1 Notification

This tex file is based on
`https://www.w3.org/TR/rdf11-mt/`
Strongly suggest reading the introduction and abstract model of RDF first.

Terminology.

- A **name** is an IRI or literal. (a non-logical symbol plus its arity)
- A **subgraph** of an RDF graph is a subset of the triples in the graph.
- An **empty** RDF graph is a empty set of triples.
- A **ground** triple is one without blank nodes. A ground RDF graph consists only of ground triples.

2 Syntax

Here the syntax means the concrete syntax of RDF rather than an abstract one which can be seen in
`https://www.w3.org/TR/rdf11-concepts/#dfn-blank-node-identifier`
There are so currently so many of them(JSON-LD, Turtle, RDFa, RDF/XML, etc.) and may be more in the future, so they are not the topic of this tex file.

Briefly speaking, the alphabet of RDF is the alphabet of unicode strings, and the sentences(parameter-free formulas) of RDF are ground triples.

3 Semantics

In this note, the term *semantics* means a model-theoretic semantics. So the semantics of RDF and RDFS are based on the concept of interpretation, and focus on the truth value of triples and RDF graphs(sets of triples).
Following W3C, and as the book *Foundations of Semantic Web Technologies* says[1],

> we start by the comparably easy definition of simple interpretations of graphs. After that, we provide additional criteria which qualify these interpretations as RDF-interpretations. Finally we give further constraints to be fulfilled by an RDF-interpretation in order to be acknowledged as an RDFS-interpretation. As a natural consequence of this approach, every RDFS-interpretation is a valid RDF-interpretation and every RDF-interpretation constitutes a simple interpretation.

3.1 Simple Interpretations

Definition. A **vocabulary** V is a set of names.

Definition. (Simple Interpretation)
A simple Interpretation $\mathcal{I}$ is a mathematical structure consisting of

- IR, a non-empty set of resources, which is the domain of $\mathcal{I}$
- a set IP called the set of properties of $\mathcal{I}$
- a function $\mathrm{I}_{\mathrm{EXT}} : IP \mapsto 2^{IR \times IR}$, where $\mathrm{I}_{\mathrm{EXT}}(p)$ is called the **extension** of $p \in IP$
- a function I_{S} from IRIs into $IR \cup IP$
- a partial function I_{L} from literals into IR

Denotation of Simple Interpretation.

- For a literal l, $l^{\mathcal{I}} = \mathrm{I}_{\mathrm{L}}(l)$
- For an IRI r, $r^{\mathcal{I}} = \mathrm{I}_{\mathrm{S}}(r)$
- For a ground triple $t = (s, p, o)$, $t^{\mathcal{I}} =$ true iff $p^{\mathcal{I}} \in IP$ and $(s^{\mathcal{I}}, o^{\mathcal{I}}) \in \mathrm{I}_{\mathrm{EXT}}(p)$. Otherwise $t^{\mathcal{I}} =$ false
- For a ground graph $G = \{t = (s, p, o)\}$, $G^{\mathcal{I}} =$ true iff $\forall t \in G$, $t^{\mathcal{I}} =$ true. Otherwise $G^{\mathcal{I}} =$ false.

The following definition stems from the W3C's RDF semantics 1.0:

Definition. (Simple interpretation of a vocabulary)
A simple interpretation $\mathcal{I}$ of a vocabulary V is a simple interpretation such that

- The domain of I_{S} is the set of IRIs from V
- The domain of I_{L} is the set of literals from V

Definition. A datatype map is a function from IRIs to datatypes.

The current semantics of RDF assumes that a recognized IRI identifies a unique datatype, so a datatype map can also be regarded as a set of datatype IRIs(a kind of vocabulary).

Definition. A literal is called **ill-typed** for a datatype D if its datatype IRI is in D but the lexical form is assigned no value by the lexical-to-value mapping for that datatype.

Definition. (Simple D-interpretation) For a datatype map D(a set of IRIs identifying datatypes), a (simple) D-interpretation or an interpretation recognizing D, is a simple interpretation satisfying

- if the datatype IRI `rdf:langString` is in D, then for every language-tagged string E with lexical form s and language tag t, $E^{\mathcal{I}} = \mathrm{I_L}(E) = (s, t')$, where t' is t converted to lower case using US-ASCII rules

- for every other datatype IRI $a \in D$, $a^{\mathcal{I}}$ is the datatype(a resource) identified by a.

- for every literal l not ill-typed for D with datatype IRI $a \in D$ and lexical form s, suppose the lexical-to-value map is $L2V(a^{\mathcal{I}})$, then $l^{\mathcal{I}} = \mathrm{I_L}(l) = L2V(a^{\mathcal{I}})(s)$

For a ground triple $t = (s, p, o)$, if o is a ill-typed literal for D, then o has no denotation and $t^{\mathcal{I}} = \text{false}$.

Definition. (Satisfiability and Validity)
For a datatype D, an RDF graph G is simply $D-$satisfiable iff for some D-interpretation $\mathcal{I}$, $G^{\mathcal{I}} = \text{true}$, and in this case we say $\mathcal{I}$ **satisfies** G. If $G^{\mathcal{I}} = \text{true}$ for all D-interpretation, then G is called $D-$valid.

Definition. (Simple Entailment)
For a datatype D, an RDF graph G_1 is said to $D-$entail another RDF graph G_2 iff for every D-interpretation $\mathcal{I}$, $G_1^{\mathcal{I}} = \text{true}$ implies that $G_2^{\mathcal{I}} = \text{true}$, denoted as $G_1 \vDash G_2$.

3.2 RDF Interpretations

Definition. (RDF vocabulary)
The RDF vocabulary, denoted as V_{RDF}, is a set containing the following IRIs:
`rdf:type rdf:subject rdf:predicate rdf:object rdf:first rdf:rest rdf:value rdf:nil rdf:List rdf:langString rdf:Property rdf:_1 rdf:_2 ...`

Definition. (RDF interpretation)
An RDF interpretation recognizing a datatype D which includes `rdf:langString` and `xsd:string` is a $D-$interpretation $\mathcal{I}$ of V_{RDF} satisfying extra semantic conditions:

- $p \in IP$ iff $(p, \texttt{rdf:Property}^{\mathcal{I}}) \in \mathrm{I_{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$ (so $IP \subset IR$)

- for every datatype IRI $r \in V$, x is in the value space of the datatype $r^{\mathcal{I}}$ iff $(x, r^{\mathcal{I}}) \in \mathrm{I_{EXT}}(\texttt{rdf:type}^{\mathcal{I}})$

- $\mathcal{I}$ satisfies all the triples called **axiomatic triples** in the following infinite set:
  ```
  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:nil rdf:type rdf:List .
rdf:_1 rdf:type rdf:Property .
rdf:_2 rdf:type rdf:Property .
...
```

Definition. (RDF Entailment)
For a datatype D which includes `rdf:langString` and `xsd:string`, an RDF graph S **RDF entails** another RDF graph E **recognizing** D iff every RDF interpretation recognizing D which satisfies S also satisfies E. When $D = \{\texttt{rdf:langString}, \texttt{xsd:string}\}$, we simply say S **RDF entails**E.

3.3 RDFS Interpretations

Definition. (RDFS Vocabulary) The RDFS vocabulary is a set containing the following IRIs:
`rdfs:domain rdfs:range rdfs:Resource rdfs:Literal rdfs:Datatype rdfs:Class rdfs:subClassOf rdfs:subPropertyOf rdfs:member rdfs:Container rdfs:label rdfs:ContainerMembershipProperty rdfs:comment rdfs:seeAlso rdfs:isDefinedBy`

Definition. (RDFS Interpretation)
An RDFS-interpretation of a datatype D which includes `xsd:string` and `rdf:langString` is an RDF interpretation recognizing D satisfying the following extra semantic conditions on V_{RDFS}(extended by the vocabulary V_{RDFS}):

- there is a function $\mathrm{I_{CEXT}} : IR \mapsto 2^{IR}$ s.t. $\mathrm{I_{CEXT}}(r) = \{x \in IR | (x, r) \in \mathrm{I_{EXT}}(\texttt{rdf:type}^{\mathcal{I}})\}, \forall r \in IR$. $\mathrm{I_{CEXT}}(r)$ is called the **class extension** of $r \in IR$. (Based on the first requirement of RDF interpretation, we know that $IP = \mathrm{I_{CEXT}}(\texttt{rdf:Property}^{\mathcal{I}})$)
- there is a set $IC = \mathrm{I_{CEXT}}(\texttt{rdfs:Class}^{\mathcal{I}})$, i.e. IC is the set of all classes.
- $IR = \mathrm{I_{CEXT}}(\texttt{rdfs:Resource}^{\mathcal{I}})$ i.e. every resource has a property type with a value `rdfs:Resource`.
- $LV \triangleq \mathrm{I_{CEXT}}(\texttt{rdfs:Literal}^{\mathcal{I}})$ is the set of all literal values
- $\mathrm{I_{CEXT}}(\texttt{rdf:langString}^{\mathcal{I}}) = \{E^{\mathcal{I}} : E \text{ a language-tagged string}\}$
- for every datatype IRI $a \in D$, $\mathrm{I_{CEXT}}(a^{\mathcal{I}})$ is the value space of $a^{\mathcal{I}}$ and $a^{\mathcal{I}} \in \mathrm{I_{CEXT}}(\texttt{rdfs:Datatype}^{\mathcal{I}})$
- if $(x, y) \in \mathrm{I_{EXT}}(\texttt{rdfs:domain}^{\mathcal{I}})$(the domain of property x is y) and $(u, v) \in \mathrm{I_{EXT}}(x)$, then $u \in \mathrm{I_{CEXT}}(y)$
- if $(x, y) \in \mathrm{I_{EXT}}(\texttt{rdfs:range}^{\mathcal{I}})$ and $(u, v) \in \mathrm{I_{EXT}}(x)$ then $v \in \mathrm{I_{CEXT}}(y)$
- $\mathrm{I_{EXT}}(\texttt{rdfs:subPropertyOf}^{\mathcal{I}})$ is transitive and reflexive on IP.
- if $(x, y) \in \mathrm{I_{EXT}}(\texttt{rdfs:subPropertyOf}^{\mathcal{I}})$, then $x, y \in IP$ and $\mathrm{I_{EXT}}(x) \subset \mathrm{I_{EXT}}(y)$
- if $x \in IC$ then $(x, \texttt{rdfs:Resource}^{\mathcal{I}}) \in \mathrm{I_{EXT}}(\texttt{subClassOf}^{\mathcal{I}})$.
- $\mathrm{I_{EXT}}(\texttt{subClassOf}^{\mathcal{I}})$ is transitive and reflexive on IC.
- if $(x, y) \in \mathrm{I_{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$, then $x, y \in IC$ and $\mathrm{I_{CEXT}}(x) \subset \mathrm{I_{CEXT}}(y)$
- if $x \in \mathrm{I_{CEXT}}(\texttt{rdfs:ContainerMemebershipProperty}^{\mathcal{I}})$, then $(x, \texttt{rdfs:member}^{\mathcal{I}}) \in \mathrm{I_{EXT}}\ \texttt{rdfs:subPropertyOf}^{\mathcal{I}}$
- if $x \in \mathrm{I_{CEXT}}(\texttt{rdfs:Datatype}^{\mathcal{I}})$ then $(x, \texttt{rdfs:Literal}^{\mathcal{I}}) \in \mathrm{I_{EXT}}(\texttt{rdfs:subClassOf}^{\mathcal{I}})$

and $\mathcal{I}$ satisfies all the following 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: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 .

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 .

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

rdfs:isDefinedBy rdfs:subPropertyOf rdfs:seeAlso .

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 .
rdf:_2 rdfs:domain rdfs:Resource .
rdf:_2 rdfs:range rdfs:Resource .
...
```

Definition. (RDFS entailment)
For a datatype D which includes `xsd:string` and `rdf:langString`, an RDF graph RDFS entails another RDF graph E recognizing D iff every RDFS interpretation recognizing D which satisfies S also satisfies E.

3.4 Semantics Extension

In formal languages, for a interpretation $\mathcal{I}$, a set of additional semantic assumptions made about constants and relation symbols is called a **semantic extension** of $\mathcal{I}$. For example, the set of extra semantic requirements for a simple interpretation to be a RDF-interpretation is a semantic extension.

How can we construct a simple interpretation for an RDF graph with a new vocabulary V? Generally, there are two steps:

- define a simple interpretation for the RDF graph
- add the IRIs in V into the set IP and IR correspondingly, redefine I_S(add mappings from IRIs in V to IR) and $\mathrm{I}_{\mathrm{EXT}}$(add mappings for new elements of IP and meet the semantic extension)

For example, to construct a RDFS-interpretation for an RDF graph, we do the following:

(1) define a simple interpretation for the RDF graph

(2) add the IRIs in RDF vocabularies V_{RDF} into the set IP and IR correspondingly, redefine I_S and $\mathrm{I}_{\mathrm{EXT}}$

(3) again add the IRIs V_{RDFS} into the set IP and IR correspondingly, redefine I_S and $\mathrm{I}_{\mathrm{EXT}}$

The semantics listed in previous sections is the **standard semantics** or **intensional semantics**, which is a minimal requirement for RDF(S)-compatible system. By a semantic extension of the intensional semantics, the semantics obtained is called a **extensional semantics**.

Besides, each semantic extension defines an **entailment regime**, which specifies:

- A subset of RDF graphs called well-formed for the regime.
- An entailment relation between subsets of well-formed graphs and well-formed graphs.

4 Natural Deduction

It seems that W3C doesn't provide an official specification for this section. All of the inference rules for simple, RDF, RDFS and datatype entailments are well introduced in [1].

4.1 Inference Rules

Using the notation

- s, any IRI or blank node ID
- p, any IRI for predicate/property in a triple
- o, any IRI, blank node ID or literal
- $\_ : n$, the ID of any blank node
- l, any literal
- "f"^^d, a typed literal with datatype d and lexical form "f".

the following is a whole list for them:

(1) simple entailment

$$\frac{s \quad p \quad o \quad .}{s \quad p \quad \_:n \quad .}\ \text{se1}$$

$$\frac{s \quad p \quad o \quad .}{\_:n \quad p \quad o \quad .}\ \text{se2}$$

(2) RDF entailment

$$\frac{}{s \quad p \quad o \quad .}\ \text{rdfax}$$

$$\frac{s \quad p \quad l \quad .}{s \quad p \quad \_:n \quad .}\ \text{lg}$$

$$\frac{s \quad p \quad o \quad .}{p \quad \texttt{rdf:type} \quad \texttt{rdf:Property}}\ \text{rdf1}$$

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

(3) RDFS entailment

$$\frac{}{s \quad p \quad o \quad .}\ \text{rdfsax}$$

$$\frac{s \quad p \quad l \quad .}{\_:n \quad \texttt{rdf:type} \quad \texttt{rdfs:Literal} \quad .}\ \text{rdfs1}$$

$$\frac{p \quad \texttt{rdfs:domain} \quad o' \quad . \qquad s \quad p \quad o \quad .}{s \quad \texttt{rdf:type} \quad o' \quad .}\ \text{rdfs2}$$

$$\frac{p \quad \texttt{rdfs:range} \quad o \quad . \qquad s \quad p \quad s' \quad .}{s' \quad \texttt{rdf:type} \quad o \quad .}\ \text{rdfs3}$$

$$\frac{s \quad p \quad o \quad .}{s \quad \texttt{rdf:type} \quad \texttt{rdfs:Resource} \quad .}\ \text{rdfs4a}$$

$$\frac{s \quad p \quad s' \quad .}{s' \quad \texttt{rdf:type} \quad \texttt{rdfs:Resource} \quad .}\ \text{rdfs4b}$$

$$\frac{s \quad \texttt{rdfs:subPropertyOf} \quad s' \quad . \qquad s' \quad \texttt{rdfs:subPropertyOf} \quad o \quad .}{s \quad \texttt{rdfs:subPropertyOf} \quad o \quad .}\ \text{rdfs5}$$

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

$$\frac{p_1 \quad \texttt{rdfs:subPropertyOf} \quad p_2 \quad . \qquad s \quad p_1 \quad o \quad .}{s \quad p_2 \quad o \quad .}\ \text{rdfs7}$$

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

$$\frac{s \quad \texttt{rdf:type} \quad s' \quad . \qquad s' \quad \texttt{rdfs:subClassOf} \quad o \quad .}{s \quad \texttt{rdf:type} \quad o \quad .}\ \text{rdfs9}$$

$$\frac{s \quad \texttt{rdf:type} \quad \texttt{rdfs:Class} \quad .}{s \quad \texttt{rdfs:subClassOf} \quad s \quad .}\ \text{rdfs10}$$

$$\frac{s\ \ \texttt{rdfs:subClassOf}\ \ s'\ \ .\qquad s'\ \ \texttt{rdfs:subClassOf}\ \ o\ \ .}{s\ \ \texttt{rdfs:subClassOf}\ \ o\ \ .}\ \text{rdfs11}$$

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

$$\frac{s\ \ \texttt{rdf:type}\ \ \texttt{rdfs:Datatype}\ \ .}{s\ \ \texttt{rdfs:subClassOf}\ \ \texttt{rdfs:Literal}\ \ .}\ \text{rdfs13}$$

$$\frac{s\ \ p\ \ \_:n\ \ .}{s\ \ p\ \ l\ \ .}\ \text{gl}$$

here(in gl) $\_:n$ identifies a blank node introduced by an earlier weakening of the literal l via the rule lg.

(4) additional rules for datatypes

$$\frac{d\ \ \texttt{rdf:type}\ \ \texttt{rdfs:Datatype}\ \ .\qquad s\ \ p\ \ \text{“}f\text{”}\texttt{\^{}\^{}}d\ \ .}{\_:n\ \ \texttt{rdf:type}\ \ d\ \ .}\ \text{rdfD1}$$

if the value of lexical form f_1 in datatype d_1 is the same as that of lexical form f_2 in datatype d_2,

$$\frac{\begin{array}{c} d_1\ \ \texttt{rdf:type}\ \ \texttt{rdfs:Datatype}\ \ . \\ d_2\ \ \texttt{rdf:type}\ \ \texttt{rdfs:Datatype}\ \ . \\ s\ \ p\ \ \text{“}f_1\text{”}\texttt{\^{}\^{}}d_1\ \ . \end{array}}{s\ \ p\ \ \text{“}f_2\text{”}\texttt{\^{}\^{}}d_2\ \ .}\ \text{rdf2}$$

if the value space of the datatype d_1 is contained in the value space of another datatype d_2,

$$\frac{}{d_1\ \ \texttt{rdfs:subClassOf}\ \ d_2\ \ .}\ \text{rdfDAx}$$

4.2 Comments

some inference rules deserve mentioning:

(1) for all the inference rules which turn an RDF term into a blank node in the conclusion(like se1, se2, lg), two different RDF terms can not be turned into the same blank node, and the identifier of new blank node must not be used in the input RDF graph.

(2) each axiomatic triple correspond to an axiom(an inference rule without premises) in the deductive system. The collection of axioms in RDF entailment is called rdfax, and that in RDFS entailment is called rdfsax.

(3) Simple entailment and RDF entailment are sound and complete:

> **Theorem.** (Soundness and Completeness of simple entailment rules)
> A graph G_1 simply entails a graph G_2, iff G_1 can be extended to a graph G_1' such that $G_2 \subseteq G_1'$ by applying the inference rules se1 and se2.
>
> here "extend" means $G_1 \subseteq G_1'$ and $G_1 \vdash (G_1' - G_1)$
>
> **Theorem.** (Soundness and Completeness of RDF entailment rules)
> A graph G_1 RDF entails a graph G_2 iff there is a graph G_1' that can be derived from G_1 by applying the inference rules lg, rdf1, rdf2, as well as rdfax such that G_1' simply entails G_2.
>
> here "derive" means $G_1 \vdash_{RDF} G_1'$

(4) RDFS entailment is sound but not complete. An RDF graph is said to have an **XML clash** if there is an ill-typed literal in the graph, and we have the following theorem:

> **Theorem.** (Soundness of RDFS entailment)
> 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

(5) External datatypes may be introduced by new vocabularies; however, it is impossible to completely characterize their semantic behavior only by RDFS-internal means(additional deduction rules may need to add). Still, it is possible to state how frequently occurring interdependencies related to datatypes should be expressed by deduction rules: they are rdfD1, rdfD2, and rdfDAx. Note that the preconditions for these rules are not just syntactical, but also semantical(like rdfD2 and rdfDAx).

References

[1] Hitzler, Pascal, Markus Krotzsch, and Sebastian Rudolph. *Foundations of semantic web technologies.* Chapman and Hall/CRC, 2009.

[2] Werner, Nutt."Semantic Technologies Part 9:RDF(S) Semantics". Semantic Technologies, Free University of Bozen-Bolzano. Fall/Winter, 2014/2015. Course handout.