COMP318: RDFS entailment

www.csc.liv.ac.uk/~valli/Comp318

Dr Valentina Tamma
Room: Ashton 2.12
Dept of computer science
University of Liverpool
V.Tamma@liverpool.ac.uk

Where were we

- SPARQL
  - RDF query language
    - Based on simple entailment
  - Syntax and exercises

Semantics

- RDF(S) vocabulary has built-in "meaning"

- RDF(S) Semantics
  - Makes meaning explicit
  - Defines what follows from an RDF graph

- Goals
  - Evaluate the truth of a triple/graph
  - Characterise the state of the world that makes a triple/graph true.

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
  - ***Interpretations*** assign special meaning to the symbols in a particular vocabulary
  - ***Interpretations*** (Normative)
    - 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
  - ***Entailments*** (Informative)
    - Transformation rules to derive new assertions from existing ones
    - May be proven complete and consistent with the formal interpretation

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

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 $\Gamma \vDash \mu$

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

Notation

- a, b,
  - refer to any arbitrary URI
    - (i.e. anything that can appear in the predicate of a triple)
- u, v,
  - refer to any arbitrary URI or blank node ID
    - (i.e. anything that can appear in the subject of a triple)
- x, y,
  - refer to an arbitrary URI, blank node ID or literal
    - (i.e. anything that can appear in the object of a triple)
- _:n,
  - refer to the ID of a blank node
    - (i.e. appearing as a subject or object)
- l,
  - refers to a literal
    - (i.e. a string that is sometimes found in the object)

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 G1 RDF-entails a graph G2 if and only if there is a graph G1' that can be derived from G1 by using the rules rdfax, lg, rdf1 and rdf2 such that G1' simply entails G2.

Inference Rules for RDFS-entailment

- Assign “meaning” to the RDFS vocabulary
- **(rdfsx)** Infer the triple `u a x.` for every RDFS axiomatic triple `u a x.`
- **(RDFS1, literal)** If G contains `u a l.` where `l` is a plain literal (with or without language information), then infer the triple:
  - `_:n rdf:type rdfs:literal.`

Domain and range restrictions

- **(rdfs2)** If G contains a triples `a rdfs:domain x.` `u a y.` then we can infer
  - `u rdf:type x`.
- **(rdf3)** If G contains a triples `a rdfs:range x.` `u a v.` then we can infer
  - `v rdf:type x`.

Everything is a resource

- (rdfs4a) If G contains a triple `u a x.` then we can infer
  - `u rdf:type rdfs:Resource`.
- (rdfs4b) If G contains a triple `u a v.` then we can infer
  - `v rdf:type rdfs:Resource`.
- We do not need an inference rule for predicates:
  - the relevant triple can be derived using rdf1 and rdfs4

Important property of binary relations

- binary relation R on a set A is said to be:
  - Reflexive: if x R x, for all x ∈ A
    - A number **is equal to** itself
  - Irreflexive: if not x R x, for all x ∈ A
    - A number x **is not equal to** (x+1)
  - Symmetric: if x R y implies y R x, for all x, y ∈ A
    - **marriedTo**: if Ross **marriedTo** Rachel then Rachel **marriedTo** Ross
  - Asymmetric: if x R y not A implies y R x, for all x, y ∈ A
    - **parentOf**: if Rachel **parentOf** Emma then it does not imply Emma **parentOf** Rachel
  - Transitive: if x R y and y R z implies x R z, for all x, y, z ∈ A
    - **friendOf**: if Monica **friendOf** Joey and Joey **friendOf** Phoebe then Monica **friendOf** Phoebe

Reflexivity and Transitivity of `rdfs:subPropertyOf`

- (rdfs5) If G contains the triples `u rdfs:subPropertyOf v.` and `v rdfs:subPropertyOf x.` we can infer
  - `u rdfs:subPropertyOf x.`

- (rdfs6) If G contains the triple `u rdf:type rdf:Property.` we can infer
  - `u rdfs:subPropertyOf u.`

More on Subproperties

- **(rdfs7)** If G contains the triples `a rdfs:subPropertyOf b.` and `u a y.` we can infer
  - `u b y`.

Classes and instances

- **(rdfs8)** If G contains the triple `u rdf:type rdfs:Class.` we can infer
    - `u rdfs:subClassOf rdfs:Resource.`

- **(rdfs9)** If G contains the triples `u rdfs:subClassOf x.` and `v rdf:type u.` we can infer
    - `v rdf:type x.`

Reflexivity and Transitivity of `rdfs:subClassOf`

- **(rdfs10)** If G contains the triple `u rdf:type rdfs:Class.` we can infer
  - `u rdfs:subClassOf u.`

- **(rdfs11)** If G contains the triples `u rdfs:subClassOf v.` and `v rdfs:subClassOf x.` we can infer
  - `u rdfs:subClassOf x.`

Containers

- (rdfs12) If G contains the triple `u rdf:type rdfs:ContainerMembershipProperty.` we can infer
  - `u rdfs:subPropertyOf rdfs:member.`

Datatypes

- (rdfs13) If G contains the triple
  `u rdf:type rdfs:Datatype.` we can infer
  - `u rdfs:subClassOf rdfs:Literal.`

Bijection between literals and surrogate bonds

- **(gl inverse of lg)** If G contains the triple
  `u a _:n.` we can infer
  - `u a l.` However:
  - this rule can be applied only when `_:n` identifies a bnode that was introduced earlier by weakening the literal `l` via the rule lg
  - This inference rule is necessary to bring back a literal that has been substituted by a blank node using rule lg, then some other inference rule produced a triple with this blank node (`_:n`) in the object position, and rule gl can now be used to bring this literal back!
    - E.g. `:Murray atp:name “Andy Murray".`
      `:atp:name rdfs:range atp:PlayerName.`
    - Would entail `"Andy Murray" a atp:PlayerName.` which is problematic. Why?

Bijection between literals and surrogate bonds

- The latter triple is not a valid RDF triple
  - a literal should not appear in the subject position!

- thus it will not be inferred (the domain of the ?v variable in the rdfs3 rule would prevent the inference). And so, to achieve the valid inference:
  - `_:AndyMurray a atp:PlayerName .`

    Requires the surrogate blank node (`_:AndyMurray`) to be used through the rule lg.

    - The inverse rule gl then allows surrogates to “travel” back as literals into the object position, though examples of such behaviour are not necessarily intuitive.

Conclusion

- RDF entailment rules
- RDFS entailment rules