

Knowledge Representation and Semantic Technologies

Ontologies and OWL

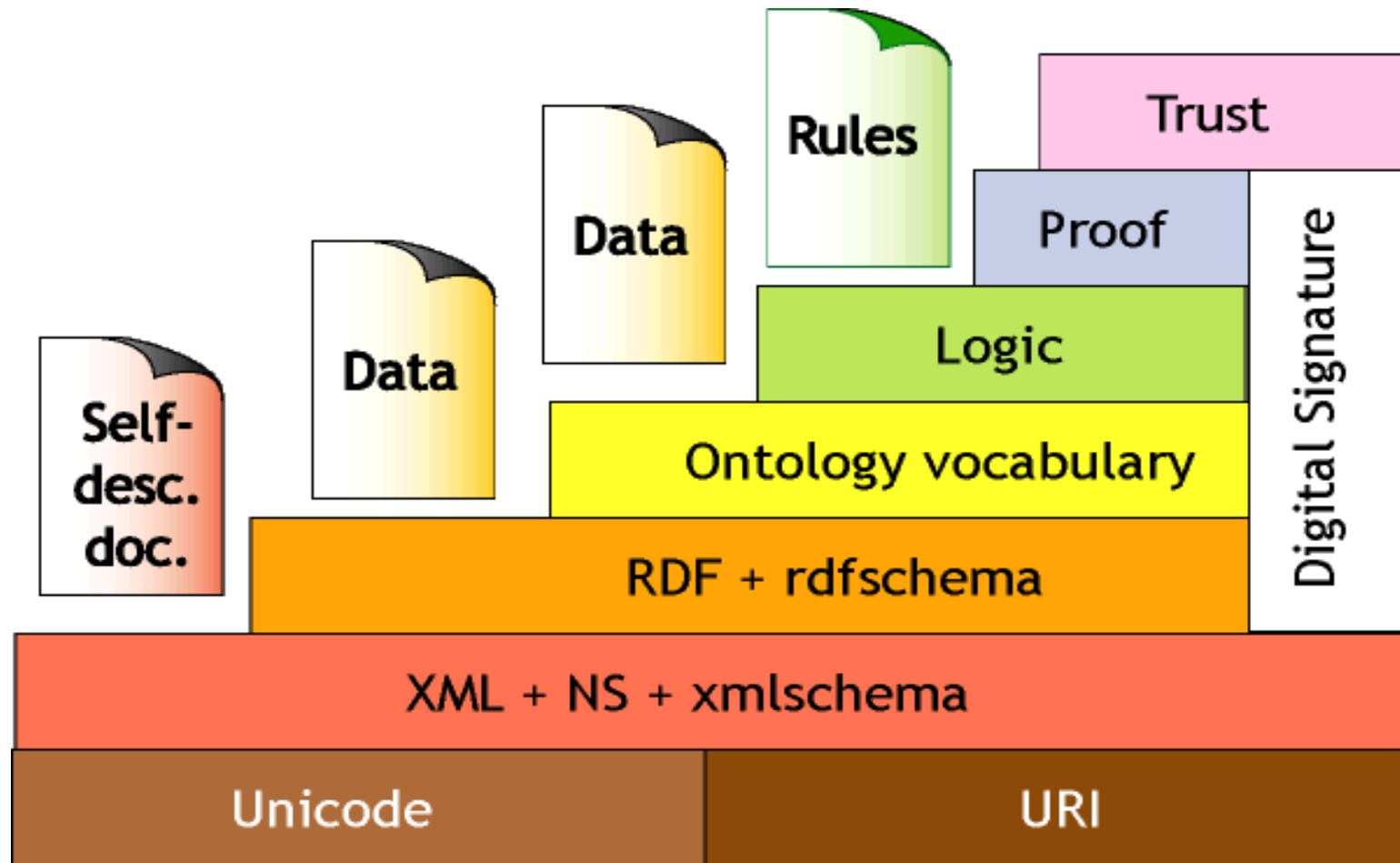
Riccardo Rosati

Corso di Laurea Magistrale in Ingegneria Informatica

Sapienza Università di Roma

2021/2022

The Semantic Web Tower



Ontology in computer science

- ontology = **shared conceptualization** of a domain of interest (Gruber, 1993)
- **shared vocabulary** (set of terms)
 - ⇒ simple (shallow) ontology
- **(complex) relationships between terms**
 - ⇒ deep ontology
- AI view:
 - **ontology = logical theory (knowledge base)**
- DB view:
 - **ontology = conceptual data model**

Structure of an ontology

- **Terms** = names for important concepts in the domain
 - **Elephant** is a concept whose members are a kind of animal
 - **Herbivore** is a concept whose members are exactly those animals who eat only plants or parts of plants
 - **Adult_Elephant** is a concept whose members are exactly those elephants whose age is greater than 20 years
- **Relationships between terms** = background knowledge/constraints on the domain
 - **Adult_Elephants** weigh at least 2,000 kg
 - All **Elephants** are either **African_Elephants** or **Indian_Elephants**
 - No individual can be both a **Herbivore** and a **Carnivore**

Ontology languages

Kinds of potential ontology languages:

- Graphical notations
- Logic-based languages
- Object-oriented languages
- Web schema languages

Ontology languages

- Graphical notations:
 - Semantic networks
 - Topic Maps
 - UML
 - RDF

Ontology languages

- Logic based languages:
 - **Description Logics**
 - **Rules** (e.g., RuleML, Logic Programming/Prolog)
 - **First Order Logic** (e.g., KIF)
 - **Conceptual graphs**
 - (Syntactically) **higher order logics** (e.g., LBase)
 - **Non-classical logics** (e.g., F-logic, Non-Monotonic Logics, Modal Logics)

Object-oriented languages

many languages use object-oriented models based on:

- **Objects/Instances/Individuals**
 - Elements of the domain of discourse
 - Equivalent to constants in FOL
- **Types/Classes/Concepts**
 - Sets of objects sharing certain characteristics
 - Equivalent to unary predicates in FOL
- **Relations/Properties/Roles**
 - Sets of pairs (tuples) of objects
 - Equivalent to binary predicates in FOL

Web schema languages

- Existing Web languages extended to facilitate content description
 - XML → XML Schema (XMLS)
 - RDF → RDF Schema (RDFS)
- XMLS *not* an ontology language
 - Changes format of DTDs (document schemas) to be XML
 - Adds an extensible type hierarchy
 - Integers, Strings, etc.
 - Can define sub-types, e.g., positive integers
- RDFS *is* recognizable as an ontology language
 - Classes and properties
 - Sub/super-classes (and properties)
 - Range and domain (of properties)

Limitations of RDFS

- **RDFS too weak to describe resources in sufficient detail**
 - No **localised range and domain** constraints
 - Can't say that the range of `hasChild` is `person` when applied to persons and `elephant` when applied to elephants
 - No **existence/cardinality** constraints
 - Can't say that all *instances* of `person` have a mother that is also a person, or that persons have exactly 2 parents
 - No **transitive, inverse or symmetrical** properties
 - Can't say that `isPartOf` is a transitive property, that `hasPart` is the inverse of `isPartOf` or that `touches` is symmetrical
 - ...

Web ontology language requirements

Desirable features identified for Web Ontology Language:

- Extends existing Web standards (XML, RDF, RDFS)
- Easy to understand and use (should be based on familiar KR idioms)
- Formally specified
- Of “adequate” expressive power
- Possible to provide automated reasoning support

Two languages developed to satisfy above requirements: DAML and OIL

The OWL language (based on DAML+OIL) became a W3C recommendation in 2004



OWL

- **OWL** = Web Ontology Language
- the **OWL** family is constituted by **3 different languages** (with different expressive power):
 - **OWL Full**
 - **union of OWL syntax and RDF**
 - **OWL-DL**
 - **“DL fragment” of OWL Full**
 - **OWL-Lite**
 - **“easier to implement” subset of OWL DL**

OWL

- OWL standards and technology:
 - first version of OWL standardized in 2004
 - reasoning techniques and tools are recent
 - “optimization” of reasoning not fully explored
 - 2009: W3C standardization of OWL 2

OWL class constructors

Constructor	DL Syntax	Example	Modal Syntax
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	Human \sqcap Male	$C_1 \wedge \dots \wedge C_n$
unionOf	$C_1 \sqcup \dots \sqcup C_n$	Doctor \sqcup Lawyer	$C_1 \vee \dots \vee C_n$
complementOf	$\neg C$	\neg Male	$\neg C$
oneOf	$\{x_1\} \sqcup \dots \sqcup \{x_n\}$	{john} \sqcup {mary}	$x_1 \vee \dots \vee x_n$
allValuesFrom	$\forall P.C$	\forall hasChild.Doctor	$[P]C$
 someValuesFrom	 $\exists P.C$	\exists hasChild.Lawyer	$\langle P \rangle C$
maxCardinality	$\leq_n P$	≤ 1 hasChild	$[P]_{n+1}$
minCardinality	$\geq_n P$	≥ 2 hasChild	$\langle P \rangle_n$

- **XMLS datatypes as well as classes in $\forall P.C$ and $\exists P.C$**
 - E.g., \exists hasAge.nonNegativeInteger
- **Arbitrarily complex nesting of constructors**
 - E.g., $\text{Person} \sqcap \forall \text{hasChild.Doctor} \sqcup \exists \text{hasChild.Doctor}$

OWL axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human \sqsubseteq Animal \sqcap Biped
equivalentClass	$C_1 \equiv C_2$	Man \equiv Human \sqcap Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$ Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} \equiv {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg$ {peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter \sqsubseteq hasChild
equivalentProperty	$P_1 \equiv P_2$	cost \equiv price
inverseOf	$P_1 \equiv P_2^-$	hasChild \equiv hasParent ⁻
transitiveProperty	$P^+ \sqsubseteq P$	ancestor ⁺ \sqsubseteq ancestor
functionalProperty	$\top \sqsubseteq \leq 1P$	$\top \sqsubseteq \leq 1$ hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leq 1P^-$	$\top \sqsubseteq \leq 1$ hasSSN ⁻

Axioms (mostly) reducible to inclusion (\sqsubseteq)

$C \equiv D$ iff both $C \sqsubseteq D$ and $D \sqsubseteq C$

XML syntax for OWL

E.g., concept $\text{Person} \sqcap \forall \text{hasChild}.\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor}$:

```
<owl:Class>
  <owl:intersectionOf rdf:parseType="collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:toClass>
        <owl:unionOf rdf:parseType="collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:hasClass rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:toClass>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```


OWL functional and RDF syntax

- In the next slides, we introduce the **functional** and **RDF** syntax of OWL (OWL 2)
- Functional syntax:
 - Functional-style syntax for OWL statements
 - E.g.: `DisjointClasses(C1 C2)`
- RDF syntax:
 - Expresses every OWL statement as a set of RDF triples
 - E.g.: `C1 owl:disjointWith C2.`

Abbreviations

Letters	Meaning	Letters	Meaning	Letters	Meaning	Letters	Meaning
C	class expression	CN	class name	D	data range	DN	datatype name
P	object property expression	PN	object property name	R	data property	A	annotation property
a	individual	aN	individual name	_:a	anonymous individual (a blank node label)	v	literal
n	non-negative integer	f	facet	ON	ontology name	U	IRI
s	IRI or anonymous individual	t	IRI, anonymous individual, or literal	p	prefix name	_:x	blank node
(a ₁ ... a _n)	RDF list						

Declarations

Language Feature	Functional Syntax	RDF Syntax
class	Declaration(Class(CN))	CN rdf:type owl:Class.
datatype	Declaration(Datatype(DN))	DN rdf:type rdfs:Datatype.
object property	Declaration(ObjectProperty(PN))	PN rdf:type owl:ObjectProperty.
data property	Declaration(DataProperty(R))	R rdf:type owl:DatatypeProperty.
annotation property	Declaration(AnnotationProperty(A))	A rdf:type owl:AnnotationProperty.
named individual	Declaration(NamedIndividual(aN))	aN rdf:type owl:NamedIndividual.

Boolean operators and enumeration

Language Feature	Functional Syntax	RDF Syntax
intersection	ObjectIntersectionOf($C_1 \dots C_n$)	<code>_:x rdf:type owl:Class. _:x owl:intersectionOf ($C_1 \dots C_n$).</code>
union	ObjectUnionOf($C_1 \dots C_n$)	<code>_:x rdf:type owl:Class. _:x owl:unionOf ($C_1 \dots C_n$).</code>
complement	ObjectComplementOf(C)	<code>_:x rdf:type owl:Class. _:x owl:complementOf C.</code>
enumeration	ObjectOneOf($a_1 \dots a_n$)	<code>_:x rdf:type owl:Class. _:x owl:oneOf ($a_1 \dots a_n$).</code>

Object property restrictions

Language Feature	Functional Syntax	RDF Syntax
universal	ObjectAllValuesFrom(P C)	<code>_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:allValuesFrom C</code>
existential	ObjectSomeValuesFrom(P C)	<code>_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:someValuesFrom C</code>
individual value	ObjectHasValue(P a)	<code>_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:hasValue a.</code>
local reflexivity	ObjectHasSelf(P)	<code>_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:hasSelf "true"^^xsd:boolean.</code>
exact cardinality	ObjectExactCardinality(n P)	<code>_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:cardinality n.</code>
qualified exact cardinality	ObjectExactCardinality(n P C)	<code>_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:qualifiedCardinality n. _:x owl:onClass C.</code>

Object property restrictions (contd.)

Language Feature	Functional Syntax	RDF Syntax
maximum cardinality	ObjectMaxCardinality(n P)	<code>_:x rdf:type owl:Restriction.</code> <code>_:x owl:onProperty P.</code> <code>_:x owl:maxCardinality n.</code>
qualified maximum cardinality	ObjectMaxCardinality(n P C)	<code>_:x rdf:type owl:Restriction.</code> <code>_:x owl:onProperty P.</code> <code>_:x owl:maxQualifiedCardinality n.</code> <code>_:x owl:onClass C.</code>
minimum cardinality	ObjectMinCardinality(n P)	<code>_:x rdf:type owl:Restriction.</code> <code>_:x owl:onProperty P.</code> <code>_:x owl:minCardinality n.</code>
qualified minimum cardinality	ObjectMinCardinality(n P C)	<code>_:x rdf:type owl:Restriction.</code> <code>_:x owl:onProperty P.</code> <code>_:x owl:minQualifiedCardinality n.</code> <code>_:x owl:onClass C.</code>

Data property restrictions

Language Feature	Functional Syntax	RDF Syntax
universal	DataAllValuesFrom(R D)	<code>_:x rdf:type owl:Restriction.</code> <code>_:x owl:onProperty R.</code> <code>_:x owl:allValuesFrom D.</code>
existential	DataSomeValuesFrom(R D)	<code>_:x rdf:type owl:Restriction.</code> <code>_:x owl:onProperty R.</code> <code>_:x owl:someValuesFrom D.</code>
literal value	DataHasValue(R v)	<code>_:x rdf:type owl:Restriction.</code> <code>_:x owl:onProperty R.</code> <code>_:x owl:hasValue v.</code>
exact cardinality	DataExactCardinality(n R)	<code>_:x rdf:type owl:Restriction.</code> <code>_:x owl:onProperty R.</code> <code>_:x owl:cardinality n.</code>
qualified exact cardinality	DataExactCardinality(n R D)	<code>_:x rdf:type owl:Restriction.</code> <code>_:x owl:onProperty R.</code> <code>_:x owl:qualifiedCardinality n.</code> <code>_:x owl:onClass D.</code>

Data property restrictions (contd.)

Language Feature	Functional Syntax	RDF Syntax
maximum cardinality	DataMaxCardinality(n R)	<code>_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:maxCardinality n.</code>
qualified maximum cardinality	DataMaxCardinality(n R D)	<code>_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:maxQualifiedCardinality n. _:x owl:onClass D.</code>
minimum cardinality	DataMinCardinality(n R)	<code>_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:minCardinality n.</code>
qualified minimum cardinality	DataMinCardinality(n R D)	<code>_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:minQualifiedCardinality n. _:x owl:onClass D.</code>

Object and data property expressions

Language Feature	Functional Syntax	RDF Syntax
named object property	PN	PN
universal object property	owl:topObjectProperty	owl:topObjectProperty
empty object property	owl:bottomObjectProperty	owl:bottomObjectProperty
inverse property	ObjectInverseOf(PN)	_:x owl:inverseOf PN

Language Feature	Functional Syntax	RDF Syntax
named data property	R	R
universal data property	owl:topDataProperty	owl:topDataProperty
empty data property	owl:bottomDataProperty	owl:bottomDataProperty

Class axioms

Language Feature	Functional Syntax	RDF Syntax
subclass	SubClassOf(C_1 C_2)	C_1 rdfs:subClassOf C_2 .
equivalent classes	EquivalentClasses($C_1 \dots C_n$)	C_j owl:equivalentClass C_{j+1} . $j=1 \dots n-1$
disjoint classes	DisjointClasses(C_1 C_2)	C_1 owl:disjointWith C_2 .
pairwise disjoint classes	DisjointClasses($C_1 \dots C_n$)	$_ :x$ rdf:type owl:AllDisjointClasses. $_ :x$ owl:members ($C_1 \dots C_n$).
disjoint union	DisjointUnionOf(CN $C_1 \dots C_n$)	CN owl:disjointUnionOf ($C_1 \dots C_n$).

Object property axioms

Language Feature	Functional Syntax	RDF Syntax
subproperty	SubObjectPropertyOf($P_1 P_2$)	P_1 rdfs:subPropertyOf P_2 .
property chain inclusion	SubObjectPropertyOf(ObjectPropertyChain($P_1 \dots P_n$) P)	P owl:propertyChainAxiom ($P_1 \dots P_n$).
property domain	ObjectPropertyDomain($P C$)	P rdfs:domain C .
property range	ObjectPropertyRange($P C$)	P rdfs:range C .
equivalent properties	EquivalentObjectProperties($P_1 \dots P_n$)	P_j owl:equivalentProperty P_{j+1} , $j=1 \dots n-1$
disjoint properties	DisjointObjectProperties($P_1 P_2$)	P_1 owl:propertyDisjointWith P_2 .
pairwise disjoint properties	DisjointObjectProperties($P_1 \dots P_n$)	$_ :x$ rdf:type owl:AllDisjointProperties. $_ :x$ owl:members ($P_1 \dots P_n$).
inverse properties	InverseObjectProperties($P_1 P_2$)	P_1 owl:inverseOf P_2 .
functional property	FunctionalObjectProperty(P)	P rdf:type owl:FunctionalProperty.
inverse functional property	InverseFunctionalObjectProperty(P)	P rdf:type owl:InverseFunctionalProperty.

Object property axioms (contd.)

Language Feature	Functional Syntax	RDF Syntax
reflexive property	ReflexiveObjectProperty(P)	P rdf:type owl:ReflexiveProperty.
irreflexive property	IrreflexiveObjectProperty(P)	P rdf:type owl:IrreflexiveProperty.
symmetric property	SymmetricObjectProperty(P)	P rdf:type owl:SymmetricProperty.
asymmetric property	AsymmetricObjectProperty(P)	P rdf:type owl:AsymmetricProperty.
transitive property	TransitiveObjectProperty(P)	P rdf:type owl:TransitiveProperty.

Data property axioms

Language Feature	Functional Syntax	RDF Syntax
subproperty	$\text{SubDataPropertyOf}(R_1 R_2)$	$R_1 \text{ rdfs:subPropertyOf } R_2.$
property domain	$\text{DataPropertyDomain}(R C)$	$R \text{ rdfs:domain } C.$
property range	$\text{DataPropertyRange}(R D)$	$R \text{ rdfs:range } D.$
equivalent properties	$\text{EquivalentDataProperties}(R_1 \dots R_n)$	$R_j \text{ owl:equivalentProperty } R_{j+1}.$ $j=1 \dots n-1$
disjoint properties	$\text{DisjointDataProperties}(R_1 R_2)$	$R_1 \text{ owl:propertyDisjointWith } R_2.$
pairwise disjoint properties	$\text{DisjointDataProperties}(R_1 \dots R_n)$	$_x \text{ rdf:type}$ $\text{owl:AllDisjointProperties}.$ $_x \text{ owl:members } (R_1 \dots R_n).$
functional property	$\text{FunctionalDataProperty}(R)$	$R \text{ rdf:type owl:FunctionalProperty}.$

Assertions (ABox statements)

Language Feature	Functional Syntax	RDF Syntax
individual equality	SameIndividual($a_1 \dots a_n$)	a_j owl:sameAs a_{j+1} . $j=1 \dots n-1$
individual inequality	DifferentIndividuals($a_1 \ a_2$)	a_1 owl:differentFrom a_2 .
pairwise individual inequality	DifferentIndividuals($a_1 \dots a_n$)	$_x$ rdf:type owl:AllDifferent. $_x$ owl:members ($a_1 \dots a_n$).
class assertion	ClassAssertion($C \ a$)	a rdf:type C .
positive object property assertion	ObjectPropertyAssertion($PN \ a_1 \ a_2$)	a_1 $PN \ a_2$.
positive data property assertion	DataPropertyAssertion($R \ a \ v$)	$a \ R \ v$.
negative object property assertion	NegativeObjectPropertyAssertion($P \ a_1 \ a_2$)	$_x$ rdf:type owl:NegativePropertyAssertion. $_x$ owl:sourceIndividual a_1 . $_x$ owl:assertionProperty P . $_x$ owl:targetIndividual a_2 .
negative data property assertion	NegativeDataPropertyAssertion($R \ a \ v$)	$_x$ rdf:type owl:NegativePropertyAssertion. $_x$ owl:sourceIndividual a . $_x$ owl:assertionProperty R . $_x$ owl:targetValue v .

XML Schema datatypes in OWL

- OWL supports XML Schema primitive datatypes
 - E.g., integer, real, string, ...
- Strict separation between “object” classes and datatypes
 - Disjoint interpretation domain Δ_D for datatypes
 - For a datavalue d , $d^I \subseteq \Delta_D$
 - And $\Delta_D \cap \Delta^I = \emptyset$
 - Disjoint “object” and datatype properties
 - For a datatype property P , $P^I \subseteq \Delta^I \times \Delta_D$
 - For object property S and datatype property P , $S^I \cap P^I = \emptyset$
- Equivalent to the “ (D_n) ” in $\mathcal{SHOIN}(D_n)$

OWL DL semantics

- Mapping OWL to equivalent DL ($\mathcal{SHOIN}(\mathcal{D}_n)$):
 - Facilitates provision of reasoning services (using DL systems)
 - Provides well defined semantics
- DL semantics defined by interpretations: $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where
 - $\Delta^{\mathcal{I}}$ is the domain (a non-empty set)
 - $\cdot^{\mathcal{I}}$ is an interpretation function that maps:
 - Concept (class) name $A \rightarrow$ subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$
 - Role (property) name $R \rightarrow$ binary relation $R^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$
 - Individual name $i \rightarrow i^{\mathcal{I}}$ element of $\Delta^{\mathcal{I}}$

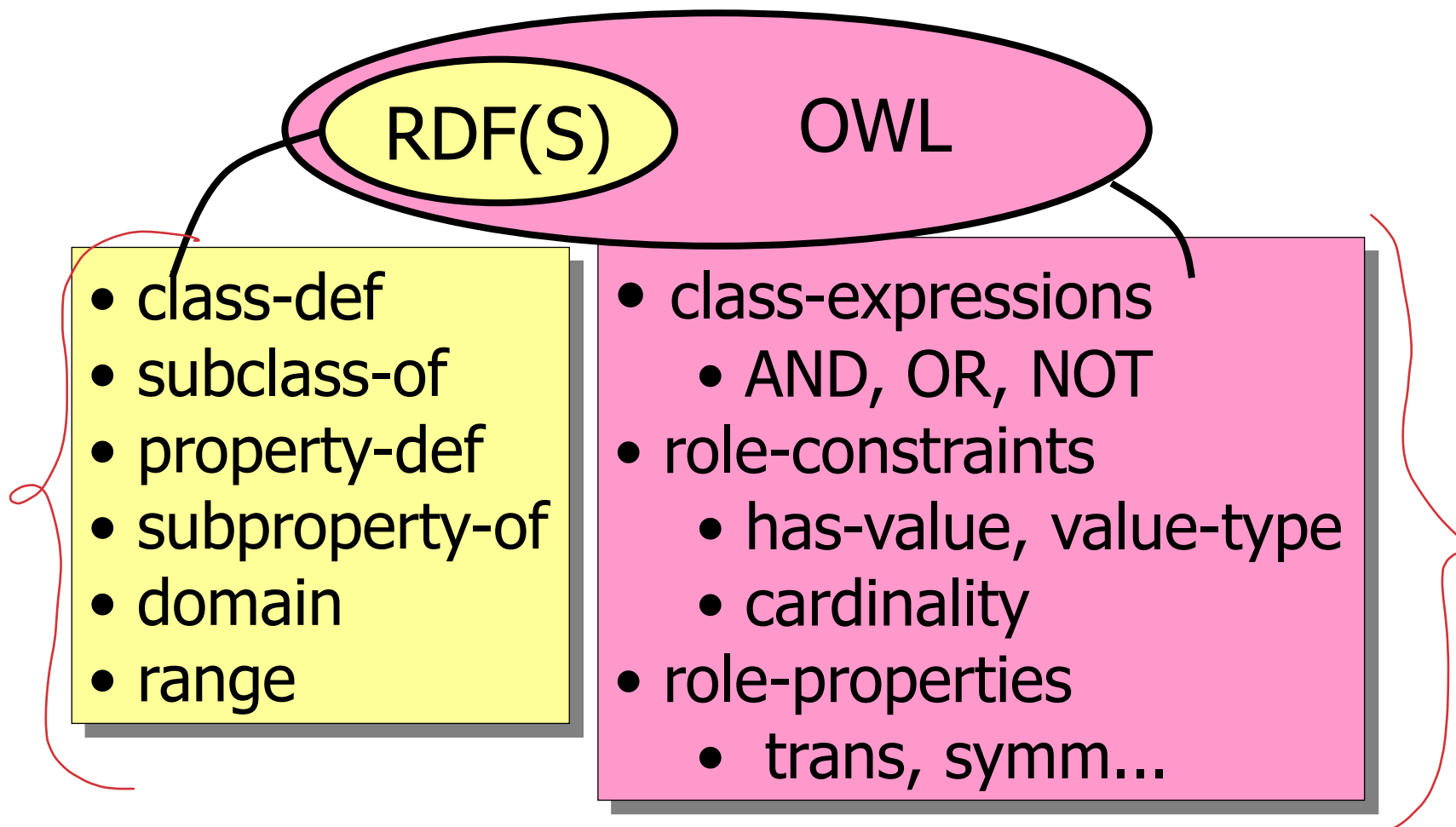
OWL DL ontologies are DL knowledge bases

- An OWL ontology maps to a DL Knowledge Base

$$\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$$

- $\mathcal{T}(\text{Tbox})$ is a set of axioms of the form:
 - $C \sqsubseteq D$ (concept inclusion)
 - $C \equiv D$ (concept equivalence)
 - $R \sqsubseteq S$ (role inclusion)
 - $R \equiv S$ (role equivalence)
 - $R^+ \sqsubseteq R$ (role transitivity)
- $\mathcal{A}(\text{Abox})$ is a set of axioms of the form
 - $x \in D$ (concept instantiation)
 - $\langle x, y \rangle \in R$ (role instantiation)

OWL vs. RDFS



OWL vs. First-Order Logic

- in general, DLs correspond to decidable subclasses of first-order logic (FOL)
- DL KB = first-order theory
- OWL Full is NOT a FOL fragment!
 - reasoning in OWL Full is undecidable
- OWL-DL and OWL-Lite are decidable fragments of FOL

OWL vs. First-Order Logic

let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an ontology about persons where:

- \mathcal{T} contains the following inclusion assertions:

MALE \sqsubseteq PERSON

FEMALE \sqsubseteq PERSON

MALE $\sqsubseteq \neg$ FEMALE

PERSON $\sqsubseteq \exists \text{Father}^- . \text{MALE}$

- \mathcal{A} contains the following instance assertions:

MALE(Bob)

PERSON (Mary)

PERSON(Paul)

OWL vs. First-Order Logic

- \mathcal{T} corresponds to the following FOL sentences:

$\forall x. \text{MALE}(x) \rightarrow \text{PERSON}(x)$

$\forall x. \text{FEMALE}(x) \rightarrow \text{PERSON}(x)$

$\forall x. \text{MALE}(x) \rightarrow \neg \text{FEMALE}(x)$

$\forall x. \text{PERSON}(x) \rightarrow \exists y. \text{Father}(y,x) \text{ and } \text{MALE}(y)$

- \mathcal{A} corresponds to the following FOL ground atoms:

$\text{MALE}(\text{Bob})$

$\text{PERSON}(\text{Mary})$

$\text{PERSON}(\text{Paul})$

Inference tasks in OWL

- **Ontology consistency** (corresponds to **KB consistency** in DL)
- Concept/role consistency (same as DL)
- Concept/role subsumption and equivalence (same as DL)
- Instance checking (same as DL)
- ...

Inference tasks

- OWL-DL ontology = first-order logical theory
- verifying the formal properties of the ontology corresponds to **reasoning** over a first-order theory

Inference tasks

- OWL-DL ontology = first-order logical theory
- verifying the formal properties of the ontology corresponds to **reasoning** over a first-order theory
- main reasoning tasks over ontologies:
 - consistency of the ontology
 - concept (and role) consistency
 - concept (and role) subsumption
 - instance checking
 - instance retrieval
 - query answering

Consistency of the ontology

- Is the ontology $K=(T,A)$ consistent (non-self-contradictory)?
- i.e., is there at least a model for K ?
- intensional + extensional reasoning task
- fundamental formal property:
- inconsistent ontology \Rightarrow there is a semantic problem in K !
- K must be repaired

Consistency of the ontology

Example TBox:

$\text{MALE} \sqsubseteq \text{PERSON}$

$\text{FEMALE} \sqsubseteq \text{PERSON}$

$\text{MALE} \sqsubseteq \neg \text{FEMALE}$

$\text{PERSON} \sqsubseteq \exists \text{hasFather.MALE}$

$\text{PERSON} \sqsubseteq \exists \text{hasMother.FEMALE}$

$\text{hasMother} \sqsubseteq \text{hasParent}$

$\text{hasFather} \sqsubseteq \text{hasParent}$

$\exists \text{hasParent.BLACK-EYES} \sqsubseteq \text{BLACK-EYES}$

Consistency of the ontology

Example ABox:

MALE(Bob)

MALE(Paul)

FEMALE(Ann)

hasFather(Ann,Paul)

hasMother(Paul,Mary)

BLACK-EYES(Mary)

\neg BLACK-EYES(Ann)

\Rightarrow **TBox + ABox inconsistent** (Ann should have black eyes)

Concept consistency

- is a concept definition C consistent in a TBox T ?
- i.e., is there a model of T in which C has a non-empty extension?
- intensional (schema) reasoning task
- detects a fundamental modeling problem in T :
 - if a concept is not consistent, then it can never be populated!

Concept subsumption

- is a concept C subsumed by another concept D in T?
- i.e., is the extension of C contained in the extension of D in every model of T?
- intensional (schema) reasoning task
- allows to do classification of concepts (i.e., to construct the concept ISA hierarchy)

Instance checking

- is an individual a a member of concept C in K ?
- i.e., is the fact $C(a)$ satisfied by every interpretation of K ?
- intensional + extensional reasoning task
- basic “instance-level query” (tell me if object a is in class C)

Instance retrieval

- find all members of concept C in K
- i.e., compute all individuals a such that $C(a)$ is satisfied by every interpretation of K
- intensional + extensional reasoning task
- (slight) generalization of instance checking

Query answering

- compute the answers to a **query** q in K (expressed in some query language)
- i.e., compute all tuples of individuals t such that $q(t)$ is entailed by K ($= q(t)$ is satisfied by every interpretation of K)
- extensional + extensional reasoning task
- generalization of instance checking and instance retrieval
- e.g.: database queries (SQL-like) over ontologies (or SPARQL-like queries)

Queries over ontologies

classes of queries over DL ontologies considered:

- **conjunctive queries** = subclass of SQL queries
 - correspond to select-project-join queries
- **unions of conjunctive queries**
 - correspond to select-project-join-union queries
- **more expressive queries** (e.g., epistemic queries)
- **SPARQL queries**
 - restrictions/extensions of SPARQL

SPARQL 1.1

- SPARQL 1.1 is the W3C standard query language over OWL ontologies (released in 2013)
- SPARQL 1.1 has different associated **entailment regimes** that define the semantics of queries over different datasets (RDF models, RDFS+RDF graphs, OWL ontologies)
- the semantics of SPARQL queries for OWL is defined by two entailment regimes for SPARQL:
 - **OWL 2 RDF-based semantics** entailment regime
 - **OWL 2 direct semantics** entailment regime (corresponds to DL semantics)

Computational aspects of reasoning

- reasoning in OWL-DL is decidable (and the complexity is characterized)
- however: high computational complexity (EXPTIME)
- (optimized) reasoning algorithms developed
- OWL-DL reasoning tools implemented

Current OWL technology

two kinds of tools:

- OWL editors (“environments”)
- OWL reasoners

OWL editors

- allow for visualizing/browsing/editing OWL ontologies
- able to connect to an external OWL reasoner
=> OWL “environments”
- main current tools:
 - Protege
 - SWOOP
 - OWLed2

OWL reasoning tools

two categories:

- OWL-DL reasoners, e.g.:
 - Hermit
 - Pellet
 - Konclude
 - Racer, RacerPro
 - Fact++
- reasoners for “tractable fragments” of OWL-DL, e.g.:
 - ELK (OWL 2 EL)
 - Mastro, Ontop (OWL 2 QL)
 - RDFox (OWL 2 RL)

OWL-DL reasoning tools

- all tools support “standard” reasoning tasks, i.e.:
 - consistency of the ontology
 - concept consistency
 - concept subsumption and classification
 - instance checking and retrieval
 - query answering (SPARQL)

References

- OWL W3C Web site:
<http://www.w3.org/2004/OWL/>
- OWL 2 overview:
<http://www.w3.org/TR/owl2-overview/>
- OWL 2 primer:
<http://www.w3.org/TR/2012/REC-owl2-primer-20121211/>
- OWL 2 profiles:
<http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/>
- OWL 2 quick reference guide:
<https://www.w3.org/TR/owl2-quick-reference/>

References

- **SPARQL 1.1:**
<http://www.w3.org/TR/sparql11-query/>
- **SPARQL 1.1 entailment regimes:**
<http://www.w3.org/TR/2013/REC-sparql11-entailment-20130321/>
- **Web page on Description Logic reasoners:**
<http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/>
- **Protege (OWL ontology editor):**
<http://protege.stanford.edu/>

References

- Hermit (OWL reasoning tool):
<http://hermit-reasoner.com/>
- Konclude (OWL 2 DL reasoner):
<http://derivo.de/produkte/konclude/>
- ELK (OWL 2 EL ontology reasoner):
<http://www.cs.ox.ac.uk/isg/tools/ELK/>
- Stardog (OWL 2 profiles and OWL2 DL reasoner):
<http://stardog.com/>
- RacerPro (OWL reasoning tool):
<http://franz.com/agraph/racer/>
- Mastro (DL-Lite ontology-based data access system)
<https://www.obdasystems.com/mastro>
<https://www.obdasystems.com/mastro-protege-plugin>