Logic and Reasoning in the Semantic Web (part I – RDF/RDFS)

Fulvio Corno, Laura Farinetti
Politecnico di Torino
Dipartimento di Automatica e Informatica
e-Lite Research Group – http://elite.polito.it
Outline

- Reasoning in Semantic Web Knowledge Bases
- RDF/RDFS Semantics and Entailments
- OWL Semantics
- Pellet
Outline

- Reasoning in Semantic Web Knowledge Bases
- RDF/RDFS Semantics and Entailments
- OWL Semantics
- Pellet
Reasoning

- Reasoning is required when a program must determine some information or some action that has not been explicitly told about.
- It must figure out what it needs to know from what it already knows.
Example

- **Facts:**
  - Robins are birds
  - All birds have wings

- **Questions:**
  - Are Robins birds? → Yes
  - Do all birds have wings? → Yes
  - Do robins have wings? → ?????
Semantic Web peculiarities

- Traditional KBS used to be developed top down
- The Semantic Web is developed bottom-up
- Problems:
  - **Scalability**: Everybody having a web page is a potential Knowledge Engineer
  - **Distribution**: Pieces of knowledge can be all over the place, finding them can be hard
  - **Heterogeneity**: People use different ways of describing the same information
  - **Quality**: Knowledge will be incomplete and even inconsistent across different sources of mapping approaches
Reasoning in the Semantic Web

- New methods are needed that
  - Scale to large amounts of data and knowledge
  - Are tolerant to errors, incompleteness and inconsistency between and within sources
  - Work on distributed representations and ideally are distributed themselves
  - ...and are compatible with Semantic web standards
Outline

- Reasoning in Semantic Web Knowledge Bases
- RDF/RDFS Semantics and Entailments
- OWL Semantics
- Pellet
Recall: RDF Schema

RDF vs RDFS

Source: http://www.dcs.bbk.ac.uk/~michael/sw/sw.html
Michael Zakharyaschev
Some RDFS inference rules

- \((X \ R \ Y), (R \ \text{subPropertyOf} \ Q) \Rightarrow (X \ Q \ Y)\)
- \((X \ R \ Y), (R \ \text{domain} \ C) \Rightarrow (X \ \text{type} \ C)\)
- \((X \ \text{type} \ C), (C \ \text{subClassOf} \ D) \Rightarrow (X \ \text{type} \ D)\)
- …
RDF Semantics

- Semantics is not about the “meaning” of assertions
  - The ‘meaning’ of an assertion in RDF or RDFS may depend on many factors, including social conventions, comments in natural language or links to other content-bearing documents, … (non machine-processable information)
  - Semantics restricts itself to a formal notion of meaning which could be characterized as the part that is common to all other accounts of meaning, and can be captured in mechanical inference rules
Model Semantics

- The RDF Semantics W3C Recommendation uses model theory for specifying the semantics of the RDF language.
- Model theory assumes that the language refers to a ‘world’, and describes the minimal conditions that a world must satisfy in order to assign an appropriate meaning for every expression in the language.
- A particular world is called an interpretation.
- The idea is to provide an abstract, mathematical account of the properties that any such interpretation must have, making as few assumptions as possible about its actual nature or intrinsic structure, thereby retaining as much generality as possible.
Goals of the inference process

- To provide a **technical** way to determine when inference processes are valid, i.e., when they preserve truth
- Starting from a set of assertions that are *regarded as* true in an RDF model, derive whether a new RDF model contains *provably* true assertions
- We never known about the “real” truth of any assertion in the “real” world.
Formalization

- **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
Interpretation: minimum notions

- Let $V$ be a vocabulary containing all names (URIs and literals) occurring in RDF triples.

- An RDF **interpretation** of $V$ consists of:
  - $\mathcal{IR}$, a non-empty set of **resources** (domain or universe).
  - $\mathcal{IP} \subseteq \mathcal{IR}$, a set of **properties** (each property is also a resource).
    
    Symbol $\nu \in V$ is a property if, and only if, $\mathcal{IS}(\nu) \in \mathcal{IP}$. 

  - $\mathcal{IS} : V \rightarrow \mathcal{IR}$, an interpretation of resources (with each symbol from $V$ a resource is associated).

  - $\mathcal{IEXT} : \mathcal{IP} \rightarrow 2^{\mathcal{IR} \times \mathcal{IR}}$, an interpretation of properties (each property is a binary relation, i.e., a subset of $\mathcal{IR} \times \mathcal{IR}$).

An RDF triple $\langle s, p, o \rangle$ is **true** in $I$ if, and only if,

$$s, p, o \in V, IS(p) \in \mathcal{IP} \text{ and } (IS(s), IS(o)) \in \mathcal{IEXT}(IS(p))$$

An RDF triple is false in $I$ if it is not true in $I$.

An RDF graph is **true** in $I$ if, and only if, every triple of it is true in $I$.

In particular, it is false in $I$ if some triple is not true in $I$. 
Conceptual framework

IS assigns one thing to each name in the vocabulary

1 is the only property in the set IP

IEXT maps 1 to a property extension

The property extension IEXT(1) maps 1 to 2 and 2 to 1
Interpretation: minimum notions

- $I$: interpretation operator
- $E$: fragment of RDF syntax

If $E$ is a graph:

- $I(E) = \text{false}$ if $I(E') = \text{false}$ for some triple $E'$ in $E$, otherwise $I(E) = \text{true}$

If $E$ is a triple $<s, p, o>$:

- $I(E) = \text{true}$ if $s$, $p$ and $o$ are in $V$, $I(p)$ is in $IP$ and $<I(s), I(o)>$ is in $IEXT(I(p))$
- otherwise $I(E) = \text{false}$

- $IP$: set of properties, $IR$: set of resources
- $IEXT(I(p))$: mapping from $IP$ into the powerset of $IR \times IR$ i.e. the set of sets of pairs $<x, y>$ with $x$ and $y$ in $IR$
Construct an interpretation in which the following RDF graph is true:

\[ IR = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \quad \text{and} \quad IP = \{1, 2, 3, 4\} \]

\[ IS(\text{name}) = 1, \quad IS(\text{isTaughtBy}) = 3, \quad IS(\text{homepage}) = 4, \quad IS(\text{sw}) = 5, \]
\[ IS(\text{MZ}) = 6, \quad IS(\text{http://www.dcs.bbk.ac.uk/~michael/sw/sw.html}) = 9, \]
\[ IS(\text{"The Semantic Web"}) = 7, \quad IS(\text{"Michael Zakharyaschev"}) = 8 \]

\[ IEXT(1) = \{(5, 7), (6, 8)\}, \quad IEXT(2) = \emptyset, \quad IEXT(3) = \{(5, 6)\}, \]
\[ IEXT(4) = \{(5, 9)\} \]
Example
Example of interpretation

The mappings colored red show that this interpretation satisfies the triple \( \text{rdf:type} \) \( \text{rdf:type} \) \( \text{rdf:Property} \).
Entailment

- I satisfies E if I(E)=true
- A set S of RDF graphs entails a graph E if every interpretation which satisfies every member of S also satisfies E

In human words:
- assertion = a claim that the world is an interpretation which assigns the value true to the assertion
- If A entails B, then
  - any interpretation that makes A true also makes B true
  - an assertion of A already contains the same "meaning" as an assertion of B
  - the meaning of B is somehow contained in, or subsumed by, that of A
Entailment regime

- RDF defines 4 entailment regimes:
  - Simple entailment: does not interpret any RDF or RDFS vocabulary (just structural matching, by re-naming blank nodes)
  - RDF entailment: interprets RDF vocabulary
  - RDFS entailment: interprets RDF and RDFS vocabularies
  - D entailment: additional support for datatypes
RDF Entailment rules (legend)

- aaa, bbb: any URI reference
- uuu, vvv: any URI reference or blank node identifier
- xxx, yyy: any URI reference, blank node identifier or literal
- lll: any literal
- _:nnn: blank node identifiers
# Simple Entailment rules

<table>
<thead>
<tr>
<th>Rule name</th>
<th>If E contains</th>
<th>then add</th>
</tr>
</thead>
<tbody>
<tr>
<td>se1</td>
<td>uuu aaa xxx .</td>
<td>uuu aaa _:nnn . where _:nnn identifies a blank node allocated to xxx by rule se1 or se2.</td>
</tr>
<tr>
<td>se2</td>
<td>uuu aaa xxx .</td>
<td>_:nnn aaa xxx . where _:nnn identifies a blank node allocated to uuu by rule se1 or se2.</td>
</tr>
<tr>
<td>lg</td>
<td>uuu aaa lll .</td>
<td>uuu aaa _:nnn . where _:nnn identifies a blank node allocated to the literal lll by this rule.</td>
</tr>
<tr>
<td>gl</td>
<td>uuu aaa _:nnn . where _:nnn identifies a blank node allocated to the literal lll by rule lg.</td>
<td>uuu aaa lll .</td>
</tr>
</tbody>
</table>
Role of Entailment rules (1/3)

- Simple entailment satisfies the “Interpolation Lemma”:
  - S entails a graph E if and only if a subgraph of S is an instance of E
  - It completely characterizes simple RDF entailment in syntactic terms

- RDF entailment satisfies the “RDF entailment lemma”:
  - S rdf-entails E if and only if there is a graph which can be derived from S plus the RDF axiomatic triples by the application of rule lg and the RDF entailment rules and which simply entails E
  - This means that the entailment rules are “complete”
## RDF Entailment rules

<table>
<thead>
<tr>
<th>Rule name</th>
<th>If E contains</th>
<th>then add</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf1</td>
<td><code>uuu aaa yyy</code> .</td>
<td><code>aaa rdf:type rdf:Property</code> .</td>
</tr>
<tr>
<td>rdf2</td>
<td><code>uuu aaa lll</code> . where <code>lll</code> is a well-typed XML literal .</td>
<td><code>_:nnn rdf:type rdf:XMLLiteral</code> . where <code>_:nnn</code> identifies a blank node allocated to <code>lll</code> by rule lg.</td>
</tr>
</tbody>
</table>
### RDF Axiomatic Triples

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf:type</td>
<td>rdf:type</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdf:subject</td>
<td>rdf:type</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdf:predicate</td>
<td>rdf:type</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdf:object</td>
<td>rdf:type</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdf:rest</td>
<td>rdf:type</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdf:value</td>
<td>rdf:type</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdf:_1</td>
<td>rdf:type</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdf:_2</td>
<td>rdf:type</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rdf:nil</td>
<td>rdf:type</td>
<td>rdf:List</td>
</tr>
</tbody>
</table>
Role of Entailment rules (2/3)

- RDF entailment satisfies the “RDF entailment lemma”:
  - S rdf-entails E if and only if there is a graph which can be derived from S plus the RDF axiomatic triples by the application of rule lg and the RDF entailment rules and which simply entails E
  - This means that the entailment rules are “complete”
## RDFS Entailment rules (1/2)

<table>
<thead>
<tr>
<th>Rule name</th>
<th>If E contains</th>
<th>then add</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs1</td>
<td>uuu aaa lll.</td>
<td><img src="#" alt="_:nnn rdf:type rdfs:Literal" /> where _:_nnn identifies a blank node allocated to lll by rule rule lg.</td>
</tr>
<tr>
<td></td>
<td>where lll is a plain literal (with or without a language tag)</td>
<td><img src="#" alt="_:nnn rdf:type rdfs:Literal" /></td>
</tr>
<tr>
<td>rdfs2</td>
<td>aaa rdfs:domain xxx . uuu aaa yyy .</td>
<td>uuu rdf:type xxx .</td>
</tr>
<tr>
<td>rdfs3</td>
<td>aaa rdfs:range xxx . uuu aaa vvv .</td>
<td>vvv rdf:type xxx .</td>
</tr>
<tr>
<td>rdfs4a</td>
<td>uuu aaa xxx .</td>
<td>uuu rdf:type rdfs:Resource .</td>
</tr>
<tr>
<td>rdfs4b</td>
<td>uuu aaa vvv.</td>
<td>vvv rdf:type rdfs:Resource .</td>
</tr>
<tr>
<td>rdfs5</td>
<td>uuu rdfs:subPropertyOf vvv . vvv rdfs:subPropertyOf xxx .</td>
<td>uuu rdfs:subPropertyOf xxx .</td>
</tr>
</tbody>
</table>
## RDFS Entailment rules (2/2)

<table>
<thead>
<tr>
<th>Rule name</th>
<th>If E contains</th>
<th>then add</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs6</td>
<td>uuu rdf:type rdf:Property .</td>
<td>uuu rdfs:subPropertyOf uuu .</td>
</tr>
<tr>
<td>rdfs7</td>
<td>aaa rdfs:subPropertyOf bbb uuu aaa yyyy .</td>
<td>uuu bbb yyyy .</td>
</tr>
<tr>
<td>rdfs8</td>
<td>uuu rdf:type rdfs:Class .</td>
<td>uuu rdfs:subClassOf rdfs:Resource .</td>
</tr>
<tr>
<td>rdfs9</td>
<td>uuu rdfs:subClassOf xxx . vvv rdf:type uuu .</td>
<td>vvv rdf:type xxx .</td>
</tr>
<tr>
<td>rdfs10</td>
<td>uuu rdf:type rdfs:Class .</td>
<td>uuu rdfs:subClassOf uuu .</td>
</tr>
<tr>
<td>rdfs11</td>
<td>uuu rdfs:subClassOf vvv . vvv rdfs:subClassOf xxx .</td>
<td>uuu rdfs:subClassOf xxx .</td>
</tr>
<tr>
<td>rdfs12</td>
<td>uuu rdf:type rdfs:ContainerMembershipProperty .</td>
<td>uuu rdfs:subPropertyOf rdfs:member .</td>
</tr>
<tr>
<td>rdfs13</td>
<td>uuu rdf:type rdfs:Datatype</td>
<td>uuu rdfs:subClassOf rdfs:Literal .</td>
</tr>
</tbody>
</table>
RDFS Axiomatic triples (1/3)

\[
\begin{align*}
\text{rdf:type} & \quad \text{rdfs:domain} & \quad \text{rdfs:Resource} . \\
\text{rdfs:domain} & \quad \text{rdfs:domain} & \quad \text{rdf:Property} . \\
\text{rdfs:range} & \quad \text{rdfs:domain} & \quad \text{rdf:Property} . \\
\text{rdfs:subPropertyOf} & \quad \text{rdfs:domain} & \quad \text{rdf:Property} . \\
\text{rdfs:subClassOf} & \quad \text{rdfs:domain} & \quad \text{rdfs:Class} . \\
\text{rdf:subject} & \quad \text{rdfs:domain} & \quad \text{rdf:Statement} . \\
\text{rdf:predicate} & \quad \text{rdfs:domain} & \quad \text{rdf:Statement} . \\
\text{rdf:object} & \quad \text{rdfs:domain} & \quad \text{rdf:Statement} . \\
\text{rdfs:member} & \quad \text{rdfs:domain} & \quad \text{rdfs:Resource} . \\
\text{rdf:first} & \quad \text{rdfs:domain} & \quad \text{rdf:List} . \\
\text{rdf:rest} & \quad \text{rdfs:domain} & \quad \text{rdf:List} . \\
\text{rdfs:seeAlso} & \quad \text{rdfs:domain} & \quad \text{rdfs:Resource} . \\
\text{rdfs:isDefinedBy} & \quad \text{rdfs:domain} & \quad \text{rdfs:Resource} . \\
\text{rdfs:comment} & \quad \text{rdfs:domain} & \quad \text{rdfs:Resource} . \\
\text{rdfs:label} & \quad \text{rdfs:domain} & \quad \text{rdfs:Resource} . \\
\text{rdf:value} & \quad \text{rdfs:domain} & \quad \text{rdfs:Resource} . 
\end{align*}
\]
### RDFS Axiomatic Triples (2/3)

<table>
<thead>
<tr>
<th>Property</th>
<th>Range</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf:type</td>
<td>rdfs:range</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdfs:domain</td>
<td>rdfs:range</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdfs:range</td>
<td>rdfs:range</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdfs:subPropertyOf</td>
<td>rdfs:range</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdfs:subClassOf</td>
<td>rdfs:range</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdf:subject</td>
<td>rdfs:range</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:predicate</td>
<td>rdfs:range</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:object</td>
<td>rdfs:range</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdfs:member</td>
<td>rdfs:range</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:first</td>
<td>rdfs:range</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:rest</td>
<td>rdfs:range</td>
<td>rdf:List</td>
</tr>
<tr>
<td>rdfs:seeAlso</td>
<td>rdfs:range</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdfs:isDefinedBy</td>
<td>rdfs:range</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdfs:comment</td>
<td>rdfs:range</td>
<td>rdfs:Literal</td>
</tr>
<tr>
<td>rdfs:label</td>
<td>rdfs:range</td>
<td>rdfs:Literal</td>
</tr>
<tr>
<td>rdf:value</td>
<td>rdfs:range</td>
<td>rdfs:Resource</td>
</tr>
</tbody>
</table>
RDFS Axiomatic triples (3/3)

<table>
<thead>
<tr>
<th>rdf:Alt</th>
<th>rdfs:subClassOf</th>
<th>rdfs:Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf:Bag</td>
<td>rdfs:subClassOf</td>
<td>rdfs:Container</td>
</tr>
<tr>
<td>rdf:Seq</td>
<td>rdfs:subClassOf</td>
<td>rdfs:Container</td>
</tr>
<tr>
<td>rdfs:ContainerMembershipProperty</td>
<td>rdfs:subClassOf</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdfs:isDefinedBy</td>
<td>rdfs:subPropertyOf</td>
<td>rdfs:seeAlso</td>
</tr>
<tr>
<td>rdf:XMLLiteral</td>
<td>rdf:type</td>
<td>rdfs:Datatype</td>
</tr>
<tr>
<td>rdf:XMLLiteral</td>
<td>rdfs:subClassOf</td>
<td>rdfs:Literal</td>
</tr>
<tr>
<td>rdfs:Datatype</td>
<td>rdfs:subClassOf</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdf:_1</td>
<td>rdf:type</td>
<td>rdfs:ContainerMembershipProperty</td>
</tr>
<tr>
<td>rdf:_1</td>
<td>rdfs:domain</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:_1</td>
<td>rdfs:range</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:_2</td>
<td>rdf:type</td>
<td>rdfs:ContainerMembershipProperty</td>
</tr>
<tr>
<td>rdf:_2</td>
<td>rdfs:domain</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:_2</td>
<td>rdfs:range</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Role of Entailment rules (3/3)

- RDFS entailment satisfies the “RDFS entailment lemma”:
  - S rdfs-entails E if and only if there is a graph which can be derived from S plus the RDF and RDFS axiomatic triples by the application of rule lg, rule gl and the RDF and RDFS entailment rules and which either simply entails E or contains an XML clash.
Effect of Entailment rules (1/3)

- The outputs of these rules will often trigger others.

<table>
<thead>
<tr>
<th>rule:</th>
<th>triggers rule:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4a</th>
<th>4b</th>
<th>5a</th>
<th>5b</th>
<th>6</th>
<th>7a</th>
<th>7b</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4a</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4b</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5a</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5b</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7a</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7b</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Dependencies between RDFS entailment rules

Effect of Entailment rules (2/3)

- Rules propagate all rdf:type assertions in the graph up the subproperty and subclass hierarchies
- rdf1 generates type assertions for all the property names used in the graph
  - uuu aaa yyy → aaa rdf:type rdf:Property
- rdfs3 together with the last RDFS axiomatic triple adds all type assertions for all the class names used
  - → rdf:_2 rdfs:range rdfs:Resource
  - aaa rdfs:range xxx ∧ uuu aaa vvv → vvv
  - rdf:type xxx
Effect of Entailment rules (3/3)

- Any subproperty or subclass assertion generates type assertions for its subject and object via rdfs2 and rdfs3 and the domain and range assertions in the RDFS axiomatic triple set
  - \( \text{aaa rdfs:domain } \text{xxx} \land \text{uuu aaa yyy} \rightarrow \text{uuu rdf:type } \text{xxx} \)
  - \( \text{aaa rdfs:range } \text{xxx} \land \text{uuu aaa vvv} \rightarrow \text{vvv rdf:type } \text{xxx} \)

- For every \( \text{uuu} \) in \( \text{V} \), the rules generate all assertions
  - \( \text{uuu rdf:type rdfs:Resource} \)

- For every class name \( \text{uuu} \), the rules generate
  - \( \text{uuu rdfs:subClassOf rdfs:Resource} \)
Example (axiomatic nodes)
Example (rdf:type and rdfs:subclassOf)
Example (properties, domains, ranges)
Example (reification of properties)
Example (reification of properties+axiomatic properties)
Closure and Reduction

- The **closure** of a graph is the graph defined by all triples that are **inferred** by the deduction system.
  - Using the closure of a graph for storing minimizes query processing time.

- The **reduction** of a graph is the **minimal** subset of triples needed to compute its closure.
  - Using the reduction of a graph for storing minimizes the storage space needed.
Example
Example: closure
Example: reduction

Diagram showing the relationships between classes and properties in a RDF model.
Discussion on RDF

- RDF is a simple, but useful language to make relations between resources explicit.
- It allows to define a simple form of schema which mostly consists of class and property hierarchies as well as relations with domain and range restriction.
- There is a simple rule based reasoning mechanism for RDF schema that in most cases can be used offline because the increase is only factor 1.5 – this can further be reduced by only completing the schema offline.
- Unlike most traditional KR languages RDF schema allows to freely combine modeling primitives which affects reasoning.
Problems with RDF

- Not always meaningful representation
Need logical axioms!

Teacher $\Leftrightarrow \exists \text{teaches.\text{Course}}$

Teacher $\Rightarrow$ Person

Person $\Rightarrow$ Thing $\land \neg$Town
We can detect inconsistencies
What properties do we need?

- Subclass relations
- Sub-properties
- Set operations

- Intersection (logical conjunction)
  - Painter ∩ Writer = \{x | x⊂Painter AND x⊂Writer\}
- Union (logical disjunction)
  - Painter ∪ Writer = \{x | x⊂Painter OR x⊂Writer\}
- Complement
  - complementOf(III,Healthy)
- Disjoint
  - disjointWith(Lung,Liver,Kidney)
What properties do we need? /2

- Cardinality restriction

- Existential quantifier

- Universal quantifier
What properties do we need? /3

- Inverse properties
- Symmetric properties
- Transitivity
References

- RDF Semantics - W3C Recommendation 10 February 2004
  - [http://www.w3.org/TR/rdf-mt/](http://www.w3.org/TR/rdf-mt/)
- Course material for “Practical Reasoning for the Semantic Web” course at the 17th European Summer School in Logic, Language and Information (ESSLLI)
  - [http://www.few.vu.nl/~schlobac/](http://www.few.vu.nl/~schlobac/)
References

- http://media.cwi.nl/survey/
- http://www.mkbergman.com
- http://www.dcs.bbk.ac.uk/~michael/sw/sw.html
- http://www.inf.unibz.it/~debruijn/teaching/swt/
License

- This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License.

- To view a copy of this license, visit [http://creativecommons.org/licenses/by-nc-sa/3.0/](http://creativecommons.org/licenses/by-nc-sa/3.0/) or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.