Semantic Technologies

Part 9: RDF(S) Semantics

Werner Nutt

Acknowledgment

These slides are based on the Latex version of slides
by Sebastian Rudolph of TU Dresden

Why Formal Semantics?

- After the introduction of RDF(S), criticism of tool developers: different tools were incompatible (despite the existing specification)
- E.g., query engines:
  - same RDF document
  - same SPARQL query
  - different answers

$\rightsquigarrow$ Thus a model-theoretic formal semantics was defined for RDF(S)

This means, RDF and RDFS are viewed als logic languages

See “RDF 1.1 Semantics” (http://www.w3.org/TR/rdf11-mt/)

Building Blocks of a Logic 1: Syntax

Syntax defines the formulas of the logic

- Propositional logic: propositions like

$$p \wedge q$$
$$p \wedge q \rightarrow q \vee \neg q$$

- First order (predicate) logic: formulas with predicates/relations, variables, and quantifiers, e.g.,

$$hasFriend(john, mary)$$
$$\exists x(hasFriend(john, x))$$
$$hasFriend(john, x)$$
$$\forall x(hasFriend(john, x) \rightarrow likes(x, john))$$
$$\forall x(Person(x) \rightarrow \exists y(Person(y) \wedge hasFriend(x, y)))$$

Sentences (= formulas with bound variables)

Building Blocks of a Logic 2: Interpretations

Interpretations define about what scenarios the logic talks:

- Propositional logic: Formulas are interpreted by **truth assignments** which assign a truth value to every proposition, e.g.

$$\alpha \colon p \mapsto \mathit{true},\ q \mapsto \mathit{false}$$

- Predicate logic interpretations $\mathcal{I}$ consist of
  - a **domain** $\Delta^{\mathcal{I}}$ of the interpretation (a nonempty set)
  - an **interpretation function** $\cdot^{\mathcal{I}}$, which maps every constant c to a domain element $c^{\mathcal{I}}$
  - **predicate interpretations**, which are relations on $\Delta^{\mathcal{I}}$, e.g.
    - Person is interpreted by $\mathcal{I}$ as a unary relation $\mathit{Person}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
    - likes is interpreted by $\mathcal{I}$ as a binary relation $\mathit{likes}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

  In addition, there are **variable assignments** α, which assign to each variable a domain element

Building Blocks of a Logic 3: Satisfaction

For every logic, there is a **satisfaction** relationship ("$\models$") that defines when an interpretation satisfies a formula

Case 1: Satisfaction in propositional logic

Let $\alpha \colon p \mapsto \mathit{true},\ q \mapsto \mathit{false}$.
Then

- $\alpha \models p \land q \to q \lor \neg q$
- $\alpha \not\models p \land q$

Note: This is due to the definition of

- when α satisfies a proposition p (base case)
- how satisfaction of a complex formula $\phi \land \psi$, $\phi \lor \psi$, $\neg\phi$, etc. depends on the satisfaction of the components ϕ, ψ (inductive step)

Building Blocks of a Logic 3: Satisfaction

Case 2: Satisfaction in predicate logic

- Let $\Delta^{\mathcal{I}} = \{1,2,3\}$, let $john^{\mathcal{I}} = 1$, $mary^{\mathcal{I}} = 2$, and let
  $Person^{\mathcal{I}} = \{1,2,3\}, \quad hasFriend^{\mathcal{I}} = \{(1,2),\ (2,1)\}$, and
  $likes^{\mathcal{I}} = \{(1,2),\ (2,1),\ (2,3),\ (3,1)\}$.

  Also, let $\alpha\colon x \mapsto 3, y \mapsto 2$. Then

$$
\begin{aligned}
\mathcal{I}, \alpha &\models hasFriend(john, mary)\\
\mathcal{I}, \alpha &\models \exists x(hasFriend(john, x))\\
\mathcal{I}, \alpha &\not\models hasFriend(john, x)\\
\mathcal{I}, \alpha &\models \forall x(hasFriend(john, x) \to likes(x, john))\\
\mathcal{I}, \alpha &\not\models \forall x(Person(x) \to \exists y(Person(y) \land hasFriend(x, y)))
\end{aligned}
$$

Note: For sentences (= formulas without free variables),
α has no influence (and could be dropped)

Reminder on Satisfaction

Let ϕ, ψ be formulas in predicate logic. Then

- $\mathcal{I}, \alpha \models R(x, y)$ iff $(\alpha(x), \alpha(y)) \in R^{\mathcal{I}}$

  ... similarly for constants and mixtures of constants and variables
- $\mathcal{I}, \alpha \models \phi \land \psi$ iff $\mathcal{I}, \alpha \models \phi$ and $\mathcal{I}, \alpha \models \psi$
- $\mathcal{I}, \alpha \models \phi \lor \psi$ iff $\mathcal{I}, \alpha \models \phi$ or $\mathcal{I}, \alpha \models \psi$
- $\mathcal{I}, \alpha \models \neg\phi$ iff $\mathcal{I}, \alpha \not\models \phi$
- $\mathcal{I}, \alpha \models \exists x(\phi)$ iff there exists a $d \in \Delta^{\mathcal{I}}$ such that $\mathcal{I}, \alpha[x/d] \models \phi$

  where $\alpha[x/d](x) = d$ and $\alpha[x/d](y) = \alpha(y)$ for $y \neq x$
- $\mathcal{I}, \alpha \models \forall x(\phi)$ iff for all $d \in \Delta^{\mathcal{I}}$ it holds that $\mathcal{I}, \alpha[x/d] \models \phi$

  where $\alpha[x/d]$ is defined as above

Satisfaction in propositional logic is defined similarly

Properties of Logical Formulas

Let ϕ be a formula in predicate logic. We say that

- $\mathcal{I}, \alpha$ is a **model** of ϕ if $\mathcal{I}, \alpha$ satisfies ϕ
- ϕ is **satisfiable** if ϕ has **some** model
- ϕ is **unsatisfiable** if ϕ has **no** model
- ϕ is **valid** if for all $\mathcal{I}$, α it holds that

$$\mathcal{I}, \alpha \models \phi,$$

  that is, every pair $\mathcal{I}, \alpha$ is a model of ϕ

The same properties are defined analogously for propositional logic

What can you say about $\neg\phi$ if ϕ is satisfiable, unsatisfiable, or valid?

Entailment

What does it mean that a formula ψ logically follows from a formula ϕ?

We say that ψ **follows from** ϕ, written

$$\phi \models \psi$$

if every model of ϕ is a also a model of ψ.

We then also say that ϕ **entails** ψ.

This can be generalized to sets Φ of formulas.

- $\mathcal{I}, \alpha$ is a model of Φ if $\mathcal{I}, \alpha$ is a model for every $\phi \in \Phi$
- Φ entails ψ, written $\Phi \models \psi$, if every model of Φ is also model of ψ

Intuition: Φ is considered as the conjunction of all its elements.

Entailment: Example

Consider

$$\phi_1 = hasFriend(john, mary)$$
$$\phi_1 = \forall x(hasFriend(john, x) \rightarrow likes(x, john))$$
$$\psi_1 = likes(john, mary)$$
$$\psi_2 = likes(mary, john)$$

What can you say about $\Phi = \{\phi_1, \phi_2\}$ entailing ψ_1, ψ_2?

Inference Rules

How can we find out, for arbitrary Φ and ψ, whether

$$\Phi \models \psi \quad ?$$

Trying out all interpretations (and assignments) is

- complex in propositional logic
- impossible in predicate logic.

However, sometimes rules allow us to **infer** that a formula follows from other formulas.

Example:

$$\frac{\phi \qquad \phi \rightarrow \psi}{\psi}$$

Inference Rules/2

An inference rule has the form

$$\frac{\phi_1 \quad \ldots \quad \phi_n}{\psi}$$

We call $\phi_1, \ldots, \phi_n$ the **premises** of the rule ψ the conclusion of the rule.

An inference rul is **sound** if

- every model of $\phi_1, \ldots, \phi_n$ is also a model of ψ

Intuition: With a sound rule, we infer true conclusions from true premises.

How is RDF(S) Linked to a Logic?

- To start with: what are the sentences/formulas in RDF(S)?
  - Basic syntactic elements (vocabulary V):
    IRIs, **bnodes** and **literals**
    (these are not sentences/formulas themselves)
  - Every triple

    $$(\mathrm{s}, \mathrm{p}, \mathrm{o}) \in (\mathrm{IRI} \cup \mathrm{bnode}) \times \mathrm{IRI} \times (\mathrm{IRI} \cup \mathrm{bnode} \cup \mathrm{literal})$$

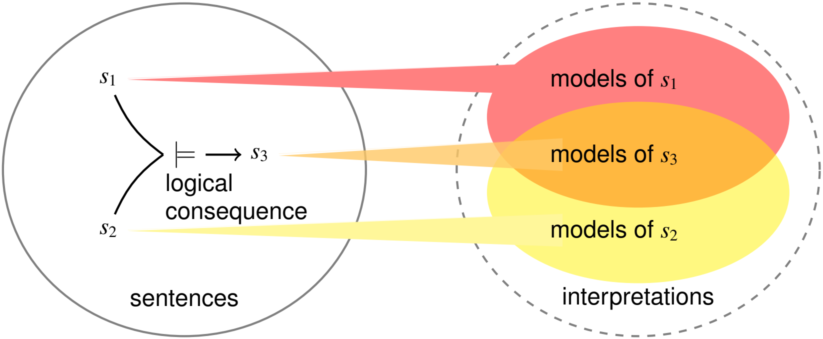
    is a sentence
  - Every finite set of triples (denoted: graph) is a sentence

How is RDF(S) Linked to a Logic?

What is the semantics? A **consequence relation** that defines

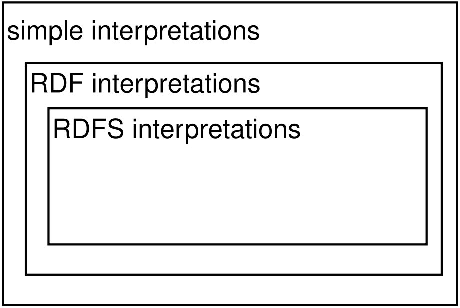
- when an RDF(S) graph G' **logically follows** from another RDF(S) graph G (written $G \models G'$)

To introduce this semantics, we define a set of interpretations and specify under which conditions an interpretation is a model of a graph.



Semantics of RDF(S)

- We proceed stepwise:



- The more we restrict the set of interpretations, the stronger the consequence relation becomes

Semantics of the Simple Entailment

Definition (Simple Interpretation)

A **simple interpretation** $\mathcal{I}$ for a vocabulary V consists of

- IR, a non-empty set of *resources*, also referred to as domain, with
- $\mathrm{LV} \subseteq \mathrm{IR}$, the set of *literal values*, that contains (at least) all untyped literals from V, and
- IP, the set of *properties* of $\mathcal{I}$;
- $\mathrm{I_S}$, a function, mapping IRIs from V to the union of the sets IR and IP, i.e., $\mathrm{I_S} : \mathrm{V} \to \mathrm{IR} \cup \mathrm{IP}$,
- $\mathrm{I_{EXT}}$, a function, mapping every property to a set of pairs from IR, i.e., $\mathrm{I_{EXT}} : \mathrm{IP} \to 2^{\mathrm{IR} \times \mathrm{IR}}$ and
- $\mathrm{I_L}$, a function mapping typed literals from V into the set IR of resources.

Semantics of the Simple Entailment

- $\mathbb{R}$, the set of resources, is also called domain or universe of discourse of $\mathcal{I}$
- $\mathsf{I}_{\mathsf{EXT}}(p)$ is also referred to as the *extension* of the property p

Remark

In summary:

- There is a domain $\mathbb{R}$, consisting of *resources*, which may include numbers, booleans, and other values
- There are also properties
- Some IRIs are interpreted as domain elements, others as *or properties*
- Properties are interpreted as binary relations on the domain

Semantics of the Simple Entailment

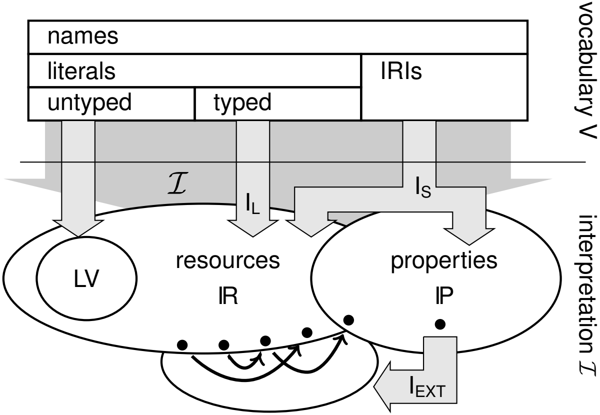
Definition (Interpretation Function)

Based on $\mathsf{I_L}$ and $\mathsf{I_S}$, we define $\cdot^{\mathcal{I}}$ as follows:

- every untyped literal `"a"` is mapped to a : $(\texttt{"a"})^{\mathcal{I}} = \mathsf{a}$
- every untyped literal with language information `"a"@t` is mapped to the pair $\langle \mathsf{a}, \mathsf{t} \rangle$, that is: $(\texttt{"a"@t})^{\mathcal{I}} = \langle \mathsf{a}, \mathsf{t} \rangle$,
- every typed literal l is mapped to $\mathsf{I_L}(l)$, that is: $l^{\mathcal{I}} = \mathsf{I_L}(l)$ and
- every IRI i is mapped to $\mathsf{I_S}(i)$, hence: $i^{\mathcal{I}} = \mathsf{I_S}(i)$.

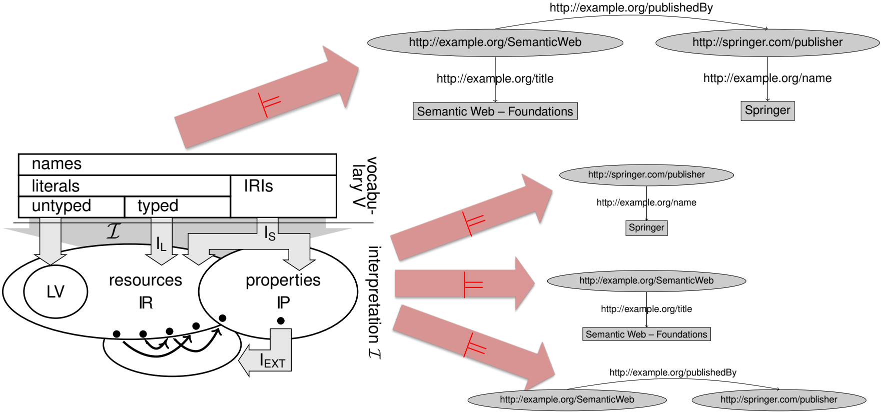
Semantics of the Simple Entailment

Interpretation (schematic):



Semantics of the Simple Entailment

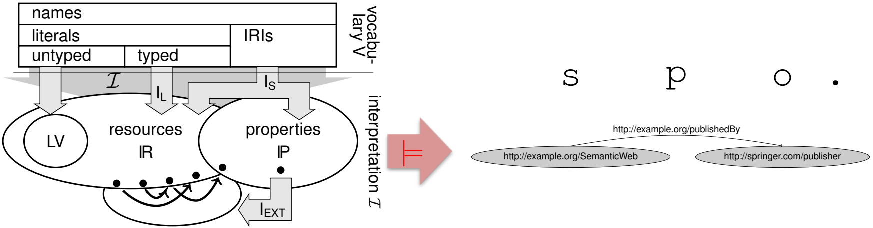
- Question: When is a given interpretation a model of a graph?
- . . . if it is a model for every triple of the graph!



Semantics of the Simple Entailment

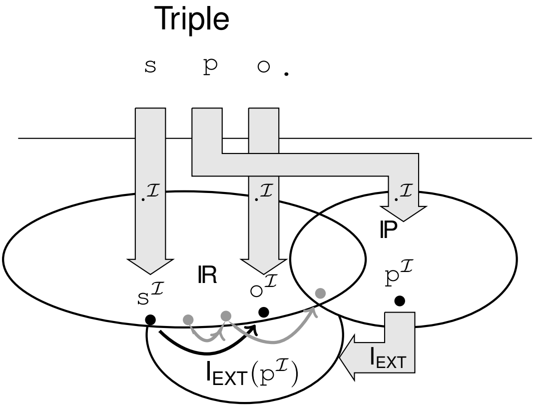
Question: When is a given interpretation a model of a triple?

- ... if subject, predicate, and object are contained in V
- ... and additionally $\langle s^{\mathcal{I}}, o^{\mathcal{I}} \rangle \in I_{EXT}(p^{\mathcal{I}})$ holds



Semantics of Simple Entailment

Schematically:



Semantics of Simple Entailment

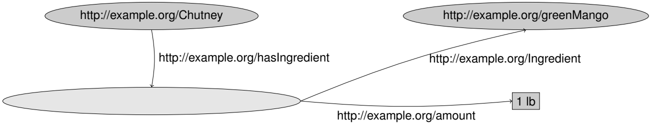
…oops, we forgot the bnodes!

- Assume, A is a function mapping all bnodes to elements of IR
- Given an interpretation $\mathcal{I}$,
  - let $\mathcal{I} + A$ behave just like $\mathcal{I}$ on the vocabulary,
  - and additionally, for every bnode `_:label`,
    let $(\texttt{\_:label})^{\mathcal{I}+A} = A(\texttt{\_:label})$
- Now, an interpretation $\mathcal{I}$ is a model of an RDF graph G,
  if there exists an A such that all triples are satisfied w.r.t. $\mathcal{I} + A$

In other words, we have extended $\mathcal{I}$ by an interpretation A for the bnodes

Simple Interpretations: Example

Given graph G:

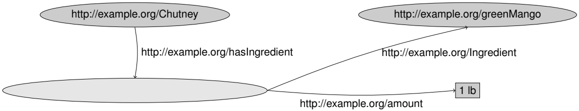


and interpretation $\mathcal{I}$:

$$
\begin{array}{rcl}
\mathrm{IR} & = & \{c, g, h, z, l, m, \text{1 lb}\} \\
\mathrm{IP} & = & \{h, z, m\} \\
\mathrm{LV} & = & \{\text{1 lb}\} \\
\mathrm{I_{EXT}} & = & h \mapsto \{\langle c, l\rangle\} \\
 & & z \mapsto \{\langle l, g\rangle\} \\
 & & m \mapsto \{\langle l, \text{1 lb}\rangle\}
\end{array}
\qquad
\begin{array}{lll}
\mathrm{I_S} = & \texttt{ex:Chutney} & \mapsto c \\
 & \texttt{ex:greenMango} & \mapsto g \\
 & \texttt{ex:hasIngredient} & \mapsto h \\
 & \texttt{ex:ingredient} & \mapsto z \\
 & \texttt{ex:amount} & \mapsto m \\
\mathrm{I_L} & \text{is the “empty function”} &
\end{array}
$$

Is $\mathcal{I}$ a model of G?

Simple Interpretations: Example



$$
\begin{array}{rcl}
\mathsf{IR} & = & \{c, g, h, z, l, m, \text{1 lb}\} \\
\mathsf{IP} & = & \{h, z, m\} \\
\mathsf{LV} & = & \{\text{1 lb}\} \\
\mathsf{I_{EXT}} & = & h \mapsto \{\langle c, l\rangle\} \\
 & & z \mapsto \{\langle l, g\rangle\} \\
 & & m \mapsto \{\langle l, \text{1 lb}\rangle\}
\end{array}
\qquad
\begin{array}{rcll}
\mathsf{I_S} & = & \texttt{ex:Chutney} & \mapsto c \\
 & & \texttt{ex:greenMango} & \mapsto g \\
 & & \texttt{ex:hasIngredient} & \mapsto h \\
 & & \texttt{ex:ingredient} & \mapsto z \\
 & & \texttt{ex:amount} & \mapsto m \\
\mathsf{I_L} & & \text{is the “empty function”} &
\end{array}
$$

- If we pick $A\colon \texttt{\_:id1} \mapsto l$, then we get

$$
\begin{array}{lllll}
\langle \texttt{ex:Chutney}^{\mathcal{I}+A}, \texttt{\_:id1}^{\mathcal{I}+A}\rangle & = \langle c, l\rangle & \in \mathsf{I_{EXT}}(h) & = \mathsf{I_{EXT}}(\texttt{ex:hasIngredient}^{\mathcal{I}+A}) \\
\langle \texttt{\_:id1}^{\mathcal{I}+A}, \texttt{ex:greenMango}^{\mathcal{I}+A}\rangle & = \langle l, g\rangle & \in \mathsf{I_{EXT}}(z) & = \mathsf{I_{EXT}}(\texttt{ex:ingredient}^{\mathcal{I}+A}) \\
\langle \texttt{\_:id1}^{\mathcal{I}+A}, \texttt{"1 lb"}^{\mathcal{I}+A}\rangle & = \langle l, \texttt{1 lb}\rangle & \in \mathsf{I_{EXT}}(m) & = \mathsf{I_{EXT}}(\texttt{ex:amount}^{\mathcal{I}+A})
\end{array}
$$

- Therefore, $\mathcal{I}$ is a model of G.

Simple Entailment

- The definition of simple interpretations fixes the notion of simple entailment for RDF graphs
- Question: How can this (abstractly defined) semantics be turned into something computable?
- Answer: Deduction rules

Simple Entailment

Deduction rules for simple entailment:

$$\frac{\texttt{u} \quad \texttt{a} \quad \texttt{x} \ .}{\texttt{u} \quad \texttt{a} \quad \texttt{\_:n} \ .} \ \text{se1}$$

$$\frac{\texttt{u} \quad \texttt{a} \quad \texttt{x} \ .}{\texttt{\_:n} \quad \texttt{a} \quad \texttt{x} \ .} \ \text{se2}$$

- Precondition for applying these rules:
  the bnode has not yet been associated with another IRI or literal

Simple Entailment

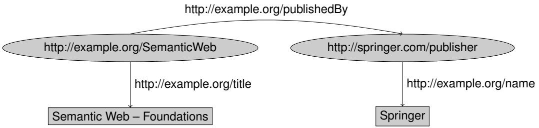
Theorem

[Soundness and Completeness of Inference Rules] A graph G_2 is simply entailed by a graph G_1

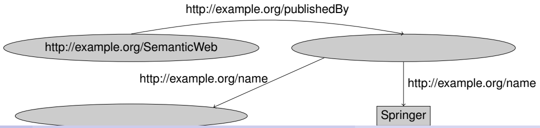
if and only if

G_1 can be extended to a graph G_1' by applying the rules se1 and se2 such that G_2 is contained in G_1'.

Example.: the graph



simply entails



RDF Interpretations

RDF interpretations are specific simple interpretations,
where additional conditions are imposed on the URIs of the RDF vocabulary

```
rdf:type  rdf:Property  rdf:XMLLiteral  rdf:nil
rdf:List  rdf:Statement  rdf:subject  rdf:predicate
rdf:object  rdf:first  rdf:rest  rdf:Seq  rdf:Bag
rdf:Alt  rdf:_1  rdf:_2  ...
```

in order to realize their intended semantics.

We present the conditions together with corresponding inference rules

Conditions for RDF Interpretations

An RDF interpretation for a vocabulary V is a simple interpretation for the vocabulary $V \cup V_{RDF}$ that additionally satisfies the following conditions:

1. $x \in \mathrm{IP}$ exactly if $\langle x, \texttt{rdf:Property}^{\mathcal{I}} \rangle \in \mathsf{I}_{\mathsf{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$.

“For every triple predicate we can infer that
it is an member of the class of all properties.”

$$\frac{\texttt{u a y}}{\texttt{a rdf:type rdf:Property}} \ \text{rdf1}$$

Conditions for RDF Interpretations

2. If `"s"^^rdf:XMLLiteral` is contained in V and `s` is a well-formed XML literal, then
   - $\mathsf{I}_\mathsf{L}(\texttt{"s"\^{}\^{}rdf:XMLLiteral})$ is the XML value of `s`;
   - $\mathsf{I}_\mathsf{L}(\texttt{"s"\^{}\^{}rdf:XMLLiteral}) \in \mathsf{LV}$;
   - $\langle \mathsf{I}_\mathsf{L}(\texttt{"s"\^{}\^{}rdf:XMLLiteral}), \texttt{rdf:XMLLiteral}^{\mathcal{I}} \rangle \in \mathsf{I}_\mathsf{EXT}(\texttt{rdf:type}^{\mathcal{I}})$

$$\frac{\texttt{u a l}}{\texttt{l rdf:type rdf:XMLLiteral}}$$ **???** für `l` a well-formed XML literal

Oops, literals must not occur in subject position!

Conditions for RDF Interpretations

2. If `"s"^^rdf:XMLLiteral` is contained in V and `s` is a well-formed XML literal, then
   - $\mathsf{I_L}(\texttt{"s"\^{}\^{}rdf:XMLLiteral})$ is the XML value of `s`;
   - $\mathsf{I_L}(\texttt{"s"\^{}\^{}rdf:XMLLiteral}) \in \mathsf{LV}$;
   - $\langle \mathsf{I_L}(\texttt{"s"\^{}\^{}rdf:XMLLiteral}), \texttt{rdf:XMLLiteral}^{\mathcal{I}} \rangle \in \mathsf{I_{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$

$$\frac{\texttt{u a l}}{\texttt{u a \_:n}} \quad \textbf{lg}$$

`l` a literal, `_:n` not bound otherwise

$$\frac{\texttt{u a \_:n}}{\texttt{\_:n rdf:type rdf:XMLLiteral}} \quad \textbf{rdf2}$$

If rule lg has assigned `_:n` to the XML Literal `l`

Rule lg is called the “literal generalization rule”

Conditions for RDF Interpretations

3. If $\texttt{"s"\^{}\^{}rdf:XMLLiteral}$ is contained in V and $\texttt{s}$ is an *ill-formed* XML literal, then
   - $\mathsf{I_L}(\texttt{"s"\^{}\^{}rdf:XMLLiteral}) \notin$ LV and
   - $\langle \mathsf{I_L}(\texttt{"s"\^{}\^{}rdf:XMLLiteral}), \texttt{rdf:XMLLiteral}^{\mathcal{I}} \rangle \notin \mathsf{I_{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$.

RDF Interpretations

- Note: x is a property exactly if it is linked to the resource denoted by `rdf:Property` via the `rdf:type` property

  (this has the direct consequence that in every RDF interpretation $\mathrm{IP} \subseteq \mathrm{IR}$ holds).

- The value space of the `rdf:XMLLiteral` datatype contains for every well-formed XML string exactly one so-called XML value. The RDF specs only require that this value is neither an XML string itself nor a data value of any XML Schema datatype nor a Unicode string.

RDF Interpretations

- Additional requirement: every RDF interpretation must be a model of the following “axiomatic” triples:

```
rdf:type        rdf:type   rdf:Property .
rdf:subject     rdf:type   rdf:Property .
rdf:predicate   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:_1          rdf:type   rdf:Property .
rdf:_2          rdf:type   rdf:Property .
...             rdf:type   rdf:Property .
rdf:nil         rdf:type   rdf:List .
```

$$\frac{}{\texttt{u a x}}\ \text{rdfax}$$

every axiomatic triple “u a x .” can always be derived

RDF Entailment

Theorem (Soundness and Completeness of RDF Inference Rules)

A graph G_2 is RDF-entailed by a graph G_1, written $G_1 \models G_2$,
if and only if there is a graph G_1', such that

- G_1' can be derived from G_1 via lg, rdf1, rdf2 and rdfax and
- G_2 is simply entailed by G_1'.

Note: The deduction process has two stages

RDFS Interpretations

RDFS interpretations are specific RDF interpretations, where additional constraints are imposed for the URIs of the RDFS vocabulary

```
rdfs:domain        rdfs:range              rdfs:Resource
rdfs:Literal       rdfs:Datatype           rdfs:Class
rdfs:subClassOf    rdfs:subPropertyOf      rdfs:Container
rdfs:member        rdfs:ContainerMembershipProperty
rdfs:comment       rdfs:seeAlso            rdfs:isDefinedBy
rdfs:label
```

such that the intended semantics of these URIs is realized.

RDFS Interpretations/2

For the sake of easier representation, we introduce for each interpretation $\mathcal{I}$

- the function $\mathsf{I}_{\mathsf{CEXT}}$ and
- the set IC.

They are defined as follows:

- $\mathsf{I}_{\mathsf{CEXT}}$ maps resources to sets of resources, i.e.,

$$\mathsf{I}_{\mathsf{CEXT}} : \mathsf{IR} \to 2^{\mathsf{IR}},$$

  where $\mathsf{I}_{\mathsf{CEXT}}(y)$ contains exactly those elements x,
  for which $\langle x, y\rangle$ is contained in $\mathsf{I}_{\mathsf{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$.
  We call $\mathsf{I}_{\mathsf{CEXT}}(y)$ the *(class) extension* of y.

- IC is the extension of the specific IRI `rdfs:Class`, hence:

$$\mathsf{IC} = \mathsf{I}_{\mathsf{CEXT}}(\texttt{rdfs:Class}^{\mathcal{I}}).$$

Note: both $\mathsf{I}_{\mathsf{CEXT}}$ as well as IC are fully determined by $\cdot^{\mathcal{I}}$ and $\mathsf{I}_{\mathsf{EXT}}$.

RDFS Interpretations/3

An *RDFS interpretation* for a vocabulary V is an RDF interpretation for the vocabulary $\mathsf{V} \cup \mathsf{V}_{\mathsf{RDFS}}$ that additionally satisfies the following criteria:

- $\mathsf{IR} = \mathsf{I}_{\mathsf{CEXT}}(\texttt{rdfs:Resource}^{\mathcal{I}})$

  Every resource is of type `rdfs:Resource`.

- $\mathsf{LV} = \mathsf{I}_{\mathsf{CEXT}}(\texttt{rdfs:Literal}^{\mathcal{I}})$

  Every untyped and every well-formed typed literal
  is of type `rdfs:Literal`.

- If $\langle x, y\rangle \in \mathsf{I}_{\mathsf{EXT}}(\texttt{rdfs:domain}^{\mathcal{I}})$ and $\langle u, v\rangle \in \mathsf{I}_{\mathsf{EXT}}(x)$, then $u \in \mathsf{I}_{\mathsf{CEXT}}(y)$.

  If the property `rdfs:domain` connects x with y, and
  the property x connects the resources u and v,
  then u is of type y.

RDFS Interpretations/4

- If $\langle x, y\rangle \in \mathsf{I}_{\mathsf{EXT}}(\texttt{rdfs:range}^{\mathcal{I}})$ and $\langle u, v\rangle \in \mathsf{I}_{\mathsf{EXT}}(x)$, then $v \in \mathsf{I}_{\mathsf{CEXT}}(y)$.

  If the property `rdfs:range` connects x with y and
  the property x connects the resources u and v,
  then v is of type y.

- $\mathsf{I}_{\mathsf{EXT}}(\texttt{rdfs:subPropertyOf}^{\mathcal{I}})$ is reflexive and transitive on $\mathbb{P}$.

  The `rdfs:subPropertyOf` property connects
  every property with itself.

  Moreover, if `rdfs:subPropertyOf` connects
  a property x with a property y and additionally y with a property z,
  then `rdfs:subPropertyOf` also connects x directly with z.

RDFS Interpretations/5

- If $\langle x, y\rangle \in \mathsf{I}_{\mathsf{EXT}}(\texttt{rdfs:subPropertyOf}^{\mathcal{I}})$,
  then $x, y \in \mathsf{IP}$ and $\mathsf{I}_{\mathsf{EXT}}(x) \subseteq \mathsf{I}_{\mathsf{EXT}}(y)$.

  If `rdfs:subPropertyOf` connects x with y,
  then both x and y are properties.

  Every pair of resources contained in the extension of x,
  is also contained in the extension of y.

- If $x \in \mathsf{IC}$, then $\langle x, \texttt{rdfs:Resource}^{\mathcal{I}}\rangle \in \mathsf{I}_{\mathsf{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$.

  If x represents a class,
  then it has to be a subclass of the class of all resources,
  i.e., the pair containing x and `rdfs:Resource`
  is in the extension of `rdfs:subClassOf`.

RDFS Interpretations/6

- If $\langle x, y \rangle \in \mathsf{I}_{\mathsf{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$,
  then $x, y \in \mathsf{IC}$ and $\mathsf{I}_{\mathsf{CEXT}}(x) \subseteq \mathsf{I}_{\mathsf{CEXT}}(y)$.

  If x and y are connected via the `rdfs:subClassOf` property,
  then both x and y are classes and
  the (class) extension of x is a subset of the (class) extension of y.

- $\mathsf{I}_{\mathsf{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$ is reflexive and transitive on IC.

  The `rdfs:subClassOf` property connects every class to itself.

  Moreover, whenever this property connects
  a class x with a class y and a class y with a class z,
  then it also directly connects x with z.

RDFS Interpretations/7

- If $x \in \mathsf{I}_{\mathsf{CEXT}}(\texttt{rdfs:ContainerMembershipProperty}^{\mathcal{I}})$,
  then $\langle x, \texttt{rdfs:member}^{\mathcal{I}} \rangle \in \mathsf{I}_{\mathsf{EXT}}(\texttt{rdfs:subPropertyOf}^{\mathcal{I}})$.

  If x is a property of the type `rdfs:ContainerMembershipProperty`,
  then it is `rdfs:subPropertyOf`-connected
  with the property `rdfs:member`.

- If $x \in \mathsf{I}_{\mathsf{CEXT}}(\texttt{rdfs:Datatype}^{\mathcal{I}})$,
  then $\langle x, \texttt{rdfs:Literal}^{\mathcal{I}} \rangle \in \mathsf{I}_{\mathsf{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$.

  If some x is typed as element of the class `rdfs:Datatype`,
  then it must be a subclass of the class of all literal values
  (denoted by `rdfs:Literal`).

- . . . additionally we require satisfaction of the following axiomatic triples:

RDFS Interpretations/8

rdf:type	rdfs:domain
rdfs:domain	rdfs:domain
rdfs:range	rdfs:domain
rdfs:subPropertyOf	rdfs:domain
rdfs:subClassOf	rdfs:domain
rdf:subject	rdfs:domain
rdf:predicate	rdfs:domain
rdf:object	rdfs:domain
rdfs:member	rdfs:domain
rdf:first	rdfs:domain
rdf:rest	rdfs:domain
rdfs:seeAlso	rdfs:domain
rdfs:isDefinedBy	rdfs:domain
rdfs:comment	rdfs:domain
rdfs:label	rdfs:domain
rdf:value	rdfs:domain

RDFS Interpretations/8

```
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 .
rdfs:comment        rdfs:domain          rdfs:Resource .
rdfs:label          rdfs:domain          rdfs:Resource .
rdf:value           rdfs:domain          rdfs:Resource .
```

RDFS Interpretations/9

rdf:type	rdfs:range
rdfs:domain	rdfs:range
rdfs:range	rdfs:range
rdfs:subPropertyOf	rdfs:range
rdfs:subClassOf	rdfs:range
rdf:subject	rdfs:range
rdf:predicate	rdfs:range
rdf:object	rdfs:range
rdfs:member	rdfs:range
rdf:first	rdfs:range
rdf:rest	rdfs:range
rdfs:seeAlso	rdfs:range
rdfs:isDefinedBy	rdfs:range
rdfs:comment	rdfs:range
rdfs:label	rdfs:range
rdf:value	rdfs:range

RDFS Interpretations/9

```
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 .
rdf:subject          rdfs:range         rdfs:Resource .
rdf:predicate        rdfs:range         rdfs:Resource .
rdf:object           rdfs:range         rdfs:Resource .
rdfs:member          rdfs:range         rdfs:Resource .
rdf:first            rdfs:range         rdfs:Resource .
rdf:rest             rdfs:range         rdf:List .
rdfs:seeAlso         rdfs:range         rdfs:Resource .
rdfs:isDefinedBy     rdfs:range         rdfs:Resource .
rdfs:comment         rdfs:range         rdfs:Literal .
rdfs:label           rdfs:range         rdfs:Literal .
rdf:value            rdfs:range         rdfs:Resource .
```

RDFS Interpretations/10

```
rdfs:ContainerMembershipProperty
                    rdfs:subClassOf
rdf:Alt             rdfs:subClassOf
rdf:Bag             rdfs:subClassOf
rdf:Seq             rdfs:subClassOf

rdfs:isDefinedBy    rdfs:subPropertyOf

rdf:XMLLiteral      rdf:type
rdf:XMLLiteral      rdfs:subClassOf
rdfs:Datatype       rdfs:subClassOf

rdf:_1              rdf:type

rdf:_1              rdfs:domain
rdf:_1              rdfs:range
rdf:_2              rdf:type

...
```

RDFS Interpretations/10

```
rdfs:ContainerMembershipProperty
                    rdfs:subClassOf     rdf:Property .
rdf:Alt             rdfs:subClassOf     rdfs:Container .
rdf:Bag             rdfs:subClassOf     rdfs:Container .
rdf:Seq             rdfs:subClassOf     rdfs:Container .

rdfs:isDefinedBy    rdfs:subPropertyOf  rdfs:seeAlso .

rdf:XMLLiteral      rdf:type            rdfs:Datatype .
rdf:XMLLiteral      rdfs:subClassOf     rdfs:Literal .
rdfs:Datatype       rdfs:subClassOf     rdfs:Class .

rdf:_1              rdf:type
                       rdfs:ContainerMembershipProperty .
rdf:_1              rdfs:domain         rdfs:Resource .
rdf:_1              rdfs:range          rdfs:Resource .
rdf:_2              rdf:type
                       rdfs:ContainerMembershipProperty .
...
```

RDFS Entailment

Automatic inference is again realized via deduction rules:

$$\frac{}{\texttt{u a x .}}\ \text{rdfsax}$$

every axiomatic triple “u a x .” can always be derived

$$\frac{\texttt{u a \_:n .}}{\texttt{u a l .}}\ \text{gl}$$

the converse of Rule lg: applies if `_:n` has been assigned (via Rule lg) to the untyped literal `l`

$$\frac{\texttt{u a l .}}{\texttt{\_:n rdf:type rdfs:Literal}}\ \text{rdfs1}$$

applies if `_:n` has been assigned (via Rule lg) to the untyped literal `l`

$$\frac{\texttt{a rdfs:domain x .}\quad \texttt{u a y .}}{\texttt{u rdf:type x .}}\ \text{rdfs2}$$

implements the semantics of property domains

$$\frac{\texttt{a rdfs:range x .}\quad \texttt{u a v .}}{\texttt{v rdf:type x .}}\ \text{rdfs3}$$

implementis the semantics of property ranges

`a`, `b`	IRIs	`x`, `y`	IRI, blank node or literal		
`u`, `v`	IRI or blank node	`l`	literal	`_:n`	blank nodes

RDFS Entailment/2

$$\frac{\texttt{u a x .}}{\texttt{u rdf:type rdfs:Resource .}}\ \text{rdfs4a}$$

the subject of every triple is a resource

$$\frac{\texttt{u a v .}}{\texttt{v rdf:type rdfs:Resource .}}\ \text{rdfs4b}$$

objects that are not literals are resources as well

$$\frac{\texttt{u rdfs:subPropertyOf v.}\quad \texttt{v rdfs:subPropertyOf x .}}{\texttt{u rdfs:subPropertyOf x .}}\ \text{rdfs5}$$

transitivity

$$\frac{\texttt{u rdf:type rdf:Property .}}{\texttt{u rdfs:subPropertyOf u .}}\ \text{rdfs6}$$

reflexivity

$$\frac{\texttt{a rdfs:subPropertyOf b .}\quad \texttt{u a y .}}{\texttt{u b y .}}\ \text{rdfs7}$$

subproperty inferences for instances

$$\frac{\texttt{u rdf:type rdfs:Class .}}{\texttt{u rdf:subClassOf rdfs:Resource .}}\ \text{rdfs8}$$

classes contain only resources

RDFS Entailment/3

$$\frac{\texttt{u rdfs:subClassOf x . \ \ v rdf:type u .}}{\texttt{v rdf:type x .}}\ \text{rdfs9}$$ subclassen inferences for instances

$$\frac{\texttt{u rdf:type rdfs:Class .}}{\texttt{u rdfs:subClassOf u .}}\ \text{rdfs10}$$ reflexivity

$$\frac{\texttt{u rdfs:subClassOf v. \ \ v rdfs:subClassOf x .}}{\texttt{u rdfs:subClassOf x .}}\ \text{rdfs11}$$ transitivity

$$\frac{\texttt{u rdf:type rdfs:ContainerMembershipProperty .}}{\texttt{u rdfs:subPropertyOf rdfs:member .}}\ \text{rdfs12}$$

$$\frac{\texttt{u rdf:type rdfs:Datatype .}}{\texttt{u rdfs:subClassOf rdfs:Literal .}}\ \text{rdfs10}$$ every datatype is a subclass of `rdfs:Literal`

RDFS Entailment: XML Clash

There is one possibility for a data graph to be inconsistent:

```
ex:hasSmiley    rdfs:range     rdfs:Literal .
ex:evilRemark   ex:hasSmiley   ">:->"^^rdf:XMLLiteral .
```

a node of type `rdfs:Literal` gets assigned an ill-formed literal value
This is called an *XML clash*

RDFS Entailment

Theorem:

A graph G_2 is RDFS entailed by G_1, if there is a graph G_1' obtained by applying the rules lg, gl, rdfax, rdf1, rdf2, rdfs1 – rdfs13 and rdfsax to G_1, such that

- G_2 is simply entailed by G_1' or
- G_1' contains an XML clash.

What RDF(S) Cannot Do

- Certain seemingly sensible consequences are not RDFS-entailed, e.g.

```
ex:talksTo  rdfs:domain      ex:Homo .
ex:Homo     rdfs:subClassOf  ex:Primates .
```

  should imply

```
ex:talksTo  rdfs:domain  ex:Primates .
```

- possible solution: use a stronger, so-called "extensional" semantics (but this would be outside the standard)
- No possibility to express negation