INFO-H-509 XML TECHNOLOGIES

Lecture 7: RDF and RDF Schema

Stijn Vansummeren

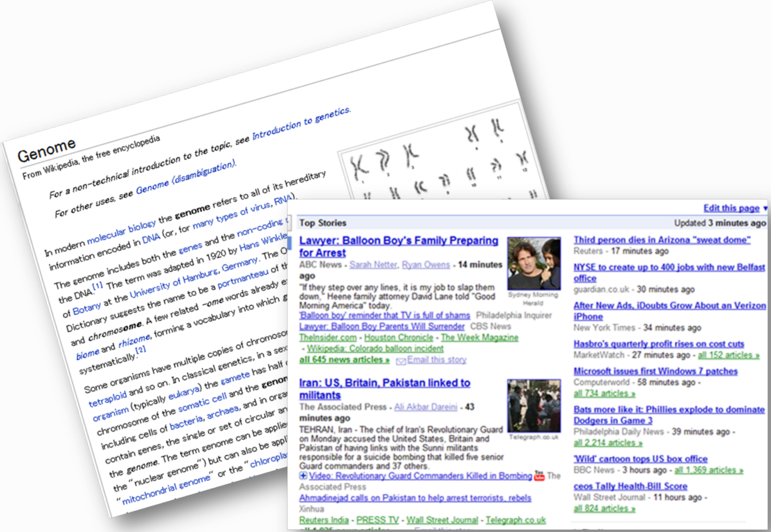
February 14, 2017

LECTURE OUTLINE

1. Motivation for RDF: the Semantic Web, Linked Data
2. The RDF Data Model
3. RDF Serialization formats
4. Simple ontologies in RDFS

Part I: Motivation

THE WEB IN THE RECENT PAST



Genome

From Wikipedia, the free encyclopedia

For a non-technical introduction to the topic, see Introduction to genetics.

For other uses, see Genome (disambiguation).

In modern molecular biology, the **genome** is the entirety of an organism's hereditary information. It is encoded either in DNA or, for many types of virus, in RNA.

The genome includes both the genes and the non-coding sequences of the DNA.[1] The term was adapted in 1920 by Hans Winkler, Professor of Botany at the University of Hamburg, Germany. The Oxford English Dictionary suggests the name to be a portmanteau of the words ***gene*** and *chromos**ome***. A few related *-ome* words already existed, such as *biome* and *rhizome*, forming a vocabulary into which *genome* fits systematically.[2]

- Natural language
- No structure
- Difficult to automatically process the knowledge encoded in these pages

Doesn’t HTML have structure?

```
<p>In modern <a href="/wiki/Molecular_biology" title="Molecular biology">molecular biology</a>,
<p>The genome includes both the <a href="/wiki/Gene" title="Gene">genes</a> and the <a href="/w
<p>Some organisms have multiple copies of chromosomes, <a href="/wiki/Diploid" title="Diploid"
<p>Both the number of <a href="/wiki/Base_pair" title="Base pair">base pairs</a> and the numbe
<p>An analogy to the human genome stored on DNA is that of instructions stored in a library:</p
<ul>
<li>The library would contain 46 books (chromosomes)</li>
<li>The books range in size from 400 to 3340 pages (genes)</li>
<li>which is 48 to 250 million letters (A,C,G,T) per book.</li>
<li>Hence the library contains over six billion letters total;</li>
<li>The library fits into a cell nucleus the size of a pinpoint;</li>
<li>A copy of the library (all 46 books) is contained in almost every cell of our body.</li>
</ul>
<table id="toc" class="toc">
<tr>
```

- Yes, but that structure only describes presentation
- It does not structure the information (DNA, genes, ...)

WHAT DO YOU MEAN: NO STRUCTURE?

Doesn't XML have structure?

2 Examples with same "structure":

```
<lecturer name="John Doe">
  <teaches>XML Technologies</teaches>
</lecturer>

<course name="XML Technologies">
  <lecturer>John Doe</lecturer>
</course>
```

- Yes, but that structure specifies only tag nesting (i.e., hierarchy)
- The tag names themselves are meaningless to a computer
- And it is impossible to unambiguously assign meaning to them …
- … unless you explicitly program a specific application (e.g. lecturers and courses)

WHAT DO YOU MEAN: NO STRUCTURE?

Doesn't XML have structure?

For a computer, this is the same as:

```
<padoijfa dkfjao="Imla Loqa">
  <Blablahdo>LMX Idjflal dld</Blablahdo>
</padoijfa>

<Rfjaomdla dkfjao="LMX Idjflal dld">
  <padoijfa>Imla Loqa</padoijfa>
</Rfjaomdla>
```

- Yes, but that structure specifies only tag nesting (i.e., hierarchy)
- The tag names themselves are meaningless to a computer
- And it is impossible to unambiguously assign meaning to them ...
- ... unless you explicitly program a specific application (e.g. lecturers and courses)

IN SUMMARY

> *"most of the Web's content... is designed for humans to read, not for computer programs to manipulate meaningfully. Computers can adeptly parse Web pages for layout and routine processing – here a header, there a link to another page but, in general, computers have no reliable way to process the Semantics...', or the meaning of the content of the page."*

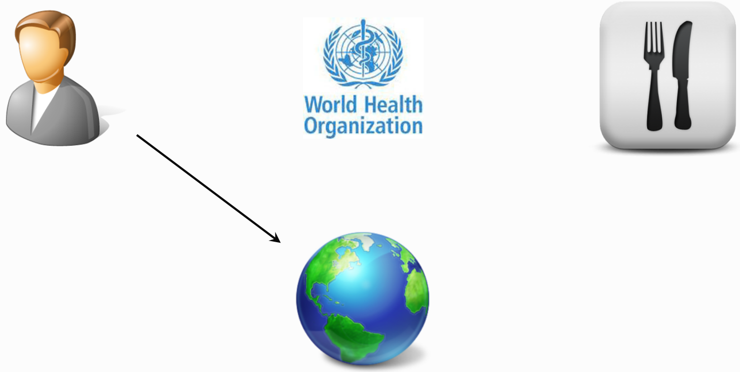
(Berners-Lee, Hendler and Lissila, 2001)

THE SEMANTIC WEB VISION

- Move from a Web of Documents to a Web of Knowledge
- This knowledge should be represented to have unambiguous semantics so that computers can process it unambiguously.
- This should make reasoning less programming-intensive: "Agents" (computer programs) should be able to reason with this knowledge without (too much) explicit programming.
- See e.g. "The Semantic Web" by Berners-Lee et. al. in Scientific American Magazine (link available on Course Webpage).

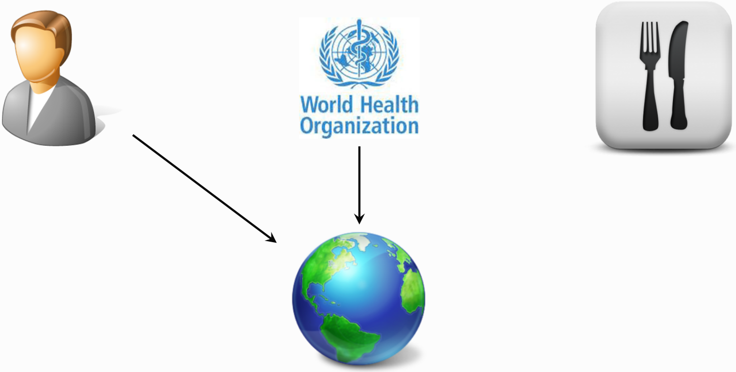
- John posts a profile on the Web (name, email address, profession) ...he also mentions that he has a tree nut allergy

EXAMPLE REASONING SCENARIO



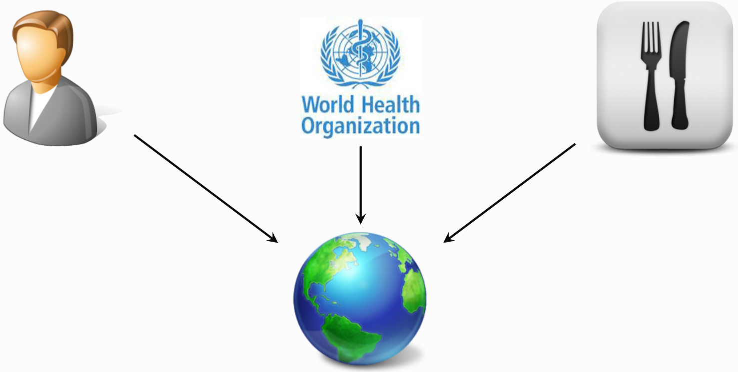
- John posts a profile on the Web (name, email address, profession) ...he also mentions that he has a tree nut allergy

EXAMPLE REASONING SCENARIO

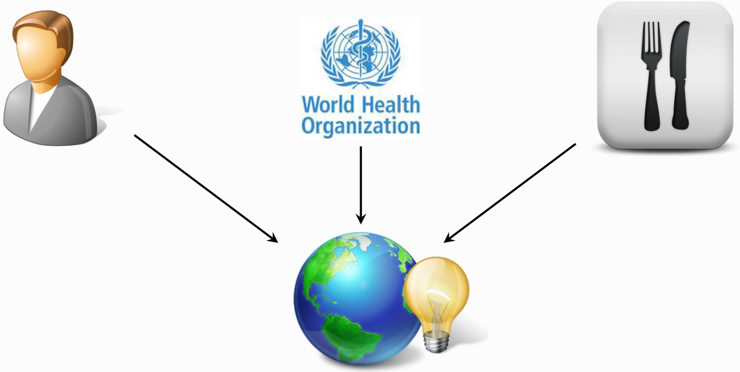


- The World Health Organization mentions on its site that people with tree nut allergy are usually also allergic to seeds, including sesamy seeds

EXAMPLE REASONING SCENARIO



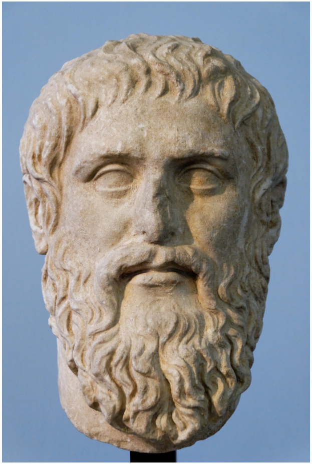
- Restaurant “The good life” mentions that its house salad contains sesame seeds.



- John's web agent concludes that when eating at the "The good life", John should not take the house salad.

SO ...

- We want to describe knowledge in such a way that (computers) can automatically reason with this.
- This has been the research topic in AI for decades ...
- ...and a fundamental question in science throughout history ...
- ...and it is really, really hard! (even impossible).

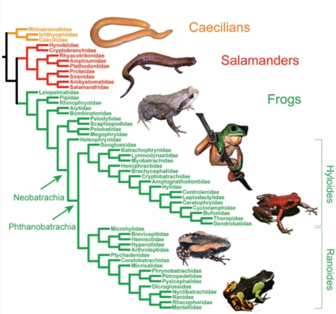


400 BC: birth of philosophical ontology

- Greek philosopher Plato asked: What is reality? Which things can be said to “exist”. What is the true nature of things?
- Philosophical ontology = the study of existence and being as such, and of the fundamental classes and relationships of existing things.

350 BC: first approach to classification

- Aristotle (384-322 BC) held the view that reallity is not given in universal ideas; we should carefully observe reallity, and describe our observations.
- Hereto, Aristotle developed ten (exhaustive) categories to classify all things that may exist, and subcategories to further specify them.
- Example: Aristotle's category of animals was composed of rational ones (humans) and irrational ones (beasts).



Taxonomy: the science

- The basic idea of classification has fueled many influential scientific models in later centuries.
- Carolus Linnaeus (1707-1778) laid the basis for modern biological classification by introducing Linnaen taxonomy.
- The term taxonomy now refers to the science of classification.
- Classifications are often tree-shaped, although not necessarily so (e.g., periodic table of elements).

Automated reasoning

- Traditionally, knowledge was recorded to be accessed and processed by human beings.
- Starting with Aristotle, however, the observation grew that logical deduction can itself be formalized and cast into a set of rules in a way reminiscent of arithmetics.
- Leibniz (1646-1716) formulated the dream to resolve conflicts in scientific discourse by just calculating the correct answer.

"If controversies were to arise, there would be no more need of disputation between two philosophers than between two accountants. For it would suffice to take their pencils in hands, sit down to their slates, and to say to each other: let us calculate."

Automated reasoning

- In summer 1956, John McCarthy and other leading researchers, inspired by the accessibility of digital computers, explored the possibility of using computers to simulate or generate intelligent behavior.
- In the course of this event, the term artificial intelligence was coined.
- Predominant in this context was the task of deducing new knowledge from known facts.

"A body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are presumed to exist in some area of interest and the relationships that hold them. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose.

Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly.

An ontology is an explicit specification of a conceptualization.

Thomas R. Gruber, 1993

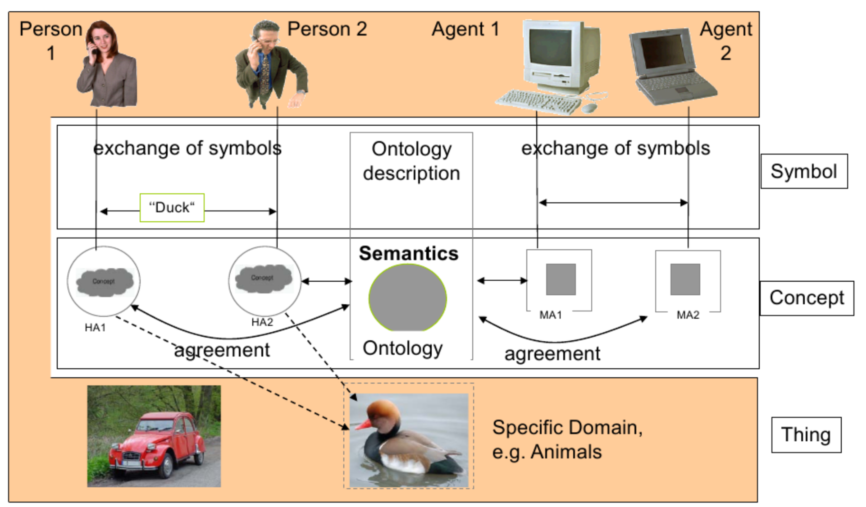
Ontologies in computer science

- Ultimately, computers are only working with symbols (a.k.a terms) that, to them do not have meaning.
- Therefore, we need to rigorously describe, for a given set of symbols, what properties these symbols are, how they relate to each other, and how they are collected in groups ("classes").
- Such rigorous descriptions are called ontologies in computer science/AI.

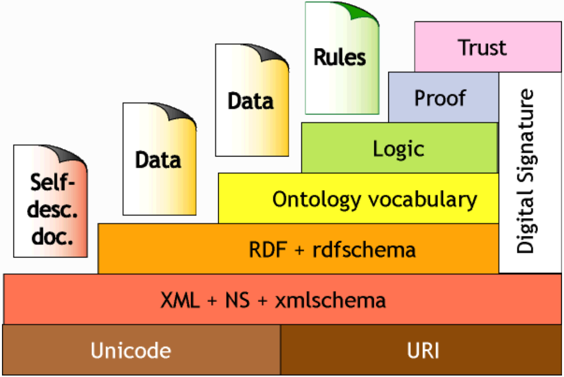
What do ontologies talk about?

- We usually have:
  - Symbols that represent individuals (John, sesame seed)
  - Symbols that represent classes (Groups of individuals like people with tree nut allgergy; all people with seed allergy)
  - Relationships between instances/groups and instances/groups (people with nut allergy are also allergic to seeds).

BASIC IDEA OF SEMANTIC WEB

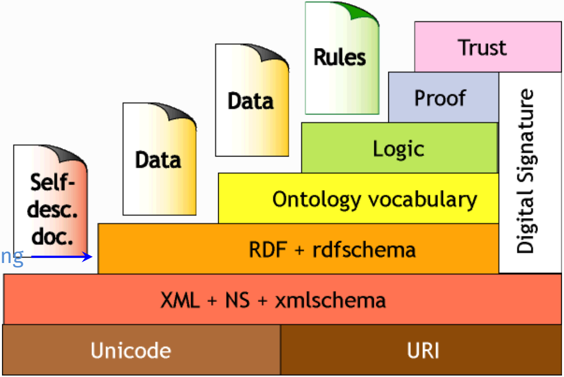


Copyright Pascal Hitzler; reused under the terms of the Creative Commons CC-BY license.


Rules
Trust
Data
Proof
Data
Logic
Self-desc. doc.
Ontology vocabulary
Digital Signature
RDF + rdfschema
XML + NS + xmlschema
Unicode
URI

KNOWLEDGE REPRESENTATION ON THE SEMANTIC WEB (2005 VISION)

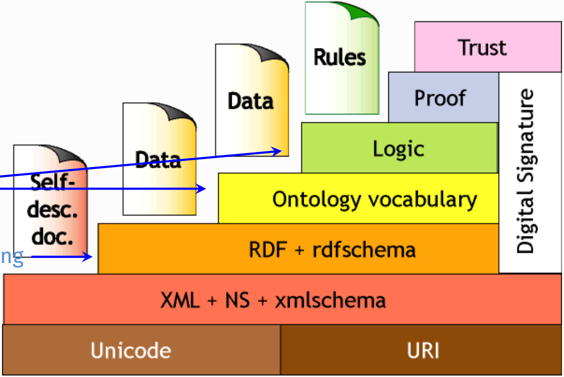
data exchange/simple reasoning
(this lecture)



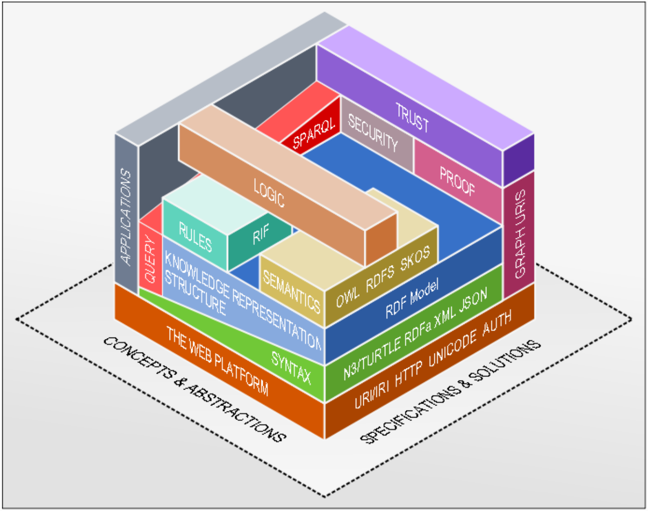
KNOWLEDGE REPRESENTATION ON THE SEMANTIC WEB (2005 VISION)

advanced reasoning
(next lectures)

data exchange/simple reasoning
(this lecture)



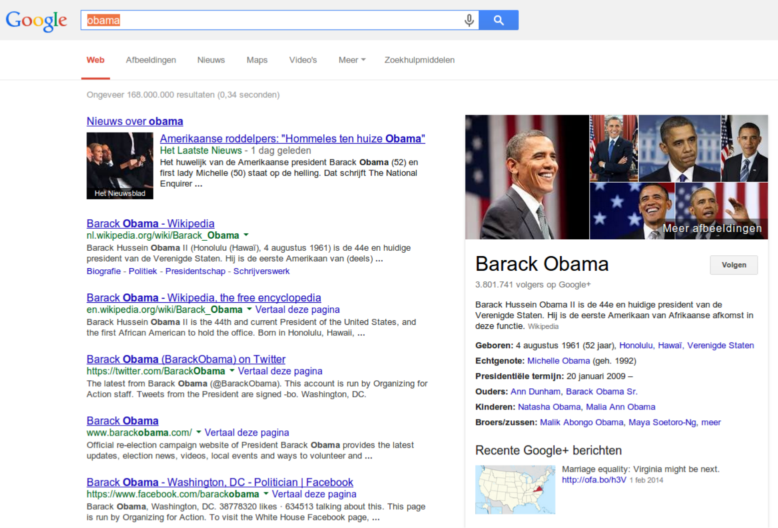
KNOWLEDGE REPRESENTATION ON THE SEMANTIC WEB (2013)



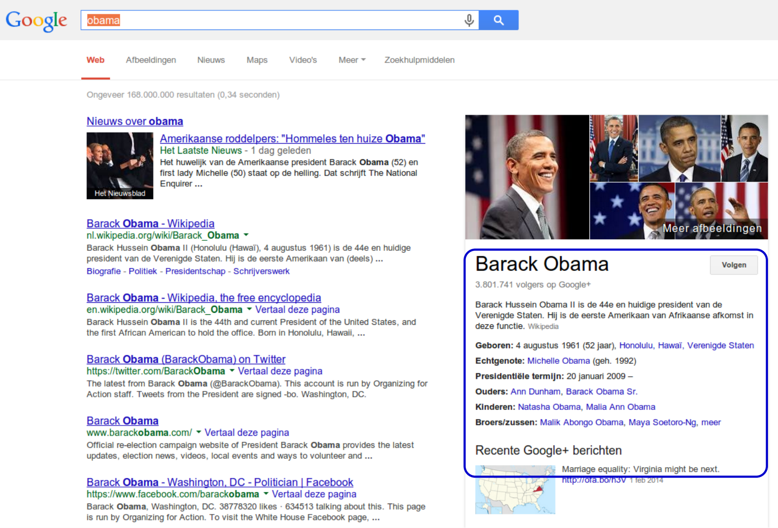
BEFORE WE DIVE IN ...

- The original vision of the Semantic Web was too futuristic and created the wrong expectations; attaining this vision would require solving AI-complete problems.
- But the research and development that was spurred after the publication of the Semantic Web article has gone a significant way towards realizing the vision of machine-treatable data through the Linking Open Data initiative.
- Linked Data is a more principled way of publishing data on the web, linking it to other data (without the sci-fi stuff).

THE CURRENT WEB: WITH STRUCTURED DATA

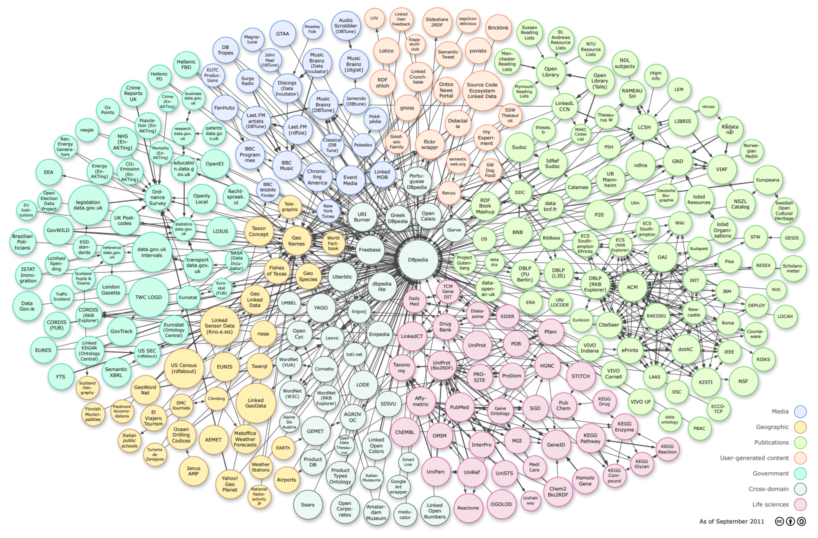


THE CURRENT WEB: WITH STRUCTURED DATA



Structured information, based on Linked Data

Many datasets and ontologies are openly made available and interlinked as RDF data



Part II: The RDF Data Model

Disclaimer

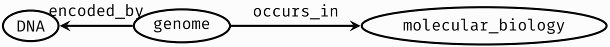
Some of the following examples have been copied from:

- The W3C RDF Primer at `http://www.w3.org/TR/rdf-primer/`
- Joshua Tauberer RDF intro at `http://rdfabout.com/intro/`
- The book Foundations of Semantic Web Technologies (Hitzler, Krötsch, Rudolph).

THE RDF GRAPH

- The Resource Description Framework (RDF) is a general method for describing knowledge in the form of a directed, labeled graph.
- Nodes represent the “entities” or “resources” we are describing; or literal values;
- Edges relate resources to other resources, or to literal values.

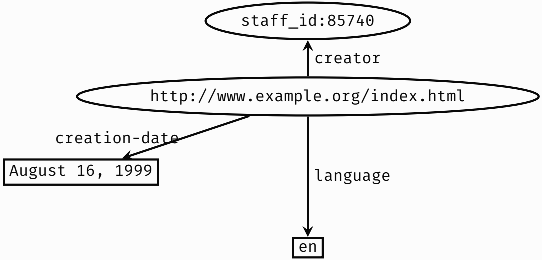
Example (not valid RDF):



THE RDF GRAPH

- The Resource Description Framework (RDF) is a general method for describing knowledge in the form of a directed, labeled graph.
- Nodes represent the “entities” or “resources” we are describing; or literal values;
- Edges relate resources to other resources, or to literal values.

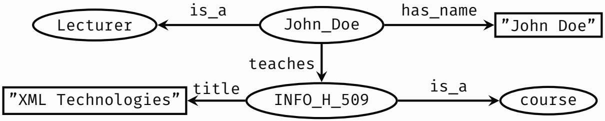
Example (not valid RDF):



THE RDF GRAPH

- The Resource Description Framework (RDF) is a general method for describing knowledge in the form of a directed, labeled graph.
- Nodes represent the "entities" or "resources" we are describing; or literal values;
- Edges relate resources to other resources, or to literal values.

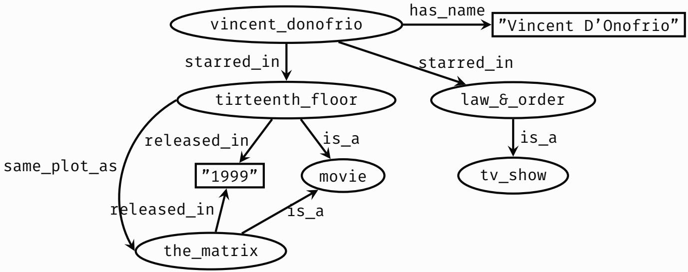
Example (not valid RDF):



THE RDF GRAPH

- The Resource Description Framework (RDF) is a general method for describing knowledge in the form of a directed, labeled graph.
- Nodes represent the “entities” or “resources” we are describing; or literal values;
- Edges relate resources to other resources, or to literal values.

Example (not valid RDF):

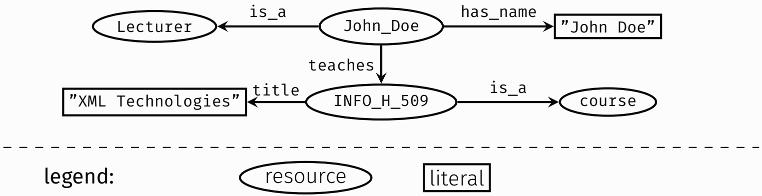


THE RDF GRAPH: RESOURCES AND LITERALS

There are two kinds of nodes:

- Nodes that represent the 'things' we are describing: these are called resources or entities
- Nodes that represent particular values of a property (like the string `John Doe`, the integer `1999`, the data `August 16, 1999` and so on): these are called literals or literal nodes
- Literals can only have incoming edges, whereas resources can have both incoming and outgoing edges

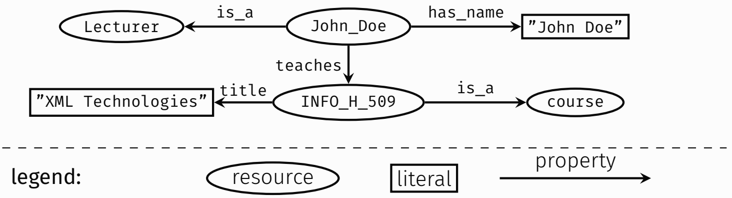
Example (not valid RDF):



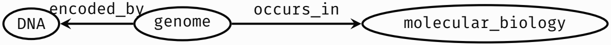
THE RDF GRAPH: PREDICATES AND PROPERTIES

The edge labels are called predicates or properties

Example (not valid RDF):



Consider again our genome information



The term "genome" can mean different things:

- Genome — the totality of genetic material carried by an organism
- Genome (novel) — science fiction novel by Sergey Lukyanenko
- Genome — a superior humanoid race in Square's console role-playing game Final Fantasy IX
- ...

In order to unambigously refer to an entity, it needs a a unique name

THE RDF GRAPH: UNIQUE IDENTIFICATION

Question:

How do we unabmiguously name things in a way that allows everyone to invent their own names without centralized agreement?

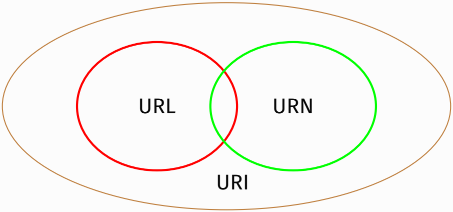
THE RDF GRAPH: UNIQUE IDENTIFICATION

Question:

How do we unabmiguously name things in a way that allows everyone to invent their own names without centralized agreement?

Answer:

- Use Uniform Resource Identifiers!
- For instance, if I own the domain `http://www.example.org` I am free to create URLs starting with that address.



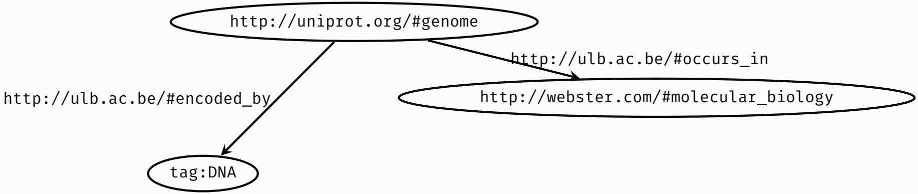
- URN example: `urn:isbn:0-486-27557-4`
- URL example: `http://www.example.org/staffid/85740`

THE RDF GRAPH: UNIQUE IDENTIFICATION

For this reason, RDF requires:

- every resource to be a URI
- every property to be a URI

Example (this time a valid RDF graph):



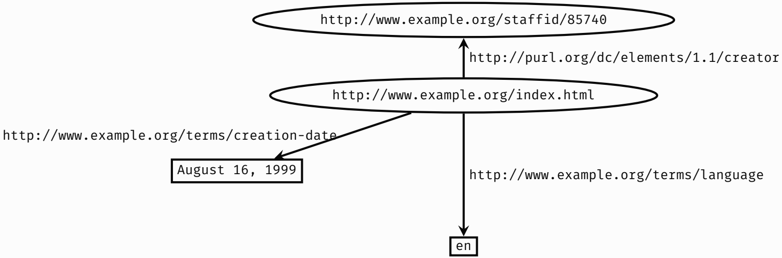
Note that these URIs are used to identify entities and properties. Even if they take the form of a URL, they need not be dereferencable: if you type them in the location bar of a browser you may get an error!

THE RDF GRAPH: UNIQUE IDENTIFICATION

For this reason, RDF requires:

- every resource to be a URI
- every property to be a URI

Another Example (also a valid RDF graph):

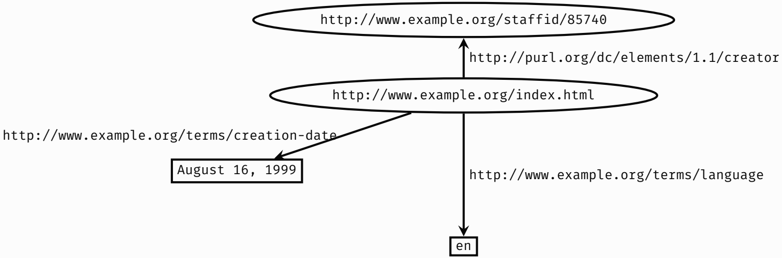


THE RDF GRAPH: UNIQUE IDENTIFICATION

For this reason, RDF requires:

- every resource to be a URI
- every property to be a URI

Another Example (also a valid RDF graph):

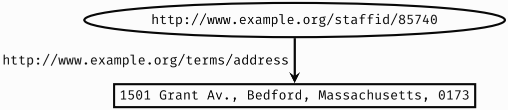


In essence: similar to namespaces in XML

THE RDF GRAPH: N-ARY RELATIONS

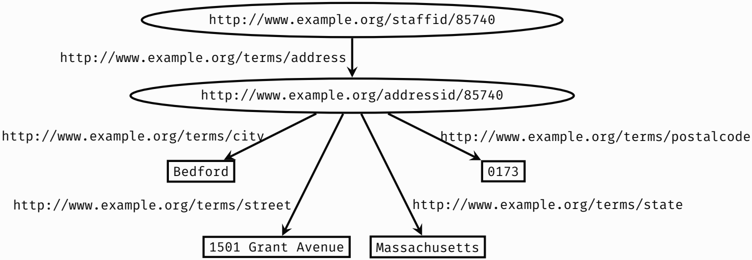
- In RDF, the only relationships that we can make is between two URIs (resources), or between a URI and a literal.
- To express *n*-ary relationships, we need to create extra resources.

Example: street address as a literal value



- In RDF, the only relationships that we can make is between two URIs (resources), or between a URI and a literal.
- To express *n*-ary relationships, we need to create extra resources.

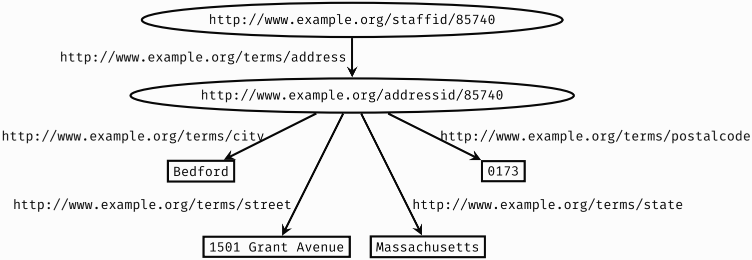
Example: street address as its own resource (n-ary relation)



THE RDF GRAPH: BLANK NODES

- Sometimes inventing a new URI for every resource becomes tedious

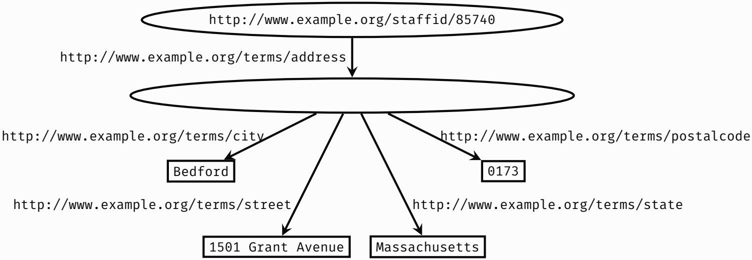
Example: street address



THE RDF GRAPH: BLANK NODES

- Sometimes inventing a new URI for every resource becomes tedious
- RDF also allows for nodes without URIs: these are called blank nodes or anonymous resources
- A blank node hence denotes an entity for which we do not have or do not want to create a globally unique name

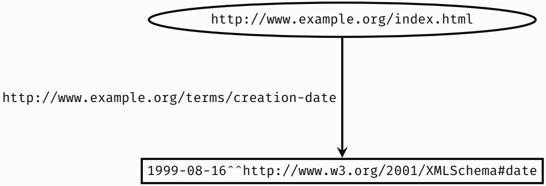
Example: street address with blank node



THE RDF GRAPH: TYPED LITERALS

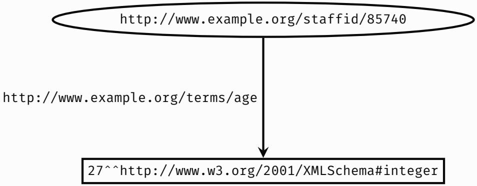
- Without further information, it is often difficult to discern how a literal should be interpreted: is it an integer, a string, a date, ...?
- An RDF Typed Literal is formed by pairing a literal with a URI that identifies a particular datatype.
- RDF itself does not specify datatypes, but one can use for example the XML Schema datatypes
- Some XML Schema datatypes, like `xsd:duration` and `xsd:QName` cannot be used in RDF

Example:



- Without further information, it is often difficult to discern how a literal should be interpreted: is it an integer, a string, a date, ...?
- An RDF Typed Literal is formed by pairing a literal with a URI that identifies a particular datatype.
- RDF itself does not specify datatypes, but one can use for example the XML Schema datatypes
- Some XML Schema datatypes, like `xsd:duration` and `xsd:QName` cannot be used in RDF

Example:



THE RDF GRAPH: THE “IS A” RELATIONSHIP

- To express statements of the form

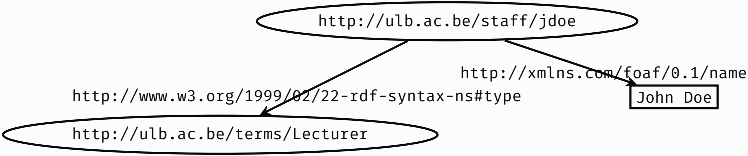
  ”Resource *A* is an instance of resource *B*”

  like “John is a lecturer” or “Law & Order is a TV show”, RDF has reserved the following URI:

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

- We will see later that, in combination with RDF Schema or OWL, this allows us to infer new statements not explicitly present in the RDF Graph

Example:



THE RDF GRAPH: CONTAINERS

Example: Course 509 has the students Amy, Mohamed, and John

THE RDF GRAPH: CONTAINERS

There is often a need to describe groups of things.

- RDF Defines three predefined types to denote containers:
  - Unordered groups with duplicates (a Bag)

    `http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag`
  - Ordered groups with duplicates (a Sequence)

    `http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq`
  - A group of alternatives (an Alternative)

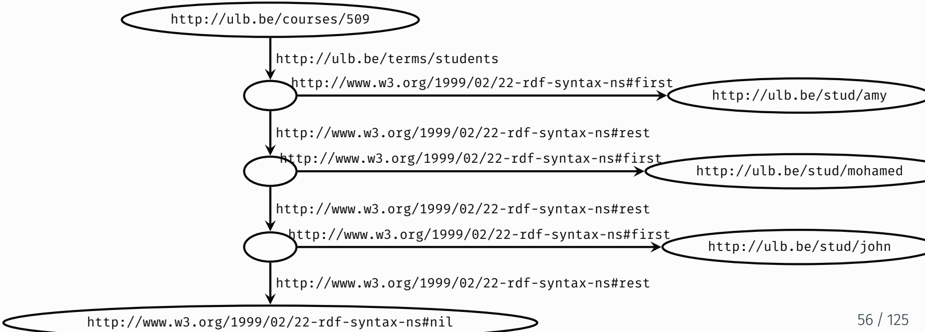
    `http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt`
- The *n*-th item in a collection is specified by the property

  `http://www.w3.org/1999/02/22-rdf-syntax-ns#_`*n*

THE RDF GRAPH: COLLECTIONS

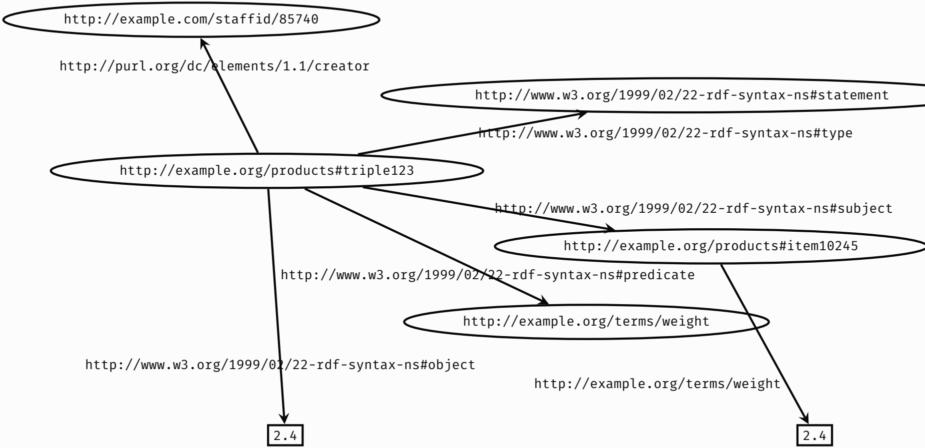
- Containers cannot be “closed”
- To specify “closed” groups RDF introduces collections, which are list structures formed by:
  - `http://www.w3.org/1999/02/22-rdf-syntax-ns#first`
  - `http://www.w3.org/1999/02/22-rdf-syntax-ns#rest`
  - `http://www.w3.org/1999/02/22-rdf-syntax-ns#nil`

Example: Course 509 has exactly the students Amy, Mohamed, and John



REIFICATION: MAKING STATEMENTS ABOUT STATEMENTS

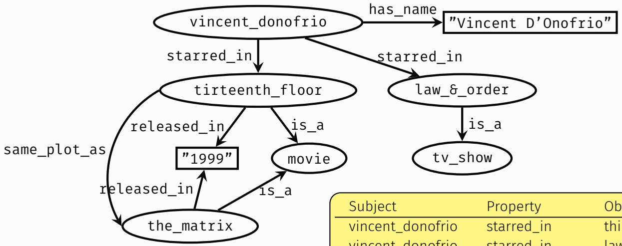
Example: staf 85740 created the edge “item10245 has weight 2.4”



- To make statements about statements RDF provides the type
  - `http://www.w3.org/1999/02/22-rdf-syntax-ns#statement`

  and the properties
  - `http://www.w3.org/1999/02/22-rdf-syntax-ns#subject`
  - `http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate`
  - `http://www.w3.org/1999/02/22-rdf-syntax-ns#object`

AN RDF GRAPH ≡ SET OF TRIPLES

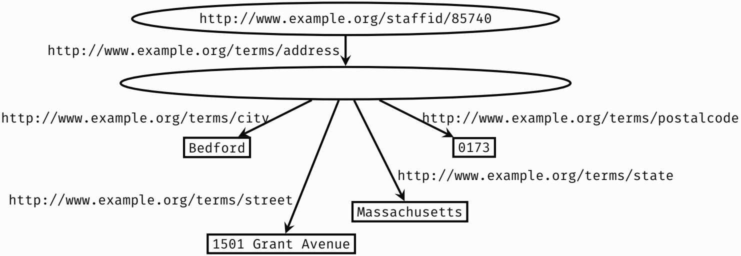


Subject	Property	Object
vincent_donofrio	starred_in	thirteenth_floor
vincent_donofrio	starred_in	law_&_order
vincent_donofrio	has_name	"Vincent D'Onofrio"
thirteenth_floor	released_in	"1999"
thirteenth_floor	is_a	movie
the_matrix	is_a	movie
the_matrix	released_in	"1999"
thirteenth_floor	same_plot_as	the_matrix
law_&_order	is_a	tv_show

- A triple is also called a statement
- To store a (piece of) an RDF graph, it suffices to store its set of triples
- This can be done in multiple serialization formats

Part III: RDF Serialization Formats

NTRIPLES: EXAMPLE




```
<http://www.example.org/staffid/85740>   <http://www.example.org/terms/address>      _:addr               .
_:addr                                   <http://www.example.org/terms/city>         ”Bedford”            .
_:addr                                   <http://www.example.org/terms/street>       ”1501 Grant Avenue”  .
_:addr                                   <http://www.example.org/terms/state>        ”Massachusetts”      .
_:addr                                   <http://www.example.org/terms/postalcode>   ”0713”               .
```

NTRIPLES: DEFINTION

NTriples is a simple but verbose serialization format

- Each line represents a single statement containing a subject, predicate, and an object, terminated by a dot
- URIs are enclosed in angle brackets <…>
- Anonymous nodes are represented as `_:name`. This `name` is only valid within the NTriples file.
- Literals are enclosed between quotes **"…"**, optionally followed by ^^ and a datatype

TURTLE (1/2)

Turtle extends the NTriples format with some convenient features:

- the `@prefix` directive allows abbreviation of URIs
- the `a` property abbreviates `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`
- XML Schema datatypes may be abreviated using the `xsd:` prefix
- The comma symbol can be used to repeat the subject and predicate of triples that only differ in the object
- The semicolon symbol can be used to repeat the subject of triples that only differ in the predicate and object v

Example:

```
@prefix staff: <http://www.example.org/staffid/> .
@prefix terms: <http://www.example.org/terms/> .

staff:85740    terms:address       _:addr
_:addr         terms:city          "Bedford"^^xsd:string     .
_:addr         terms:street        "1501 Grant Avenue"       .
_:addr         terms:state         "Massachusetts"           .
_:addr         terms:postalcode    "0713"                    .
staff:85740    a                   terms:employee            .
```

TURTLE (1/2)

Turtle extends the NTriples format with some convenient features:

- the `@prefix` directive allows abbreviation of URIs
- the `a` property abbreviates `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`
- XML Schema datatypes may be abreviated using the `xsd:` prefix
- The comma symbol can be used to repeat the subject and predicate of triples that only differ in the object
- The semicolon symbol can be used to repeat the subject of triples that only differ in the predicate and object v

Example:

```
@prefix staff: <http://www.example.org/staffid/> .
@prefix terms: <http://www.example.org/terms/> .

staff:85740   terms:address      _:addr
_:addr        terms:city         "Bedford"^^xsd:string   ;
              terms:street       "1501 Grant Avenue"     ;
              terms:state        "Massachusetts"         ;
              terms:postalcode   "0713"                  .
staff:85740   a                  terms:employee          .
```

TURTLE (1/2)

Turtle extends the NTriples format with some convenient features:

- the `@prefix` directive allows abbreviation of URIs
- the `a` property abbreviates `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`
- XML Schema datatypes may be abreviated using the `xsd:` prefix
- The comma symbol can be used to repeat the subject and predicate of triples that only differ in the object
- The semicolon symbol can be used to repeat the subject of triples that only differ in the predicate and object v

Example:

```
@prefix staff: <http://www.example.org/staff id/> .
@prefix : <http://www.example.org/terms/> .

staff:85740    :address      _:addr
_:addr         :city         "Bedford"^^xsd:string    ;
               :street       "1501 Grant Avenue"      ;
               :state        "Massachusetts"          ;
               :postalcode   "0713"                   .
staff:85740    a             :employee                .
```

TURTLE (2/2)

Turtle extends the NTriples format with some convenient features:

- Blank nodes can be described by nesting Turtle statements in [ ]
- Collections can be described by resources between parenthesis (...)

Example:

```
@prefix staff:  <http://www.example.org/staff id/> .
@prefix :  <http://www.example.org/terms/> .

staff:85740     :address        _:addr
_:addr          :city           "Bedford"^^xsd:string       ;
                :street         "1501 Grant Avenue"         ;
                :state          "Massachusetts"             ;
                :postalcode     "0713"                      .
staff:85740     a               :employee                   .
```

TURTLE (2/2)

Turtle extends the NTriples format with some convenient features:

- Blank nodes can be described by nesting Turtle statements in [ ]
- Collections can be described by resources between parenthesis (...)

Example:

```
@prefix staff: <http://www.example.org/staff id/> .
@prefix : <http://www.example.org/terms/> .

staff:85740   :address   [ :city         "Bedford"^^xsd:string ;
                           :street       "1501 Grant Avenue" ;
                           :state        "Massachusetts" ;
                           :postalcode   "0713"                 ] .
staff:85740   a          :employee                               .
```

TURTLE (2/2)

Turtle extends the NTriples format with some convenient features:

- Blank nodes can be described by nesting Turtle statements in [ ]
- Collections can be described by resources between parenthesis (...)

Example:

```
@prefix courses: <http://ulb.be/courses/> .
@prefix terms: <http://ulb.be/terms/> .
@prefix : <http://ulb.be/students/> .

courses:509    terms:students    _:a         .
_:a            rdf:first         :amy        .
_:a            rdf:rest          _:b         .
_:b            rdf:first         :mohamed    .
_:b            rdf:rest          _:c         .
_:b            rdf:first         :john       .
_:b            rdf:rest          rdf:nil     .
```

TURTLE (2/2)

Turtle extends the NTriples format with some convenient features:

- Blank nodes can be described by nesting Turtle statements in [ ]
- Collections can be described by resources between parenthesis (...)

Example:

```
@prefix courses: <http://ulb.be/courses/> .
@prefix terms: <http://ulb.be/terms/> .
@prefix : <http://ulb.be/students/> .

courses:509     terms:students     ( :amy :mohamed :john ) .
```

NOTATION 3 (N3)

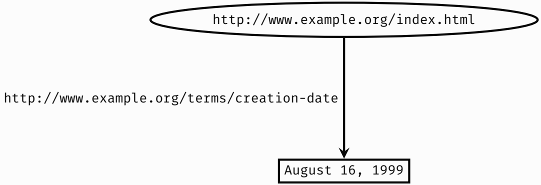
Notation 3 (N3) is an extension of Turtle

- Allows the specification of blank nodes without making up temporary identifiers of the form `_:name` for them
- It allows the definition of reasoning rules …
- …but those rules are not part of the RDF data model!

RDF/XML

RDF/XML is a description of RDF Triples and Graphs in XML

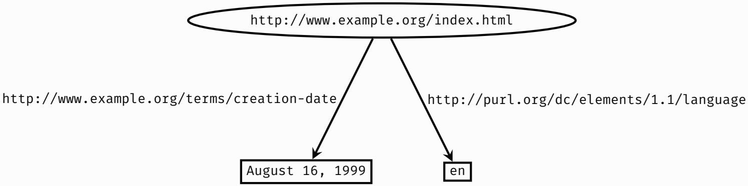
Example:




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:exterms="http://www.example.org/terms/">

  <rdf:Description rdf:about="http://www.example.org/index.html">
    <exterms:creation-date>August 16, 1999</exterms:creation-date>
  </rdf:Description>

</rdf:RDF>
```

RDF/XML

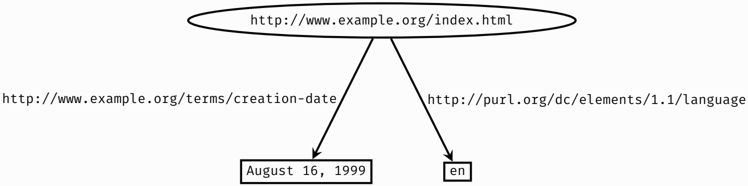
Example of multiple statements :




```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:exterms="http://www.example.org/terms/">

  <rdf:Description rdf:about="http://www.example.org/index.html">
    <exterms:creation-date>August 16, 1999</exterms:creation-date>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.example.org/index.html">
    <dc:language>en</dc:language>
  </rdf:Description>

</rdf:RDF>
```

RDF/XML

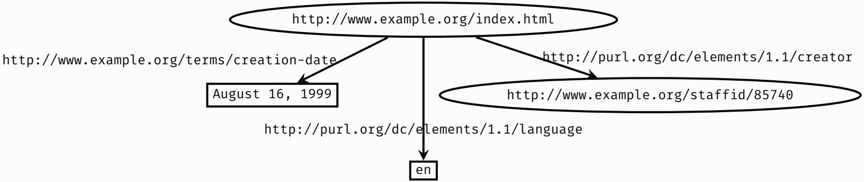
Example of multiple statements with abbreviation:




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:exterms="http://www.example.org/terms/">

  <rdf:Description rdf:about="http://www.example.org/index.html">
    <exterms:creation-date>August 16, 1999</exterms:creation-date>
    <dc:language>en</dc:language>
  </rdf:Description>

</rdf:RDF>
```

RDF/XML

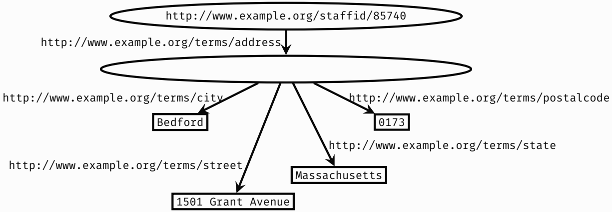
Example of a statement with a resource as object:




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:exterms="http://www.example.org/terms/">

  <rdf:Description rdf:about="http://www.example.org/index.html">
    <exterms:creation-date>August 16, 1999</exterms:creation-date>
    <dc:language>en</dc:language>
    <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
  </rdf:Description>

</rdf:RDF>
```

RDF/XML

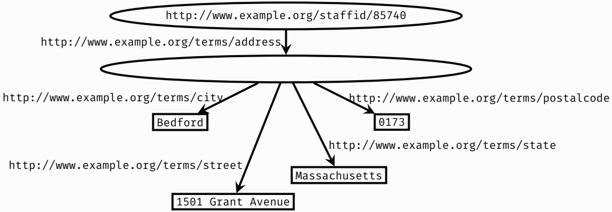
Example of a blank node:




```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:exterms="http://www.example.org/terms/">

  <rdf:Description rdf:about="http://www.example.org/staffid/85740">
    <exterms:addresss rdf:nodeID="abc"/>
  </rdf:Description>

  <rdf:Description rdf:nodeID="abc">
    <exterms:city>Bedford</exterms:city>
    <exterms:street>1501 Grant Avenue</exterms:street>
    <exterms:state>Massachusetts</exterms:state>
    <exterms:postalcode>0173</exterms:postalcode>
  </rdf:Description>
</rdf:RDF>
```

RDF/XML

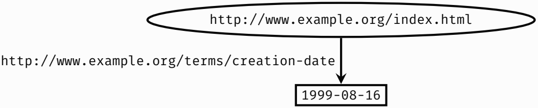
Example of a blank node (alternate syntax):




```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:exterms="http://www.example.org/terms/">

  <rdf:Description rdf:about="http://www.example.org/staffid/85740">
    <exterms:addresss rdf:parseType="Resource">
      <exterms:city>Bedford</exterms:city>
      <exterms:street>1501 Grant Avenue</exterms:street>
      <exterms:state>Massachusetts</exterms:state>
      <exterms:postalcode>0173</exterms:postalcode>
    </exterms:addresss>
  </rdf:Description>
</rdf:RDF>
```

RDF/XML

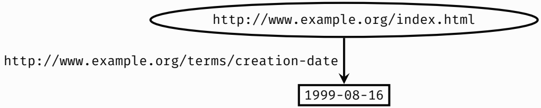
Example of a typed literal:




```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:exterms="http://www.example.org/terms/">

  <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>

</rdf:RDF>
```

RDF/XML

Example of a typed literal:

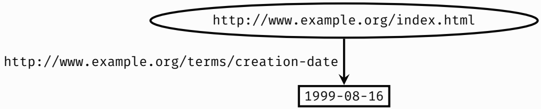



```
<rdf:RDF xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
         xmlns:exterms=”http://www.example.org/terms/”>

  <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>

</rdf:RDF>
```

Note that URIs that occur in attributes must be written in full!

RDF/XML

Example of using entities to abbreviate URIs:




```
<?xml version=”1.0”?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd ”http://www.w3.org/2001/XMLSchema#”>]>
<rdf:RDF xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
         xmlns:exterms=”http://www.example.org/terms/”>

  <rdf:Description rdf:about=”http://www.example.org/index.html”>
    <exterms:creation-date rdf:datatype=”&xsd;date”>
      1999-08-16
    </exterms:creation-date>
  </rdf:Description>

</rdf:RDF>
```

Example of container - first possiblity:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">

  <rdf:Description rdf:about="http://example.org/courses/6.001">
    <s:students>
      <rdf:Bag>
        <rdf:_1 rdf:resource="http://example.org/students/Amy"/>
        <rdf:_2 rdf:resource="http://example.org/students/Mohamed"/>
        <rdf:_3 rdf:resource="http://example.org/students/Johann"/>
        <rdf:_4 rdf:resource="http://example.org/students/Maria"/>
        <rdf:_5 rdf:resource="http://example.org/students/Phuong"/>
      </rdf:Bag>
    </s:students>
  </rdf:Description>
</rdf:RDF>
```

RDF/XML

Example of container - second possiblity:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">

  <rdf:Description rdf:about="http://example.org/courses/6.001">
    <s:students>
      <rdf:Bag>
        <rdf:li rdf:resource="http://example.org/students/Amy"/>
        <rdf:li rdf:resource="http://example.org/students/Mohamed"/>
        <rdf:li rdf:resource="http://example.org/students/Johann"/>
        <rdf:li rdf:resource="http://example.org/students/Maria"/>
        <rdf:li rdf:resource="http://example.org/students/Phuong"/>
      </rdf:Bag>
    </s:students>
  </rdf:Description>
</rdf:RDF>
```

RDF/XML

XML/RDF has many other useful abbreviations

- For collections
- For reification
- …

See the handouts available on website!

JSON

RDF graphs can also be serialized as JSON objects.

- Recent standard, outside scope of this class.
- See `http://www.w3.org/TR/json-ld/` if you are interested.

Part IV: RDF Schema

Genome

From Wikipedia, the free encyclopedia

For a non-technical introduction to the topic, see Introduction to genetics.

For other uses, see Genome (disambiguation).

In modern molecular biology, the **genome** is the entirety of an organism's hereditary information. It is encoded either in DNA or, for many types of virus, in RNA.

The genome includes both the genes and the non-coding sequences of the DNA.[1] The term was adapted in 1920 by Hans Winkler, Professor of Botany at the University of Hamburg, Germany. The Oxford English Dictionary suggests the name to be a portmanteau of the words ***gen**e* and *chromos**ome***. A few related *-ome* words already existed, such as *biome* and *rhizome*, forming a vocabulary into which *genome* fits systematically.[2]

- Natural language
- No structure
- Difficult to process automatically

Recent past – no structure

In modern molecular biology, the **genome** is the entirety of an organism's hereditary information. It is encoded either in DNA or, for many types of virus, in RNA.

The genome includes both the genes and the non-coding sequences of the DNA.[1] The term was adapted in 1920 by Hans Winkler, Professor of Botany at the University of Hamburg, Germany. The Oxford English Dictionary suggests the name to be a portmanteau of the words ***gene*** and *chromos**ome***. A few related *-ome* words already existed, such as *biome* and *rhizome*, forming a vocabulary into which *genome* fits systematically.[2]

Current/Future structured by RDF (subject, predicate, object)

b:genome	b:field	b:molecular-bio	.
b:DNA	b:encode	b:genes	.
b:DNA	b:encode	b:non-coding-seq	.
b:genome	b:include	b:non-coding-seq	.
b:genome	b:include	b:gene	.
b:genome	b:related-to	b:rhizome	.

- RDF asserts knowledge (statements) about entities (resources)
- By convention is clear what the subject, predicate, and object are
- Easier to process automatically, but a computer still does not know their meaning ...
- How do we add some semantics to the statements?

WHAT DO YOU MEAN: SEMANTICS?

Input

```
@prefix prod: <http://www.example.org/products/> .
@prefix terms: <http://www.example.org/terms/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
prod:cam1    rdf:type      terms:digital-camera    .
prod:cam1    terms:price   150                     .
prod:nb1     rdf:type      terms:netbook           .
prod:nb1     terms:price   300                     .
prod:book1   rdf:type      terms:book              .
prod:book1   terms:price   2.50                    .
```

How do we find all products that are digital devices?

WHAT DO YOU MEAN: SEMANTICS?

Input

```
@prefix prod: <http://www.example.org/products/> .
@prefix terms: <http://www.example.org/terms/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
prod:cam1    rdf:type      terms:digital-camera   .
prod:cam1    terms:price   150                    .
prod:nb1     rdf:type      terms:netbook          .
prod:nb1     terms:price   300                    .
prod:book1   rdf:type      terms:book             .
prod:book1   terms:price   2.50                   .
```

How do we find all products that are digital devices?

Hmm, digital cameras are digital devices

select all x such that
x `rdf:type terms:digital-camera` .

Input

```
@prefix prod: <http://www.example.org/products/> .
@prefix terms: <http://www.example.org/terms/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
prod:cam1    rdf:type      terms:digital-camera   .
prod:cam1    terms:price   150                    .
prod:nb1     rdf:type      terms:netbook          .
prod:nb1     terms:price   300                    .
prod:book1   rdf:type      terms:book             .
prod:book1   terms:price   2.50                   .
```

How do we find all products that are digital devices?

Hmm, digital cameras are digital devices
… so are netbooks

```
select all x such that
  x rdf:type terms:digital-camera .
OR
  x rdf:type terms:netbook .
```

WHAT DO YOU MEAN: SEMANTICS?

Input

```
@prefix prod: <http://www.example.org/products/> .
@prefix terms: <http://www.example.org/terms/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
prod:cam1    rdf:type      terms:digital-camera   .
prod:cam1    terms:price   150                    .
prod:nb1     rdf:type      terms:netbook          .
prod:nb1     terms:price   300                    .
prod:book1   rdf:type      terms:book             .
prod:book1   terms:price   2.50                   .
```

- The computer has no “knowledge of the world” stating that cameras and netbooks are digital devices
- So we have to manually encode this “knowledge of the world” in the query
- This solution is inadequate: error-prone and difficult to maintain
- It would be better if we could tell the computer our “knowledge of the world” and let him do the reasoning!

Explicitly Asserted Knowledge

I am a man
John is a man
Jane is a woman

Explicitly Asserted Knowledge

I am a man
John is a man
Jane is a woman

Knowledge About the World
(Ontology)

Every man is human
Every woman is human

REASONING BY MEANS OF INFERENCE: THE GENERAL IDEA

Explicitly Asserted Knowledge

I am a man
John is a man
Jane is a woman

Knowledge About the World (Ontology)

Every man is human
Every woman is human

Inference

I am a man
John is a man
Jane is a woman
I am human
John is human
Jane is human

Enriched Knowledge

“Knowledge about the world” for our example:

- Every camera is a digital device
- Every netbook is a digital device
- Every computer is a digital device
- Every book is human-readable

Such knowledge is set-based (also called class-based)

- The set of cameras is a subset of the set of digital devices
- The set of netbooks is a subset of the set of digital devices
- The set of books is a subset of the set of human-readable objects

Ontologies

Ontologies provide formal specifications of the classes of objects that inhabit “the world”, the relationships between individual and classes, and their properties.

REASONING BY MEANS OF INFERENCE: THE GENERAL IDEA (2)

Explicitly Asserted Knowledge

prod:cam1	rdf:type	terms:digital-camera
prod:cam1	terms:price	150
prod:nb1	rdf:type	terms:netbook
prod:nb1	terms:price	300
prod:book1	rdf:type	terms:book
prod:book1	terms:price	2.50

REASONING BY MEANS OF INFERENCE: THE GENERAL IDEA (2)

Explicitly Asserted Knowledge

prod:cam1	rdf:type	terms:digital-camera
prod:cam1	terms:price	150
prod:nb1	rdf:type	terms:netbook
prod:nb1	terms:price	300
prod:book1	rdf:type	terms:book
prod:book1	terms:price	2.50

Knowledge About the World (Ontology)

Every digital camera is a digital device

Every netbook is a digital device

REASONING BY MEANS OF INFERENCE: THE GENERAL IDEA (2)

Explicitly Asserted Knowledge

prod:cam1	rdf:type	terms:digital-camera
prod:cam1	terms:price	150
prod:nb1	rdf:type	terms:netbook
prod:nb1	terms:price	300
prod:book1	rdf:type	terms:book
prod:book1	terms:price	2.50

Knowledge About the World (Ontology)

Every digital camera is a digital device

Every netbook is a digital device

Inference

prod:cam1	rdf:type	terms:digital-camera
prod:cam1	terms:price	150
prod:nb1	rdf:type	terms:netbook
prod:nb1	terms:price	300
prod:book1	rdf:type	terms:book
prod:book1	terms:price	2.50
prod:cam1	rdf:type	terms:digital-device
prod:nb1	rdf:type	terms:digital-device

Enriched Knowledge

REASONING BY MEANS OF INFERENCE: THE GENERAL IDEA (2)

Explicitly Asserted Knowledge

prod:cam1	rdf:type	terms:digital-camera
prod:cam1	terms:price	150
prod:nb1	rdf:type	terms:netbook
prod:nb1	terms:price	300
prod:book1	rdf:type	terms:book
prod:book1	terms:price	2.50

Knowledge About the World (Ontology)

Every digital camera is a digital device

Every netbook is a digital device

Inference

prod:cam1	rdf:type	terms:digital-camera
prod:cam1	terms:price	150
prod:nb1	rdf:type	terms:netbook
prod:nb1	terms:price	300
prod:book1	rdf:type	terms:book
prod:book1	terms:price	2.50
prod:cam1	rdf:type	terms:digital-device
prod:nb1	rdf:type	terms:digital-device

Query → ...

Enriched Knowledge

How do we specify ontologies?

- RDF Schema and the Ontology Web Language (OWL) allow ontologies to be specified in RDF itself: they provide resources and properties in a standard namespace to talk about class relationships, properties of classes, …
- They also specify how these resources and properties should be interpreted (in terms of inference)
- OWL allows more fine-grained knowledge of the world to be specified, at the cost of less efficient inference and reasoning

Example:

```
@prefix terms: <http://www.example.org/terms/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

terms:digital-camera       rdfs:subClassOf    terms:digital-device .
terms:netbook              rdfs:subClassOf    terms:digital-device .
terms:book                 rdfs:subClassOf    terms:media .
terms:digital-device       rdfs:subClassOf    terms:electronic-device .
terms:electronic-device    rdfs:subClassOf    terms:device .
```

Example Ontology in RDF Schema:

```
@prefix terms: <http://www.example.org/terms/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

terms:digital-camera   rdfs:subClassOf   terms:digital-device .
terms:netbook          rdfs:subClassOf   terms:digital-device .
terms:book             rdfs:subClassOf   terms:media .
```

Starting from the following asserted triples ...

```
@prefix prod: <http://www.example.org/products/> .
@prefix terms: <http://www.example.org/terms/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
prod:cam1    rdf:type      terms:digital-camera   .
prod:cam1    terms:price   150                    .
prod:nb1     rdf:type      terms:netbook          .
prod:nb1     terms:price   300                    .
prod:book1   rdf:type      terms:book             .
prod:book1   terms:price   2.50                   .
```

Example Ontology in RDF Schema:

```
@prefix terms: <http://www.example.org/terms/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

terms:digital-camera   rdfs:subClassOf   terms:digital-device .
terms:netbook          rdfs:subClassOf   terms:digital-device .
terms:book             rdfs:subClassOf   terms:media .
```

...an RDF Schema-aware processor will infer the following triples (in purple)

```
@prefix prod: <http://www.example.org/products/> .
@prefix terms: <http://www.example.org/terms/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
prod:cam1    rdf:type      terms:digital-camera    .
prod:cam1    terms:price   150                     .
prod:nb1     rdf:type      terms:netbook           .
prod:nb1     terms:price   300                     .
prod:book1   rdf:type      terms:book              .
prod:book1   terms:price   2.50                    .
prod:cam1    rdf:type      terms:digital-device    .
prod:nb1     rdf:type      terms:digital-device    .
prod:book1   rdf:type      terms:media             .
```

Example Ontology in RDF Schema:

```
@prefix terms: <http://www.example.org/terms/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

terms:digital-camera    rdfs:subClassOf    terms:digital-device .
terms:netbook           rdfs:subClassOf    terms:digital-device .
terms:book              rdfs:subClassOf    terms:media .
```

So finding all products that are digital devices becomes easy (assuming our SPARQL engine is RDFS-aware):

```
PREFIX prod: <http://www.example.org/products/>
PREFIX terms: <http://www.example.org/terms/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?prod
WHERE {
    ?prod rdf:type terms:digital-device .
}
```

In what follows:

- The prefix `rdf` abbreviates the RDF namespace
  `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
- The prefix `rdfs` abbreviates the RDFS namespace
  `http://www.w3.org/2000/01/rdf-schema#`
- We range over arbitrary URIs by a and b (i.e., anything admissible for the predicate position of a triple)
- u and v refer to arbitrary URIs or blank node IDs by (i.e., anything admissible for the subject position of a triple)
- x and y can be used for arbitrary URIs, blank node IDs or literals

CLASSES AND CLASS HIERARCHIES

- Classes stand for sets of things (in RDF: sets of URIs)
- Use `rdf:type` to indicate class membership
- A URI can belong to several classes.

Example

:mary	rdf:type	:Woman	.
:mary	rdf:type	:Student	.
			.

- Classes stand for sets of things (in RDF: sets of URIs)
- Use `rdf:type` to indicate class membership
- A URI can belong to several classes.
- `X rdfs:subClassOf Y` is used to specify that all members of a class *X* are also members of a class *Y*
- This allows to arrange classes in hierarchies.
- And allows expressing that two classes are the same by mutual inclusion: C_1 subClassOf C_2 and C_2 subClassOf C_1

Deduction Rule

If u `rdfs:subClassOf` x .
And y `rdf:type` u .
Then infer y `rdf:type` x .

Example

:mary	rdf:type	:Woman	.
:mary	rdf:type	:Student	.
:Woman	rdfs:subClassOf	:Person	.
:Person	rdfs:subClassOf	:Animal	.

- Classes stand for sets of things (in RDF: sets of URIs)
- Use `rdf:type` to indicate class membership
- A URI can belong to several classes.
- `X rdfs:subClassOf Y` is used to specify that all members of a class X are also members of a class Y
- This allows to arrange classes in hierarchies.
- And allows expressing that two classes are the same by mutual inclusion: C_1 subClassOf C_2 and C_2 subClassOf C_1

Deduction Rule

If	u `rdfs:subClassOf` x .
And	y `rdf:type` u .
Then infer	y `rdf:type` x .

Example

:mary	rdf:type	:Woman	.
:mary	rdf:type	:Student	.
:Woman	rdfs:subClassOf	:Person	.
:Person	rdfs:subClassOf	:Animal	.
:mary	rdf:type	:Woman	.

CLASSES AND CLASS HIERARCHIES

- Classes stand for sets of things (in RDF: sets of URIs)
- Use `rdf:type` to indicate class membership
- A URI can belong to several classes.
- `X rdfs:subClassOf Y` is used to specify that all members of a class X are also members of a class Y
- This allows to arrange classes in hierarchies.
- And allows expressing that two classes are the same by mutual inclusion: C_1 subClassOf C_2 and C_2 subClassOf C_1

Deduction Rule

If	u `rdfs:subClassOf` x .
And	y `rdf:type` u .
Then infer	y `rdf:type` x .

Example

:mary	rdf:type	:Woman	.
:mary	rdf:type	:Student	.
:Woman	rdfs:subClassOf	:Person	.
:Person	rdfs:subClassOf	:Animal	.
:mary	rdf:type	:Woman	.
:mary	rdf:type	:Animal	.

PROPERTIES AND PROPERTY HIERARCHIES

- Everything that occurs in the middle of a triple is a property.
- `X rdfs:subPropertyOf Y` is used to specify that all resources with property X to some object also have property Y to that object
- This allows to arrange properties also in hierarchies.
- And expressing that two properties are the same (by mutual inclusion).

Deduction Rule

If	a `rdfs:subPropertyOf` b .
And	$x\ a\ y$.
Then add	$x\ b\ y$.

Example

:john	:happilyMariedTo	:mary	.
:happilyMariedTo	rdfs:subPropertyOf	:marriedTo	.

- Everything that occurs in the middle of a triple is a property.
- `X rdfs:subPropertyOf Y` is used to specify that all resources with property *X* to some object also have property *Y* to that object
- This allows to arrange properties also in hierarchies.
- And expressing that two properties are the same (by mutual inclusion).

Deduction Rule

If	*a* `rdfs:subPropertyOf` *b* .
And	*x a y* .
Then add	*x b y*.

Example

:john	:happilyMariedTo	:mary	.
:happilyMariedTo	rdfs:subPropertyOf	:marriedTo	.
:john	:marriedTo	:mary	.

- Property restrictions allow expressing that a certain property can only be between things of a certain rdf:type.
- P `rdfs:domain` C is used to specify that all subjects with (outgoing) property *P* belong to class *C*
- P `rdfs:range` C is used to specify that all objects with (incoming) property *P* belong to class *C*

Deduction Rule

If	*a* `rdfs:domain` *x* .
And	*u a y* .
Then add	*u* `rdf:type` *x* .

If	*a* `rdfs:range` *x* .
And	*u a v* .
Then add	*v* `rdf:type` *x* .

Example

:john	:isMarriedTo	:mary	.
:isMarriedTo	rdfs:domain	:Person	.
:isMarriedTo	rdfs:range	:Person	.

PROPERTY RESTRICTION

- Property restrictions allow expressing that a certain property can only be between things of a certain rdf:type.
- P `rdfs:domain` C is used to specify that all subjects with (outgoing) property *P* belong to class *C*
- P `rdfs:range` C is used to specify that all objects with (incoming) property *P* belong to class *C*

Deduction Rule

If	*a* `rdfs:domain` *x* .
And	*u a y* .
Then add	*u* `rdf:type` *x* .

If	*a* `rdfs:range` *x* .
And	*u a v* .
Then add	*v* `rdf:type` *x* .

Example

:john	:isMarriedTo	:mary	.
:isMarriedTo	rdfs:domain	:Person	.
:isMarriedTo	rdfs:range	:Person	.
:john	rdf:type	:Person	.

PROPERTY RESTRICTION

- Property restrictions allow expressing that a certain property can only be between things of a certain rdf:type.
- `P rdfs:domain C` is used to specify that all subjects with (outgoing) property *P* belong to class *C*
- `P rdfs:range C` is used to specify that all objects with (incoming) property *P* belong to class *C*

Deduction Rule

If	*a* `rdfs:domain` *x* .
And	*u a y* .
Then add	*u* `rdf:type` *x* .

If	*a* `rdfs:range` *x* .
And	*u a v* .
Then add	*v* `rdf:type` *x* .

Example

:john	:isMarriedTo	:mary	.
:isMarriedTo	rdfs:domain	:Person	.
:isMarriedTo	rdfs:range	:Person	.
:john	rdf:type	:Person	.
:mary	rdf:type	:Person	.

Consider the following example. What new triples are inferred?

:MarriedWoman	rdfs:subClassOf	:Woman	.
:maidenName	rdfs:domain	:MarriedWoman	.
:Karen	:maidenName	"Stephens"	.

Consider the following example. What new triples are inferred?

:MarriedWoman	rdfs:subClassOf	:Woman	.
:maidenName	rdfs:domain	:MarriedWoman	.
:Karen	:maidenName	”Stephens”	.
:Karen	rdf:type	:MarriedWoman	.

Consider the following example. What new triples are inferred?

```
:MarriedWoman   rdfs:subClassOf   :Woman          .
:maidenName     rdfs:domain       :MarriedWoman   .

:Karen          :maidenName       "Stephens"      .
:Karen          rdf:type          :MarriedWoman   .
:Karen          rdf:type          :Woman          .
```

Beware of the following when modeling using RDFs:

- multiple property restrictions are allowed.

Example

```
ex:authorOf   rdfs:range   ex:TextBook   .
ex:authorOf   rdfs:range   ex:Storybook  .
```

States that everything in the range of `ex:authorOf` is both a textbook and a storybook!

MODELING PITFALLS

Beware of the following when modeling using RDFs:

- multiple property restrictions are allowed.
- property restrictions do not depend on the context

Example

ex:isMarriedTo	rdfs:domain	ex:Person	.
ex:isMarriedTo	rdfs:range	ex:Person	.
ex:instULB	rdf:type	ex:Institution	.
ex:john	ex:isMarriedTo	ex:instULB	.

Beware of the following when modeling using RDFs:

- multiple property restrictions are allowed.
- property restrictions do not depend on the context

Example

ex:isMarriedTo	rdfs:domain	ex:Person	.
ex:isMarriedTo	rdfs:range	ex:Person	.
ex:instULB	rdf:type	ex:Institution	.
ex:john	ex:isMarriedTo	ex:instULB	.
ex:instULB	rdf:type	:Person	.

Now we have ex:instULB is both an institute and a Person.

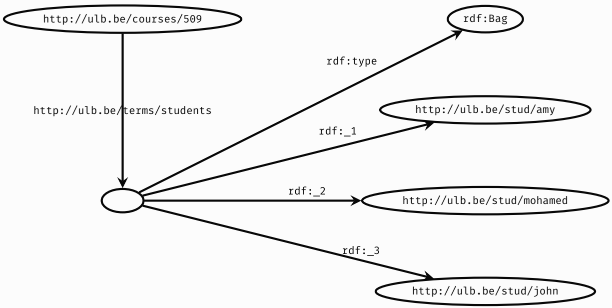
RDF Schema also provides the following resources:

- `rdfs:Resource` — The class of all resources. Every resource is an instance of this class
- `rdfs:Class` — The class of all classes.
- `rdfs:Literal` — The class of all literals.
- `rdfs:Property` — The class of all properties.
- `rdfs:Datatype` — The class of all datatypes.

RDF Schema is equipped with the following built-in triples (among others)

rdfs:subClassOf	rdfs:domain	rdfs:Class	.
rdfs:subClassOf	rdfs:range	rdfs:Class	.
rdfs:subClassOf	rdf:type	rdfs:Property	.
rdfs:subPropertyOf	rdfs:domain	rdfs:Property	.
rdfs:subPropertyOf	rdfs:range	rdfs:Property	.
rdfs:subPropertyOf	rdf:type	rdfs:Property	.
rdfs:domain	rdf:type	rdfs:Property	.
rdfs:range	rdf:type	rdfs:Property	.
rdfs:Resource	rdf:type	rdfs:Class	.
rdfs:Class	rdfs:subClassOf	rdfs:Resource	.
rdfs:Class	rdf:type	rdfs:Class	.

OPEN LISTS IN RDFS



For modeling open lists, RDF schema introduces:

- `rdfs:Container` as the superclass of `rdf:Seq`, `rdf:Bag`, `rdf:Alt`
- `rdfs:ContainerMembershipProperty` as the property that contains all properties used with containers (`rdf:_1, rdf:_2,` ...are all of this type).

In addition, it introduces

- a new property, `rdfs:member`
- which is the superproperty of all properties contained in `rdfs:ContainerMembershipProperty`
- this allows one to introduce their own properties to describe membership in a container.
- and to easily determine whether something is a member of a container without knowing the property used (just look for *u* `rdfs:member` *x*).

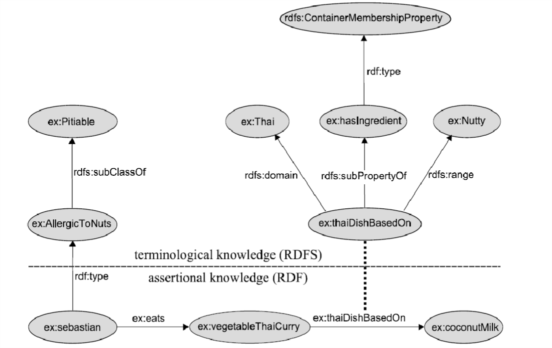
If	*a* `rdf:type` `rdfs:ContainerMembershipProperty .`
And	*u a x* `.`
Then add	*a* `rdfs:member` *x* `.`

SUPPLEMENTARY INFORMATION IN RDFS

In addition, RDFs proposes a standard set of URIs to document RDF or RDFs. These are not part of the actual facts or ontology, but are for the human reader/user/developer.

- `rdfs:label` to give, e.g., a human-readable name for a URI;
- `rdfs:comment` used for lengthy commentary/explanatory text;
- `rdfs:seeAlso` and `rdfs:definedBy`: properties pointing to URIs where further information or definitions can be found.

A SIMPLE ONTOLOGY



ex:vegetableThaiCurry	ex:thaiDishBasedOn	ex:coconutMilk	.
ex:sebastian	rdf:type	ex:AllergicToNuts	.
ex:sebastian	ex:eats	ex:vegetableThaiCurry	.
ex:AllergicToNuts	rdfs:subClassOf	ex:Pitiable	.
ex:thaiDishBasedOn	rdfs:domain	ex:Thai	.
ex:thaiDishBasedOn	rdfs:range	ex:Nutty	.
ex:thaiDishBasedOn	rdfs:subPropertyOf	ex:hasIngredient	.
ex:hasIngredient	rdf:type	rdfs:containerMembershipProperty	.

LIMITATIONS OF RDF SCHEMA

RDF Schema allows us to represent some ontological knowledge:

- Typed hierarchies using classes and subclasses, properties and subproperties
- Domain and range restrictions
- Describing instances of classes (through subclasses and `rdf:type`)

Sometimes we want more:

- Local scope of properties Using `rdfs:range` and `rdfs:domain` we can't state that cows only eat plants while other animals may eat meat too.
- Disjointness of classes. We can't state, for example, that `terms:male` and `terms:female` do not have any members in common.
- Special characteristics of properties. Sometimes it is convenient to be able to say that a property is transitive (like "greater than"), unique (like "father of"), or the inverse of another property (like "father of" and "child of").
- Cardinality restrictions like "a person has exactly 2 parents"

OWL will remedy this.

REFERENCES

- P. Hitzler, M. Krötzsch, S. Rudolph. *Foundations of Semantic Web technologies*. Chapters 1 and 2.
- RDF 1.1. Primer (`http://www.w3.org/TR/rdf11-primer/`)
- RDF 1.1. Turtle (`http://www.w3.org/TR/turtle/`), section 1,2, 3.
- RDF 1.1. XML Syntax (`http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/`), section 1 and 2.
- RDF 1.1. N-Triples (`http://www.w3.org/TR/n-triples/`), section 1, 2
- RDF Schema 1.1. (`http://www.w3.org/TR/2014/PER-rdf-schema-20140109/`)