RDF and RDF Schema

Resource Description Framework
Outline

• RDF Design objectives
• RDF General structure
• RDF Vocabularies
• Serialization: XML
• Semantic features
• RDF Schema
• RDF Semantics and Reasoning
SW Technology Stack

- User interface and applications
- Trust
- Proof
- Unifying Logic
- Querying: SPARQL
- Ontologies: OWL
- Rules: RIF/SWRL
- Taxonomies: RDFS
- Data interchange: RDF
- Identifiers: URI
- Character Set: UNICODE
- Syntax: XML
- Cryptography
A common language for describing resources

• Resource Description Framework (RDF)
  – a language for representing information about resources
  – in the Semantic Web
  – in the World Wide Web
  – things that can be identified on the Web, even when they cannot be directly retrieved on the Web
RDF Design goals

- **Simple** data model
- Formal **semantics** and provable **inference**
- Extensible URI-based vocabulary
- Using an XML-based syntax (but not only...)
  - Supporting use of XML schema datatypes
- Allowing **anyone** to make statements about **any** resource
Simple yet powerful

- RDF has an **abstract syntax** that reflects a simple **graph-based** data model
- RDF has formal **semantics** with a rigorously defined notion of **entailment** providing a basis for well-founded deductions
Basic principles (1/2)

• Clearly separate
  – Model structure (RDF graph)
  – Interpretation Semantics (Entailment)
  – Concrete Syntaxes (XML, TN, N3, ...)

• Only two datatypes
  – URI/URIref: everything is a URI
  – Literal
    • String or other XSD datatype
Basic principles (2/2)

• Integrated with the Web
  – Uses XMLSchema datatypes
  – May reference http-retrievable resources

• Open world assumption
  – Allows anyone to make statements about any resource
  – No guaranteed completeness
  – No guaranteed consistency
RDF GENERAL STRUCTURE
Key concepts

• Graph data model
• URI-based vocabulary
• Datatypes
• Literals
• XML serialization syntax
• Expression of simple facts
• Entailment
Graph data model

- Triple: subject, predicate, object
- Expression: collection of triples
  - RDF graph
Example

```
Ridley Scott

directed

Blade Runner

blade_runner

name

Blade Runner
```

Terminology and constraints

- Subject and Object are called Nodes
- Predicate and Property are synonyms
- Special unnamed nodes: Blank Nodes
- Subject may be: URI reference or blank node
- Predicate must be: URI reference
- Object may be: URI reference, literal or blank node
The Triples and the Graph

• The **assertion** of an RDF triple says that some relationship, indicated by the predicate, holds between the things denoted by subject and object of the triple.

• The **assertion** of an RDF graph amounts to asserting all the triples in it, so the meaning of an RDF graph is the conjunction (logical AND) of the statements corresponding to all the triples it contains.
Expression of Simple Facts

• Some simple facts indicate a relationship between two things → one triple
  – the predicate names the relationship
  – the subject and object denote the two things
Information in triples

http://directory.com/people#FulvioCorno

http://xmlns.com/foaf/0.1/workplaceHomepage

http://www.polito.it/

PersonID | Homepage
----------|----------
FulvioCorno | http://www.polito.it/

RDF

Relational database

First order logic predicate

HasCompanyHomePage('FulvioCorno', 'http://www.polito.it/');
# Triples vs Databases

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>RELATIONAL DATABASE</th>
<th>KNOWLEDGEBASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>Schema</td>
<td>Ontology statements</td>
</tr>
<tr>
<td>Data</td>
<td>Rows</td>
<td>Instance statements</td>
</tr>
<tr>
<td>Administration language</td>
<td>DDL</td>
<td>Ontology statements</td>
</tr>
<tr>
<td>Query language</td>
<td>SQL</td>
<td>SPARQL</td>
</tr>
<tr>
<td>Relationships</td>
<td>Foreign keys</td>
<td>Multidimensional</td>
</tr>
<tr>
<td>Logic</td>
<td>External of database/triggers</td>
<td>Formal logic statements</td>
</tr>
<tr>
<td>Uniqueness</td>
<td>Key for table</td>
<td>URI</td>
</tr>
</tbody>
</table>
But...

- Relational database tables may have an arbitrary number of columns
- First order logic predicates may have an arbitrary number of places (arguments)
- RDF triples may only have one subject and one object
  - Complex statements have to be decomposed for representation as RDF triples
Example
Example

• Represent in RDF the following statement
• "there is a Person identified by http://www.w3.org/People/EM/contact#me, whose name is Eric Miller, whose email address is em@w3.org, and whose title is Dr."
Example

http://www.w3.org/2000/10/swap/pim/contact#Person

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

http://www.w3.org/People/EM/contact#me

http://www.w3.org/2000/10/swap/pim/contact#fullName

http://www.w3.org/2000/10/swap/pim/contact#mailbox

mailto:em@w3.org

http://www.w3.org/2000/10/swap/pim/contact#personalTitle

Dr.

Eric Miller
URIs represent (almost) everything

- **Nodes (subject or object)**
  - individuals: Eric Miller, identified by http://www.w3.org/People/EM/contact#me
  - kinds of things: Person, identified by http://www.w3.org/2000/10/swap/pim/contact#Person
  - values of properties: mailto:em@w3.org as the value of the mailbox property

- **Predicates**
  - properties of things: mailbox, identified by http://www.w3.org/2000/10/swap/pim/contact#mailbox
Non-URI information

- Literals (only as objects, never as subjects)
  - The name "Eric Miller"
  - The title "Dr."
  - May be localized
    - "Dr."@en
    - "Dott."@it
  - May be typed with XMLSchema data types
    - "27"^^<http://www.w3.org/2001/XMLSchema#integer>
    - "37"^^xsd:integer
    - "1999-08-16"^^xsd:date
URIs are more than URLs

- **URL** = uniform resource locator
  - Designed to locate, and retrieve, resources on the web
- **URI** = uniform resource identifier
  - More general
  - Identifies also resources that do not have a network location
  - Every person or organization can independently create URIs, and use them to identify “things” (either concrete or abstract)
URIref = URI#fragment

• URIref = URI reference
• A single URI may define many different resources
  – E.g., the URI references an RDF file with many definitions
• To identify a single fragment inside the URI, we use the ‘#’ notation
  – E.g., http://example.org/index#person
Graph Merging

Graph 1

Graph 2

Graphs 1 and 2
RDF Syntaxes

• Multiple allowed serializations
• Embeddable in documents
• Embeddable in programming languages
• Compatible with relational views
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>
</rdf:RDF>
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>
</rdf:RDF>

Name space shortcut.
Equivalent to
http://www.w3.org/2000/10/swap/pim/contact#fullName
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
    <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
        <contact:fullName>Eric Miller</contact:fullName>
        <contact:mailbox rdf:resource="mailto:em@w3.org"/>
        <contact:personalTitle>Dr.</contact:personalTitle>
    </contact:Person>
</rdf:RDF>
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
    <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
        <contact:fullName>Eric Miller</contact:fullName>
        <contact:mailbox rdf:resource="mailto:em@w3.org"/>
        <contact:personalTitle>Dr.</contact:personalTitle>
    </contact:Person>
</rdf:RDF>
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>
</rdf:RDF>
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
    <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
        <contact:fullName>Eric Miller</contact:fullName>
        <contact:mailbox rdf:resource="mailto:em@w3.org"/>
        <contact:personalTitle>Dr.</contact:personalTitle>
    </contact:Person>
</rdf:RDF>
Hands-on exercise

• The Geonames service (http://www.geonames.org) is based on the semantic description of all geographic elements, as shown in http://www.geonames.org/ontology

• Find Torino, and show its RDF representation

• Check and visualize the triples on https://www.w3.org/RDF/Validator/

• Find your vacation destination, and show its RDF representation
"Turtle" notation
(Terse RDF Triple Language)

```turtle
<http://www.w3.org/People/EM/contact#me>  
<http://www.w3.org/2000/10/swap/pim/contact#fullName>  
"Eric Miller" .

<http://www.w3.org/People/EM/contact#me>  
<http://www.w3.org/2000/10/swap/pim/contact#mailbox>  
<mailto:em@w3.org> .

<http://www.w3.org/People/EM/contact#me>  
<http://www.w3.org/2000/10/swap/pim/contact#personalTitle>  
"Dr." .

<http://www.w3.org/People/EM/contact#me>  
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>  
<http://www.w3.org/2000/10/swap/pim/contact#Person> .
```
Turtle basic syntax
Abbreviating prefixes

@prefix w3people: <http://www.w3.org/People/>
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#>

w3people:EM#me contact:fullName "Eric Miller" .

w3people:EM#me contact:mailbox <mailto:em@w3.org> .

w3people:EM#me contact:personalTitle "Dr." .

w3people:EM#me rdf:type contact:Person .
Abbreviating common subjects

@prefix w3people: <http://www.w3.org/People/>
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#>

w3people:EM#me contact:fullName "Eric Miller" ;

  contact:mailbox <mailto:em@w3.org> ;

  contact:personalTitle "Dr." ;

  rdf:type contact:Person .
Abbreviating common subjects and predicates

@prefix w3people: <http://www.w3.org/People/>
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#>

w3people:EM#me contact:fullName "Eric Miller" ;
  contact:mailbox <mailto:em@w3.org>, <mailto:miller@gmail.com> ;
  contact:personalTitle "Dr.", "Prof." ;
  rdf:type contact:Person .
N-Triples

• The N-Triples format is the same as Turtle, except
  – Does not allow @prefixes
  – Does not allow ; or , abbreviations
• Much more verbose
• Simpler to parse
Hands-on exercise

• Model as an RDF graph a subset of the following assertions:
  – Oracle Corporation (NASDAQ: ORCL) and Sun Microsystems (NASDAQ: JAVA) announced today they have entered into a definitive agreement under which Oracle will acquire Sun common stock for $9.50 per share in cash.
  – [...] Sun Microsystems, Inc. (NASDAQ: JAVA) develops the technologies that power the global marketplace. [...] Sun can be found in more than 100 countries and on the Web at http://www.sun.com.
  – Oracle (NASDAQ: ORCL) is the world's largest enterprise software company. For more information about Oracle, please visit our Web site at http://www.oracle.com.

RDF VOCABULARIES
RDF vocabularies

• A set of URIref is called vocabulary
• Common vocabularies collect URIrefs under the same name space, so that all nodes may be reached with QNames such as:
  – prefix:nodeName
• The name space is chosen to represent the organization responsible for the definitions
• Every elaboration in RDF must first resolve all prefixes, so that only absolute URIs are used by the algorithms
Common prefixes

- prefix rdf:, namespace URI: http://www.w3.org/1999/02/22-rdf-syntax-ns#
- prefix rdfs:, namespace URI: http://www.w3.org/2000/01/rdf-schema#
- prefix dc:, namespace URI: http://purl.org/dc/elements/1.1/
- prefix owl:, namespace URI: http://www.w3.org/2002/07/owl#
- prefix xsd:, namespace URI: http://www.w3.org/2001/XMLSchema#
- prefix ex:, namespace URI: http://www.example.org/ (or http://www.example.com/)
Vocabulary reuse

• Extremely easy to re-use other vocabularies in our RDF graph... just define a prefix to point to the proper name space

• When using a predicate, always check if its semantics is already satisfied by some property defined in well-known vocabularies
  – Never re-define, with a different URIref, some already existing predicate

• The same applies for names, but with somewhat less importance.
Hands-on: let's explore some useful vocabularies...

• Dublin Core

• Recent Dublin Core enhancement: DCMI Metadata Terms
Dublin Core Registry

http://dcmi.kc.tsukuba.ac.jp/dcregistry/navigateServlet
Hands-on: let's explore some useful vocabularies...

- **FOAF**
  - Namespace: xmlns:foaf="http://xmlns.com/foaf/0.1/"

- **RSS 1.0**

- **Curious?**
  - [http://richard.cyganiak.de/blog/2011/02/top-100-most-popular-rdf-namespace-prefixes/](http://richard.cyganiak.de/blog/2011/02/top-100-most-popular-rdf-namespace-prefixes/)
Blank nodes

- RDF just supports triples, i.e., binary relationships
- Higher-order relationships must be broken down into many binary pieces
- Breaking down means creating additional nodes
- Such additional nodes will never be referenced from outside the current sub-graph → they don’t need a name!
- A subject or object may be left “blank”
Example

• Anonymous resources
• No "strong" URIs
• Preserves attributes and relationships
Example (multi-dimensional attributes)
Example

exaddressid:85740  exterms:street  "1501 Grant Avenue" .
exaddressid:85740  exterms:city  "Bedford" .
exaddressid:85740  exterms:state  "Massachusetts" .
**Example – with blank node**

```
exstaff:85740  exterms:address  _:johnaddress .
_:johnaddress  exterms:street  "1501 Grant Avenue" .
_:johnaddress  exterms:city    "Bedford" .
_:johnaddress  exterms:state   "Massachusetts" .
_:johnaddress  exterms:postalCode "01730" .
```
More complex Example

@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntaxns#.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix : <http://example.org/>.

<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/XML Syntax Specification (Revised)";
  :editor [
    :fullName "Dave Beckett";
    :homePage <http://purl.org/net/dajobe/>
  ].
XML SERIALIZATION
Details on the XML serialization

• The XML document has a root node `<rdf:RDF>`
• Specifying the subject:
  – `<rdf:Description rdf:about="SubjectURIref">`
• Specifying properties, in the body of the `rdf:Description` tag
  – `<ex:propertyName>ObjectLiteral</ex:propertyName>`
  – `<ex:otherProperty rdf:resource="ObjectURIref"/>`
• Several triples sharing the same subject may be collected in the same `rdf:Description` body
Examples

1. `<?xml version="1.0"?>`
2. `<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"`
   3. `xmlns:exterms="http://www.example.org/terms/"`>
4. `<rdf:Description rdf:about="http://www.example.org/index.html">
5.   `<exterms:creation-date>August 16, 1999</exterms:creation-date>
6.   `</rdf:Description>`
7. `</rdf:RDF>`
Examples

1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.       xmlns:exterms="http://www.example.org/terms/">
4.   <rdf:Description rdf:about="http://www.example.org/index.html">
5.       <exterms:creation-date>August 16, 1999</exterms:creation-date>
6.   </rdf:Description>
7. </rdf:RDF>

1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.       xmlns:dc="http://purl.org/dc/elements/1.1/">
4.       xmlns:exterms="http://www.example.org/terms/">
5.   <rdf:Description rdf:about="http://www.example.org/index.html">
6.       <exterms:creation-date>August 16, 1999</exterms:creation-date>
7.       <dc:language>en</dc:language>
8.       <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
9.   </rdf:Description>
10. </rdf:RDF>
Blank nodes in XML: rdf:nodeID

5. <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
6.   <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
7.   <exterms:editor rdf:nodeID="abc"/>
8. </rdf:Description>

9. <rdf:Description rdf:nodeID="abc">
10.  <exterms:fullName>Dave Beckett</exterms:fullName>
12. </rdf:Description>
Typed literals in XML

```xml
<rdf:Description rdf:about="http://www.example.org/index.html">
  <exterms:creation-date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1999-08-16</exterms:creation-date>
</rdf:Description>
```

4. `<rdf:Description rdf:about="http://www.example.org/index.html">
5.   `<exterms:creation-date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1999-08-16
6.   `</exterms:creation-date>
7. </rdf:Description>
RDF SEMANTIC FEATURES
RDF Data structures

- Containers (unbounded)
  - rdf:Bag (unordered)
  - rdf:Seq (ordered)
  - rdf:Alt (one-of)
    - Semantically equivalent, the different between Bag/Seq/Alt is only in its “intended usage”
    - Does not limit the member elements to the ones declared

- Collections (bounded)
  - rdf:List
    - Only the mentioned elements are part of the collection
Reification

• It may be sometimes useful to assert a statement about another statement.
  – For example, I want to say who added a fact (a triple) to my set of statements
• In this case, instead of writing the triple, we describe the triple by
  – Giving a name to the statement (rdf:Statement)
  – Giving the elements of the triple with rdf:subject, rdf:predicate, rdf:object
Example

exproducts:item10245  exterms:weight  "2.4"^^xsd:decimal .

... and now the statement has a URIref: this.rdf#triple12345
We expressed the dc:creator of the previous statement!
Entailment

• An RDF expression A is said to entail another RDF expression B if every possible arrangement of things in the world that makes A true also makes B true. On this basis, if the truth of A is presumed or demonstrated then the truth of B can be inferred.

• The mechanism for defining formal semantics for RDF
• The ultimate mechanism for creating reasoning engines in the semantic web
• Never asserts anything about “the things in the world”, only about the propagation of truth in RDF statements/assertions

More on this in the RDF Semantics chapter!
RDF SCHEMA
RDF Schema

• Special RDF vocabulary for describing the properties and the content of... RDF vocabularies
• Think of a definition (schema) of the nodes and predicates used in an RDF document.
  – However, this definition is expressed in RDF, too, by using the RDFS vocabulary
• With RDFS we may restrict the usage of RDF nodes and predicates, by introducing coherency and a sort of data types
• RDF Schema provides a type system for RDF
RDFS nature

- RDFS does not specify a vocabulary of descriptive properties such as “author”
- RDFS specifies mechanisms that may be used to name and describe properties and the classes of resource they describe
- Similar to the type systems of object-oriented programming languages, but:
  - OO languages define a class in terms of the properties its instances may have
  - RDFS describes properties in terms of the classes of resource to which they apply (domain & range)
Example

- **OO language**
  - define a class eg:Book
  - with an attribute called eg:author
  - of type eg:Person

- **RDFS**
  - define the eg:author property
  - to have a domain of eg:Document
  - and a range of eg:Person

- **Why?**
  - Easy for others to subsequently define additional properties with a domain of eg:Document or a range of eg:Person
  - This can be done without the need to re-define the original description of these classes
  - It allows anyone to extend the description of existing resources, one of the architectural principles of the Web
Defining Classes in RDFS

• \texttt{rdf:type}
  – Defines the ‘type’ of the subject node
  – The object of ‘type’ must be a class

• \texttt{rdfs:Class}
  – The set of all possible classes
  – A class is any resource having an \texttt{rdf:type} property whose value is the resource \texttt{rdfs:Class}

\begin{verbatim}
ex:MotorVehicle rdf:type rdfs:Class .
\end{verbatim}
Defining class hierarchies

• rdfs:subClassOf
  – Defines a narrower class
  – Any instance of class ex:Van is also an instance of class ex:MotorVehicle
  – A transitive predicate

ex:MotorVehicle rdf:type rdfs:Class .

ex:Van rdf:type rdfs:Class .
ex:Truck rdf:type rdfs:Class .
ex:Van rdfs:subClassOf ex:MotorVehicle .
Class hierarchies

http://www.example.org/schemas/vehicles#MotorVehicle

http://www.example.org/schemas/vehicles#Truck

http://www.example.org/schemas/vehicles#Van

http://www.example.org/schemas/vehicles#PassengerVehicle

http://www.example.org/schemas/vehicles#MiniVan
Defining properties in RDFS

- **rdf:Property**
  - Any URIref used as a predicate has an rdf:type of rdf:Property

- **rdfs:domain, rdfs:range**
  - Define the domain and the range of the property
  - Domain and range are Classes

- **rdfs:subPropertyOf**
  - Defines hierarchies of properties
Example

```xml
<rdf:Property rdf:ID="registeredTo">
  <rdfs:domain rdf:resource="#MotorVehicle"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Property>

<rdf:Property rdf:ID="rearSeatLegRoom">
  <rdfs:domain rdf:resource="#PassengerVehicle"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
</rdf:Property>

<rdfs:Class rdf:ID="Person"/>

<rdfs:Datatype rdf:about="&xsd;integer"/>
```
## RDF/RDFS Classes

<table>
<thead>
<tr>
<th>Class name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs:Resource</td>
<td>The class resource, everything.</td>
</tr>
<tr>
<td>rdfs:Literal</td>
<td>The class of literal values, e.g. textual strings and integers.</td>
</tr>
<tr>
<td>rdf:XMLLiteral</td>
<td>The class of XML literals values.</td>
</tr>
<tr>
<td>rdfs:Class</td>
<td>The class of classes.</td>
</tr>
<tr>
<td>rdf:Property</td>
<td>The class of RDF properties.</td>
</tr>
<tr>
<td>rdfs:Datatype</td>
<td>The class of RDF datatypes.</td>
</tr>
<tr>
<td>rdf:Statement</td>
<td>The class of RDF statements.</td>
</tr>
<tr>
<td>rdf:Bag</td>
<td>The class of unordered containers.</td>
</tr>
<tr>
<td>rdf:Seq</td>
<td>The class of ordered containers.</td>
</tr>
<tr>
<td>rdf:Alt</td>
<td>The class of containers of alternatives.</td>
</tr>
<tr>
<td>rdfs:Container</td>
<td>The class of RDF containers.</td>
</tr>
<tr>
<td>rdfs:ContainerMembershipProperty</td>
<td>The class of container membership properties, rdf:_1, rdf:_2, ..., all of which are sub-properties of 'member'.</td>
</tr>
<tr>
<td>rdf:List</td>
<td>The class of RDF Lists.</td>
</tr>
</tbody>
</table>
# RDF/RDFS Properties

<table>
<thead>
<tr>
<th>Property name</th>
<th>comment</th>
<th>domain</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf:type</td>
<td>The subject is an instance of a class.</td>
<td>rdfs:Resource</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdfs:subClassOf</td>
<td>The subject is a subclass of a class.</td>
<td>rdfs:Class</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdfs:subPropertyOf</td>
<td>The subject is a subproperty of a property.</td>
<td>rdf:Property</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdfs:domain</td>
<td>A domain of the subject property.</td>
<td>rdf:Property</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdfs:range</td>
<td>A range of the subject property.</td>
<td>rdf:Property</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdfs:label</td>
<td>A human-readable name for the subject.</td>
<td>rdfs:Resource</td>
<td>rdfs:Literal</td>
</tr>
<tr>
<td>rdfs:comment</td>
<td>A description of the subject resource.</td>
<td>rdfs:Resource</td>
<td>rdfs:Literal</td>
</tr>
<tr>
<td>rdfs:member</td>
<td>A member of the subject resource.</td>
<td>rdfs:Resource</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:first</td>
<td>The first item in the subject RDF list.</td>
<td>rdf:List</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:rest</td>
<td>The rest of the subject RDF list after the first item.</td>
<td>rdf:List</td>
<td>rdf:List</td>
</tr>
<tr>
<td>rdfs:seeAlso</td>
<td>Further information about the subject resource.</td>
<td>rdfs:Resource</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdfs:isDefinedBy</td>
<td>The definition of the subject resource.</td>
<td>rdfs:Resource</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:value</td>
<td>Idiomatic property used for structured values (see the RDF Primer for <a href="#">an example</a> of its usage).</td>
<td>rdfs:Resource</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:subject</td>
<td>The subject of the subject RDF statement.</td>
<td>rdf:Statement</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:predicate</td>
<td>The predicate of the subject RDF statement.</td>
<td>rdf:Statement</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:object</td>
<td>The object of the subject RDF statement.</td>
<td>rdf:Statement</td>
<td>rdfs:Resource</td>
</tr>
</tbody>
</table>
Hands-on exercise

• Find on Wikipedia a topic (eg. Politecnico di Torino)
• Go to: http://dbpedia.org/page/page_name
• Browse using Faceted Browser
  – Look for Classes, types, relationships
• Export in different Formats
  – Also check the HTML
RDF SEMANTICS AND REASONING
Reasoning

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

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

- **Questions:**
  - Are Robins birds? → Yes
  - Do all birds have wings? → Yes
  - Do robins have wings? → ?????
What kind of reasoning can we expect?

- Depends on
  - The semantic level of the representation (RDFS, OWL dialect), that implies a given mathematical formalization for the knowledge base
  - The size of the involved knowledge base
  - The presence (or absence) of instances
  - The capabilities of the reasoning tool
Definitions

• **Inference** (n.) An act or process of constructing new expressions from existing expressions, or the result of such an act or process. In RDF, inferences corresponding to **entailments** are described as **correct** or **valid**.

• **Inference rule**, formal description of a type of inference; **inference system**, organized system of inference rules; also, software which generates inferences or checks inferences for validity.
Recall: RDF Schema

RDF vs RDFS

RDF Schema

RDF

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

- \((X \ R \ Y), (R \ subPropertyOf \ Q) \Rightarrow (X \ Q \ Y)\)
- \((X \ R \ Y), (R \ domain \ C) \Rightarrow (X \ type \ C)\)
- \((X \ type \ C), (C \ subClassOf \ D) \Rightarrow (X \ type \ D)\)
- ...


RDF Semantics

Semantics is not about the “meaning” of assertions

- The ‘meaning’ of an assertion in RDF or RDFS may depend on many factors, including social conventions, comments in natural language or links to other content-bearing documents, ... (non machine-processable information)
- Semantics restricts itself to a formal notion of meaning which could be characterized as the part that is common to all other accounts of meaning, and can be captured in mechanical inference rules
Model Semantics

- The RDF Semantics W3C Recommendation uses model theory for specifying the semantics of the RDF language.

- Model theory assumes that the language refers to a ‘world’, and describes the minimal conditions that a world must satisfy in order to assign an appropriate meaning for every expression in the language.

- A particular world is called an interpretation.

- The idea is to provide an abstract, mathematical account of the properties that any such interpretation must have, making as few assumptions as possible about its actual nature or intrinsic structure, thereby retaining as much generality as possible.
Goals of the inference process

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

- **Interpretations** (Normative)
  - Mapping of RDF assertions into an *abstract model*, based on set-theory
  - With an “interpretation operator” \( I() \), maps a RDF graph into a highly abstract set of high-cardinality sets
  - Highly theoretical model, useful to prove mathematical properties

- **Entailments** (Informative)
  - *Transformation rules* to derive new assertions from existing ones
  - May be *proven complete and consistent* with the formal interpretation
Definitions

• **Interpretation (of)** (n.) A minimal formal description of those aspects of a world which is just sufficient to establish the truth or falsity of any expression of a logic.

• **World** (n.) (with the:) (i) The actual world. (with a:) 
  (ii) A way that the actual world might be arranged. 
  (iii) An interpretation 
  (iv) A possible world.
Definition

- **Entail (v.), entailment (n.).** A semantic relationship between expressions which holds whenever the truth of the first guarantees the truth of the second. Equivalently, whenever it is logically impossible for the first expression to be true and the second one false. Equivalently, when any interpretation which satisfies the first also satisfies the second. (Also used between a set of expressions and an expression.)
Interpretation: minimum notions

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

- An RDF **interpretation** of $V$ consists of:
  - $IR$, a non-empty set of **resources** (domain or universe)
  - $IP \subseteq IR$, a set of **properties** (each property is also a resource; symbol $v \in V$ is a property if, and only if, $IS(v) \in IP$)
  - $IS : V \to IR$, an interpretation of resources (with each symbol from $V$ a resource is associated)
  - $IEXT : IP \to 2^{IR \times IR}$, an interpretation of properties (each property is a binary relation, i.e., a subset of $IR \times IR$)

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

\[
\begin{align*}
s, p, o & \in V, \quad IS(p) \in IP \quad \text{and} \quad (IS(s), IS(o)) \in IEXT(IS(p))
\end{align*}
\]

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

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

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

IS assigns one thing to each name in the vocabulary

1 is the only property in the set IP

IEXT maps 1 to a property extension

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

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

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

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

- $IP$: set of properties, $IR$: set of resources
- $IEXT(I(p))$: mapping from $IP$ into the powerset of $IR \times IR$
  i.e. the set of sets of pairs $<x,y>$ with $x$ and $y$ in $IR$
Example

Construct an interpretation in which the following RDF graph is true:

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

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

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

The mappings colored red show that this interpretation satisfies the triple `rdf:type rdf:type rdf:Property.`
Entailment

- I satisfies E if I(E)=true
- A set S of RDF graphs entails a graph E if every interpretation which satisfies every member of S also satisfies E
- In human words:
  - assertion = a claim that the world is an interpretation which assigns the value true to the assertion
  - If A entails B, then
    - any interpretation that makes A true also makes B true
    - an assertion of A already contains the same "meaning" as an assertion of B
    - the meaning of B is somehow contained in, or subsumed by, that of A
Entailment regime

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

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

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

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

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

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

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

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

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

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

(1/3)

```
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:_predicate 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 .
```
### RDFS Axiomatic Triples (2/3)

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

<table>
<thead>
<tr>
<th>RDF</th>
<th>RDFS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf:Alt</td>
<td>rdfs:subClassOf</td>
<td>rdfs:Container</td>
</tr>
<tr>
<td>rdf:Bag</td>
<td>rdfs:subClassOf</td>
<td>rdfs:Container</td>
</tr>
<tr>
<td>rdf:Seq</td>
<td>rdfs:subClassOf</td>
<td>rdfs:Container</td>
</tr>
<tr>
<td>rdfs:ContainerMembershipProperty</td>
<td>rdfs:subClassOf</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdfs:isDefinedBy</td>
<td>rdfs:subPropertyOf</td>
<td>rdfs:seeAlso</td>
</tr>
<tr>
<td>rdf:XMLLiteral</td>
<td>rdf:type</td>
<td>rdfs:Datatype</td>
</tr>
<tr>
<td>rdf:XMLLiteral</td>
<td>rdfs:subClassOf</td>
<td>rdfs:Literal</td>
</tr>
<tr>
<td>rdfs:Datatype</td>
<td>rdfs:subClassOf</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdf:_1</td>
<td>rdf:type</td>
<td>rdfs:ContainerMembershipProperty</td>
</tr>
<tr>
<td>rdf:_1</td>
<td>rdfs:domain</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:_1</td>
<td>rdfs:range</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:_2</td>
<td>rdf:type</td>
<td>rdfs:ContainerMembershipProperty</td>
</tr>
<tr>
<td>rdf:_2</td>
<td>rdfs:domain</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdf:_2</td>
<td>rdfs:range</td>
<td>rdfs:Resource</td>
</tr>
</tbody>
</table>

...
Role of Entailment rules (3/3)

- RDFS entailment satisfies the “RDFS entailment lemma”:
  - S rdfs-entails E if and only if there is a graph which can be derived from S plus the RDF and RDFS axiomatic triples by the application of rule lg, rule gl and the RDF and RDFS entailment rules and which either simply entails E or contains an XML clash
The outputs of these rules will often trigger others.

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

Table 1: Dependencies between RDFS entailment rules

Effect of Entailment rules (2/3)

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

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

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

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

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

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

```
ex:creates
```

```
ex:paints
```

```
ex:paints
```

```
ex:r1
```

```
ex:r2
```

```
ex:Artist
```

```
ex:Human
```

```
ex:Painter
```

```
ex:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```

```
x:paints
```

```
x:r1
```

```
x:r2
```

```
x:creates
```
Example: closure
Example: reduction

The diagram illustrates a portion of the Semantic Web, focusing on the relationships between 'ex:Artist', 'ex:Painter', 'ex:creates', and 'ex:paints'. The properties and classes are shown with RDF terminology such as `rdfs:domain`, `rdfs:subPropertyOf`, and `rdfs:subClassOf`. The diagram helps to visualize the hierarchical and relational structure of the Semantic Web.
References

- W3C Semantic Web
  - https://www.w3.org/standards/semanticweb/

- W3C Tutorial on Semantic Web
  - https://www.w3.org/Consortium/Offices/Presentations/RDFTutorial/

- Lee Feigenbaum, “The Semantic Web Landscape”
  - http://www.slideshare.net/LeeFeigenbaum/cshals-2010-w3c-semantic-web-tutorial
License

• This work is licensed under the Creative Commons “Attribution-NonCommercial-ShareAlike Unported (CC BY-NC-SA 3.0)” License.

• You are free:
  – to Share - to copy, distribute and transmit the work
  – to Remix - to adapt the work

• Under the following conditions:
  – Attribution - You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
  – Noncommercial - You may not use this work for commercial purposes.
  – Share Alike - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

• To view a copy of this license, visit http://creativecommons.org/license/by-nc-sa/3.0/