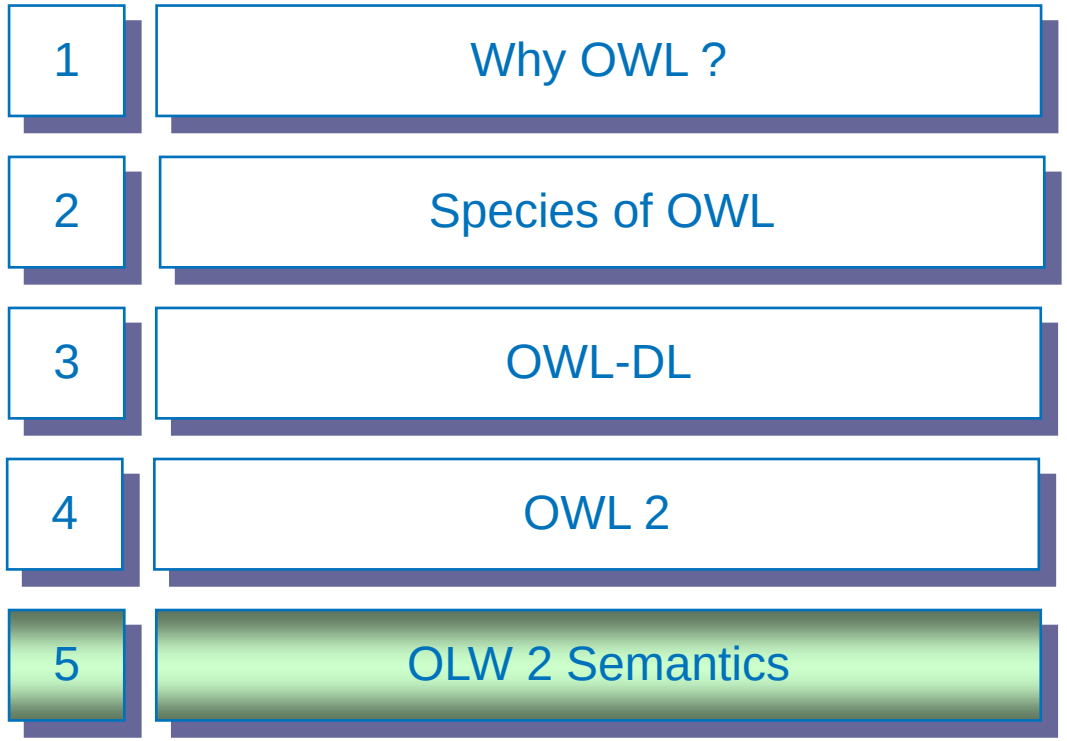
Complexity remains a key factor. OWL 2 offers three distinct tractable profiles:

- EL: fast reasoning services (ptime) for large ontologies (classes, properties).
  - Many applications, in particular in life sciences, have large ontologies : SNOMED, GO …
  - They need to represent complex entities, to propagate properties (e.g., location of diseases …).
  - EL is designed for such ontologies with a large conceptual / taxonomy part.
- QL: Efficient query answering using RDBMs via SQL.
  - Conjunctive query answering in logspace wrt size of data; consistency and subsumption reasoning in ptime.
  - For applications needing inter-operability with database technologies and tools.
  - QL is useful for large datasets already stored in RDBs.
- RL: can be implemented using rule-based technologies (rule-extended DBMSs, Prolog …)
  - Fast reasoning services (ptime).
  - For applications concerned with inter-operability with rule engines operating at RDF level.
  - RL is useful for managing large datasets of RDF triples which can be enriched with rules.

Compatibility of reasoners

- Every conforming OWL 2 DL reasoner is also a conforming reasoner for OWL 2 EL, OWL 2 RL, and OWL 2 QL !
- But they may differ in performance as the OWL 2 DL reasoner is tuned for a more general set of cases.

Agenda



OLW Semantics

- Two semantics are defined for OWL 2 :

1. Direct semantics:
   - Model-theoretic semantics similar to DL SROIQ, extended for datatypes and punning.

     Due to these extensions the direct semantics has received a separate formal definition. We will satisfy ourselves to summarize the mapping between OWL 2 DL and description logics (mapping table next slide).
   - Assigns meaning to OWL 2 DL ontologies (those meeting the restrictions necessary to be translated into DL).
   - Also provides semantics for OWL 1 Lite and OWL 1 DL ontologies and OWL 2 profiles.
   - As description logics, OWL uses the Open World assumption.
2. RDF-based semantics:
   - Assigns RDF-based meaning to all OWL 2 (full) ontologies;
   - For an OWL 2 DL ontology, inferences based on direct semantics are still valid using RDF-based semantics.
   - Will not be covered in this course.

Mapping between OWL and description logics

Abstract Syntax	DL Syntax
Descriptions (C)	
A	A
owl:Thing	$\top$
owl:Nothing	$\bot$
intersectionOf($C_1 \dots C_n$)	$C_1 \sqcap \dots \sqcap C_n$
unionOf($C_1 \dots C_n$)	$C_1 \sqcup \dots \sqcup C_n$
complementOf(C)	$\neg C$
oneOf($o_1 \dots o_n$)	$\{o_1\} \sqcup \dots \sqcup \{o_n\}$
restriction(R someValuesFrom(C))	$\exists R.C$
restriction(R allValuesFrom(C))	$\forall R.C$
restriction(R hasValue(o))	$R : o$
restriction(R minCardinality(n))	$\geqslant n\, R$
restriction(R minCardinality(n))	$\leqslant n\, R$
restriction(U someValuesFrom(D))	$\exists U.D$
restriction(U allValuesFrom(D))	$\forall U.D$
restriction(U hasValue(v))	$U : v$
restriction(U minCardinality(n))	$\geqslant n\, U$
restriction(U maxCardinality(n))	$\leqslant n\, U$
Data Ranges (D)	
D	D
oneOf($v_1 \dots v_n$)	$\{v_1\} \sqcup \dots \sqcup \{v_n\}$
Object Properties (R)	
R	R
inv(R)	R^-
Datatype Properties (U)	
U	U
Individuals (o)	
o	o
Data Values (v)	
v	v

Fig. 14.2. OWL DL Descriptions, Data Ranges, Properties, Individuals, and Data Values

Abstract Syntax	DL Syntax
Class(A partial $C_1 \dots C_n$)	$A \sqsubseteq C_1 \sqcap \dots \sqcap C_n$
Class(A complete $C_1 \dots C_n$)	$A \equiv C_1 \sqcap \dots \sqcap C_n$
EnumeratedClass(A $o_1 \dots o_n$)	$A \equiv \{o_1\} \sqcup \dots \sqcup \{o_n\}$
SubClassOf(C_1 C_2)	$C_1 \sqsubseteq C_2$
EquivalentClasses($C_1 \dots C_n$)	$C_1 \equiv \dots \equiv C_n$
DisjointClasses($C_1 \dots C_n$)	$C_i \sqcap C_j \sqsubseteq \bot, i \not\equiv j$
Datatype(D)	
ObjectProperty(R super(R_1)...super(R_n)	$R \sqsubseteq R_i$
domain(C_1)...domain(C_m)	$\geqslant 1\, R \sqsubseteq C_i$
range(C_1)...range(C_ℓ)	$\top \sqsubseteq \forall R.C_i$
[inverseOf(R_0)]	$R \equiv R_0^-$
[Symmetric]	$R \equiv R^-$
[Functional]	$\top \sqsubseteq \leqslant 1\, R$
[InverseFunctional]	$\top \sqsubseteq \leqslant 1\, R^-$
[Transitive])	$Tr(R)$
SubPropertyOf(R_1 R_2)	$R_1 \sqsubseteq R_2$
EquivalentProperties($R_1 \dots R_n$)	$R_1 \equiv \dots \equiv R_n$
DatatypeProperty(U super(U_1)...super(U_n)	$U \sqsubseteq U_i$
domain(C_1)...domain(C_m)	$\geqslant 1\, U \sqsubseteq C_i$
range(D_1)...range(D_ℓ)	$\top \sqsubseteq \forall U.D_i$
[Functional])	$\top \sqsubseteq \leqslant 1\, U$
SubPropertyOf(U_1 U_2)	$U_1 \sqsubseteq U_2$
EquivalentProperties($U_1 \dots U_n$)	$U_1 \equiv \dots \equiv U_n$
AnnotationProperty(S)	
OntologyProperty(S)	
Individual(o type(C_1)...type(C_n)	$o \in C_i$
value(R_1 o_1)...value(R_n o_n)	$\langle o, o_i \rangle \in R_i$
value(U_1 v_1)...value(U_n v_n))	$\langle o, v_i \rangle \in U_i$
SameIndividual($o_1 \dots o_n$)	$\{o_1\} \equiv \dots \equiv \{o_n\}$
DifferentIndividuals($o_1 \dots o_n$)	$\{o_i\} \sqsubseteq \neg\{o_j\}, i \not\equiv j$

Fig. 14.3. OWL DL Axioms and Facts

(source Horrocks et al. 2007. These tables use abstract syntax. The mapping with the Turtle syntax can be found in the OWL 2 Primer.)

Summary

- OWL aims to integrate the flexibility and linked data access provided by RDF graphs with the precise semantics of description logics.
- This is only possible with restrictions on RDF constructs, in particular the constructs representing a class and a property.
- These restrictions led to the OWL DL sublanguage, which can be mapped into description logics and for which efficient reasoners exist.
- The full flexibility of RDF is still available in OWL Full, for which ,however, no reasoner covering the entire language exists.
- OWL 2 DL the most expressive decidable ontology language (corresponding to description logic $\mathcal{SROIQ}(\mathcal{D})$.
- 3 additional tractable simplified languages are proposed : OWL2 EL, RL and QL.
- OWL is the language of references for ontologies for the semantic web, supported by an array of reasoners and other tools.

References

- *[Abiteboul et al. 2011]: Abiteboul S., Manolescu I, Rigaux P., Rousset M-C. and Senellart P., Web Data Management, Cambridge University Press, 2011.*
- *[Baader et al. 2017]: Baader, F., Horrocks, I. Lutz C. and Sattler, U.*, An introduction to Description Logic, *Cambridge University Press, 2017.*
- *[Horrocks et al. 2007]: Horrocks I., Patel-Schneider P., McGuinness D., and Welty C., OWL: a Description Logic Based Ontology Language for the Semantic Web, in Baader F., Calvanese D., McGuinness D., Nardi D., and Patel-Schneider P. (eds.)*, The Description Logic Handbook: Theory, Implementation, and Applications (2nd Edition), *chapter 14. Cambridge University Press, 2007.*

THANK YOU