

Semantic Web Technologies

Simple and RDFS Entailment

Jos de Bruijn
debruijn@inf.unibz.it

KRDB Research Group
Free University of Bolzano, Italy

17 October 2007

Outline

Model-Theoretic Semantics of RDF

Simple Entailment

Evaluating Simple Entailment

RDFS Entailment

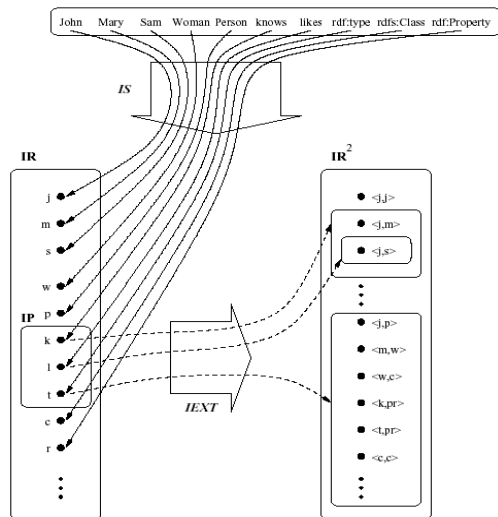
Side-step: model-theoretic semantics

- ▶ Semantics of a language is defined using a model theory
- ▶ **Interpretation**:
 - ▶ Domain D
 - ▶ Interpretation function \cdot^I
 - ▶ Maps symbols in the language to values in D
 - ▶ Maps statements in the language to truth values in $(D \times \dots \times D)$
- ▶ The model theory describes conditions under which:
 - ▶ The interpretation is model for a theory
- ▶ A model theory provides a **declarative** semantics for a language
 - ▶ Two important semantic notions:
 - ▶ **Entailment**: a theory S entails S' iff every model of S is a model of S'
 - ▶ **Satisfiability**: a theory S is satisfiable iff S has a model

Basic RDF Interpretations

- ▶ $I = \langle IR, IP, IS, IEXT \rangle$
- ▶ Interpretation domain for resources IR , which is a nonempty set
- ▶ Interpretation domain for properties IP , which is a set (which might be a subset of IR)
- ▶ Interpretation function IS which maps URIs to elements of IR
- ▶ Extension function $IEXT$ which maps between IP and $IR \times IR$
- ▶ Given a URI u , we define $I(u) = IS(u)$, i.e. URIs are interpreted using IS

Basic RDF Interpretation (cont'd)



Blank Node Assignments and Interpretation of Blank Nodes

- ▶ Set of blank nodes
 - ▶ in a triple $\langle s, p, o \rangle$ is denoted $bl(\langle s, p, o \rangle)$
 - ▶ in a graph S is denoted $bl(S)$
 - e.g. $bl(\langle _ : x, rdf:type, ex:Student \rangle) = \{ _ : x \}$, $bl(\langle ex:john, rdf:type, ex:Student \rangle) = \{ _ : x \}$
- ▶ A **blank node assignment** A is a mapping from blank nodes to resources: $A : bl(S) \rightarrow IR$
- ▶ Given an interpretation I , a blank node assignment A , and a blank node $_ : x$, we define $I(_ : x) = A(_ : x)$, i.e. blank nodes are interpreted using the blank node assignment A

Satisfaction of Triples and Graphs

- ▶ An interpretation $I = \langle IR, IP, IS, IEXT \rangle$ **satisfies** a triple $\langle s, p, o \rangle$ **relative to a blank node assignment** A if
 - ▶ $I(p) \in IP$, i.e. p denotes a property, and
 - ▶ $\langle I(s), I(o) \rangle \in IEXT(I(p))$, i.e. s, o denote the value of the property denoted by p

which is denoted $I, A \models \langle s, p, o \rangle$

- ▶ An interpretation $I = \langle IR, IP, IS, IEXT \rangle$ **satisfies** a graph S **relative to a blank node assignment** A if
 - ▶ $I, A \models \langle s, p, o \rangle$ for every triple $\langle s, p, o \rangle$ in S

which is denoted $I, A \models S$

- ▶ If I, A do not satisfy $\langle s, p, o \rangle; S$, we write $I, A \not\models \langle s, p, o \rangle; I, A \not\models S$
- ▶ I is a model of S , denoted $I \models S$, if there exists **some** blank node assignment $A : bl(S) \rightarrow IR$ such that $I, A \models S$

Adding Literals I

- ▶ Recall: there are 2 kinds of literals:
 - ▶ Plain literals, which are strings with an optional language tag
 - ▶ e.g. "John Smith", "1", "Vino Bianco"@it
 - ▶ Typed literals, which are pairs of strings and datatype URIs
 - ▶ e.g. ("John Smith", xsd:string), ("1", xsd:integer), ("2007-10-10", xsd:date)
 - ▶ We also write these as: "John Smith"8sd:string, "1"8sd:integer, "2007-10-10"8sd:date

Adding Literals II

- ▶ Interpretation domain for resources IR
- ▶ Interpretation domain for properties $IP \subseteq IR$
- ▶ Interpretation function IS which maps URIs to elements of IR
- ▶ Extension function $IEXT$ which maps between IP and $IR \times IR$
- ▶ Interpretation domain for plain literals $LV \subseteq IR$
- ▶ Interpretation function IL which maps typed literals to elements of IR
- ▶ Given a plain literal l , we define $I(l) = l$, i.e. plain literals are interpreted as themselves
- ▶ Given a typed literal (l, u) , we define $I((l, u)) = LV((l, u))$, i.e. typed literals are interpreted using LV

Satisfiability

- ▶ An RDF graph S is satisfiable iff S has a model

Do the following RDF graphs have a model?

<code><http://.../#john></code>	<code><http://.../#hasName></code>	<code>"John Smith"</code>
<code><http://.../#mary></code>	<code><http://.../#hasName></code>	<code>"Mary Jane"</code>

<code><http://.../#john></code>	<code><http://.../#marriedTo></code>	<code><http://.../#john></code>
---------------------------------------	--	---------------------------------------

--

In fact, **every RDF graph is satisfiable!**

Simple Entailment

- ▶ “ S **simple-entails** E , denoted $S \models_s E$, if every interpretation which is a model of S is also a model of E .”
- ▶ Any subgraph S' of an RDF graph S **is simple-entailed by** S , i.e. $S \models_s S'$
- ▶ Any instance E of an RDF graph S **simple-entails** S , i.e. $E \models_s S$
- ▶ Entailment is **transitive**

<code><http://.../#john></code>	<code><http://.../#hasName></code>	<code>" John Smith"</code>
<code><http://.../#mary></code>	<code><http://.../#hasName></code>	<code>" Mary Jane"</code>

instance

entails

<code><http://.../#john></code>	<code><http://.../#hasName></code>	<code>..x</code>
<code><http://.../#mary></code>	<code><http://.../#hasName></code>	<code>" Mary Jane"</code>

entails

<code><http://.../#mary></code>	<code><http://.../#hasName></code>	<code>" Mary Jane"</code>
---------------------------------------	--	---------------------------

subgraph

Simple Entailment II

- ▶ Simple entailment seems weak, because RDF and RDFS vocabularies are not interpreted
- ▶ However, checking simple entailment is “simple”
- ▶ When viewing RDF as pure data, the RDF(S) vocabulary is arguably not “useful”
- ▶ SPARQL, the query language for RDF, uses simple entailment

Blank nodes substitution

Blank node substitution

A substitution is a mapping of the form

$\theta = \{b_1 \mapsto t_1, \dots, b_n \mapsto t_n\}$, with b_1, \dots, b_n blank nodes (essentially, variables), and t_1, \dots, t_n blank nodes, URIs, or literals (essentially, terms).

Example:

$\theta = \{ _ :x \mapsto \#john, _ :z \mapsto _ : u, _ :y \mapsto _ :y, _ :w \mapsto " John Smith" \}$

Substitution Application

$S = \{ \langle \#john, \text{hasName}, _ :w \rangle, \langle _ :x, \text{marriedTo}, _ :z \rangle, \langle _ :y, \text{marriedTo}, _ :x \rangle \}$

$\theta(S) = \{ \langle \#john, \text{hasName}, " John Smith" \rangle, \langle \#john, \text{marriedTo}, _ :u \rangle, \langle _ :y, \text{marriedTo}, \#john \rangle \}$

Alternative Definition of Simple Entailment

- ▶ $S \models_s E$ iff $\theta(E) \subseteq S$ for some blank node substitution θ .
 - ▶ θ maps blank nodes in E to symbols in S
- ▶ In other words, $S \models_s E$ iff some instance of E is a subset of S .

Example

$$S = \{ \langle \#john, \text{hasName}, "John\ Smith" \rangle, \langle \#john, \text{marriedTo}, _ : u \rangle, \\ \langle _ : y, \text{marriedTo}, \#john \rangle \}$$
$$E = \{ \langle \#john, \text{hasName}, "John\ Smith" \rangle, \langle _ : y, \text{marriedTo}, \#john \rangle \}$$
$$S \models_s E$$

RDFS Interpretations

- ▶ A simple interpretation is an RDFS interpretation if it satisfies:
 - ▶ The RDF axiomatic triples
 - ▶ The RDF semantic conditions
 - ▶ The RDFS axiomatic triples
 - ▶ The RDFS semantic conditions
- ▶ Axiomatic triples
 - ▶ Statements which must **always** be true
 - ▶ Define certain relationships (e.g. typing or subclassing) within the RDF and RDFS vocabularies
- ▶ Semantic conditions
 - ▶ Define additional properties of RDF and RDFS vocabularies (e.g. typing commutes over subclass)

RDFS Entailments

S **rdfs-entails** E , denoted $S \models_{rdfs} E$, if every rdfs-interpretation which is a model of S is also a model of E .

RDFS Axiomatic Triples

⟨rdf:type, rdfs:domain, rdfs:Resource⟩

⟨rdfs:domain, rdfs:domain, rdf:Property⟩

⟨rdfs:range, rdfs:domain, rdf:Property⟩

⟨rdfs:subPropertyOf, rdfs:domain, rdf:Property⟩

⟨rdfs:subClassOf, rdfs:domain, rdfs:Class⟩

⟨rdf:type, rdfs:range, rdfs:Class⟩

⟨rdfs:domain, rdfs:range, rdfs:Class⟩

⟨rdfs:range, rdfs:range, rdfs:Class⟩

⟨rdfs:subPropertyOf, rdfs:range, rdf:Property⟩

⟨rdfs:subClassOf, rdfs:range, rdfs:Class⟩

RDFS Semantic Conditions on Ontology Vocabulary

Define class extension function $ICEXT$: $ICEXT(c) = \{i \mid \langle i, c \rangle \in IEXT_{type}\}$

1. $IR = ICEXT(IS(Resource))$
2. $LV = ICEXT(IS(Literal))$
3. x is in IP if and only if $\langle x, I(Property) \rangle$ is in $IEXT(I(type))$
 Informally: $\langle x, rdf:type, rdf:Property \rangle$
4. if $\langle x, y \rangle \in IEXT(IS(domain))$ and $\langle u, v \rangle \in IEXT(x)$, then $u \in ICEXT(y)$
5. if $\langle x, y \rangle \in IEXT(IS(range))$ and $\langle u, v \rangle \in IEXT(x)$, then $v \in ICEXT(y)$
6. $IEXT(IS(subPropertyOf))$ is transitive and reflexive on IP
7. if $\langle x, y \rangle \in IEXT(IS(subPropertyOf))$, then $IEXT(x) \subseteq IEXT(y)$
8. if $x \in ICEXT(Class)$, then $\langle \langle x, Resource \rangle \in IEXT(subClassOf)$
9. if $\langle x, y \rangle \in IEXT(IS(subClassOf))$, then $ICEXT(x) \subseteq ICEXT(y)$
10. $IEXT(IS(subClassOf))$ is transitive and reflexive on $ICEXT(Class)$

Required reading

- ▶ Document “Resource Description Framework” on website:
<http://www.debruijn.net/teaching/swt/rdf.pdf>
- ▶ RDF Semantics, Chapters 1, 2, 4: <http://www.w3.org/TR/rdf-mt/>

Further reading

- ▶ J. de Bruijn and S. Heymans. Logical foundations of (e)RDF(S): Complexity and reasoning. In *Proceedings of the 6th International Semantic Web Conference (ISWC2007)*.
- ▶ J. de Bruijn, E. Franconi, and S. Tessaris. Logical reconstruction of normative RDF. In *OWL: Experiences and Directions Workshop (OWLED-2005)*, Galway, Ireland, November 2005.
- ▶ H. J. ter Horst. Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. *Journal of Web Semantics*, 3(23):79115, 2005.