Multi-agent and Semantic Web Systems: RDF Data Structures

Fiona McNeill

School of Informatics

31st January 2013

URIs and Resources

- A resource is any entity that one can hold information about.

URIs and Resources

- A resource is any entity that one can hold information about.
- An information resource is a resource whose essential characteristics can be conveyed in a message.

URIs and Resources

- A resource is any entity that one can hold information about.
- An information resource is a resource whose essential characteristics can be conveyed in a message.
- Information resources typically have one or more representations that can be accessed using HTTP.

URIs and Resources

- A resource is any entity that one can hold information about.
- An information resource is a resource whose essential characteristics can be conveyed in a message.
- Information resources typically have one or more representations that can be accessed using HTTP.
- Any other resource is a non-information resource.

URIs and Resources

- A resource is any entity that one can hold information about.
- An information resource is a resource whose essential characteristics can be conveyed in a message.
- Information resources typically have one or more representations that can be accessed using HTTP.
- Any other resource is a non-information resource.

[A] document is an example of an information resource. It consists of words and punctuation symbols and graphics and other artifacts that can be encoded, with varying degrees of fidelity, into a sequence of bits. There is nothing about the essential information content of [a] document that cannot in principle be transferred in a message.

`http://www.w3.org/TR/webarch/#id-resources`

URIs and Resources

- Every resource can be uniquely identified by a URI.

URIs and Resources

- Every resource can be uniquely identified by a URI.
- Every URI identifies a resource.

URIs and Resources

- Every resource can be uniquely identified by a URI.
- Every URI identifies a resource.
- The act of retrieving a representation of a resource identified by a URI is known as dereferencing that URI.

URIs and Resources

- Every resource can be uniquely identified by a URI.
- Every URI identifies a resource.
- The act of retrieving a representation of a resource identified by a URI is known as dereferencing that URI.

...consider the creation of a [bank] statement ....We'll suppose that a URI identifies the resource which, in this case, is a particular set of of binary data held in a relational database. To create a representation of the resource, the appropriate data is first extracted from the database and converted to textual form. Then it is embedded in a stream of HTML markup that also references appropriate styling information. This representation flows across the Web to a browser, where it is rendered. A user is able to perceive the rendered form and to understand the activity on the account for month in question.

`http://www.w3.org/2001/tag/doc/httpRange-14/2007-05-31/HttpRange-14`

Literals and Datatypes in Turtle

Full URIs are enclosed in <, >.

Triple with Full URIs

```
# this is a comment
<http://inf.ed.ac.uk/ont#aroast>
<http://http://purl.org/dc/elements/1.1/title>
”Artisan Roast” .
```

Triple in Abbreviated Form

```
@prefix : <http://inf.ed.ac.uk/ont#> .
@prefix dc: <http://http://purl.org/dc/elements/1.1/> .

:aroast dc:title ”Artisan Roast” .
```

Literals and Datatypes in Turtle

- Literals written with double quotes.
- So-called datatype URIs consist of a literal appended by ^^ and a URI — usually from XML Schema.

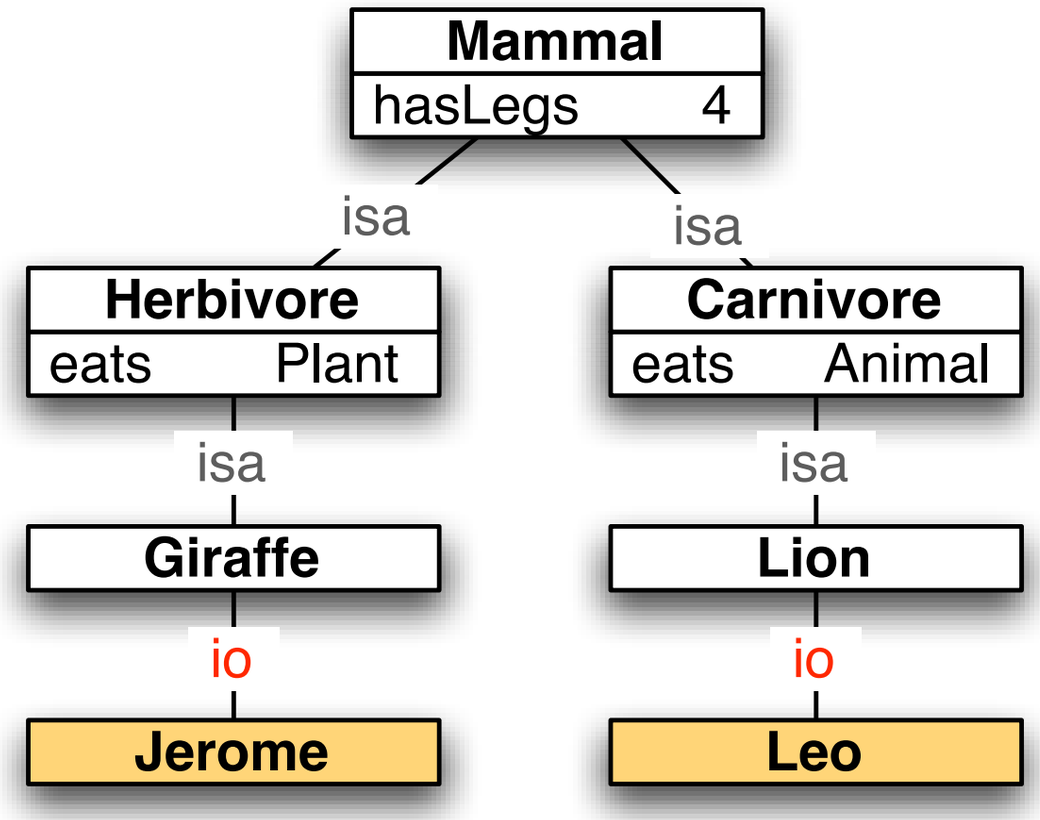
Literals and Datatype URIs

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:ebagel dc:title ”Elephant and Bagel” .
:ebagel :rating ”4”^^xsd:decimal .
```

Example datatypes: `xsd:string`, `xsd:boolean`, `xsd:decimal`, `xsd:float`, `xsd:double`, `xsd:dateTime`. *Cf.* http://www.w3.org/TR/xmlschema-2/#built-in-primitive-datatypes

Instance-of in RDF



Class membership expressed via `rdf:type`

Instance-of in RDF

RDF Type, 1

```
@prefix : <http://zoo.org/> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

:jerome rdf:type :Giraffe .
:leo rdf:type :Lion .
```

RDF Type, 2

```
@prefix : <http://inf.ed.ac.uk/ont#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

:aroast rdf:type :Cafe .
:Cafe rdf:type rdfs:Class .
```

Instance-of in RDF

`rdf:type` is often abbreviated as `a`

RDF Type, 3

```
@prefix : <http://zoo.org/> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

:jerome a :Giraffe .
:leo a :Lion .
```

RDF Type, 4

```
@prefix : <http://inf.ed.ac.uk/ont#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

:amy a foaf:Person .
```

Abbreviating groups of triples

, for repeated subjects and predicates:

Full Form

```
:aroast :lovedBy :stu .
:aroast :lovedBy :rod .
```

Abbreviated Form

```
:aroast :lovedBy :stu, :rod .
```

Abbreviating groups of triples

; for repeated subjects and predicates:

Full Form

```
:aroast dc:title ”Artisan Roast” .
:aroast db:locatedIn:eastEnd .
:aroast :rating ”5”^^xsd:decimal .
```

Abbreviated Form

```
:aroast dc:title ”Artisan Roast” ;
        db:locatedIn:eastEnd ;
        :rating ”5”^^xsd:decimal .
```

Abbreviating groups of triples

combining both abbreviations

Full Form

```
:aroast dc:title ”Artisan Roast” .
:aroast db:locatedIn:eastEnd .
:aroast :rating ”5”^^xsd:decimal .
:aroast :lovedBy :stu .
:aroast :lovedBy :rod .
```

Abbreviated Form

```
:aroast dc:title ”Artisan Roast” ;
        db:locatedIn:eastEnd ;
        :rating ”5”^^xsd:decimal ;
        :lovedBy :stu, :rod .
```

Blank nodes

- May be situations where we don't know the identity of a resource.
- E.g., Artisan Roast is run by a manager whose telephone number is 0131 229 0001.
- We could coin a new URI:

Fabricated URI

```
:aroast db:runBy :manager1 .
:manager1 pim:telno ”0131 229 0001” .
:manager a db:Manager .
```

Blank nodes

- Alternative: relevant node in the graph can be left blank.
- Blank node identifiers (also called anonymous resources) are of the form

`_:label`

- Blank node identifiers are not intended to be globally unique; only unique relative to a single 'graph'.

Blank Node Version

```
:aroast db:runBy _:a .
_:a pim:telno ”0131 229 0001” .
_:a a db:Manager .
```

Blank nodes

Alternative notation using `[, ]`:

Alternative Blank Node Notation

```
:aroast db:runBy [ pim:telno ”0131 229 0001” ;
                   a db:Manager] .
```

Blank nodes

- Blank notes are interpreted like existential quantification in first-order logic.
- Artisan Roast is run by someone whose telephone number is 0131 229 0001 and who is a manager.

$$\exists x.[\mathtt{runBy}(\mathtt{ar}, \mathtt{x}) \wedge$$
$$\mathtt{telno}(x) = \text{"}\mathtt{0131\ 229\ 0001}\text{"} \wedge$$
$$\mathtt{Manager}(x)]$$

RDF Formal Syntax

RDF Symbols:

- An RDF vocabulary $V = U \cup L$ consists of two disjoint subsets:
  - a set U of URI references, and
  - a set L of literals (textual representation of a value).
- B is a set disjoint from V containing blank nodes.

RDF Formal Syntax

RDF triples:

- An RDF triple in V ($V = U \cup L$) with blank nodes B is an expression of the from (s, p, o) where
  - $s \in U \cup B$ is the subject of the triple,
  - $p \in U$ is the predicate of the triple,
  - $o \in U \cup B \cup L$ is the object of the triple.

- An RDF triple is ground if it contains no blank node.

- An RDF graph in V with blank nodes B is a set of RDF triples. An RDF graph is ground if the triples it contains are all ground.

RDF vs first-order logic

- RDF vocabulary *V* is analogous to a signature of FOL.
- Blank nodes *B* are analogous to variables.
- Literals are analogous to individual constants, interpreted as specific data types.
- URIs are analogous to individual constants and also to binary relation symbols.
- A triple is analogous to an atomic formula.
- An RDF graph is analogous to a set of formulas. A finite RDF graph corresponds to the existential closure of the conjunction of its triples.

Given an RDF graph *G*, the set of members of $U \cup B \cup L$ occurring in subject or object position are regarded as nodes and each triple (s, p, o) is regarded as a directed edge from *s* to *o* labeled with *p*.

The Deep Web

- A lot of data on the web is only accessible through DB query.
- Deep web / invisible web / hidden web
  - estimated to be at least 500 times bigger than 'surface' web accessible to standard search engines.
  - http://library.albany.edu/internet/deepweb.html http://aip.completeplanet.com
- Semantic Web proposal: export RDBs to RDF and allow them to be integrated, searched, repurposed.

From RDB to RDF

There is a standard mapping from records in a DB to RDF triples:

- Each field (column) label is mapped to an RDF Predicate;
- the data in each corresponding field is mapped into the Object.

- the Subject of the triple can be a blank node, or possibly the primary key of the record.

DB Records as Triples

	column	
id	data	

DB Records as Triples

	Predicate	
Subject	Object	

DB Records as Triples

	Name	Artist	Place	
ID0039	"The Red Vineyard"	V. Van Gogh	Arles	

DB Records as Triples

DB2RDF Triples

```
:Paintings_0039 :name ”The Red Vinyard” ;
                :artist db:Vincent_Van_Gogh ;
                :place db:Arles .
```

- NB: some decision required about literals in fields;
- keep as literals or convert to URIs?

Summary

- We can use intermediary nodes to aggregate a subset of statements.

Summary

- We can use intermediary nodes to aggregate a subset of statements.
- Aggregates can be rooted in an ordinary resource node or in a blank node.

Summary

- We can use intermediary nodes to aggregate a subset of statements.
- Aggregates can be rooted in an ordinary resource node or in a blank node.
- Blank nodes are referred to with a special naming convention in triples: `_:label`.

Summary

- We can use intermediary nodes to aggregate a subset of statements.
- Aggregates can be rooted in an ordinary resource node or in a blank node.
- Blank nodes are referred to with a special naming convention in triples: `_:label`.
- Blank nodes can also be used for other things:
  - Referring to individuals via a cluster of properties.
  - Expressing relations of arity > 2 (via two patterns).

Summary

- We can use intermediary nodes to aggregate a subset of statements.
- Aggregates can be rooted in an ordinary resource node or in a blank node.
- Blank nodes are referred to with a special naming convention in triples: `_:label`.
- Blank nodes can also be used for other things:
  - Referring to individuals via a cluster of properties.
  - Expressing relations of arity > 2 (via two patterns).
- RDF only allows us to make statements about individuals; no quantifiers, no general statements.

Summary

- We can use intermediary nodes to aggregate a subset of statements.
- Aggregates can be rooted in an ordinary resource node or in a blank node.
- Blank nodes are referred to with a special naming convention in triples: `_:label`.
- Blank nodes can also be used for other things:
  - Referring to individuals via a cluster of properties.
  - Expressing relations of arity > 2 (via two patterns).
- RDF only allows us to make statements about individuals; no quantifiers, no general statements.
- We need something richer on top of RDF to define what counts as 'semantically well-formed' statements ⇒ RDFS

Reading

- SWWO, Ch 3
- Turtle Primer: http://www.w3.org/2007/02/turtle/primer/
- Information vs. non-information resources: http://www.w3.org/2001/ tag/doc/httpRange-14/2007-05-31/HttpRange-14

Task

- Either go back to the ontology you created in the second task, or create a new collection of things (for example: cities, countries, means of transport, people who like to travel, etc.) This does not need to be organised in an ontology!
- Create some RDF to make some statements about this. What sorts of things might you want to say? What sorts of things are difficult to say?