COMP318

Ontologies and Semantic Web

RDF - Part 12

Dr Valentina Tamma
V.Tamma@liverpool.ac.uk

Where were we

- RDFS schema language
  - RDFS as an extension of RDF
- Simple entailment

RDF(S) entailment

- An RDF(S) graph entails implicit triples
  - triples not explicitly contained in the graph, but that can be derived from an RDF(S) graph
  - using the special semantics of the vocabulary of the graph
    - vocabulary of the graph: set of names which occurs as the subject, predicate, object
  - Entailments (Informative) vs Interpretations (Normative)

- Entailments
  - Transformation rules to derive new assertions from existing ones
  - May be proven complete and consistent with the formal interpretation
- Interpretations
  - Interpretations assign special meaning to the symbols in a particular vocabulary
    - Mapping of RDF assertions into an abstract model, based on set-theory
    - With an “interpretation operator” **I()**, maps a RDF graph into a highly abstract set of high-cardinality sets
    - Highly theoretical model, useful to prove mathematical properties

Entailment regimes

- Three entailment regimes
  - simple entailment: no particular extra conditions are posed on a vocabulary, including the RDF vocabulary itself;
    - it involved only graph transformations.
  - RDF entailment: based on the interpretation of the RDF vocabulary;
  - RDFS entailment:
    - based on the interpretation of the RDFS vocabulary
    - some extra conditions are posed by in the form of axiomatic triples and semantic conditions

RDFS axiomatic triples (excerpt)

- RDF and RDFS include a set of default triples to guide the grammar of expected triples
  - Only resources have types:
    - `rdf:type rdfs:domain rdfs:Resource .`
  - Types are classes:
    - `rdf:type rdfs:range rdfs:Class .`
  - Ranges only apply to properties:
    - `rdf:range rdfs:domain rdf:Property .`
  - Ranges are classes:
    - `rdf:range rdfs:range rdfs:Class .`
    - … and many more

RDF(S) Grammar

Triples

Individuals

- indi o-prop indi .
- indi d-prop “Literal” .
- indi rdf:type class .

Class and properties

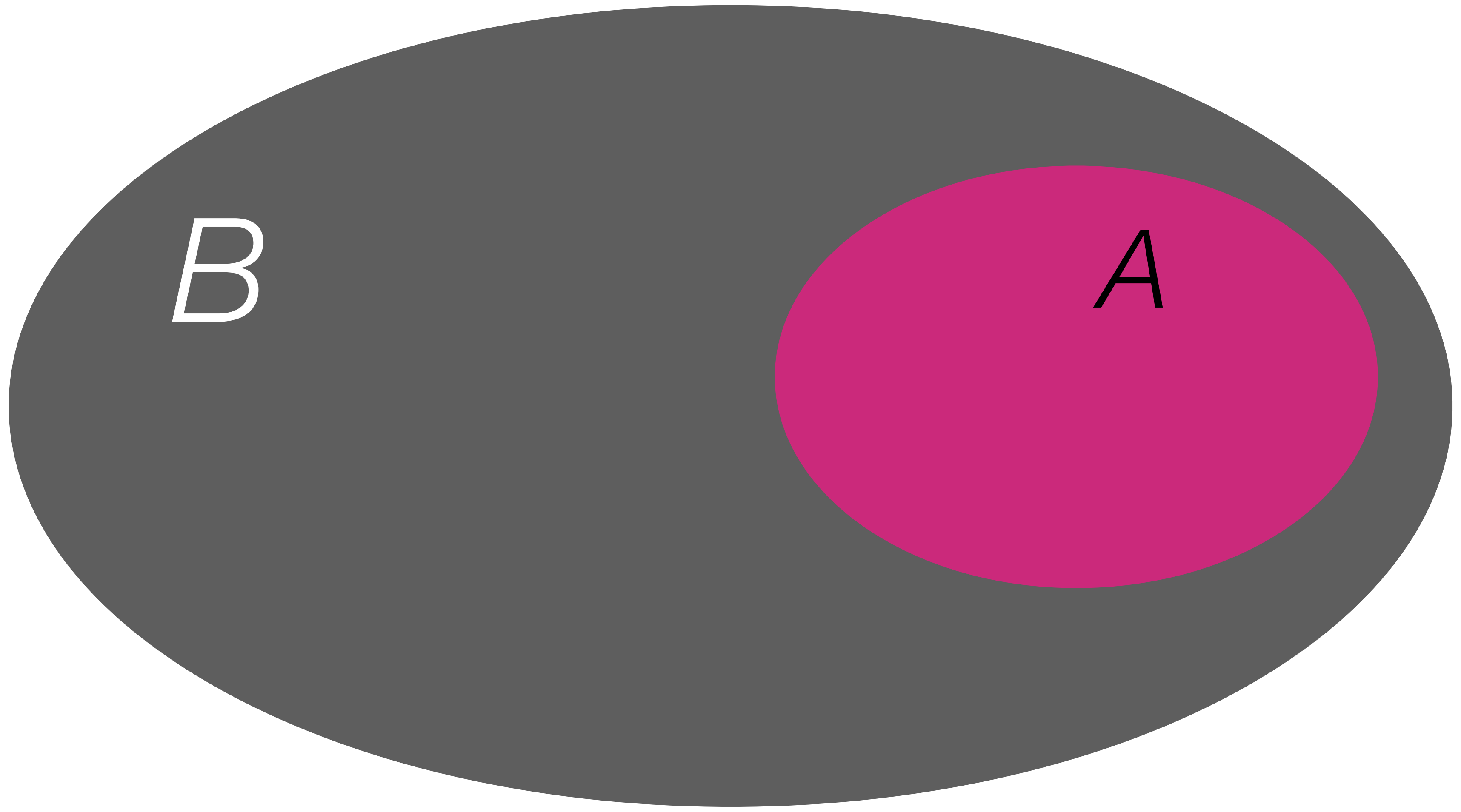
- class rdfs:subClassOf class .
- o-prop rdfs:subPropertyOf o-prop .
- d-prop rdfs:subPropertyOf d-prop .
- o-prop rdfs:domain class .
- o-prop rdfs:range class .
- d-prop rdfs:domain class .
- d-prop rdfs:range “Literal” .

Classes as sets

- A set is a mathematical object: {♠♣♥♦}
  - it only contains the symbols ♠, ♣, ♥, ♦ and nothing else
  - there is no order imposed on the elements: {♠♣♥♦} = {♥♦♠♣}
  - no element can be repeated: {♠♣♠♣♠♣} ={♠♣}
  - sets with different elements are different: {♠♣} $\neq$ {♥♦}
  - $\in$ indicates set membership: ♠ $\in$ {♠♣} but ♥ $\notin$ {♠♣}

- The empty set, $\emptyset$ or { }, is a set with no elements
  - x $\notin$ $\emptyset$, for any x

Subsets



- Let A and B be sets. A is a subset of B, $A \subseteq B$ if every element of A is in B
  - $\{1, 2, 3, 4\} \subseteq \{1, 2, 3, 4, 5, \ldots.\}$
  - $\{1, 2\} \nsubseteq \{1, 4, 5\}$
- $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$

Classes as set of resources

- We can say that `rdfs:Class` is a set of Resources
  - it gives us the intuition

RDFS	Set Theory
`A rdf:type rdfs:Class` `:Building rdf:type rdfs:Class`	A is a set of resources Building is a set of resources
`x rdf:type A` `:baronWayBuilding rdf:type :Building`	$x \in A$: baronWayBuilding $\in$: Building
`A rdfs:subClassOf B` `:Building rdfs:subClassOf :ResidentialUnit`	$A \subseteq B$: Building $\subseteq$: ResidentialUnit

Pairs

- A pair is an ordered collection of two objects: $\langle 1, 2 \rangle$
  - equality of pairs is based on components: $\langle a, b \rangle = \langle x, y \rangle$ if and only if x = a and b = y
  - the order imposed on the elements matters: $\langle a, b \rangle \neq \langle b, a \rangle$
  - elements can be repeated: $\langle 1, 1 \rangle$
    - $\langle a, b \rangle$ is a pair, irrespectively of weather a = b or not

Properties as relations

- A relation *R* between two sets A and B is a set of pairs: $\langle a, b\rangle \in A \times B$, $R \subseteq A \times B$
  - $A \times B$ set of all pairs $\langle a, b\rangle$ where $a \in A$ and $b \in B$
    - cross product between A and B
  - we write **a R b** to denote $\langle a, b\rangle \in R$
  - a relation R on some set A is a relation between A and A, A^2
- The domain of R is the set of all x such that x R …
  - dom R = { x ∈ A s.t. x R y for some y ∈ B}
- The range of R is the set of all y such that … R y
  - range R = { y ∈ B s.t. x R y for some x ∈ A}

Intuition: properties as relations

- We can say that **`rdf:Property`** is similar to defining a relation between resources

RDFS	Set Theory
`R rdf:type rdf:Property`	R is a relation on resources
`x R y`	$\langle x, y \rangle \in R$
`R rdfs:subPropertyOf S`	$R \subseteq S$
`:rents rdf:type rdf:Property`	`:rents` is a relation on resources
`:rents rdfs:subPropertyOf :residesAt`	$: rents \subseteq : residesAt$
`R rdfs:domain A`	$dom_R \subseteq A$
`R rdfs:range B`	$range_R \subseteq B$
`:rents rdfs:domain :Person`	$dom_{:rents} \subseteq : Person$
`:rents rdfs:range :ResidentialUnit`	$range_{:rents} \subseteq : ResidentialUnit$
`:john :rents :BaronWayApartment`	$\langle : john : BaronWayApartment \rangle \in : rents$

Model-theoretic semantics

- Based on the notion of **interpretations**
  - which might be thought of as potential “realities” or “worlds”

- Interpretations map values to elements
  - The intuitions behind set-theory are formally represented

- Given an interpretation I and a set of triples G,
  - then G is valid in I ($I \models G$) iff $I \models t$ for all $t \in G$
    - I is also called a **model** of G

RDF Interpretations

- RDF interpretations add some additional semantics wrt instances using the `rdf:type` property:
  - in particular, memberships of the built-in class `rdf:Property` and the datatypes supported;
- We define an interpretation function for classes, properties and datatypes in terms of their individuals (extensionally)
- The interpretations need to satisfy the RDF axiomatic triples
  - For more details refer to Foundations of Semantic Web Technologies, Chapter 3

rdf:type	rdf:type	rdf:Property .
rdf:subject	rdf:type	rdf:Property .
rdf:object	rdf:type	rdf:Property .
rdf:first	rdf:type	rdf:Property
rdf:rest	rdf:type	rdf:Property .
rdf:value	rdf:type	rdf:Property .
rdf:nil	rdf:type	rdf:Property .
rdf:_:n	rdf:type	rdf:Property .

Inference rules

- An inference rule is a rule of the form:

$$\frac{\phi_1, \phi_2, \ldots \phi_n}{\psi}$$

- where $\phi_1, \phi_2, \ldots \phi_n$ are sentences in the language (**assumptions**), whilst ψ is a new sentence derived from the assumptions (**conclusion**)

- Inference rules are a formal description of the process for constructing new expressions from existing ones.
  - In RDF, inferences corresponding to **entailments** are described are described as correct or valid.

Proof theory

- Every formal logic has a set of inference rules that can be used to "prove" some formula μ from a given set of formulas Γ

- A **formal proof** is the sequential application of the inference rules that starts with Γ and ends with μ
  - $\Gamma \vdash \mu, \mu$ can be proved from Γ

Soundness and completeness

- An inference mechanism is **sound** if it derives only sentences that are entailed.
  - If $\Gamma \vdash \mu$ then $\Gamma \models \mu$

- An inference mechanism is **complete** if derives all the sentences that are entailed.
  - If $\Gamma \models \mu$ then $\Gamma \vdash \mu$

RDF inference rules

- The W3C recommendation “RDF Semantics’ provides the inference rules that corresponds to the various form of entailments mentioned;
  - simple entailment
  - RDF entailment
  - RDFS entailment

RDF Entailment

- The RDF entailment has 4 inference rules:

- (rdfax) Infer the triple `:u :a :x.` for every RDF axiomatic triple `:u :a :x.`

- (lg, literal generalisation) If G contains `:u :a :l.` then infer the triple `:u :a _:n.`
  - Specialised version of SE1 that allows generalisation of a literal by a blank node
    - Other properties of this literal can be inferred via this blank node: e.g. the literal is an instance of a class
    - Literals can only appear as objects in a triple

RDF Entailment

- The RDF entailment has 4 inference rules:
- **(rdf1)** If G contains a triple `:u :a :y.` then we can infer `:a rdf:type rdf:Property`.
- **(rdf2)** If G contains a triple `:u :a :l.` where `:l` is a well-formed XML literal then we can infer `_:n rdf:type rdf:XMLLiteral`.

RDF entailment

- **Theorem**. A graph G_1 RDF-entails a graph G_2 if and only if there is a graph G_1' that can be derived from G_1 by using the rules *rdfax*, *lg*, *rdf1* and *rdf2* such that G_1' simply entails G_2.

COMP318

Ontologies and Semantic Web

End of RDF - Part 12

Dr Valentina Tamma
V.Tamma@liverpool.ac.uk