Semantics

**Comic**: If I have 12 tomatoes and take away two... What is the difference? Exactly, I don't like tomatoes, either. What are you in for this time? Semantics. Detention.
Knowledge Representation for the Semantic Web

Winter Quarter 2011

Slides 5 – 01/20+25/2010

Pascal Hitzler
Kno.e.sis Center
Wright State University, Dayton, OH
http://www.knoesis.org/pascal/
Textbook (required)

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

Foundations of Semantic Web Technologies

Chapman & Hall/CRC, 2010

Choice Magazine Outstanding Academic Title 2010 (one out of seven in Information & Computer Science)

http://www.semantic-web-book.org
Today: RDF(S) semantics
Today’s Session: RDF(S) semantics

1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations
Syntax and Semantics

Syntax: character strings without meaning
Semantics: meaning of the character strings

IF cond(A,B)
THEN display(_354)

Show pixel set "_354" on screen if "A" is of type "B".

Syntax
assignment of meaning
meaning, e.g., "in the world"
Semantics of Programming Languages

Syntax

FUNCTION f(n:natural):natural;
BEGIN
  IF n=0 THEN f:=1
  ELSE f:=n*f(n-1);
END;

What happens at program execution

Procedural Semantics

Formal Semantics

computing factorial

Intended Semantics

\( f : n \mapsto n! \)
Semantics of Logic

Syntax

\[ \forall X \ (p(X) \rightarrow q(X)) \]

Intended Semantics

All humans are mortal

Model-theoretic semantics

Provable in a calculus

Proof-theoretic semantics

\[ \models \]
Recall: Implicit knowledge

• if an RDFS document contains

\[ u \text{ rdf:type ex:Textbook} . \]

and

\[ \text{ex:Textbook rdfs:subClassOf ex:Book} . \]

then

\[ u \text{ rdf:type ex:Book} . \]

is implicitly also the case: it’s a logical consequence. (We can also say it is deduced (deduction) or inferred (inference). We do not have to state this explicitly. Which statements are logical consequences is governed by the formal semantics (covered in the next session).
Recall: Implicit knowledge

- From

\[
\text{ex:Textbook ~ rdfs:subClassOf ~ ex:Book .}
\]
\[
\text{ex:Book ~ rdfs:subClassOf ~ ex:PrintMedia .}
\]

the following is a logical consequence:

\[
\text{ex:Textbook ~ rdfs:subClassOf ~ ex:PrintMedia .}
\]

I.e. rdfs:subClassOf is transitive.
What Semantics Is Good For

- Opinions Differ. Here’s my take.

- Semantic Web requires a shareable, declarative and *computable* semantics.
- I.e., the semantics must be a formal entity which is clearly defined and automatically computable.

- Ontology languages provide this by means of their formal semantics.
- Semantic Web Semantics is given by a relation – the *logical consequence* relation.
In other words

We capture the meaning of information

not by specifying its meaning (which is impossible)
but by specifying

how information interacts with other information.

We describe the meaning indirectly through its effects.
Today’s Session: RDF(S) semantics

1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations
Model-theoretic Semantics

• You need:
  – a language/syntax
  – a notion of model for sentences in the language

• Models
  – are made such that each sentence is either true or false in each model
  – If a sentence $\alpha$ is true in a model $M$, then we write $M \vDash \alpha$

• Logical consequence:
  – $\beta$ is a logical consequence of $\alpha$ (written $\alpha \vDash \beta$), if for all $M$ with $M \vDash \alpha$, we also have $M \vDash \beta$
  – If $K$ is a set of sentences, we write $K \vDash \beta$ if $M \vDash \beta$ for each $M \vDash K$
  – If $J$ is another set of sentences, we write $K \vDash J$ if $K \vDash \beta$ for each $\beta \in J$
  
  (note that the notation $\vDash$ is overloaded)
Logical Consequence
Model theory (contrived) example

- Language:
  variables ...,w,x,y,z,...
symbol $\eta$
  allowed sentences: $a \eta b$ (for $a$, $b$ any variables)

- We want to know:

  What are the logical consequences of the set

  $\{x \eta y, y \eta z\}$

- To answer this, we must say what the models in our semantics are.
Model theory (contrived) example

- Say, a model I of a set K of sentences consists of
  - a set C of cars and
  - a function I(·) which maps each variable to a car in C
  such that, for each sentence a \( \eta \) b in K we have that
    \( I(a) \) has more horsepower than \( I(b) \).

- We now claim that \( \{x \eta y, y \eta z\} \models x \eta z \).
- Proof: Consider any model M of \( \{x \eta y, y \eta z\} \).
  Since \( M \models \{x \eta y, y \eta z\} \), we know that
    M(x) has more horsepower than M(y) and
    M(y) has more horsepower than M(z).
  Hence, M(x) has more horsepower than M(z), i.e. \( M \models x \eta z \).

This argument holds for all models of \( \{x \eta y, y \eta z\} \), therefore
\( \{x \eta y, y \eta z\} \models x \eta z \).
Model theory (contrived) example

• Say, a model $I$ of a set $K$ of sentences consists of
  – a set $C$ of cars and
  – a function $I(\cdot)$ which maps each variable to a car in $C$
    such that, for each sentence $a \eta b$ in $K$ we have that
    $I(a)$ has more horsepower than $I(b)$.

• An *interpretation* $I$ for our language consists of
  – a set $C$ of cars and
  – a function $I(\cdot)$ which maps each variable to a car in $C$.

(and that’s it, i.e. no information whether a sentence is true or
false with respect to $I$).
Today’s Session: RDF(S) semantics

1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations
Now let’s do this for RDF(S)

- **Language**: Whatever is valid RDF(S).
- **Sentences are triples. (Graphs are sets of triples.)**

- Interpretations are given via sets and functions from language vocabularies to these sets.
- Models are defined such that they capture the intended meaning of the RDF(S) vocabulary.
- And there are three different notions:
Simple Interpretations

So we define: a \textit{simple interpretation} $\mathcal{I}$ of a given vocabulary $V$ consists of

- $IR$, a non-empty set of \textit{resources}, alternatively called domain or universe of discourse of $\mathcal{I}$,

- $IP$, the set of \textit{properties} of $\mathcal{I}$ (which may overlap with $IR$),

- $I_{\text{EXT}}$, a function assigning to each property a set of pairs from $IR$, i.e. $I_{\text{EXT}} : IP \rightarrow 2^{IR \times IR}$, where $I_{\text{EXT}}(p)$ is called the \textit{extension} of the property $p$,

- $I_S$, a function, mapping URIs from $V$ into the union of the sets $IR$ and $IP$, i.e. $I_S : V \rightarrow IR \cup IP$,

- $I_L$, a function from the typed literals from $V$ into the set $IR$ of resources and

- $LV$, a particular subset of $IR$, called the set of \textit{literal values}, containing (at least) all untyped literals from $V$. 
Simple Interpretations

Now define an interpretation function \( \mathcal{I} \) (written as exponent).

- every untyped literal "a" is mapped to \( a \), formally: \( ("a")^\mathcal{I} = a \),
- every untyped literal carrying language information "a"@t is mapped to the pair \( \langle a, t \rangle \), i.e. \( ("a"@t)^\mathcal{I} = \langle a, t \rangle \),
- every typed literal \( l \) is mapped to \( I_L(l) \), formally: \( l^\mathcal{I} = I_L(l) \), and
- every URI \( u \) is mapped to \( I_S(u) \), i.e. \( u^\mathcal{I} = I_S(u) \).
Simple Interpretations
Simple models

- The truth value \( s \ p \ o. \mathcal{I} \) of a (grounded*) triple \( s \ p \ o. \) is true exactly if (\( s, p, o \) are contained in \( V \)) and \( \langle s^\mathcal{I}, o^\mathcal{I} \rangle \in I_{\text{EXT}}(p^\mathcal{I}) \).

* A grounded triple does not contain a blank node.
Simple models

- The truth value $s \ p \ o \ I$ of a (grounded*) triple $s \ p \ o$ is true exactly if ($s$, $p$, $o$ are contained in $V$) and $\langle s^I, o^I \rangle \in I_{EXT}(p^I)$.

* A grounded triple does not contain a blank node.
What about blank nodes?

• Say, $A$ is a function from blank nodes to URIs. [these URIs need not be contained in the graph we’re looking at]

• If, in a graph $G$, we replace each blank node $x$ by $A(x)$, then we obtain a graph $G'$ which we call a *grounding* of $G$.

• We know how to do the semantics for the grounded graphs.

• So define:
  $I \models G$ if and only if $I \models G'$ for *at least one* grounding $G'$ of $G$. 
Simple entailment

• A graph $G$ *simply entails* a graph $G'$ if every simple interpretation that is a model of $G$ is also a model of $G'$.

• (Recall that a simple interpretation is a model of a graph $G$ if it is a model of each triple in $G$.)
It’s really simple

• Basically, $G \models G'$ if and only if $G'$ can be obtained from $G$ by replacing some nodes in $G$ by blank nodes.

• It’s really simple entailment.
An RDF-interpretation of a vocabulary $V$ is a simple interpretation of the vocabulary $V \cup V_{RDF}$ that additionally satisfies the following conditions:

- $x \in IP$ exactly if $\langle x, \text{rdf:Property}^I \rangle \in I_{EXT}(\text{rdf:type}^I)$.

- If "s"^^rdf:XMLLiteral is contained in $V$ and $s$ is a well-typed XML-Literal, then

  - $I_L("s"^^\text{rdf:XMLLiteral})$ is the XML value of $s$;

  - $I_L("s"^^\text{rdf:XMLLiteral}) \in LV$;

  - $\langle I_L("s"^^\text{rdf:XMLLiteral}), \text{rdf:XMLLiteral}^I \rangle \in I_{EXT}(\text{rdf:type}^I)$

- If "s"^^rdf:XMLLiteral is contained in $V$ and $s$ is an ill-typed XML literal, then

  - $I_L("s"^^\text{rdf:XMLLiteral}) \not\in LV$ and

  - $\langle I_L("s"^^\text{rdf:XMLLiteral}), \text{rdf:XMLLiteral}^I \rangle \not\in I_{EXT}(\text{rdf:type}^I)$.
• In addition, each RDF-interpretation has to evaluate all the following triples to true:

\[
\begin{align*}
\text{rdf : type} & \quad \text{rdf : type} & \quad \text{rdf : Property}. \\
\text{rdf : subject} & \quad \text{rdf : type} & \quad \text{rdf : Property}. \\
\text{rdf : predicate} & \quad \text{rdf : type} & \quad \text{rdf : Property}. \\
\text{rdf : object} & \quad \text{rdf : type} & \quad \text{rdf : Property}. \\
\text{rdf : first} & \quad \text{rdf : type} & \quad \text{rdf : Property}. \\
\text{rdf : rest} & \quad \text{rdf : type} & \quad \text{rdf : Property}. \\
\text{rdf : value} & \quad \text{rdf : type} & \quad \text{rdf : Property}. \\
\text{rdf : } & _i & \quad \text{rdf : type} & \quad \text{rdf : Property}. \\
\text{rdf : nil} & \quad \text{rdf : type} & \quad \text{rdf : List}. \\
\end{align*}
\]
RDFS-Interpretations Part 1

• Define (for a given RDF-interpretation $\mathcal{I}$):
  
  $I_{\text{CEXT}} : IR \rightarrow 2^{IR}$: We define $I_{\text{CEXT}}(y)$ to contain exactly those elements $x$ for which $\langle x, y \rangle$ is contained in $I_{\text{EXT}}(\text{rdf:type}^\mathcal{I})$. The set $I_{\text{CEXT}}(y)$ is then also called the (class) extension of $y$.

  $IC = I_{\text{CEXT}}(\text{rdfs:Class}^\mathcal{I})$.

• $IR = I_{\text{CEXT}}(\text{rdfs:Resource}^\mathcal{I})$

• $LV = I_{\text{CEXT}}(\text{rdfs:Literal}^\mathcal{I})$

• If $\langle x, y \rangle \in I_{\text{EXT}}(\text{rdfs:domain}^\mathcal{I})$ and $\langle u, v \rangle \in I_{\text{EXT}}(x)$, then $u \in I_{\text{CEXT}}(y)$.

• If $\langle x, y \rangle \in I_{\text{EXT}}(\text{rdfs:range}^\mathcal{I})$ and $\langle u, v \rangle \in I_{\text{EXT}}(x)$, then $v \in I_{\text{CEXT}}(y)$.

• $I_{\text{EXT}}(\text{rdfs:subPropertyOf}^\mathcal{I})$ is reflexive and transitive on $IP$. 
RDFS-Interpretation Part 2

- If \( \langle x, y \rangle \in I_{\text{EXT}}(\text{rdfs:subPropertyOf}^\mathcal{I}) \), then \( x, y \in IP \) and \( I_{\text{EXT}}(x) \subseteq I_{\text{EXT}}(y) \).
- If \( x \in IC \), then \( \langle x, \text{rdfs:Resource}^\mathcal{I} \rangle \in I_{\text{EXT}}(\text{rdfs:subClassOf}^\mathcal{I}) \).
- If \( \langle x, y \rangle \in I_{\text{EXT}}(\text{rdfs:subClassOf}^\mathcal{I}) \), then \( x, y \in IC \) and \( I_{\text{CEXT}}(x) \subseteq I_{\text{CEXT}}(y) \).
- \( I_{\text{EXT}}(\text{rdfs:subClassOf}^\mathcal{I}) \) is reflexive and transitive on \( IC \).
- If \( x \in I_{\text{CEXT}}(\text{rdfs:ContainerMembershipProperty}^\mathcal{I}) \), then \( \langle x, \text{rdfs:member}^\mathcal{I} \rangle \in I_{\text{EXT}}(\text{rdfs:subPropertyOf}^\mathcal{I}) \).
- If \( x \in I_{\text{CEXT}}(\text{rdfs:Datatype}^\mathcal{I}) \), then \( \langle x, \text{rdfs:Literal}^\mathcal{I} \rangle \in I_{\text{EXT}}(\text{rdfs:subClassOf}^\mathcal{I}) \).
Furthermore, all of the following must be satisfied.

- `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:subject` : `rdfs:domain` : `rdf:Statement`
- `rdf:predicate` : `rdfs:domain` : `rdf:Statement`
- `rdf:object` : `rdfs:domain` : `rdf:Statement`
- `rdfs:member` : `rdfs:domain` : `rdfs:Resource`
- `rdf:first` : `rdfs:domain` : `rdf:List`
- `rdf:rest` : `rdfs:domain` : `rdf:List`
- `rdfs:seeAlso` : `rdfs:domain` : `rdfs:Resource`
- `rdfs:isDefinedBy` : `rdfs:domain` : `rdfs:Resource`
Furthermore, all of the following must be satisfied.

<table>
<thead>
<tr>
<th>rdfs:comment</th>
<th>rdfs:domain</th>
<th>rdfs:Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs:label</td>
<td>rdfs:domain</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:value</td>
<td>rdfs:domain</td>
<td>rdfs:Resource</td>
</tr>
<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>
Furthermore, all of the following must be satisfied.

- \texttt{rdfs:ContainerMembershipProperty}
  - \texttt{rdfs:subClassOf rdfs:Property}.
  - \texttt{rdfs:Alt}
  - \texttt{rdfs:subClassOf rdfs:Container}.
  - \texttt{rdfs:Bag}
  - \texttt{rdfs:subClassOf rdfs:Container}.
  - \texttt{rdfs:Seq}
  - \texttt{rdfs:subClassOf rdfs:Container}.

- \texttt{rdfs:isDefinedBy}
  - \texttt{rdfs:subPropertyOf rdfs:seeAlso}.

- \texttt{rdfs:XMLLiteral}
  - \texttt{rdf:type rdfs:Datatype}.
  - \texttt{rdfs:XMLLiteral}
  - \texttt{rdfs:subClassOf rdfs:Literal}.
  - \texttt{rdfs:Datatype}
  - \texttt{rdfs:subClassOf rdfs:Class}.

- \texttt{rdf:_i}
  - \texttt{rdf:type}
  - \texttt{rdfs:ContainerMembershipProperty}.

- \texttt{rdf:_i}
  - \texttt{rdfs:domain rdfs:Resource}.

- \texttt{rdf:_i}
  - \texttt{rdfs:range rdfs:Resource}. \
Today’s Session: RDF(S) semantics

1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations
Back to our contrived example

• Say, a model $I$ of a set $K$ of sentences consists of
  – a set $C$ of cars and
  – a function $I(\cdot)$ which maps each variable to a car in $C$
  such that, for each sentence $a \eta b$ in $K$ we have that
  $I(a)$ has more horsepower than $I(b)$.

• Can we find an algorithm to compute all logical consequences of
  a set of sentences?

• Algorithm Input: set $K$ of sentences
  1. The algorithm non-deterministically selects two sentences
     from $K$. If the first sentence is $a \eta b$, and the second
     sentence is $b \eta c$, then add $a \eta c$ to $K$.
     \[
     \text{IF } a \eta b \in K \text{ and } b \eta c \in K \text{ THEN } K \leftarrow \{a \eta c\}
     \]
  2. Repeat step 1 until no selection results in a change of $K$.
  3. Output: $K$
Back to the example

- The algorithm produces only logical consequences: it is sound with respect to the model-theoretic semantics.
- The algorithm produces all logical consequences: it is complete with respect to the model-theoretic semantics.
- The algorithm always terminates.
- The algorithm is non-deterministic.
- What is the computational complexity of this algorithm?

And actually, the algorithm just given is not sound and complete. Do you see, why?
What do we gain?

• Recall:
  • \( \beta \) is a logical consequence of \( \alpha \) (written \( \alpha \models \beta \)), if for all \( M \) with \( M \models \alpha \), we also have \( M \models \beta \) are

• Implementing model-theoretic semantics directly is not feasible: We would have to deal with all models of a knowledge base. Since there are a lot of cars in this world, we would have to check a lot of possibilities.

• Proof theory reduces model-theoretic semantics to symbol manipulation! It removes the models from the process.
Deduction rules

IF \(a \eta b \in K\) and \(b \eta c \in K\) THEN \(K \leftarrow \{a \eta c\}\)

is a so-called *deduction rule*. Such rules are usually written schematically as

\[
\begin{array}{c}
\begin{array}{c}
\quad a \eta b
\end{array}
\begin{array}{c}
\quad b \eta c
\end{array}
\end{array}
\quad \begin{array}{c}
\quad a \eta c
\end{array}
\]

\[
\begin{array}{c}
\quad a \eta b
\begin{array}{c}
\quad b \eta c
\end{array}
\quad a \eta c
\end{array}
\]


Today’s Session: RDF(S) semantics

1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations
First, some notation

- $a$ and $b$ can refer to arbitrary URIs (i.e. anything admissible for the predicate position in a triple),
- $_:n$ will be used for the ID of a blank node,
- $u$ and $v$ refer to arbitrary URIs or blank node IDs (i.e. any possible subject of a triple),
- $x$ and $y$ can be used for arbitrary URIs, blank node IDs or literals (i.e. anything admissible for the object position in a triple), and
- $l$ may be any literal.
Simple Entailment Rules

\[
\frac{u \quad a \quad x}{u \quad a \quad _:n} \quad \text{se1}
\]

\[
\frac{u \quad a \quad x}{_:n \quad a \quad x} \quad \text{se2}
\]

_:n must not be contained in the graph the rule is applied to
Additional RDF-entailment Rules

\[ \frac{u \ a \ x}{\text{rdf}ax} \]

For all RDF axiomatic triples \( u \ a \ x \).

\[ \frac{u \ a \ l}{\text{lg}} \]

Where \( _n \) does not yet occur in the graph.

\[ \frac{u \ a \ y}{\text{rdf}1} \]

\[ \frac{a \text{ rdf:type } \text{rdf:Property}}{\text{rdf}1} \]

\[ \frac{u \ a \ l}{\text{rdf}2} \]

Where \( _n \) does not yet occur in the graph, unless it has been introduced by a preceding application of the lg rule.
for all RDFS axiomatic triples with \(_:n\) as usual
Additional RDFS-entailment Rules - II

\[\begin{align*}
& u \ rdfs:subPropertyOf \ v . \quad v \ rdfs:subPropertyOf \ x . \\
& \quad \quad u \ rdfs:subPropertyOf \ x . \quad \text{rdfs5}
\end{align*}\]

\[\begin{align*}
& u \ rdf:type \ rdf:Property . \\
& \quad u \ rdfs:subPropertyOf \ u . \\
& \quad \text{rdfs6}
\end{align*}\]

\[\begin{align*}
& a \ rdfs:subPropertyOf \ b . \quad u \ a \ y . \\
& \quad u \ b \ y . \\
& \quad \text{rdfs7}
\end{align*}\]

\[\begin{align*}
& u \ rdf:type \ rdfs:Class . \\
& \quad u \ rdfs:subClassOf \ rdfs:Resource . \\
& \quad \text{rdfs8}
\end{align*}\]

\[\begin{align*}
& u \ rdfs:subClassOf \ x . \quad v \ rdf:type \ u . \\
& \quad v \ rdf:type \ x . \\
& \quad \text{rdfs9}
\end{align*}\]

\[\begin{align*}
& u \ rdf:type \ rdfs:Class . \\
& \quad u \ rdfs:subClassOf \ u . \\
& \quad \text{rdfs10}
\end{align*}\]
where \( \_n \) identifies a blank node introduced by an earlier “weakening” of the literal \( l \) via the rule \( l_g \)
Completeness?

• The deduction rules for simple and RDF entailment are sound and complete.

• The deduction rules for RDFS entailment are sound.

The spec says, they are also complete, but they are not:

```prolog
ex:isHappilyMarriedTo rdfs:subPropertyOf _:bnode.
_:bnode rdfs:domain ex:Person.
ex:markus ex:isHappilyMarriedTo ex:anja.
```

has as logical consequence

```prolog
ex:markus rdf:type ex:Person.
```

but this is not derivable using the deduction rules.
Complexity

Simple, RDF, and RDFS entailment are NP-complete problems.

If we disallow blank nodes, all three entailment problems are polynomial.
Does RDFS semantics do what it should?

Does entail

ex:speaksWith rdfs:domain ex:Homo .
ex:Homo rdfs:subClassOf ex:Primates .

entail

ex:speaksWith rdfs:domain ex:Primates .
A new W3C working group has just been chartered and should continue work shortly:

http://www.w3.org/2011/01/rdf-wg-charter
Today’s Session: RDF(S) semantics

1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations
Class project: next step

• keep bugfixing
• find, for your RDF Schema ontology, each of the following:
  – a triple which is RDFS-entailed, but not RDF-entailed
  – a triple which is RDF-entailed, but not simply entailed
  – a triple which is simply entailed
• For each of them, write down a justification why it is entailed.

• send to me by Sunday 30th of January
  – the current version of your Turtle RDF Schema document
  – the three entailed triples with explanations.
Today’s Session: RDF(S) semantics

1. What is Semantics?
2. What is Model-theoretic Semantics?
3. Model-theoretic Semantics for RDF(S)
4. What is Proof-theoretic Semantics?
5. Proof-theoretic Semantics for RDF(S)
6. Class Project
7. Class Presentations
Class presentations – first topics

- SPARQL 1.1 entailment regimes:  
  http://www.w3.org/2009/sparql/docs/entailment/xmlspec.xml
- Jacopo Urbani, Spyros Kotoulas, Jason Maassen, Frank van Harmelen, Henri E. Bal: OWL Reasoning with WebPIE: Calculating the Closure of 100 Billion Triples. ESWC (1) 2010: 213-227
- Yuan Ren, Jeff Z. Pan, Yuting Zhao: Soundness Preserving Approximation for TBox Reasoning. AAAI 2010
- Franz Baader, Sebastian Brandt, Carsten Lutz: Pushing the EL Envelope. IJCAI 2005: 364-369
Thursday 13th of January: RDFS Part I
Tuesday 18th of January: Exercise Session
Thursday 20th of January: RDF and RDFS Semantics
Tuesday 25th of January: RDF and RDFS Semantics
Thursday 27th of January: Description Logics
Tuesday 1st of March: Description Logic Semantics

Estimated breakdown of sessions:
Intro + XML: 2    RDF: 4    OWL and Logic: 6
SPARQL and Querying: 2    Class Presentations: 3
Exercise sessions: 3