RDF Data Model and Query Languages

Sergio Tessaris

April 21, 2006
FOAF example

#me

- foaf:title: Dr
- rdf:type: foaf:Person
- foaf:workplaceHomepage: http://www.unibz.it/inf
- foaf:homepage: http://www.inf.unibz.it/~tessaris
- foaf:event: 758128cdae69fd0fd9e880921d9f4b25259edbf5...
- foaf:name: Sergio Tessaris
- foaf:givenname: Sergio
- foaf:family_name: Tessaris
Introduction
RDF Semantics
Querying RDF

RDF Site Summary (RSS) 1.0

Slashdot

1. Google's New Calendar CLS

2. Voltage Files Regulatory Complaint Over QFD


4. Novell Returns to the SUSE Name

5. Movies Losing Popularity

6. Google Enters Web-Office

7. The Problems With Games Copy Protection

Google's official blog tells us that Google has acquired Wiretap, a collaboration workflow product. From the article: “To be clear: Wiretap is still in beta, and it's far from perfect. Deploying our first user-experience wasn't everything to us, so we're not accepting new registrations until we've smoothed Wiretap to Google's software architecture. If you're interested in giving it a try, we hope you'll post on the waitlist as we can let you know when you'll

1. Google's New Calendar CLS

Google is apparently working on its own calendar (CLG) program to integrate with Gmail. The official blog is engaging with about 250 participants—people involved—are not allowed to share any details with outsiders. There are some leaked photos of the CLG.

Fri, Mar 10, 2006 7:11

2. Security Flow Discovered in GP

WebShark writes: "A serious problem in the use of QFD is verify digital signatures have been discovered, which also affects the use of a QFD. It is possible for an attacker to take any signed message and associate arbitrary data without affecting the signed state of the message. Depending on how the QFD is used, it may be possible to output valid data as several versions of QFD have been discovered. All versions of QFD prior to 1.4.23 are affected, and it is highly recommended to update Google QFD as soon as possible to version 1.4.22."

Fri, Mar 10, 2006 3:32

3. Voltage Files Regulatory Complaint Over QFD

GnuFile writes: "A recent CBO report says that Voltage Catala has filed a complaint with the Federal Regulatory over a new $15.00 per month Quality of Service Parameter that they charged their clients. That parameter, charging customers of X(V), failed to report a recent regulatory review by the Commission. Voltage enacted the QF parameter and highlighted the poor quality of the Voltage service. That we no longer have any problems is the real point about the slow service in order to formulate an appropriate regulatory response."

Fri, Mar 10, 2006 4:40

4. Novell Returns to the SUSE Name

Current writes: "It appears that Novell has decided to resume their enterprise desktop line SUSE, once again. According to an announcement at GNOX, Novell will be releasing the next version of their desktop product under the name LINUX Enterprise Desktop. The moniker Novell Linux Desktop. Naming aside, it looks like the feature will be there to make up a strong desktop competitor."

Fri, Mar 10, 2006 3:40

5. Movies Losing Popularity

AmyF writes: "Without the slightest mention of piracy, the OFPA and box-office revenues declined by 8 percent last year. About 45 percent of the decline came from the U.S. Now if only they'd realize that the decline is from movies making more than any other. It's been a while since a film was marketed spending the money to watch it in a room full of strangers."

Fri, Mar 10, 2006 0:41

6. Google Enters Web-Office

AmyF writes: "Google's official blog tells us that Google has acquired Wiretap, a collaboration workflow product. From the article: “To be clear: Wiretap is still in beta, and it's far from perfect. Deploying our first user-experience wasn't everything to us, so we're not accepting new registrations until we've smoothed Wiretap to Google's software architecture. If you're interested in giving it a try, we hope you'll post on the waitlist as we can let you know when you'll
Introduction
- Building Blocks
- RDF Abstract Syntax
- RDF Vocabulary

RDF Semantics
- RDF model theory
- Entailment
- Casting RDF into FOL

Querying RDF
- Introduction
- Graph Patterns
- Query languages
- SPARQL
**RDF**: language for representing information about resources (e.g. Metadata)
Basic Concepts

- **RDF**: language for representing information about resources (e.g., Metadata)
- Information about things *identified* on the Web
  - Identifiable doesn’t mean *retrievable*
  - E.g., goods from an eShop, prices, availability, etc.
Basic Concepts

- **RDF**: language for representing information about resources (e.g. Metadata)
- Information about things *identified* on the Web
  - Identifiable doesn’t mean *retrievable*
  - E.g. goods from an eShop, prices, availability, etc.
- Information processed by applications, not human beings
  - Semantic Web
The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.

Design Goals

- having a simple data model
- having formal semantics and provable inference
- using an extensible URI-based vocabulary
- using an XML-based syntax
- supporting use of XML schema datatypes
- allowing anyone to make statements about any resource
RDF is about making statements about resources

- E.g. Sergio Tessaris is the author of the web page http://www.inf.unibz.it/~tessaris/index.html
- This can be stated as a property of the web page http://www.inf.unibz.it/~tessaris/index.html has an author whose value is "Sergio Tessaris"

**RDF statements**

- **subject**: e.g. URL http://www.inf.unibz.it/~tessaris/index.html
- **predicate**: e.g. property author
- **object**: e.g. string "Sergio Tessaris"
Identifying Resources

- RDF identifiers: **Uniform Resource Identifiers (URI)**
- URIs, URLs, and URNs
  - **URL** identifies resources via a representation of their primary access mechanism
  - **URN** URIs that are required to remain globally unique and persistent

**Example**

ftp://ftp.is.co.za/rfc/rfc1808.txt
http://www.math.uio.no/faq/compression-faq/part1.html
news:comp.infosystems.www.servers.unix
telnet://melvyl.ucop.edu/
mailto:someone@example.com
RDF allows the use of values
- strings
- numbers
- booleans

**Literals** are basically UNICODE strings
- **plain** just strings (with optional language tag)
- **typed** have associated datatype URI

RDF literals and typing
- literals are **not** URIs
  - e.g. "http://www.unicode.org" and http://www.unicode.org are different
Triples and Graphs

- **RDF triples** basic element of RDF model
  - subject
  - predicate
  - object
Triples and Graphs

- **RDF triples** basic element of RDF model
  - subject
  - predicate
  - object
- triple: two nodes (subject, object) connected by a labelled edge (predicate)
**Triples and Graphs**

- **RDF triples** basic element of RDF model
  - subject
  - predicate
  - object
- triple: two nodes (subject, object) connected by a labelled edge (predicate)
- set of triples: a labelled directed graph
Blank Nodes

- RDF graphs may contain additional nodes
- arbitrary set of blank nodes (bnodes)
  - infinite
  - disjoint from URIs and literals
- given two blank nodes it is possible to determine whether or not they are the same
Blank Nodes

- RDF graphs may contain additional nodes
- arbitrary set of blank nodes (bnodes)
  - infinite
  - disjoint from URIs and literals
- given two blank nodes it is possible to determine whether or not they are the same
- intuition: a bnode represents the existence of something
URIs and Literals have a global scope
- two equal URIs (Literals) always represent the same object
  - equal means that the two unicode strings are the same
  - there are not contextual to the graph
- bnodes are contextual to the graph in which they appear
Scope of RDF Terms

- URIs and Literals have a global scope
  - two equal URIs (Literals) always represent the same object
    - equal means that the two unicode strings are the same
    - there are not contextual to the graph
- bnodes are contextual to the graph in which they appear
- ...more to came on the role of bnodes
RDF Graphs

- **RDF triple**
  - subject: RDF URI reference or a bnode
  - predicate: RDF URI reference
  - object: literal, RDF URI reference or a bnode
RDF Graphs

- **RDF triple**
  - **subject**: RDF URI reference or a bnode
  - **predicate**: RDF URI reference
  - **object**: literal, RDF URI reference or a bnode

- **RDF graph**: set of RDF triples
  - **nodes** of an RDF graph are subjects and objects in the triples
no literals as subject
predicates are just URIs
URIs are used for both resources (nodes) and predicates (edges)
literals can be non well formed datatypes
no complete information about any resource
On Bnodes

- bnodes are different from other RDF terms
- starting from the syntax
On Bnodes

- Bnodes are different from other RDF terms
- Starting from the syntax
- Graph equivalence

**Definition**

$G, G'$ are equivalent if there is a bijection $M$ between the nodes of the two graphs, s.t.:

1. $M$ maps bnodes to bnodes.
2. $M(lit) = lit$ for literals $lit$ in $G$.
3. $M(uri) = uri$ for URI in $G$.
4. $\langle s, p, o \rangle$ in $G$ iff the triple $\langle M(s), p, M(o) \rangle$ in $G'$.
On Bnodes

- Bnodes are different from other RDF terms
- Starting from the syntax
- Graph equivalence

**Definition**

$G, G'$ are equivalent if there is a bijection $M$ between the nodes of the two graphs, s.t.:

1. $M$ maps bnodes to bnodes.
2. $M(lit) = lit$ for literals $lit$ in $G$.
3. $M(uri) = uri$ for URI in $G$.
4. $⟨s, p, o⟩$ in $G$ iff the triple $⟨M(s), p, M(o)⟩$ in $G'$.

- I.e. a sort of graph isomorphism
Syntax vs. Semantics

- ‘still talking about (abstract) syntax
- nothing has been said on the actual semantics of RDF
  - introduced with an RDF vocabulary
  - given by means of a Model Theory
- how do you write/exchange RDF?
- in particular bnodes and literals
- several possibilities
  - N-Triples
  - RDF/XML (normative)
  - Turtle notation (subset of N3)
documents contain a set of assertions

subject predicate object .

URI references written out completely:

<http://example.org/resource30>

Literals as strings:

plain "chat"@fr

typed "<a></a>"^^<http://www.w3.org/2000/01/rdf-schema#XMLLiteral>
N-Triples Example

```
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://xmlns.com/foaf/0.1/title> "Dr" .
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://xmlns.com/foaf/0.1/mbox_sha1sum>
    "758128cdae69fd0fd9e880921d9f4b25259edbf5" .
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://xmlns.com/foaf/0.1/name> "Sergio Tessaris" .
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
  <http://xmlns.com/foaf/0.1/family_name> "Tessaris" .
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
<http://www.inf.unibz.it/~tessaris/myfoaf.xml#me>
```
RDF/XML

- Normative XML serialisation for RDF documents
- Use namespace abbreviations: e.g.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:myfoaf="http://www.inf.unibz.it/~tessaris/myfoaf.xml">
```

- Encode paths of the RDF graph
- There are several ways of encoding the same graph!
RDF/XML

- document root is `rdf:RDF`
- `rdf:Description` to represent nodes
- predicates use the corresponding URI

```xml
<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
    <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
</rdf:Description>
```

- `bnodes` can be omitted
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/">
    <foaf:Person rdf:ID="me">
        <foaf:name>Sergio Tessaris</foaf:name>
        <foaf:title>Dr</foaf:title>
        <foaf:givenname>Sergio</foaf:givenname>
        <foaf:family_name>Tessaris</foaf:family_name>
        <foaf:mbox_sha1sum>758128cdae69fd0fd9e880921d9f4b25259edbf5</foaf:mbox_sha1sum>
        <foaf:homepage rdf:resource="http://www.inf.unibz.it/~tessaris"/>
        <foaf:workplaceHomepage rdf:resource="http://www.unibz.it/inf"/>
    </foaf:Person>
</rdf:RDF>
Turtle Notation

- Extension of N-Triples
- Compact representation of graphs
Turtle Notation

- Extension of N-Triples
- Compact representation of graphs
- I will use this notation, explaining it on the way
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix : <http://www.inf.unibz.it/~tessaris/myfoaf.xm#> .

:me rdf:type foaf:Person ;
  foaf:family_name "Tessaris" ;
  foaf:givenname "Sergio" ;
  foaf:homepage
    <http://www.inf.unibz.it/~tessaris> ;
  foaf:mbox_sha1sum
    "758128cdae69fd0fd9e880921d9f4b25259edbf5" ;
  foaf:name "Sergio Tessaris" ;
  foaf:phone
    <tel:+39-0471-016-125> ;
  foaf:title "Dr" ;
  foaf:workplaceHomepage
    <http://www.unibz.it/inf> .
up till now we have
  - abstract syntax for graphs
  - a way to exchange these graphs
where’s the semantics?
how to express “rich” semantic constructs?
up till now we have
  - abstract syntax for graphs
  - a way to exchange these graphs
where’s the semantics?
how to express “rich” semantic constructs?
everything is going to be in RDF itself
up till now we have
- abstract syntax for graphs
- a way to exchange these graphs

where’s the semantics?

how to express “rich” semantic constructs?

everything is going to be in RDF itself

URIs are *global*: RDF prescribes the *meaning* of same URIs
RDF, RDF Schema, and more

- RDF: basic vocabulary, e.g.
  - class membership

- RDFS: extends the basic vocabulary, e.g.
  - subclass relation
  - domain and range for predicates

- OWL family: “extends” to a full fledged ontology language
• RDF: basic vocabulary, e.g.
  • class membership
• RDFS: extends the basic vocabulary, e.g.
  • subclass relation
  • domain and range for predicates
• OWL family: “extends” to a full fledged ontology language
  • we wont discuss OWL
RDF Vocabulary

- **Classes**
  - `rdf:Property`
  - `rdf:XMLLiteral`

- **Properties**
  - `rdf:type`

- **Reification**
  - `rdf:Statement`
  - `rdf:subject`, `rdf:predicate`, `rdf:object`

- **Collections and Containers**
  - `rdf:Bag`, `rdf:Seq`, `rdf:Alt`
  - `rdf:List`, `rdf:first`, `rdf:rest`, `rdf:nil`
  - `rdf:_1`, `rdf:_2`, `rdf:_3`,... etc.
RDFS Vocabulary

- **Classes**
  - rdfs:Resource
  - rdfs:Class
  - rdfs:Literal
  - rdfs:Datatype

- **Properties**
  - rdfs:range, rdfs:domain
  - rdfs:subClassOf
  - rdfs:subPropertyOf
  - rdfs:label, rdfs:comment
the *meaning* of an RDF document is context dependent
- mostly not machine accessible

we are interested in the *semantics* of RDF(S) as a *formal language*
- implications of the asserted statements
- *entailment* as the basic tool (query answering)
- inference rules to decide entailment between two graphs

semantics provided by means of a *model theory* (normative)
- over the abstract syntax already described

monotonic
- no closed-world assumptions
- no defaults
Consider a set of terms $\mathcal{T}$ (URIs and Literals in a graph)

**Definition (Simple Interpretation)**

A simple interpretation $\mathcal{I}$ over $\mathcal{T}$ is composed by:

- A non-empty set $\Delta$ of resources (domain of $\mathcal{I}$).
- A set $\mathcal{P}$ of properties of $\mathcal{I}$.
  - $\mathcal{PL} \subseteq \Delta$, which contains all the plain literals in $\mathcal{T}$.
- A mapping $\mathcal{V}$ from URI references in $\mathcal{T}$ into $\Delta \cup \mathcal{P}$.
- A mapping $\mathcal{L}$ from typed literals in $\mathcal{T}$ into $\Delta$.
- A mapping $\mathcal{E}$ from $\mathcal{P}$ into the powerset of $\Delta \times \Delta$. 
RDF Interpretations

- “double level” interpretation
- the key is in $\mathcal{E}$
- literals are in the domain
- no bnodes in $\mathcal{T}$
Triples Satisfiability

- interpretation $\mathcal{I}(t)$ of a term $t \in \mathcal{T}$
  - $\mathcal{I}(t) = t$ if $t$ is a plain literal
  - $\mathcal{I}(t) = \mathcal{L}(t)$ if $t$ is a typed literal
  - $\mathcal{I}(t) = \mathcal{V}(t)$ if $t$ is a URI reference
interpretation $\mathcal{I}(t)$ of a term $t \in \mathcal{T}$

- $\mathcal{I}(t) = t$ if $t$ is a plain literal
- $\mathcal{I}(t) = \mathcal{L}(t)$ if $t$ is a typed literal
- $\mathcal{I}(t) = \mathcal{V}(t)$ if $t$ is a URI reference

a triple $s$ $p$ $o$ is satisfied by $\mathcal{I}$ iff

- $\langle \mathcal{I}(s), \mathcal{I}(o) \rangle \in \mathcal{E}(\mathcal{I}(p))$
Triples Satisfiability

- interpretation $\mathcal{I}(t)$ of a term $t \in \mathcal{T}$
  - $\mathcal{I}(t) = t$ if $t$ is a plain literal
  - $\mathcal{I}(t) = \mathcal{L}(t)$ if $t$ is a typed literal
  - $\mathcal{I}(t) = \mathcal{V}(t)$ if $t$ is a URI reference
- a triple $s p o$. is satisfied by $\mathcal{I}$ iff
  - $\langle \mathcal{I}(s), \mathcal{I}(o) \rangle \in \mathcal{E}(\mathcal{I}(p))$
- Bnodes
  - $\mathcal{B}(G)$ is the set of bnodes in $G$
  - $\mathcal{A}$ a mapping from $\mathcal{B}(G)$ to $\Delta$
  - $\mathcal{I}_\mathcal{A}$ is the extension of $\mathcal{I}$ with $\mathcal{A}$ (i.e. $\mathcal{I}_\mathcal{A}(b) = \mathcal{A}(b)$)
Models of RDF Graphs

- **Ground graphs** (without bnodes)

**Definition (Ground Satisfiability)**

$I$ is a model of a ground RDF graph $G$ iff it satisfies all the triples in $G$.

Sergio Tessaris

RDF Data Model and Query Languages
Models of RDF Graphs

- **Ground graphs** (without bnodes)

**Definition (Ground Satisfiability)**

$I$ is a model of a ground RDF graph $G$ iff it satisfies all the triples in $G$.

- **Graphs with bnodes**

**Definition (Satisfiability)**

$I$ is a model of an RDF graph $G$ iff there is a mapping $\mathcal{A}$ from $B(G)$ to $\Delta$ s.t. $I_{\mathcal{A}}$ satisfies all the triples in $G$.
Simple Entailment

- Entailment is defined in terms of models
- $G$ entails $G'$ iff every model of $G$ is a model of $G'$
Entailment is defined in terms of models.

- $G$ entails $G'$ iff every model of $G$ is a model of $G'$.
- bnodes are “existential variables”
  - $\mathcal{I}$ model of $G$ if exists $\mathcal{A}$ s.t. $\mathcal{I}_A$ satisfies all the triples in $G$.
  - $\mathcal{I}$ model of $G'$ if exists $\mathcal{A}'$ s.t. $\mathcal{I}_{A'}$ satisfies all the triples in $G$. 

What about RDF and RDFS?

Sergio Tessaris

RDF Data Model and Query Languages
Simple Entailment

- Entailment is defined in terms of models
- $G$ entails $G'$ iff every model of $G$ is a model of $G'$
- bnodes are “existential variables”
  - $\mathcal{I}$ model of $G$ if exists $\mathcal{A}$ s.t. $\mathcal{I}_\mathcal{A}$ satisfies all the triples in $G$
  - $\mathcal{I}$ model of $G'$ if exists $\mathcal{A}'$ s.t. $\mathcal{I}_{\mathcal{A}'}$ satisfies all the triples in $G$
- What about RDF and RDFS?
RDF(S) Vocabulary

- additional semantic conditions over the vocabulary
  - e.g. `rdf:type`
    - restrict the set of models
    - derived from the intended meaning (RDF data model)
      - e.g. `rdf:type` is a `rdf:Property`

- Semantic conditions
  - e.g. \( x \in \mathcal{P} \) iff \( \langle x, \mathcal{V}(\text{rdf:Property}) \rangle \in \mathcal{E}(\mathcal{V}(\text{rdf:type})) \)

- Axiomatic triples
  - e.g. `rdf:type rdf:type rdf:Property .`

- RDF-MT doesn’t cover reification, containers and collections
RDFS Semantic Conditions

- RDFS introduces classes and their properties still based on `rdf:type`
- Macro $C(\cdot)$ for unary predicates

\[ x \in C(y) \text{ iff } \langle x, y \rangle \in \mathcal{E}(\mathcal{V}(\text{rdf:type})) \]
RDFS Semantic Conditions

- RDFS introduces classes and their properties still based on `rdf:type`
- macro $C(\cdot)$ for unary predicates

\[ x \in C(y) \iff \langle x, y \rangle \in E(\mathcal{V}(\text{rdf}:\text{type})) \]

- Semantic conditions
RDFS Semantic Conditions

- RDFS introduces classes and their properties still based on `rdf:type`
- Macro $C(\cdot)$ for unary predicates

$$x \in C(y) \text{ iff } \langle x, y \rangle \in E(V(rdf:type))$$

- Semantic conditions
  - Subclass: if $\langle x, y \rangle \in E(V(rdfs:subClassOf))$ then $C(x) \subseteq C(y)$
    - $E(V(rdfs:subClassOf))$ is transitive and reflexive
RDFS Semantic Conditions

- RDFS introduces classes and their properties still based on `rdf:type`
- Macro $C(\cdot)$ for unary predicates

\[ x \in C(y) \text{ iff } \langle x, y \rangle \in \mathcal{E}(\mathcal{V}(\text{rdf:type})) \]

- **Semantic conditions**
  - **Subclass**: if $\langle x, y \rangle \in \mathcal{E}(\mathcal{V}(\text{rdfs:subClassOf}))$ then $C(x) \subseteq C(y)$
    - $\mathcal{E}(\mathcal{V}(\text{rdfs:subClassOf}))$ is transitive and reflexive
  - **Domain**: if $\langle x, y \rangle \in \mathcal{E}(\mathcal{V}(\text{rdfs:range}))$ then $\langle u, v \rangle \in \mathcal{E}(x)$ implies $v \in C(y)$
RDFS introduces classes and their properties still based on `rdf:type`.

- macro $C(\cdot)$ for unary predicates

\[
x \in C(y) \text{ iff } \langle x, y \rangle \in \mathcal{E}(\mathcal{V}(\text{rdf:type}))
\]

**Semantic conditions**

- **Subclass:** if $\langle x, y \rangle \in \mathcal{E}(\mathcal{V}(\text{rdfs:subClassOf}))$ then $C(x) \subseteq C(y)$
  - $\mathcal{E}(\mathcal{V}(\text{rdfs:subClassOf}))$ is transitive and reflexive

- **Domain:** if $\langle x, y \rangle \in \mathcal{E}(\mathcal{V}(\text{rdfs:range}))$ then $\langle u, v \rangle \in \mathcal{E}(x)$ implies $v \in C(y)$

- something’s wrong here?
Extensional Semantic Conditions

- One direction (only if) is missing!
- Semantic conditions revised
  - Subclass: \( \langle x, y \rangle \in \mathcal{E}(\forall(\text{rdfs:subClassOf})) \) iff \( \mathcal{C}(x) \subseteq \mathcal{C}(y) \)
  - Domain: \( \langle x, y \rangle \in \mathcal{E}(\forall(\text{rdfs:range})) \) iff \( \langle u, v \rangle \in \mathcal{E}(x) \) implies \( v \in \mathcal{C}(y) \)
Extensional Semantic Conditions

- One direction (only if) is missing!
- **Semantic conditions revised**
  - Subclass: \( \langle x, y \rangle \in \mathcal{E}(\forall(\text{rdfs:subClassOf})) \iff \mathcal{C}(x) \subseteq \mathcal{C}(y) \)
  - Domain: \( \langle x, y \rangle \in \mathcal{E}(\forall(\text{rdfs:range})) \iff \langle u, v \rangle \in \mathcal{E}(x) \text{ implies } v \in \mathcal{C}(y) \)
  
  \[ \text{:p1 rdfs:range :c1 .} \]
  \[ \text{:p2 rdfs:range :c2 .} \]
  \[ \text{:p1 rdfs:subPropertyOf :p2 .} \]
One direction (only if) is missing!

Semantic conditions revised

- **Subclass**: \( \langle x, y \rangle \in E(\mathcal{V}(\text{rdfs:subClassOf})) \iff C(x) \subseteq C(y) \)
- **Domain**: \( \langle x, y \rangle \in E(\mathcal{V}(\text{rdfs:range})) \iff \langle u, v \rangle \in E(x) \) implies \( v \in C(y) \)

\[
:p1 \text{ rdfs:range :c1 .} \\
:p2 \text{ rdfs:range :c2 .} \\
:p1 \text{ rdfs:subPropertyOf :p2 .} \\
\text{implies? :c1 rdfs:subClassOf :c2 .}
\]
One direction (only if) is missing!

Semantic conditions revised

- Subclass: \( \langle x, y \rangle \in E(V(\text{rdfs:subClassOf})) \iff C(x) \subseteq C(y) \)
- Domain: \( \langle x, y \rangle \in E(V(\text{rdfs:range})) \iff \langle u, v \rangle \in E(x) \implies v \in C(y) \)

\[ :p1 \text{ rdfs:range } :c1 . \]
\[ :p2 \text{ rdfs:range } :c2 . \]
\[ :p1 \text{ rdfs:subPropertyOf } :p2 . \]
\[ \text{implies? } :c1 \text{ rdfs:subClassOf } :c2 . \]

only with extensional semantic conditions (not normative)
RDFS Semantic Conditions

\[ x \in C(y) \text{ iff } \langle x, y \rangle \in E(\mathcal{V}(\text{rdf:type})) \]

\[ \langle x, y \rangle \in E(\mathcal{V}(\text{rdfs:subClassOf})) \text{ iff } C(x) \subseteq C(y) \]

\[ \langle x, y \rangle \in E(\mathcal{V}(\text{rdfs:subPropertyOf})) \text{ iff } E(x) \subseteq E(y) \]

\[ \langle x, y \rangle \in E(\mathcal{V}(\text{rdfs:range})) \text{ iff } \langle u, v \rangle \in E(x) \rightarrow v \in C(y) \]

\[ \langle x, y \rangle \in E(\mathcal{V}(\text{rdfs:domain})) \text{ iff } \langle u, v \rangle \in E(x) \rightarrow u \in C(y) \]

if \( x \in C(\mathcal{V}(\text{rdfs:Class})) \) then \( C(x) \subseteq C(\mathcal{V}(\text{rdfs:Resource})) \)

if \( x \in C(\mathcal{V}(\text{rdfs:Datatype})) \) then \( C(x) \subseteq C(\mathcal{V}(\text{rdfs:Literal})) \)
RDF(S) Literals

- Plain literals (RDFS)

\[ PL = C(\forall(\text{rdfs:Literal})) \]
RDF(S) Literals

- Plain literals (RDFS)

\[ \mathcal{PL} = C(\forall(\text{rdfs:Literal})) \]

- affects logical implication

:myself foaf:name "Sergio Tessaris" .
entails
:myself foaf:name _:b1 .
_:b2 rdf:type rdfs:Literal .

Note: "Sergio Tessaris" rdf:type rdfs:Literal . is not a valid RDF(S) triple!
RDF(S) Literals

- Plain literals (RDFS)

\[ \mathcal{PL} = C(\forall(\text{rdfs:Literal})) \]

- affects logical implication

  \[
  \text{:myself foaf:name "Sergio Tessaris" .}
  \]

  entails

  \[
  \text{:myself foaf:name _:b1 .}
  \]

  \[
  _:b2 \text{ rdf:type rdfs:Literal .}
  \]

Note

"Sergio Tessaris" rdf:type rdfs:Literal .

is not a valid RDF(S) triple!
Entailment Rules

- a set of inference rules to capture RDF(S)-entailment
- rules complete an RDF graph
  * add a triple if there is some pattern*
  - rules application terminates
  - in a polynomial number of steps
- set of rules for RDF and RDFS (informative)
Lemma (Entailment Lemma)

$G\ rdf(s)$-entails $E$ iff exists $G'$ derived from $G$ with axiomatic triples using the entailment rules s.t. $G'$ simply entails $E$.

- simple entailment is enough
- effective procedure (needs simple entailment)
Grounding a graph $G$

Def *completed*: added axiomatic triples and applied entailment rules

Def *Herbrand* model: each bnode replaced by an URIs or Literal

Def *Canonical* model ($\hat{G}$): bnodes replaced with fresh URIs
Graph entailment

**Theorem**

*RDF graphs entailment:* $G$ entails $E$ iff some herbrand model of $E$ is a subgraph of the canonical model of $G$

- Connects our definition to W3C normative semantics
- Complexity of entailment
  1. NP-complete in the size of the RDF graphs
  2. PTIME in the size of the entailing graph $G$
  3. PTIME if $E$ is acyclic or ground
RDFS and FOL

- RDF has an high order flavour
  - URIs can play different roles
  - $\langle \text{ex:o, rdf:type, ex:o} \rangle$
- Introduce a different model theoretic semantics for RDF
  - “Compatible” with FOL
  - Extensibility to more expressive languages (e.g. OWL)
  - Use standard technologies (e.g. databases or theorem provers)
key idea: polymorphic interpretation of URIs (contextual PC)

- abstract domain $\Delta$
- $u^{Io} \in \Delta$: individual + function mapping valid literals to datavalues
- $u^{IC} \subseteq \Delta$: class
- $u^{IR} \subseteq \Delta \times \Delta$: binary relation

E.g.

$$\langle \text{ex:}o, \text{rdf:}type, \text{ex:}o \rangle$$

leads to

$$\text{ex:}o^{Io} \in \text{ex:}o^{IC}$$
Def non-high order graph: no blank nodes as objects of rdf:type

Def The classical logic translation $\text{FO}(G)$ of a non-high order graph $G$

- URIs and literals are constants
- blank nodes are existentially quantified variables
- binary atomic formulas in correspondence with triples in $G$
- $\langle u_1, \text{rdf:type}, u_2 \rangle$ triples introduce $u_2(u_1)$ atomic formulae

Theorem

Given an RDF graph $G$ and a non-high order graph $E$, $G$ entails $E$ iff $\text{FO}(\hat{G}) \models_c \text{FO}(E)$
Querying RDF documents

- How do we get informations out of an RDF document?
- RDF graphs can be stored in a variety of means
  - RDF/XML documents
  - databases
  - dedicated RDF service providers (e.g. RSS 1.0)
  - generated on the fly; e.g. from web pages (Gleaning Resource Descriptions from Dialects of Languages [GRDDL])
- Often retrieving the whole graph is not feasible/desirable
  - too big
  - lot of uninteresting parts
  - dynamic
- Solution: a (standard) protocol for querying RDF graphs
entailment and query answering are strictly related

an answer to a query is a set of entailed facts
  - tuples representing variable bindings
  - complex structures like RDF graphs or XML documents
  - it depends on the query language

RDF aims at big volumes of data
  - look out for efficiency (i.e. low data complexity)

two main factors
  - representational language (RDF)
  - query language
RDF(S) is a *simple* language
- entailment can be checked on a single canonical model
- see entailment rules and lemma

more freedom on the query language
- truly graph based query language (e.g. a la XQuery)
  \textit{WilburQL}
- simple language based on graph entailment
RDF(S) is a simple language
- entailment can be checked on a single canonical model
- see entailment rules and lemma
- more freedom on the query language
  - truly graph based query language (e.g. a la XQuery)
    WilburQL
  - simple language based on graph entailment
Graph Patterns

- **Dataset**: graph to be queried
- define a *new* kind of graphs: RDF graph patterns
  - RDF graphs with variables
    - ?x foaf:nick "Alice".
- patterns may contain bnodes
  - _:b foaf:nick "Alice".

Query Answering

Find all the assignments for the variables that make the pattern a logical consequence of the Dataset.
Subgraph Matching

- can we query in an efficient way?
Subgraph Matching

- can we query in an efficient way?
- entailment lemma: simple entailment is subgraph matching
Subgraph Matching

- can we query in an efficient way?
- entailment lemma: simple entailment is subgraph matching
- subgraph matching is conjunctive query answering
Subgraph Matching

- can we query in an efficient way?
- entailment lemma: simple entailment is subgraph matching
- subgraph matching is conjunctive query answering
- Dataset can be stored in a relational database
Subgraph Matching

- can we query in an efficient way?
- entailment lemma: simple entailment is subgraph matching
- subgraph matching is conjunctive query answering
- Dataset can be stored in a relational database
- the answer is yes!
can we query in an efficient way?
entailment lemma: simple entailment is subgraph matching
subgraph matching is conjunctive query answering
Dataset can be stored in a relational database
the answer is yes!
nothing more than conjunctive queries using a single ternary predicate (e.g. \texttt{triple(x, foaf:nick, "Alice"))}
e.g. Oracle supports RDF and an extension to SQL
most SPARQL implementations rely on a database back-end
ORACLE SQL Extension

SELECT t.r reviewer, e.emailid emailid
 FROM TABLE(RDF_MATCH(
   '(?r ReviewerOf ?c)
   (?r rdf:type Faculty)',
   RDFModels('reviewers'),
   NULL, NULL)) t, employees e
 WHERE t.r = e.name;
query language for RDF becoming a W3C recommendation
- based on Graph Patterns
  - patterns “extract” a set of variable bindings (tuples)
- results from different graph patterns can be combined using an algebra
  - union of answer sets
  - left outer join
  - filtering based on XQuery operators
  - optionally, an RDF graph can be returned (using a template)
SPARQL Example

BASE <http://example.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ex: <properties/1.0#>
SELECT DISTINCT ?person ?name ?age
FROM <http://rdf.example.org/people.rdf>
WHERE { ?person a foaf:Person ;
         foaf:name ?name. 
         OPTIONAL { ?person ex:age ?age } . 
         FILTER ! REGEX(?name, "Bob") 
}
LIMIT 3 ORDER BY ASC[?name]
Semantics for pattern matching: **entailment regime**
- kind of entailment: simple, RDF, RDFS, …

Answer set is a set of mappings from variables in $Q$ to terms occurring in $G$

Given an entailment $\models_E$, a query pattern $Q$, an RDF graph $G$, then $Q$ $E$-matches graph $G$ with answer $S$ if:
Semantics for pattern matching: entailment regime
- kind of entailment: simple, RDF, RDFS, ...

Answer set is a set of mappings from variables in $Q$ to terms occurring in $G$

Given an entailment $\models_E$, a query pattern $Q$, an RDF graph $G$, then $Q$ $E$-matches graph $G$ with answer $S$ if:
1. $Q'$ graph equivalent to $Q$ (isomorphic to $Q$)
SPARQL Semantics

- Semantics for pattern matching: **entailment regime**
  - kind of entailment: simple, RDF, RDFS, ...  
- Answer set is a set of mappings from variables in $Q$ to terms occurring in $G$
- Given an entailment $|=_{E}$, a query pattern $Q$, an RDF graph $G$, then $Q$ $E$-matches graph $G$ with answer $S$ if:
  1. $Q'$ graph equivalent to $Q$ (isomorphic to $Q$)
  2. $G \cup S(Q')$ well-formed graph
Semantics for pattern matching: entailment regime

kind of entailment: simple, RDF, RDFS, ...

Answer set is a set of mappings from variables in $Q$ to terms occurring in $G$

Given an entailment $\models_E$, a query pattern $Q$, an RDF graph $G$, then $Q$ $E$-matches graph $G$ with answer $S$ if:

1. $Q'$ graph equivalent to $Q$ (isomorphic to $Q$)
2. $G \cup S(Q')$ well-formed graph
3. $G \models_E (G \cup S(Q'))$
Semantics for pattern matching: entailment regime
- kind of entailment: simple, RDF, RDFS, ...

Answer set is a set of mappings from variables in $Q$ to terms occurring in $G$

Given an entailment $\models_E$, a query pattern $Q$, an RDF graph $G$, then $Q$ $E$-matches graph $G$ with answer $S$ if:
1. $Q'$ graph equivalent to $Q$ (isomorphic to $Q$)
2. $G \cup S(Q')$ well-formed graph
3. $G \models_E (G \cup S(Q'))$

Backward compatible with subgraph homomorphism
Where’s the problem? why not $G \models E S(Q')$
Where’s the problem? why not $G \models_E S(Q')$

- **bnodes in answer sets**

  **graph:**
  
  `_:myself foaf:homepage "http://www.inf.unibz.it/"

  **query:**
  
  `_:b foaf:homepage ?x`
BNodes in Query Answers

- Where’s the problem? why not $G \models_E S(Q')$
- bnodes in answer sets
  - graph:
    - _:myself foaf:homepage "http://www.inf.unibz.it/"
  - query:
    - _:b foaf:homepage ?x
  - answers:
    - [ ?x"http://www.inf.unibz.it/" ],
    - [ ?x/_:myself]
which bnodes in answers?

graph :myself foaf:name "Sergio Tessaris" .
query ?x rdf:type rdfs:Literal
answer {}
BNodes and Literals

- which bnodes in answers?
  ```
  graph :myself foaf:name "Sergio Tessaris" .
  query ?x rdf:type rdfs:Literal
  answer ∅
  ```

- querying precompleted graph
  ```
  graph :myself foaf:name "Sergio Tessaris" .
  _:b rdf:type rdfs:Literal
  answer {[ ?x/_:b ]}
  ```
BNodes and Literals

- which bnodes in answers?

  ```
  graph :myself foaf:name "Sergio Tessaris" .
  query ?x rdf:type rdfs:Literal
  answer ∅
  ```

- querying precompleted graph

  ```
  graph :myself foaf:name "Sergio Tessaris" .
  _:b rdf:type rdfs:Literal
  answer { [ ?x/_:b ]}
  ```

- in both cases _:w rdf:type rdfs:Literal is a logical consequence
Conclusions

- RDF has a precise (model theoretic) semantics
- Several drawbacks, but it’s a standard we have to deal with it!
- Strange semantics but relatively easy to deal with (up to RDFS)
- Extensions (e.g. OWL) require a more careful rethinking of the whole framework
  - Keeping backward compatibility
  - Interoperability via SPARQL
  - Higher order style semantics (cfg HiLog)