RDF and RDF Schema

Resource Description Framework

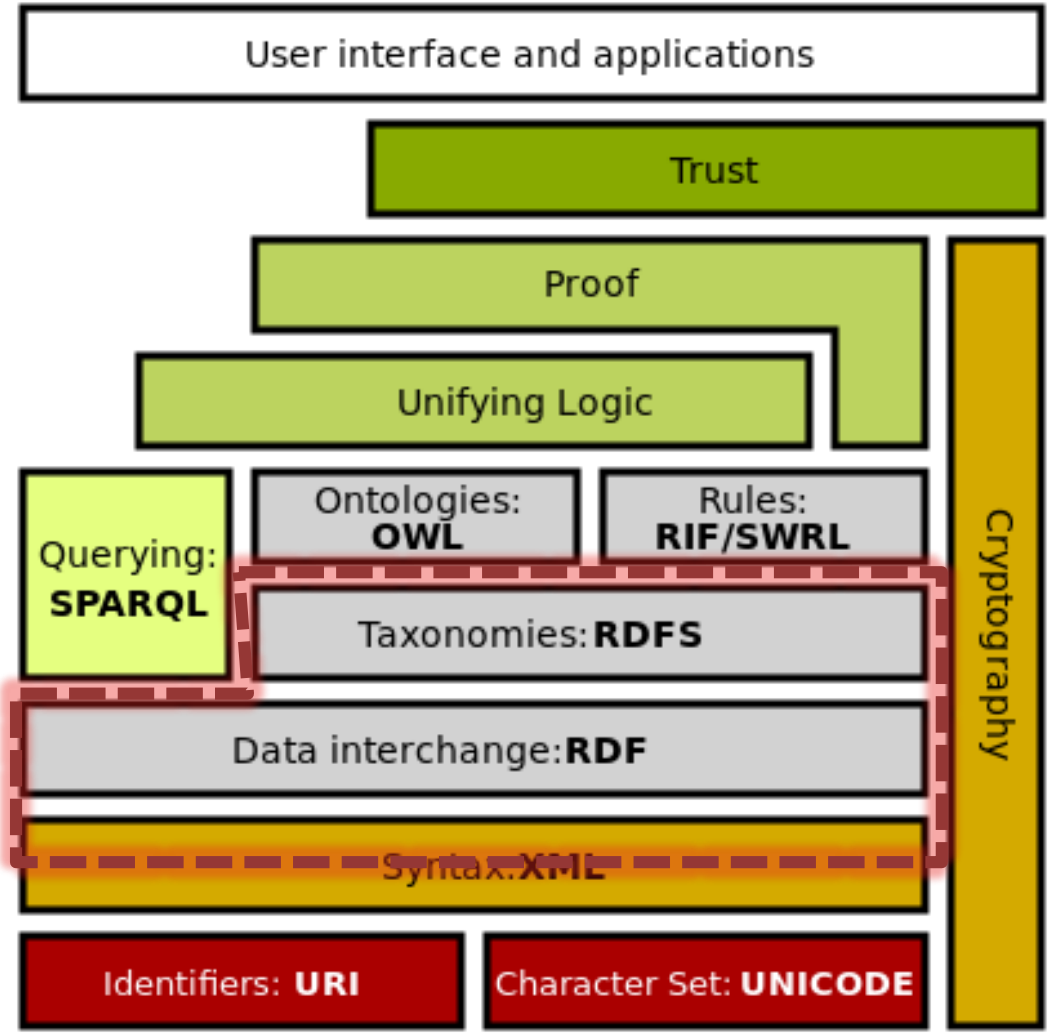
POLITECNICO DI TORINO

e-Lite

Outline

- RDF Design objectives
- RDF General structure
- RDF Vocabularies
- Serialization: XML
- Semantic features
- RDF Schema
- RDF Semantics and Reasoning

SW Technology Stack



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

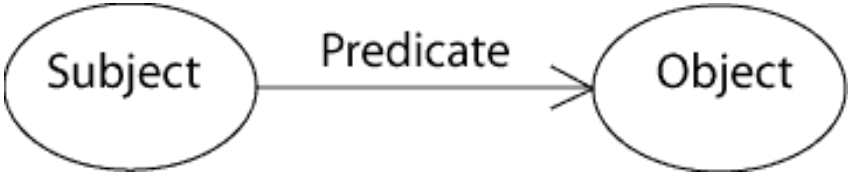
RDF GENERAL STRUCTURE

Key concepts

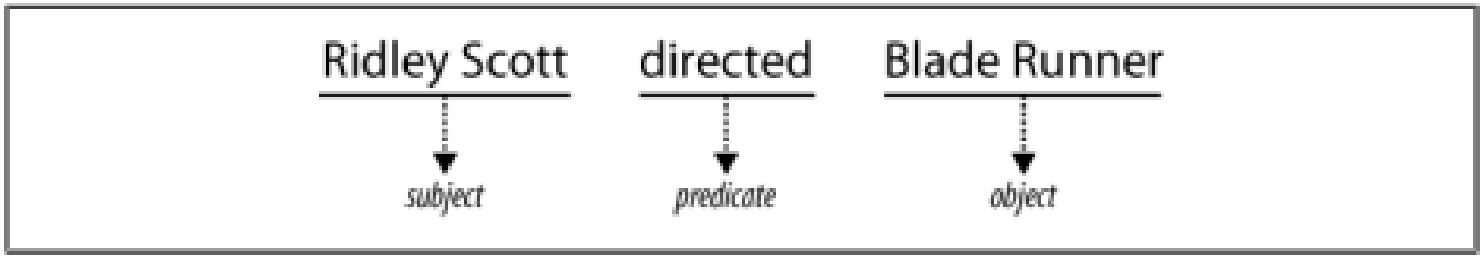
- Graph data model
- URI-based vocabulary
- Datatypes
- Literals
- XML serialization syntax
- Expression of simple facts
- Entailment

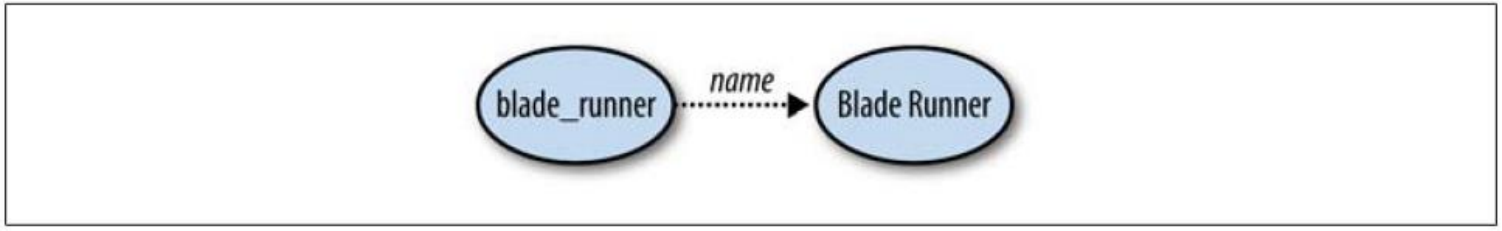
Graph data model

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



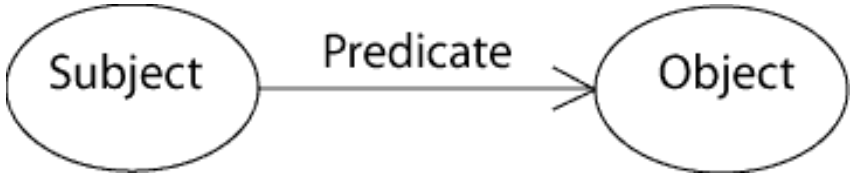
Example


Ridley Scott
directed
Blade Runner
subject
predicate
object


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



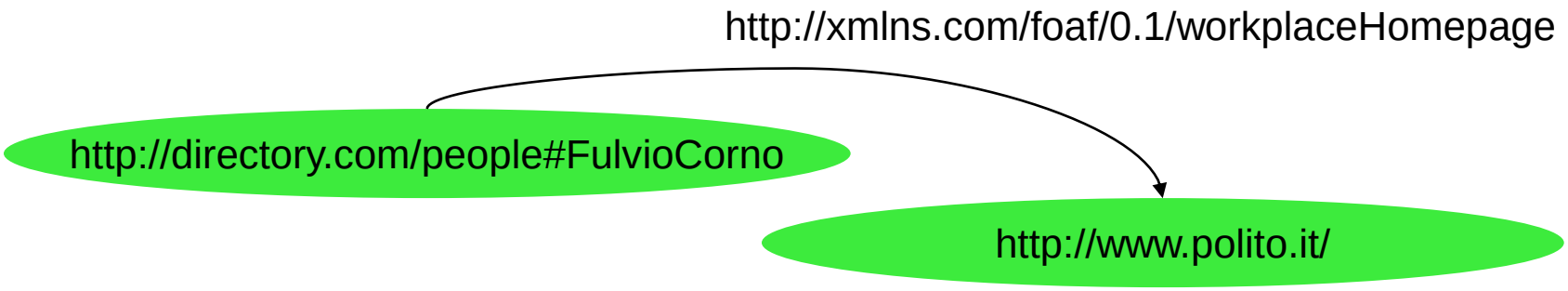
The Triples and the Graph

- 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



RDF

CompanyHomePage

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

Relational database

First order logic predicate

```
HasCompanyHomePage(
'FulvioCorno',
'http://www.polito.it/') ;
```

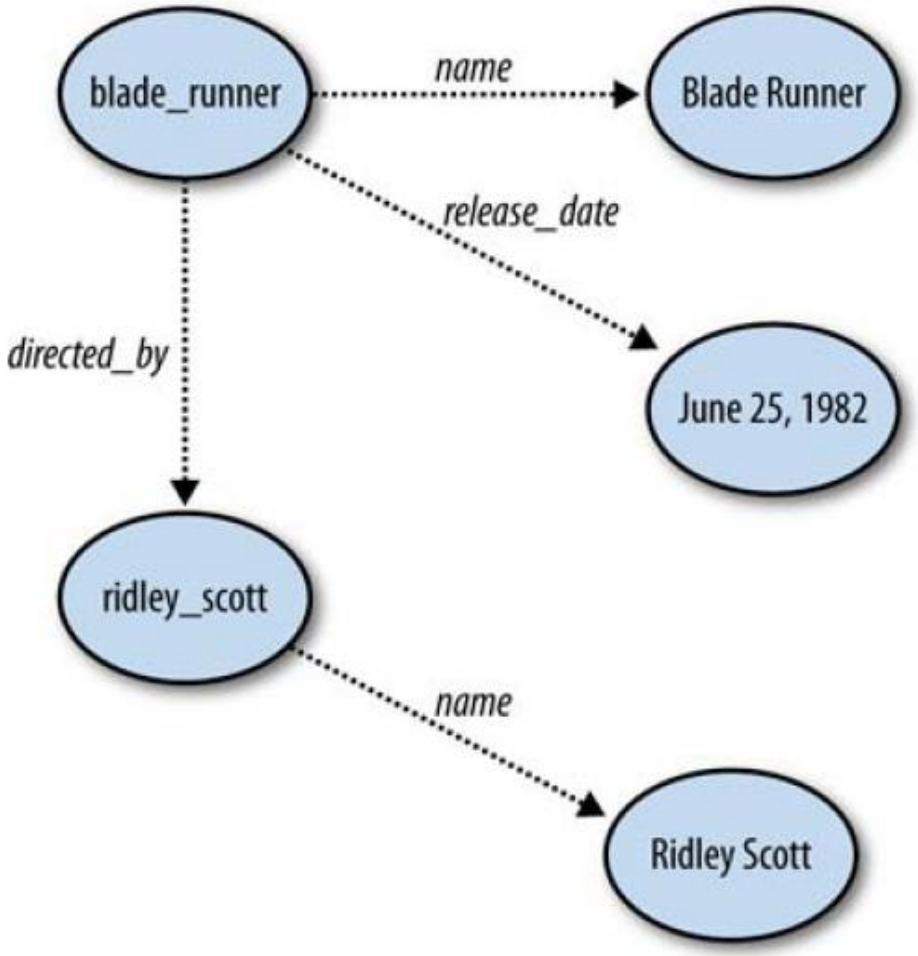
Triples vs Databases

FEATURE	RELATIONAL DATABASE	KNOWLEDGEBASE
Structure	Schema	Ontology statements
Data	Rows	Instance statements
Administration language	DDL	Ontology statements
Query language	SQL	SPARQL
Relationships	Foreign keys	Multidimensional
Logic	External of database/triggers	Formal logic statements
Uniqueness	Key for table	URI

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


blade_runner
name
Blade Runner
release_date
directed_by
June 25, 1982
ridley_scott
name
Ridley Scott

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
Eric Miller
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.

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

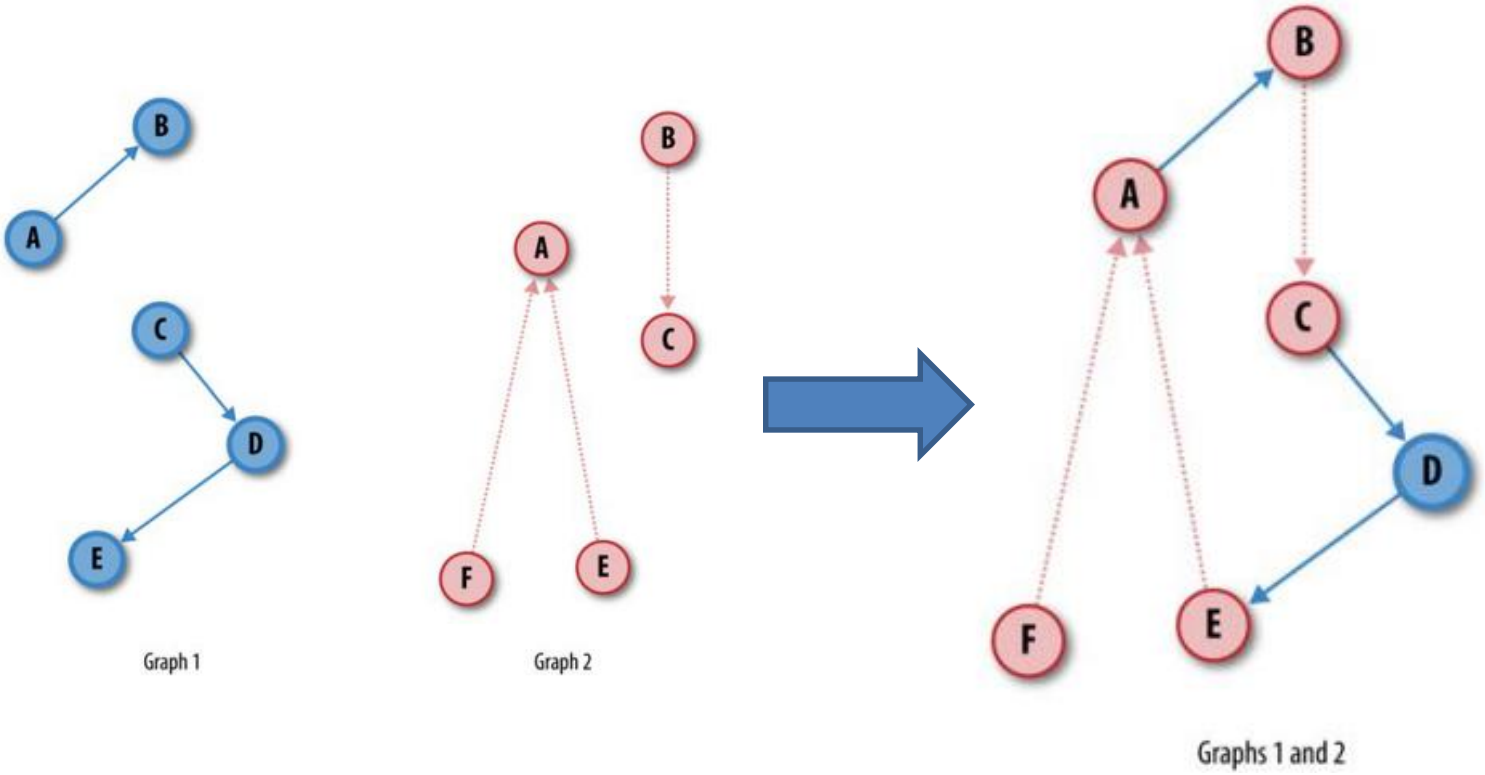
URIs are more than URLs

- 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#fragmet

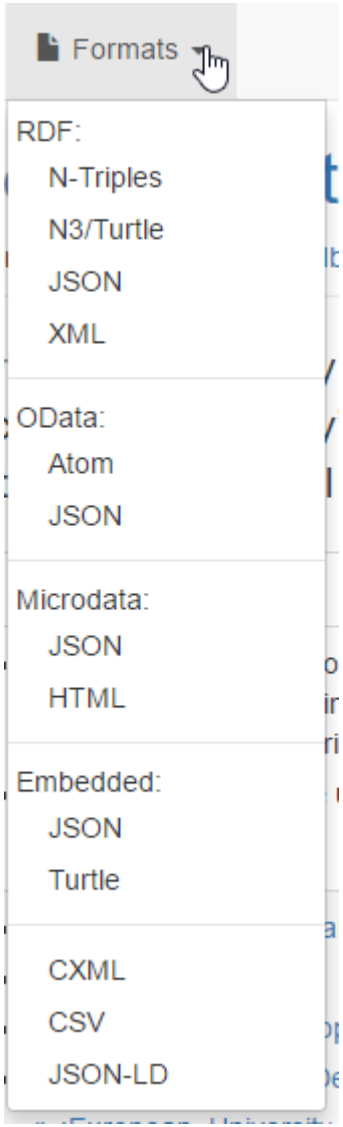
- URIref = URI reference
- A single URI may define many different resources
  - E.g., the URI references an RDF file with many definitions
- To identify a single fragment inside the URI, we use the ‘#’ notation
  - E.g., http://example.org/index#person

Graph Merging



RDF Syntaxes

- Multiple allowed serializations
- Embeddable in documents
- Embeddable in programming languages
- Compatible with relational views



RDF/XML Syntax

```
<?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>
```

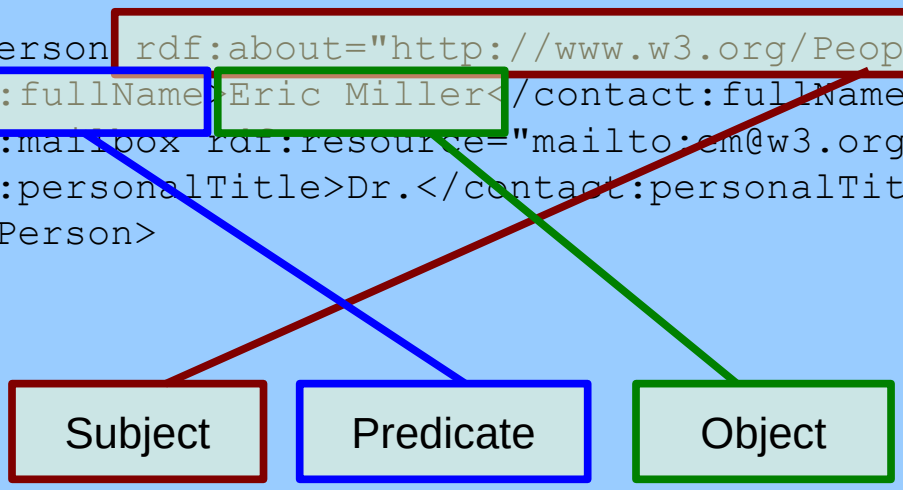
RDF/XML Syntax

```
<?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`

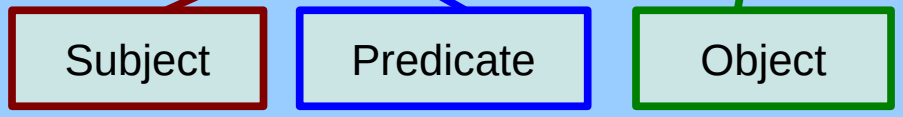
RDF/XML Syntax

```
<?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>
```



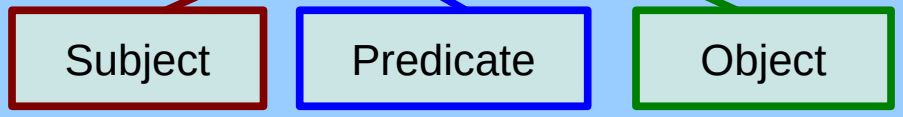
RDF/XML Syntax

```
<?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>
```



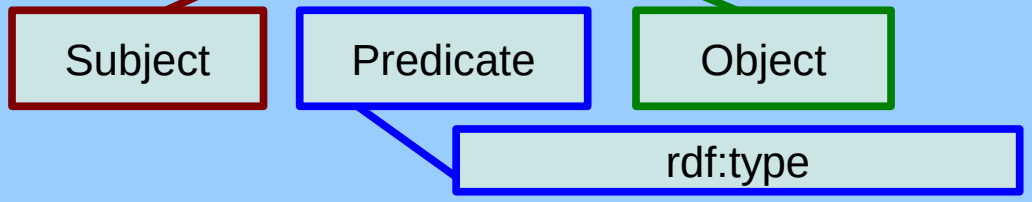
RDF/XML Syntax

```
<?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>
```



RDF/XML Syntax

```
<?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>
```



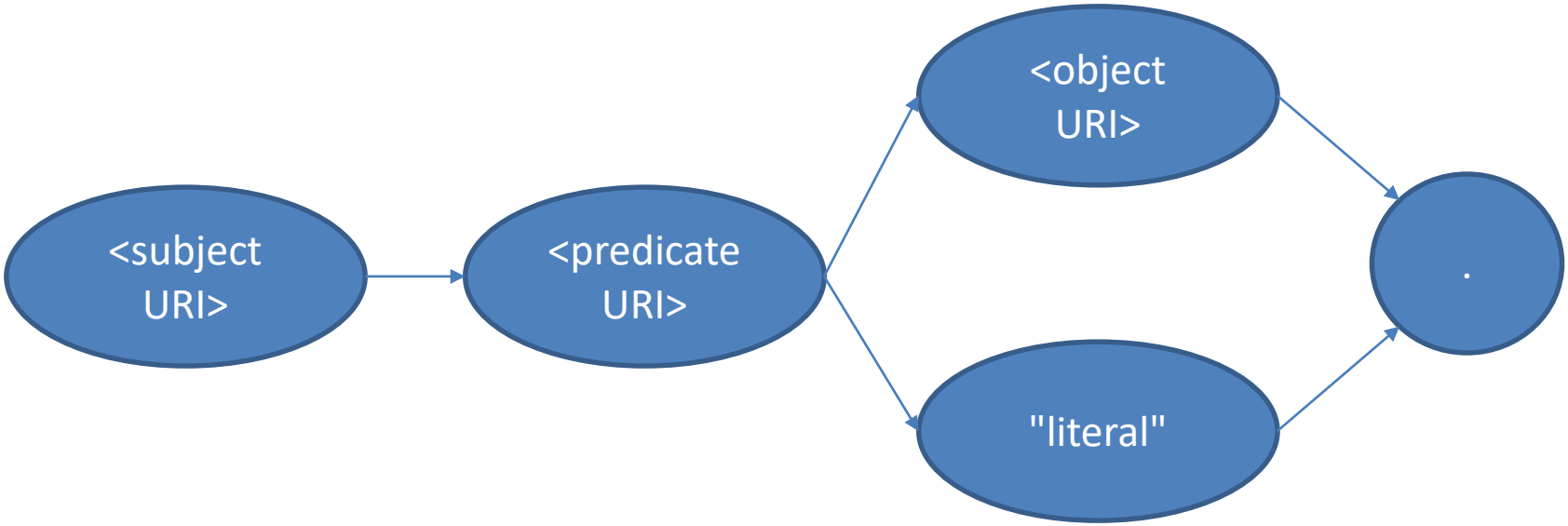
Hands-on exercise

- 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)

```
<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.

Source: http://www.oracle.com/us/corporate/press/018363

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

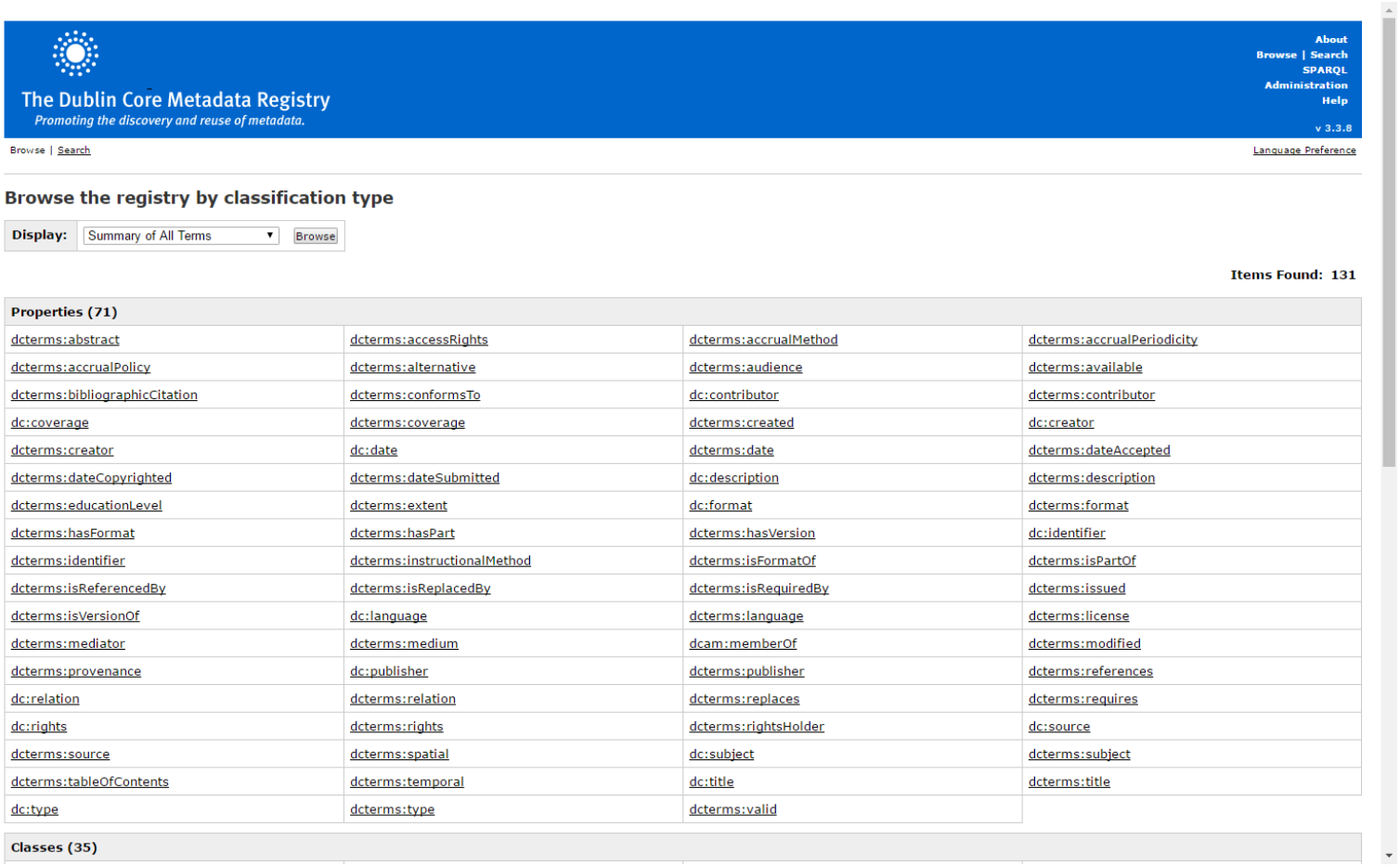
- Extremely easy to re-use other vocabularies in our RDF graph... just define a prefix to point to the proper name space
- When using a predicate, always check if its semantics is already satisfied by some property defined in well-known vocabularies
  - Never re-define, with a different URIref, some already existing predicate
- The same applies for names, but with somewhat less importance.

Hands-on: let's explore some useful vocabularies...

- Dublin Core
  - Specification: http://dublincore.org/documents/dces/
  - Namespace: xmlns:dc=http://purl.org/dc/elements/1.1/
- Recent Dublin Core enhancement: DCMI Metadata Terms
  - Specification: http://dublincore.org/documents/dcmi-terms/
  - Namespace: xmlns:dcterms="http://purl.org/dc/terms/"

Dublin Core

http://dcmi.kc.tsukuba.ac.jp/dcregistry/navigateServlet



The Dublin Core Metadata Registry

Promoting the discovery and reuse of metadata.

About
Browse | Search
SPARQL
Administration
Help
v 3.3.8

Browse | Search

Language Preference

Browse the registry by classification type

Display: Summary of All Terms ▾ Browse

Items Found: 131

Properties (71)			
dcterms:abstract	dcterms:accessRights	dcterms:accrualMethod	dcterms:accrualPeriodicity
dcterms:accrualPolicy	dcterms:alternative	dcterms:audience	dcterms:available
dcterms:bibliographicCitation	dcterms:conformsTo	dc:contributor	dcterms:contributor
dc:coverage	dcterms:coverage	dcterms:created	dc:creator
dcterms:creator	dc:date	dcterms:date	dcterms:dateAccepted
dcterms:dateCopyrighted	dcterms:dateSubmitted	dc:description	dcterms:description
dcterms:educationLevel	dcterms:extent	dc:format	dcterms:format
dcterms:hasFormat	dcterms:hasPart	dcterms:hasVersion	dc:identifier
dcterms:identifier	dcterms:instructionalMethod	dcterms:isFormatOf	dcterms:isPartOf
dcterms:isReferencedBy	dcterms:isReplacedBy	dcterms:isRequiredBy	dcterms:issued
dcterms:isVersionOf	dc:language	dcterms:language	dcterms:license
dcterms:mediator	dcterms:medium	dcam:memberOf	dcterms:modified
dcterms:provenance	dc:publisher	dcterms:publisher	dcterms:references
dc:relation	dcterms:relation	dcterms:replaces	dcterms:requires
dc:rights	dcterms:rights	dcterms:rightsHolder	dc:source
dcterms:source	dcterms:spatial	dc:subject	dcterms:subject
dcterms:tableOfContents	dcterms:temporal	dc:title	dcterms:title
dc:type	dcterms:type	dcterms:valid	

Classes (35)

Hands-on: let's explore some useful vocabularies...

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

- RSS 1.0
  - Information: http://en.wikipedia.org/wiki/RSS_(file_format)

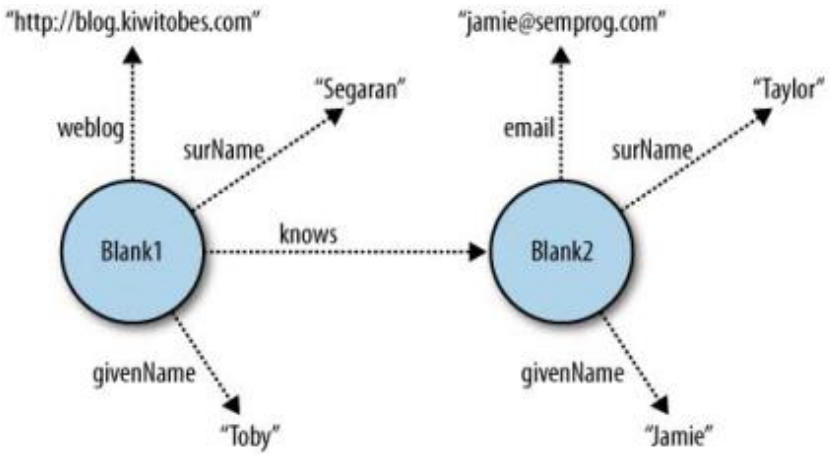
- Curious?
  - http://richard.cyganiak.de/blog/2011/02/top-100-most-popular-rdf-namespace-prefixes/

RDF BLANK NODES

Blank nodes

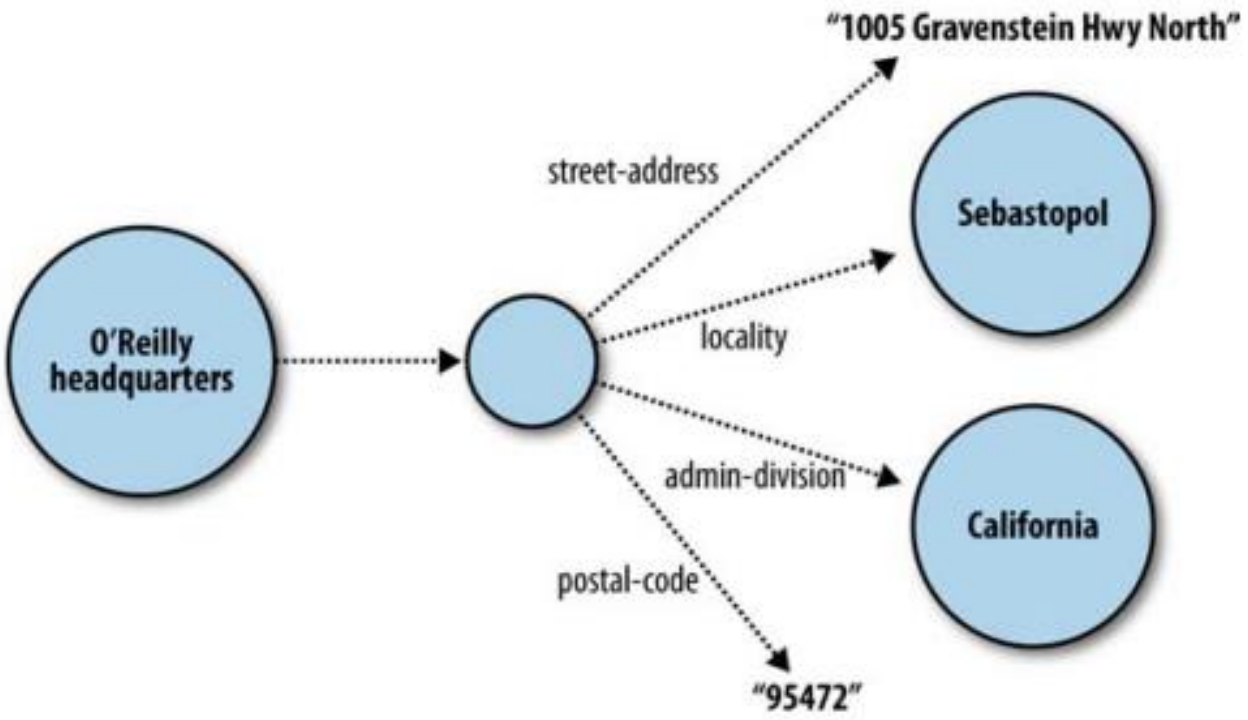
- RDF just supports triples, i.e., binary relationships
- Higher-order relationships must be broken down into many binary pieces
- Breaking down means creating additional nodes
- Such additional nodes will never be referenced from outside the current sub-graph → the don't need a name!
- A subject or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

Example

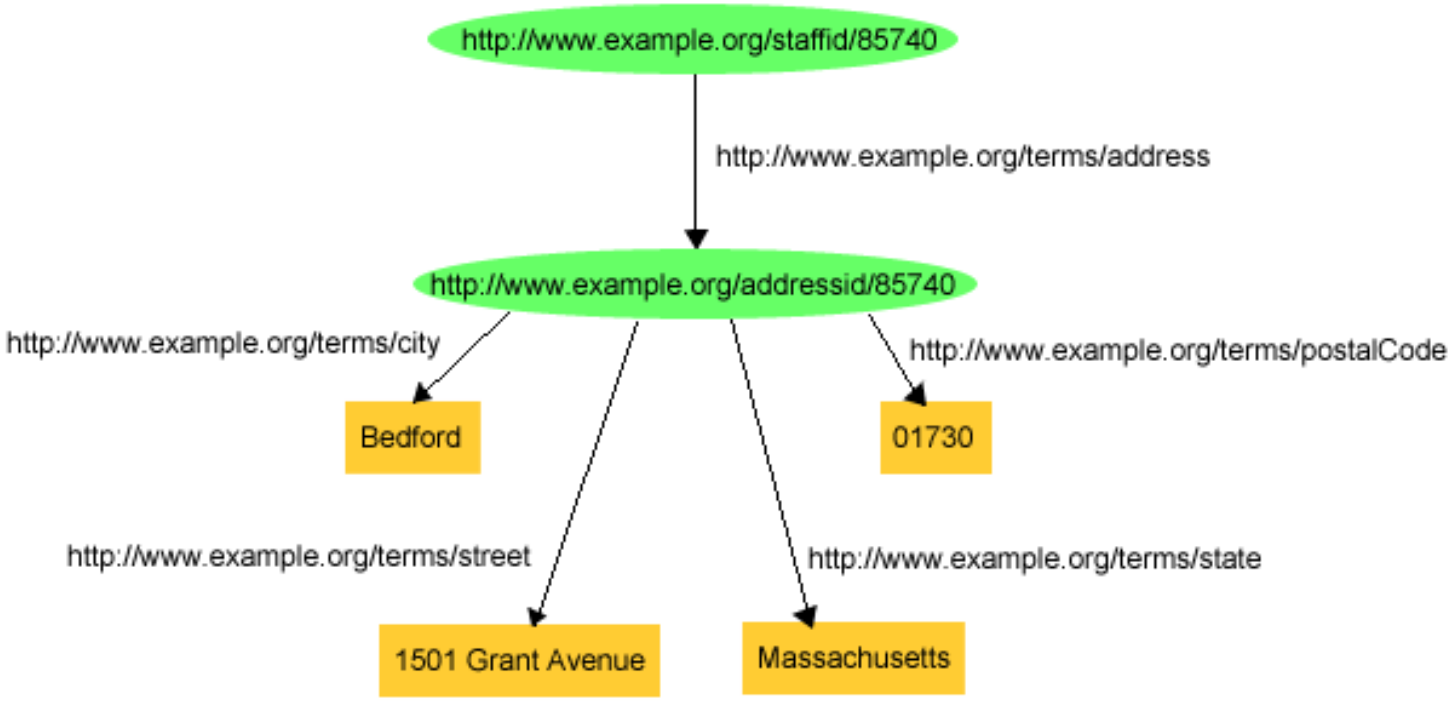


- Anonymous resources
- No "strong" URIs
- Preserves attributes and relationships

Example (multi-dimensional attributes)

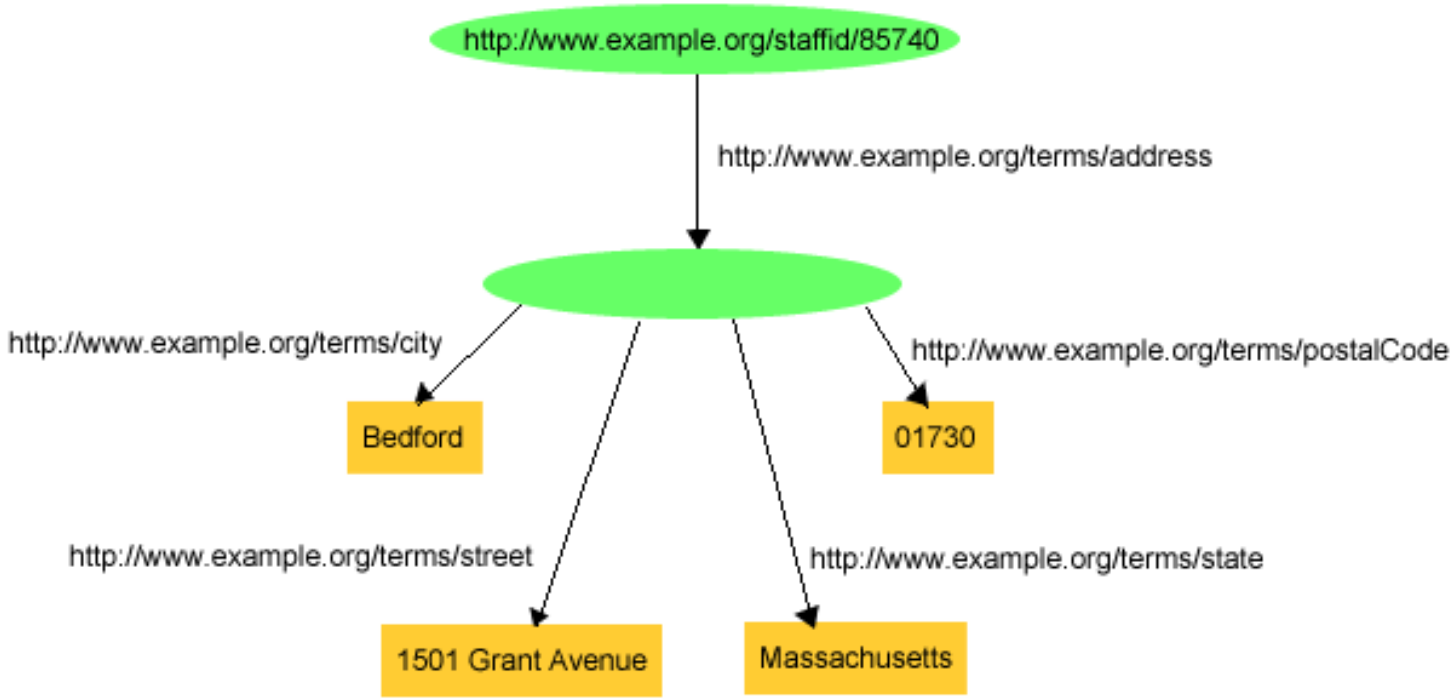


Example




```
exstaff:85740       exterms:address        exaddressid:85740 .
exaddressid:85740   exterms:street         "1501 Grant Avenue" .
exaddressid:85740   exterms:city           "Bedford" .
exaddressid:85740   exterms:state          "Massachusetts" .
exaddressid:85740   exterms:postalCode     "01730" .
```

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

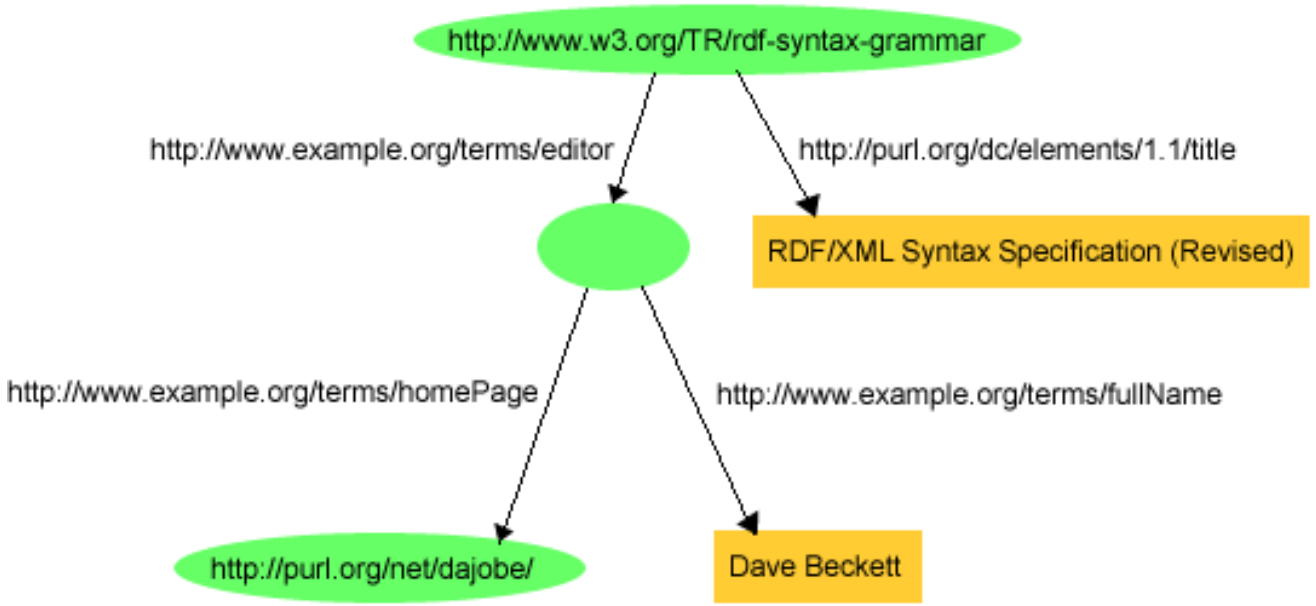
```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.            xmlns:exterms="http://www.example.org/terms/">

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>
11.         <exterms:homePage rdf:resource="http://purl.org/net/dajobe/"/>
12.     </rdf:Description>
```

Typed literals in XML

```
ex:index.html   exterms:creation-date   "1999-08-16"^^xsd:date .
```

```
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
       </exterms:creation-date>
6.   </rdf:Description>
```

RDF SEMANTIC FEATURES

RDF Data structures

- Containers (unbounded)
  - rdf:Bag (unordered)
  - rdf:Seq (ordered)
  - rdf:Alt (one-of)
  - Semantically equivalent, the different beween 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 .
```


```
exproducts:triple12345   rdf:type         rdf:Statement .
exproducts:triple12345   rdf:subject      exproducts:item10245 .
exproducts:triple12345   rdf:predicate    exterms:weight .
exproducts:triple12345   rdf:object       "2.4"^^xsd:decimal .
```

… and now the statement has a URIref: `this.rdf#triple12345`

Example (cont.)

```
exproducts:triple12345   rdf:type        rdf:Statement .
exproducts:triple12345   rdf:subject     exproducts:item10245 .
exproducts:triple12345   rdf:predicate   exterms:weight .
exproducts:triple12345   rdf:object      "2.4"^^xsd:decimal .
```

```
exproducts:triple12345   dc:creator      exstaff:85740 .
```

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

- rdf:type
  - Defines the ‘type’ of the subject node
  - The object of ‘type’ must be a class
- rdfs:Class
  - The set of all possible classes
  - A class is any resource having an rdf:type property whose value is the resource rdfs:Class

```
ex:MotorVehicle   rdf:type   rdfs:Class .

exthings:companyCar   rdf:type   ex:MotorVehicle .
```

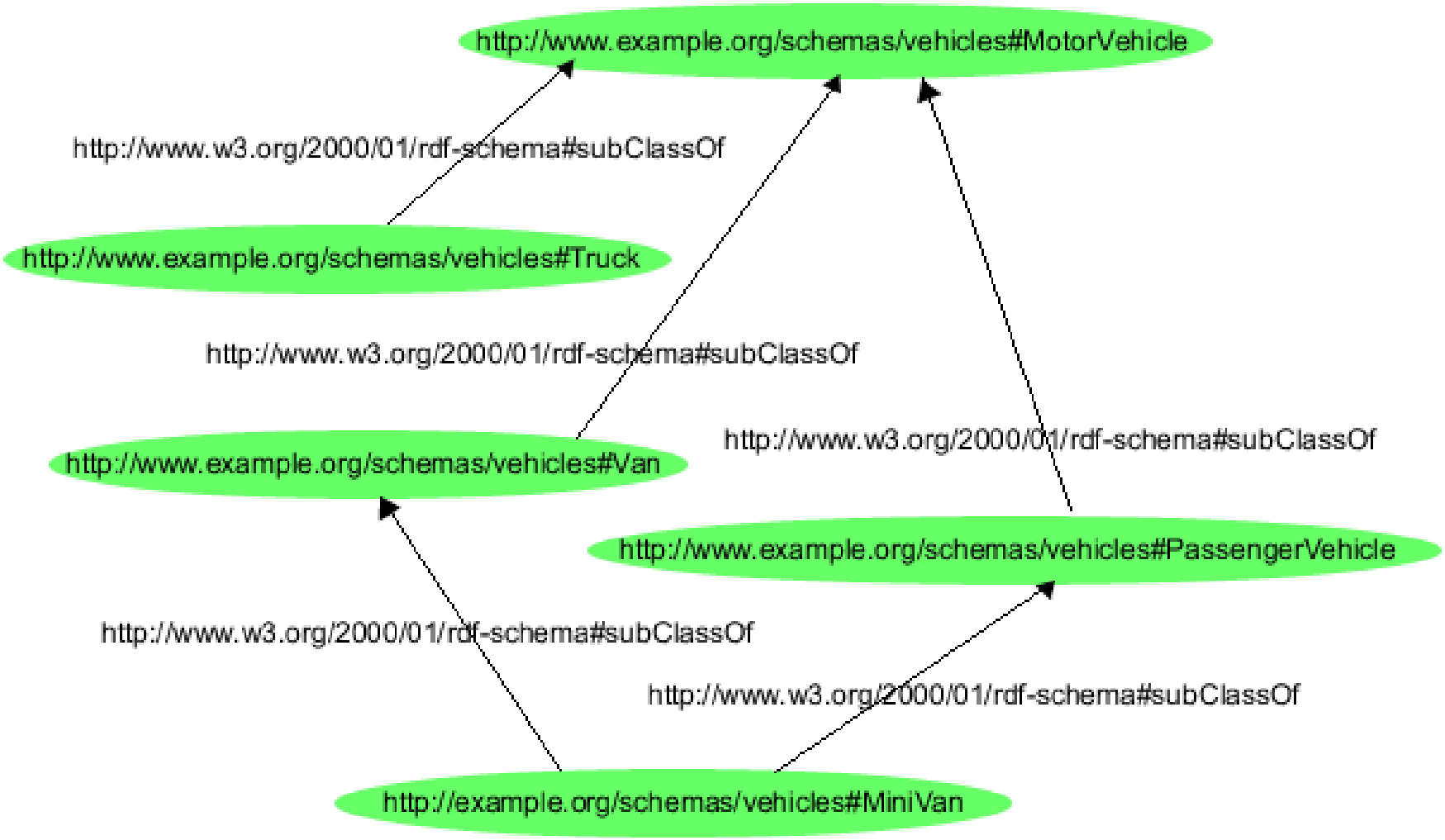
Defining class hierarchies

- 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 .

exthings:companyCar   rdf:type   ex:MotorVehicle .
```

```
ex:Van     rdf:type   rdfs:Class .
ex:Truck   rdf:type   rdfs:Class .

ex:Van   rdfs:subClassOf   ex:MotorVehicle .
```

Class hierarchies



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

```
<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

Class name	comment
rdfs:Resource	The class resource, everything.
rdfs:Literal	The class of literal values, e.g. textual strings and integers.
rdf:XMLLiteral	The class of XML literals values.
rdfs:Class	The class of classes.
rdf:Property	The class of RDF properties.
rdfs:Datatype	The class of RDF datatypes.
rdf:Statement	The class of RDF statements.
rdf:Bag	The class of unordered containers.
rdf:Seq	The class of ordered containers.
rdf:Alt	The class of containers of alternatives.
rdfs:Container	The class of RDF containers.
rdfs:ContainerMembershipProperty	The class of container membership properties, rdf:_1, rdf:_2, ..., all of which are sub-properties of 'member'.
rdf:List	The class of RDF Lists.

RDF/RDFS Properties

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

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? $\rightarrow$ Yes
  - Do all birds have wings? $\rightarrow$ Yes
  - Do robins have wings? $\rightarrow$?????

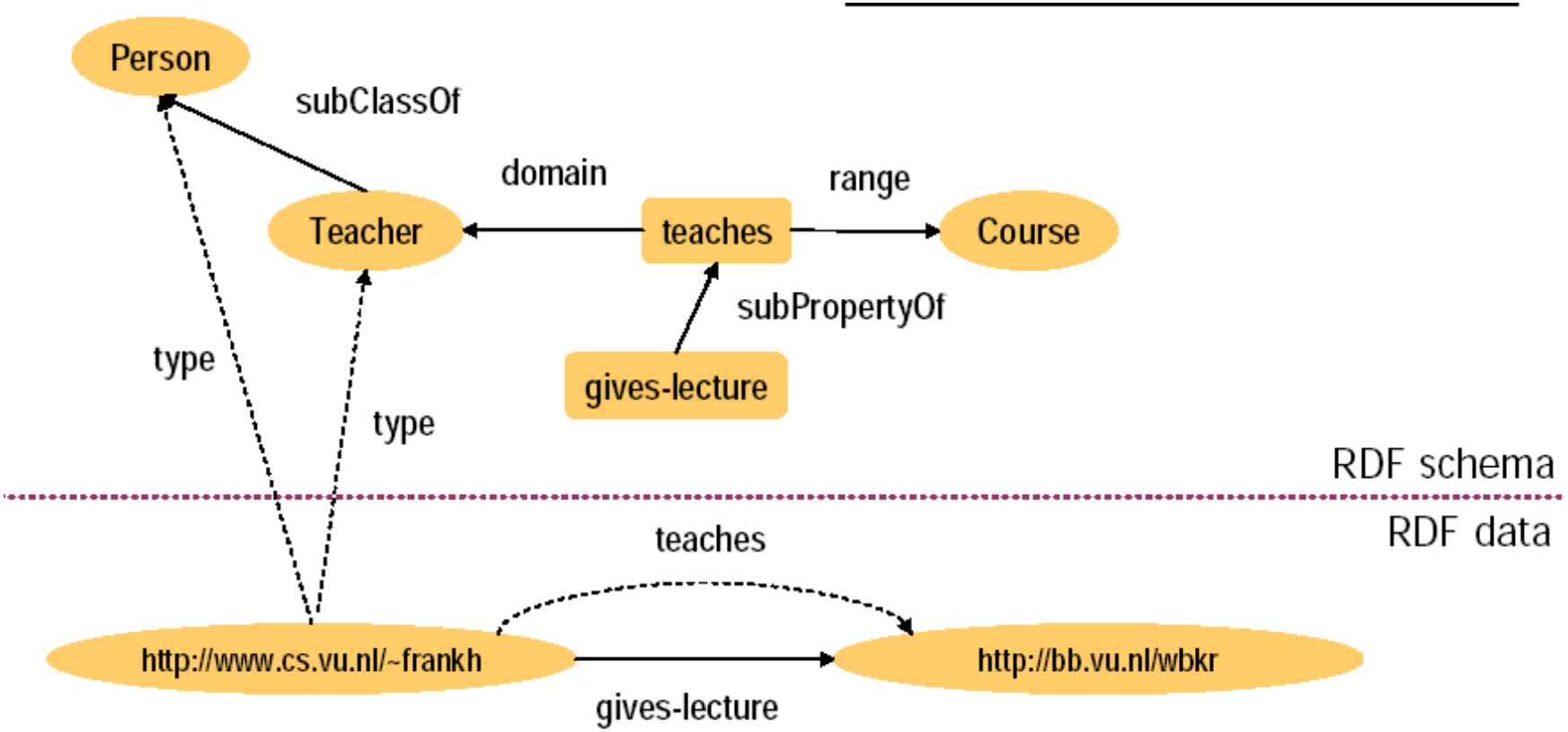
What kind of reasoning can we expect?

- 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

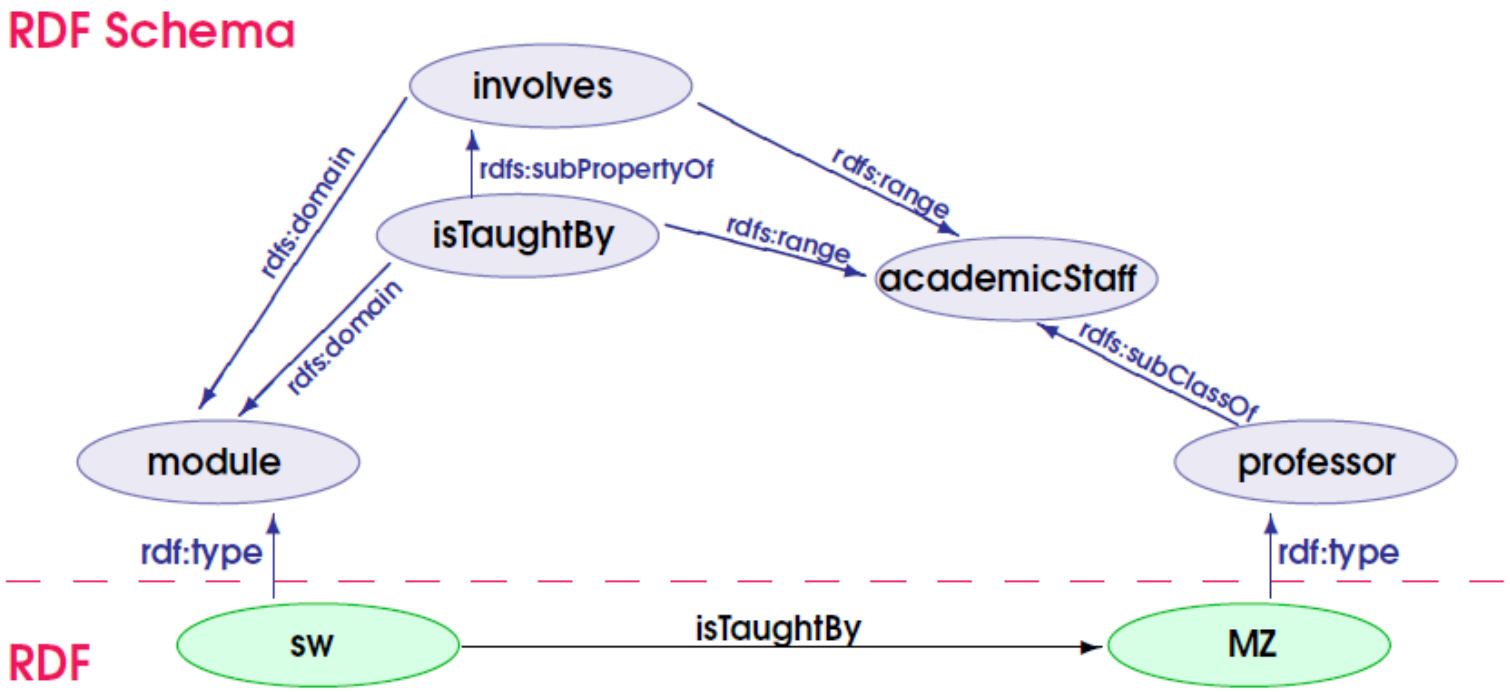
- **Inference** (n.) An act or process of constructing new expressions from existing expressions, or the result of such an act or process. In RDF, inferences corresponding to entailments are described as *correct* or *valid*.
- **Inference rule**, formal description of a type of inference; **inference system**, organized system of inference rules; also, software which generates inferences or checks inferences for validity.

Recall: RDF Schema



Source: Practical Reasoning for the Semantic Web (ESSLLI2005), S. Schlobach, H. Stuckenschmidt, http://www.few.vu.nl/~schlobac/esslli_day1.pdf

RDF vs RDFS



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

Some RDFS inference rules

- (X R Y), (R subPropertyOf Q) => (X Q Y)
- (X R Y), (R domain C) => (X type C)
- (X type C), (C subClassOf D) => (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.)

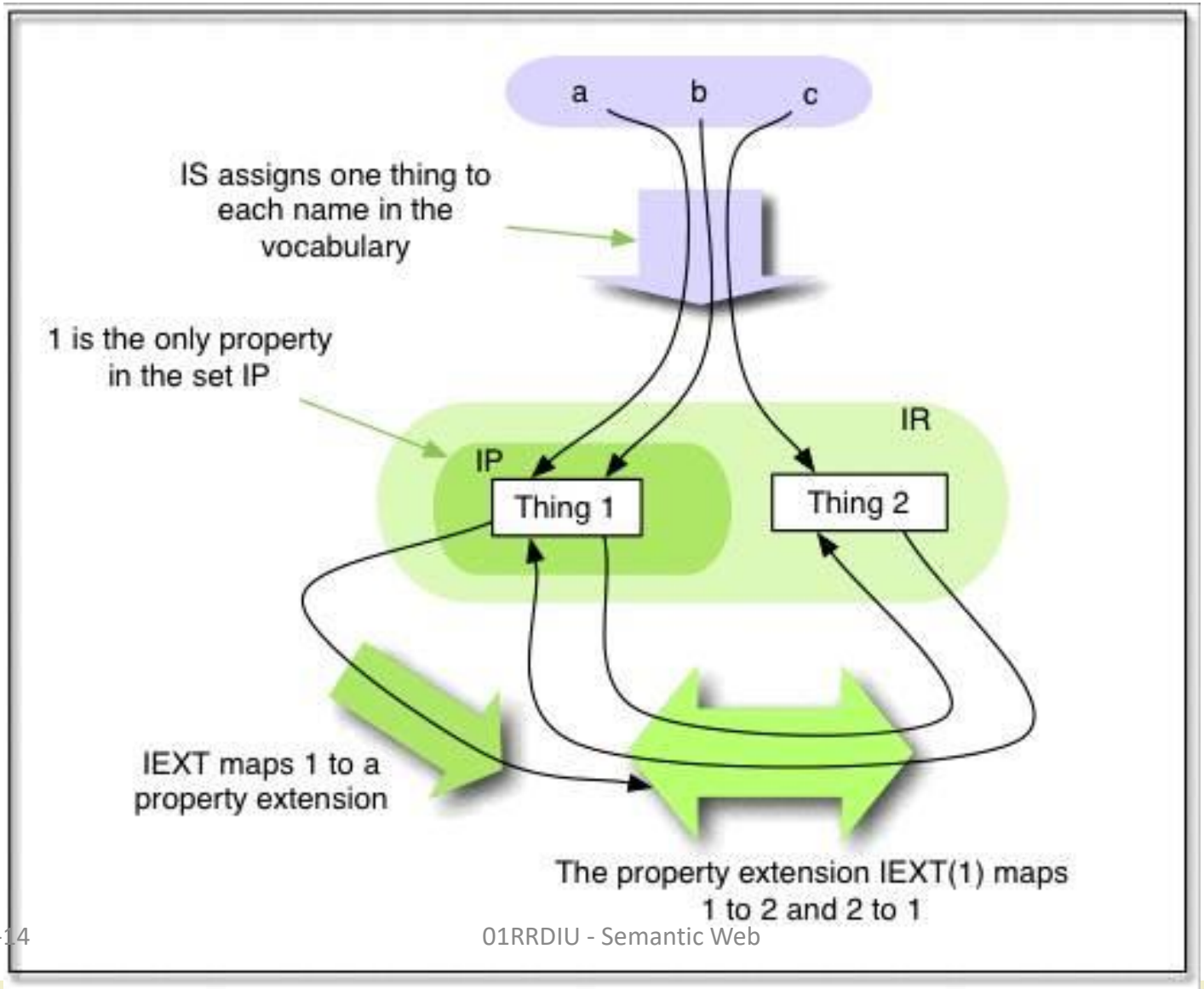
Interpretation: minimum notions

- 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, $IS(v) \in IP$)
  - $IS : V \rightarrow IR$, an interpretation of resources
    (with each symbol from V a resource is associated)
  - $IEXT : IP \rightarrow 2^{IR \times IR}$, 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,
$s, p, o \in V$, $IS(p) \in IP$ and $(IS(s), IS(o)) \in IEXT(IS(p))$
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

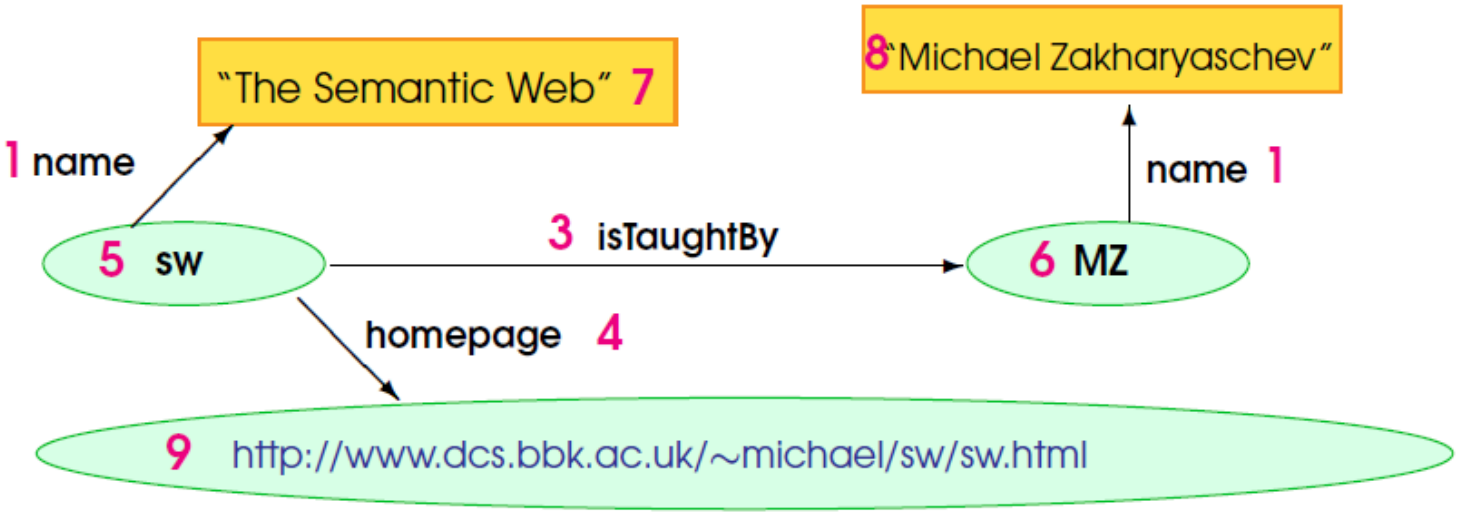


Interpretation: minimum notions

- I : interpretation operator
- E : fragment of RDF syntax
- If E is a graph:
  - I(E) = false if I(E') = false for some triple E' in E, otherwise I(E) =true
- If E is a triple <s, p, o>:
  - I(E) = 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)= false
- IP : set of properties, IR : set of resources
- IEXT(I(p)) : mapping from IP into the powerset of IR x 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:

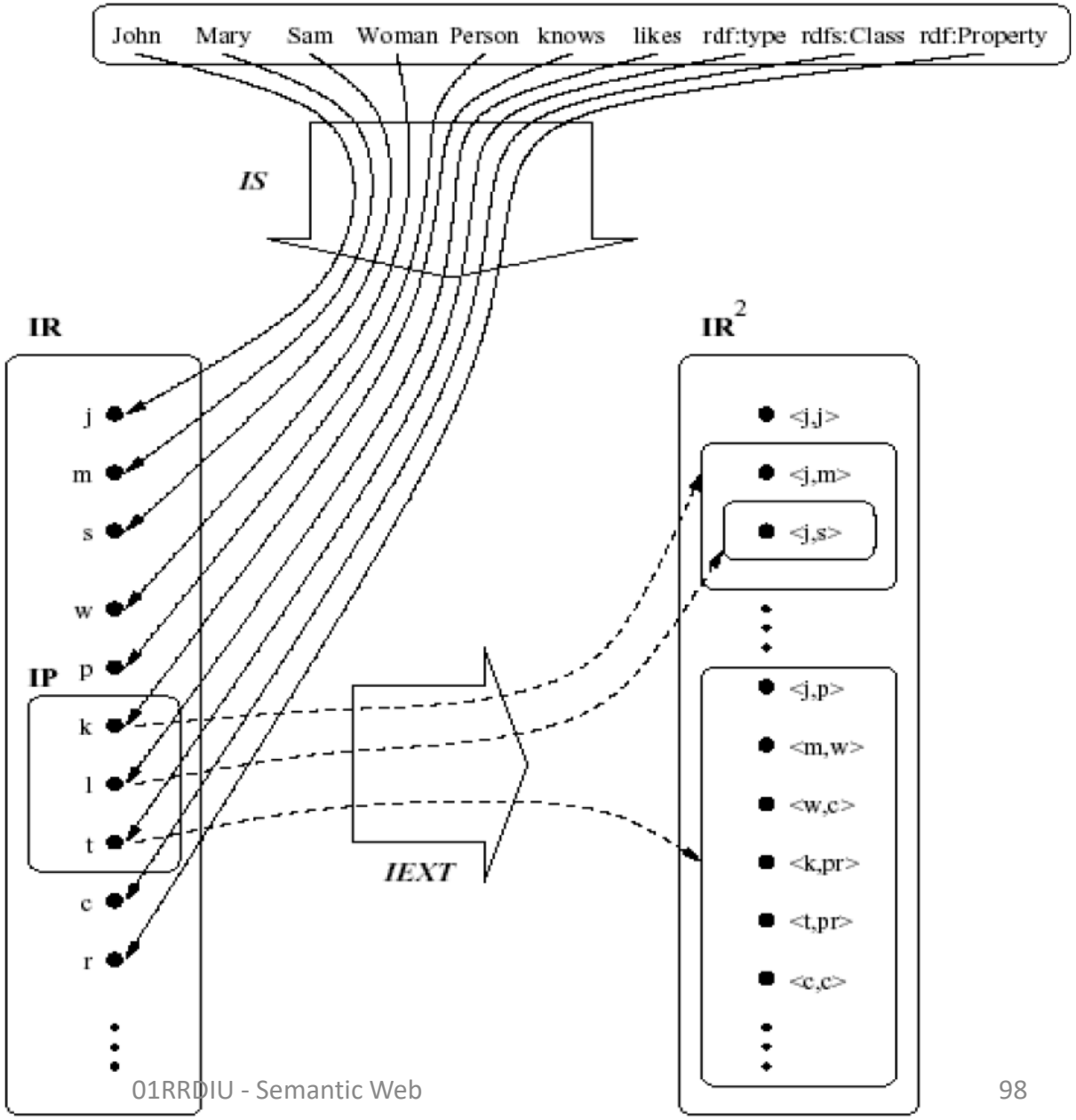


$IR = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and $IP = \{1, 2, 3, 4\}$

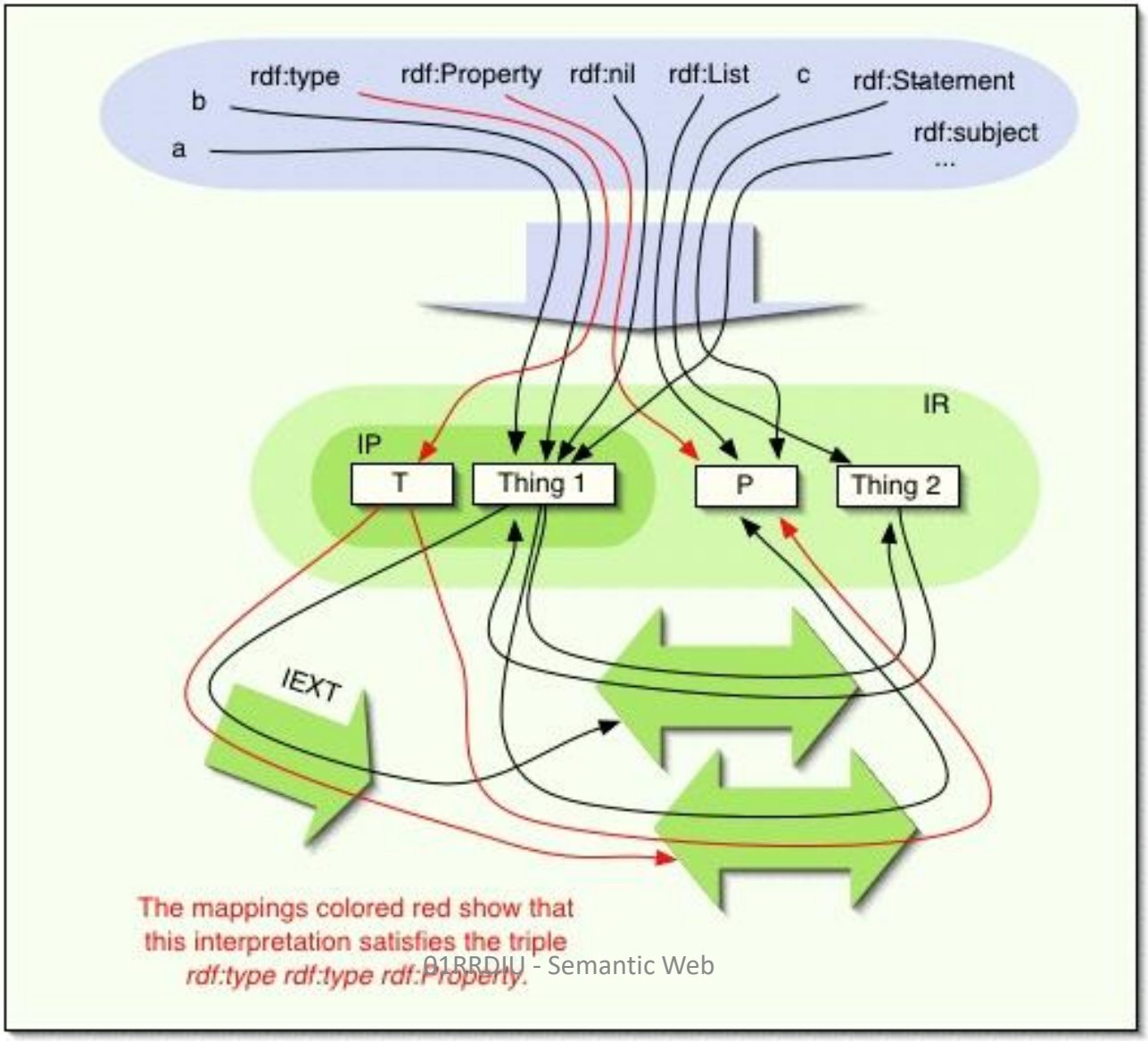
$IS(\textbf{name}) = 1$, $IS(\textbf{isTaughtBy}) = 3$, $IS(\textbf{homepage}) = 4$, $IS(\textbf{sw}) = 5$,
$IS(\textbf{MZ}) = 6$, $IS(\text{http://www.dcs.bbk.ac.uk/}\sim\text{michael/sw/sw.html}) = 9$,
$IS(\text{"The Semantic Web"}) = 7$, $IS(\text{"Michael Zakharyaschev"}) = 8$

$IEXT(1) = \{(5, 7), (6, 8)\}$, $IEXT(2) = \emptyset$, $IEXT(3) = \{(5, 6)\}$,
$IEXT(4) = \{(5, 9)\}$

Example


John
Mary
Sam
Woman
Person
knows
likes
rdf:type
rdfs:Class
rdf:Property
IS
IR
IR2
j
m
s
w
p
IP
k
l
t
c
r
IEXT
<j,j>
<j,m>
<j,s>
<j,p>
<m,w>
<w,c>
<k,pr>
<t,pr>
<c,c>

Example of interpretation



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
- `lll`: any literal
- `_:nnn`: blank node identifiers

Simple Entailment rules

Rule name	If E contains	then add
se1	`uuu aaa xxx .`	`uuu aaa _:nnn .` where _:nnn identifies a blank node allocated to xxx by rule se1 or se2.
se2	`uuu aaa xxx .`	`_:nnn aaa xxx .` where _:nnn identifies a blank node allocated to uuu by rule se1 or se2.
lg	`uuu aaa lll .`	`uuu aaa _:nnn .` where _:nnn identifies a blank node allocated to the literal lll by this rule.
gl	`uuu aaa _:nnn .` where _:nnn identifies a blank node allocated to the literal lll by rule lg.	`uuu aaa lll .`

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

Rule name	If E contains	then add
rdf1	`uuu aaa yyy .`	`aaa rdf:type rdf:Property .`
rdf2	`uuu aaa lll .` where lll is a well-typed XML literal .	`_:nnn rdf:type rdf:XMLLiteral .` where _:nnn identifies a blank node allocated to lll by rule lg.

RDF Axiomatic triples

```
rdf:type          rdf:type          rdf:Property .
rdf:subject       rdf:type          rdf:Property .
rdf:predicate     rdf:type          rdf:Property .
rdf:object        rdf:type          rdf:Property .
rdf:first         rdf:type          rdf:Property .
rdf:rest          rdf:type          rdf:Property .
rdf:value         rdf:type          rdf:Property .
rdf:_1            rdf:type          rdf:Property .
rdf:_2            rdf:type          rdf:Property .
...
rdf:nil           rdf:type          rdf:List .
```

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)

Rule name	If E contains	then add
rdfs1	`uuu aaa lll.` where lll is a plain literal (with or without a language tag)	`_:nnn rdf:type rdfs:Literal .` where _:nnn identifies a blank node allocated to lll by rule rule lg.
rdfs2	`aaa rdfs:domain xxx .` `uuu aaa yyy .`	`uuu rdf:type xxx .`
rdfs3	`aaa rdfs:range xxx .` `uuu aaa vvv .`	`vvv rdf:type xxx .`
rdfs4a	`uuu aaa xxx .`	`uuu rdf:type rdfs:Resource .`
rdfs4b	`uuu aaa vvv.`	`vvv rdf:type rdfs:Resource .`
rdfs5	`uuu rdfs:subPropertyOf vvv .` `vvv rdfs:subPropertyOf xxx .`	`uuu rdfs:subPropertyOf xxx .`

RDFS Entailment rules (2/2)

Rule name	If E contains	then add
rdfs6	`uuu rdf:type rdf:Property .`	`uuu rdfs:subPropertyOf uuu .`
rdfs7	`aaa rdfs:subPropertyOf bbb` `uuu aaa yyy .`	`uuu bbb yyy .`
rdfs8	`uuu rdf:type rdfs:Class .`	`uuu rdfs:subClassOf rdfs:Resource .`
rdfs9	`uuu rdfs:subClassOf xxx .` `vvv rdf:type uuu .`	`vvv rdf:type xxx .`
rdfs10	`uuu rdf:type rdfs:Class .`	`uuu rdfs:subClassOf uuu .`
rdfs11	`uuu rdfs:subClassOf vvv .` `vvv rdfs:subClassOf xxx .`	`uuu rdfs:subClassOf xxx .`
rdfs12	`uuu rdf:type rdfs:ContainerMembershipProperty .`	`uuu rdfs:subPropertyOf rdfs:member .`
rdfs13	`uuu rdf:type rdfs:Datatype`	`uuu rdfs:subClassOf rdfs:Literal .`

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:subject             rdfs:domain     rdf:Statement .
rdf:predicate           rdfs:domain     rdf:Statement .
rdf:object              rdfs:domain     rdf:Statement .
rdfs:member             rdfs:domain     rdfs:Resource .
rdf:first               rdfs:domain     rdf:List .
rdf:rest                rdfs:domain     rdf:List .
rdfs:seeAlso            rdfs:domain     rdfs:Resource .
rdfs:isDefinedBy        rdfs:domain     rdfs:Resource .
rdfs:comment            rdfs:domain     rdfs:Resource .
rdfs:label              rdfs:domain     rdfs:Resource .
rdf:value               rdfs:domain     rdfs:Resource .
```

RDFS Axiomatic triples (2/3)

```
rdf:type                rdfs:range      rdfs:Class .
rdfs:domain             rdfs:range      rdfs:Class .
rdfs:range              rdfs:range      rdfs:Class .
rdfs:subPropertyOf      rdfs:range      rdf:Property .
rdfs:subClassOf         rdfs:range      rdfs:Class .
rdf:subject             rdfs:range      rdfs:Resource .
rdf:predicate           rdfs:range      rdfs:Resource .
rdf:object              rdfs:range      rdfs:Resource .
rdfs:member             rdfs:range      rdfs:Resource .
rdf:first               rdfs:range      rdfs:Resource .
rdf:rest                rdfs:range      rdf:List .
rdfs:seeAlso            rdfs:range      rdfs:Resource .
rdfs:isDefinedBy        rdfs:range      rdfs:Resource .
rdfs:comment            rdfs:range      rdfs:Literal .
rdfs:label              rdfs:range      rdfs:Literal .
rdf:value               rdfs:range      rdfs:Resource .
```

RDFS Axiomatic triples (3/3)

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

rdfs:isDefinedBy           rdfs:subPropertyOf        rdfs:seeAlso .

rdf:XMLLiteral             rdf:type                  rdfs:Datatype .
rdf:XMLLiteral             rdfs:subClassOf           rdfs:Literal .
rdfs:Datatype              rdfs:subClassOf           rdfs:Class .

rdf:_1                     rdf:type  rdfs:ContainerMembershipProperty .
rdf:_1                     rdfs:domain               rdfs:Resource .
rdf:_1                     rdfs:range                rdfs:Resource .
rdf:_2                     rdf:type  rdfs:ContainerMembershipProperty .
rdf:_2                     rdfs:domain               rdfs:Resource .
rdf:_2                     rdfs:range                rdfs:Resource .
...
```

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

Effect of Entailment rules (1/3)

- The outputs of these rules will often trigger others.

rule:	triggers rule: 1	2	3	4a	4b	5a	5b	6	7a	7b	8	9	10	11
1		*	*	*			*	*				*		
2		*	*				*	*	*	*		*	*	*
3		*	*				*	*	*	*		*	*	*
4a		*	*					*				*		
4b		*	*					*				*		
5a						*		*						
5b		*	*					*						
6		*	*			*	*	*	*	*	*	*	*	*
7a		*	*					*			*	*		
7b		*	*					*						
8								*			*	*		
9			*				*	*	*	*		*	*	*
10		*	*		*	*		*						
11		*	*					*			*	*		

Table 1: Dependencies between RDFS entailment rules

Source: Inferencing and Truth Maintenance in RDF Schema, Exploring a naive practical approach, J. Broekstra, A. Kampman, http://www.openrdf.org/doc/papers/inferencing.pdf

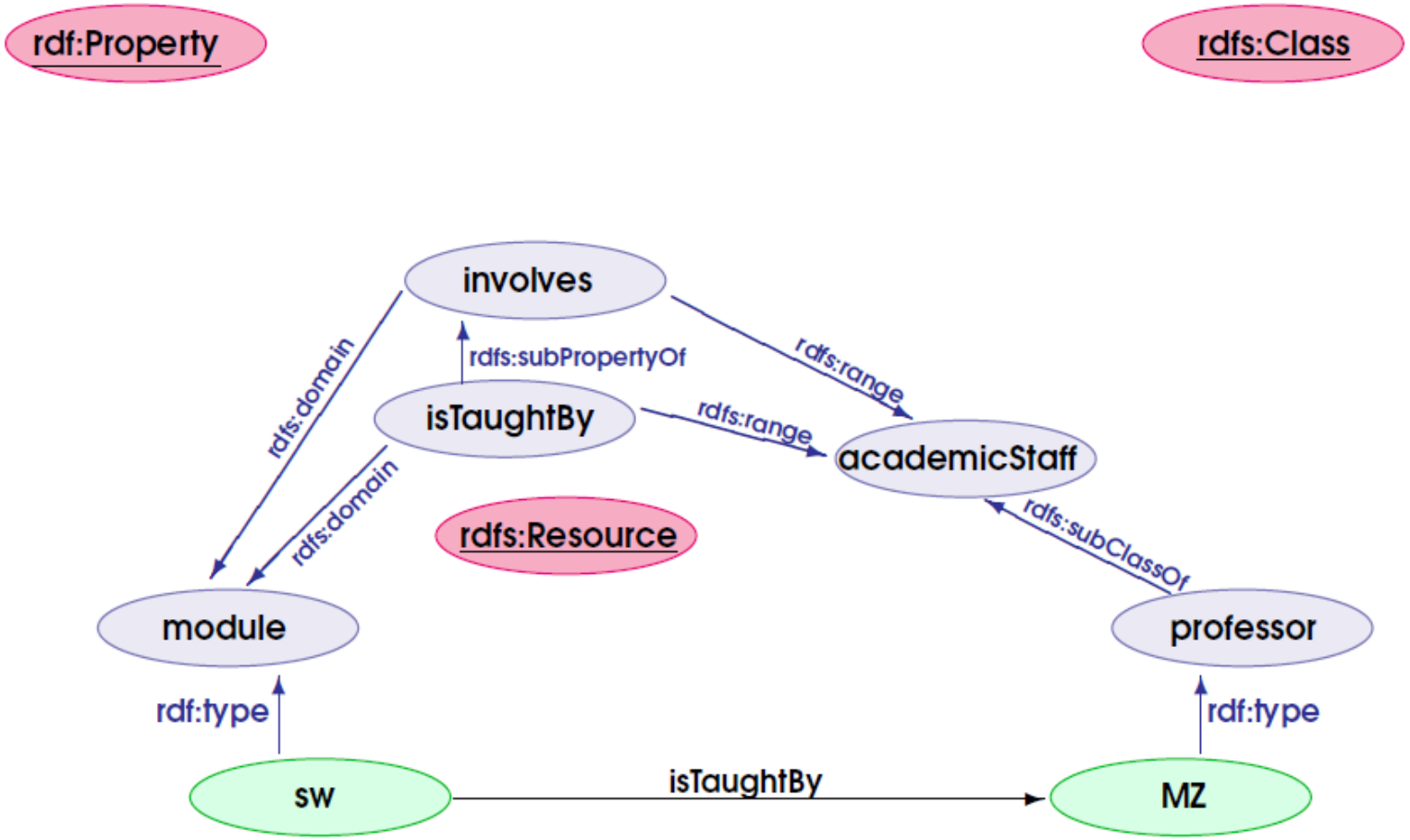
Effect of Entailment rules (2/3)

- 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`

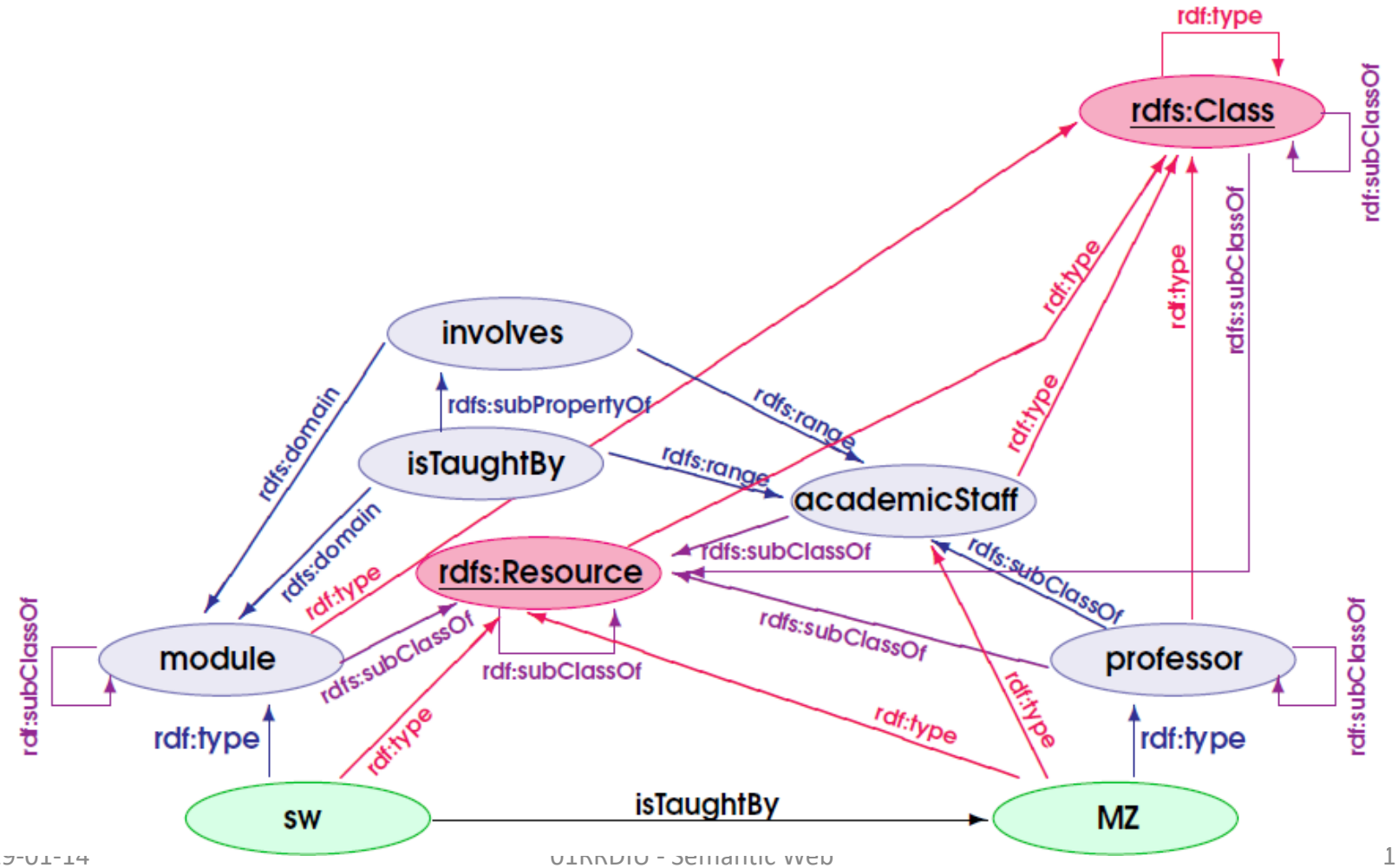
Effect of Entailment rules (3/3)

- 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
  - `aaa rdfs:domain xxx ∧ uuu aaa yyy → uuu rdf:type xxx`
  - `aaa rdfs:range xxx ∧ uuu aaa vvv → vvv rdf:type xxx`
- For every uuu in V, the rules generate all assertions
  - `uuu rdf:type rdfs:Resource`
- For every class name uuu, the rules generate
  - `uuu rdfs:subClassOf rdfs:Resource`

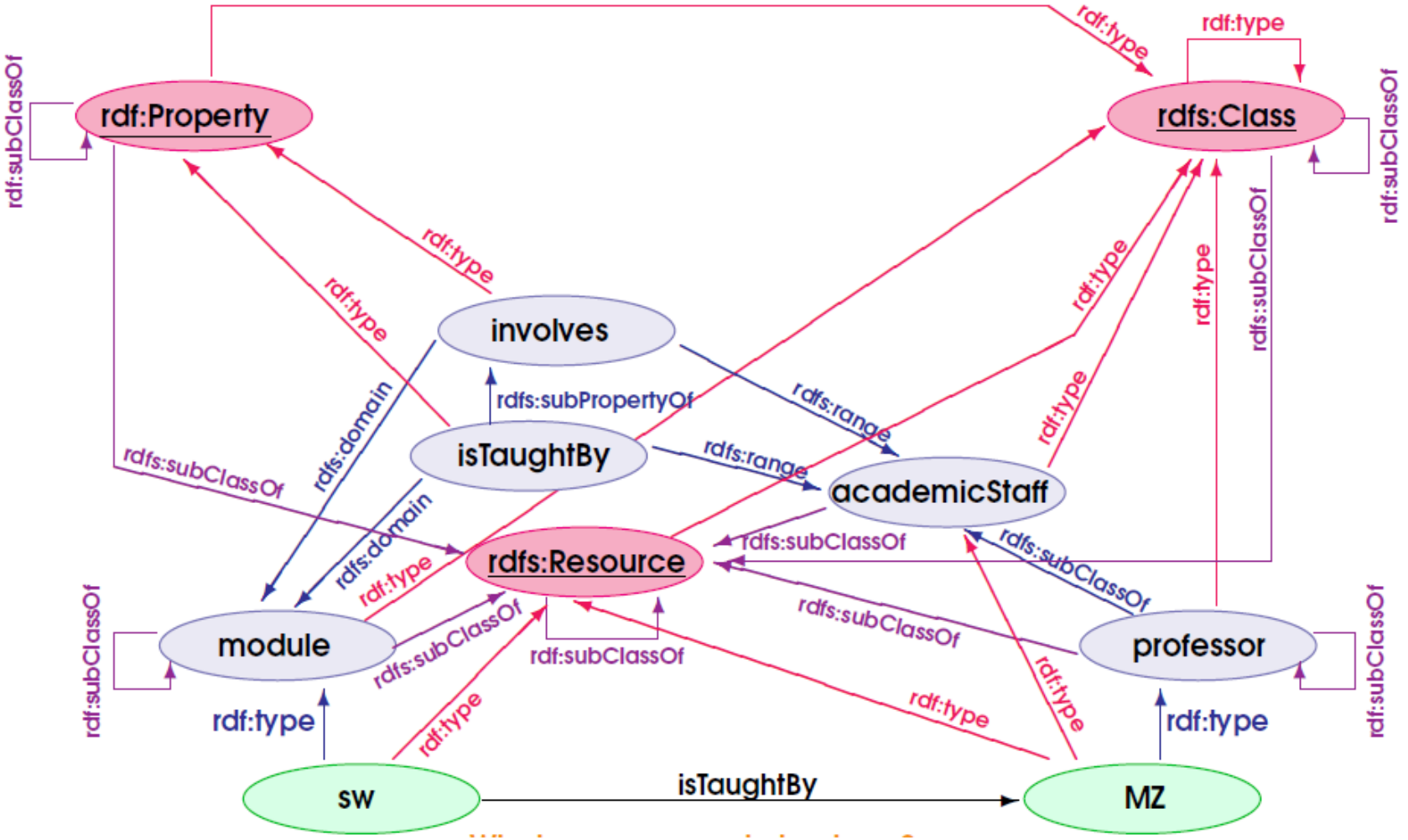
Example (axiomatic nodes)



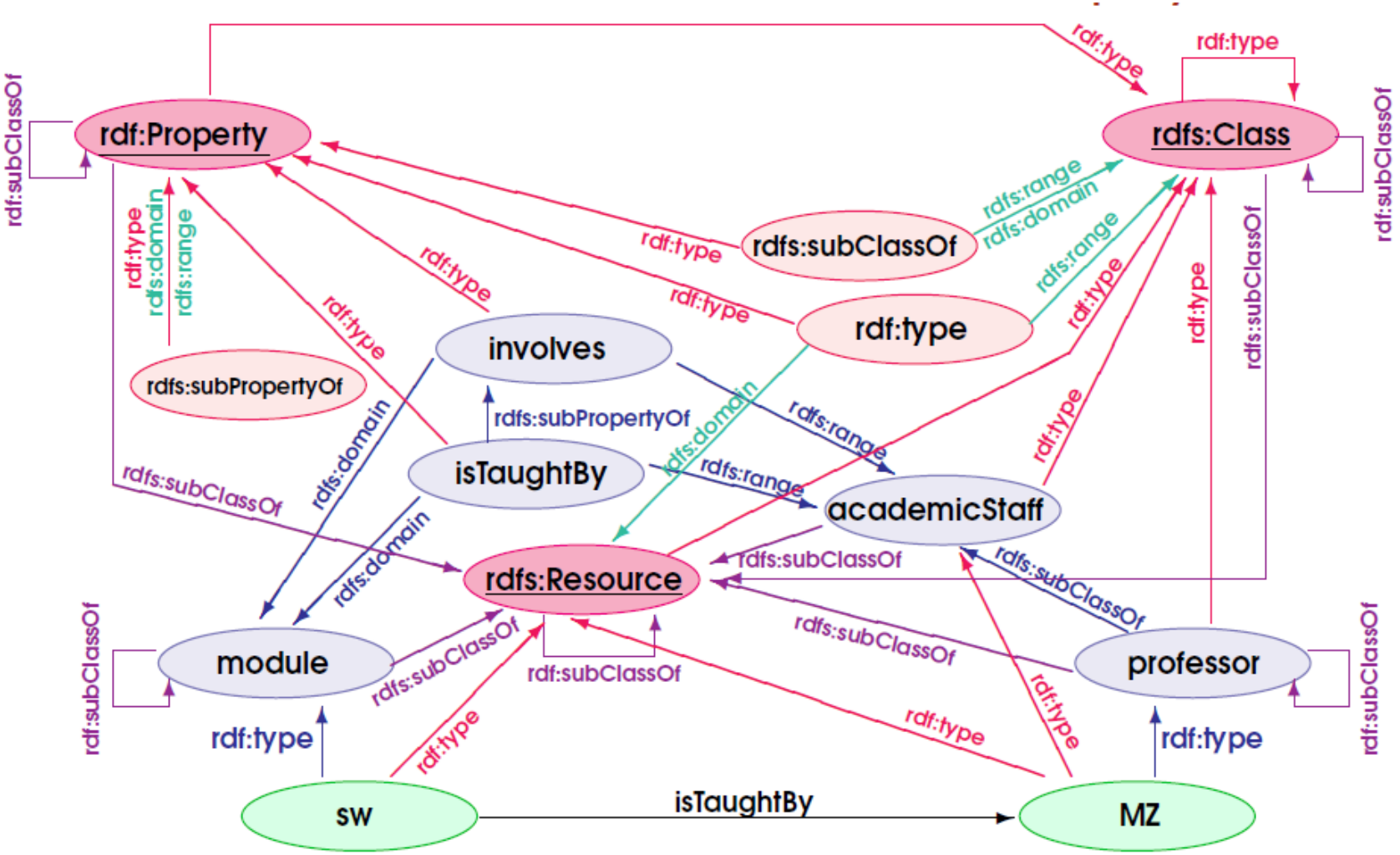
Example (rdf:type and rdfs:subclassOf)



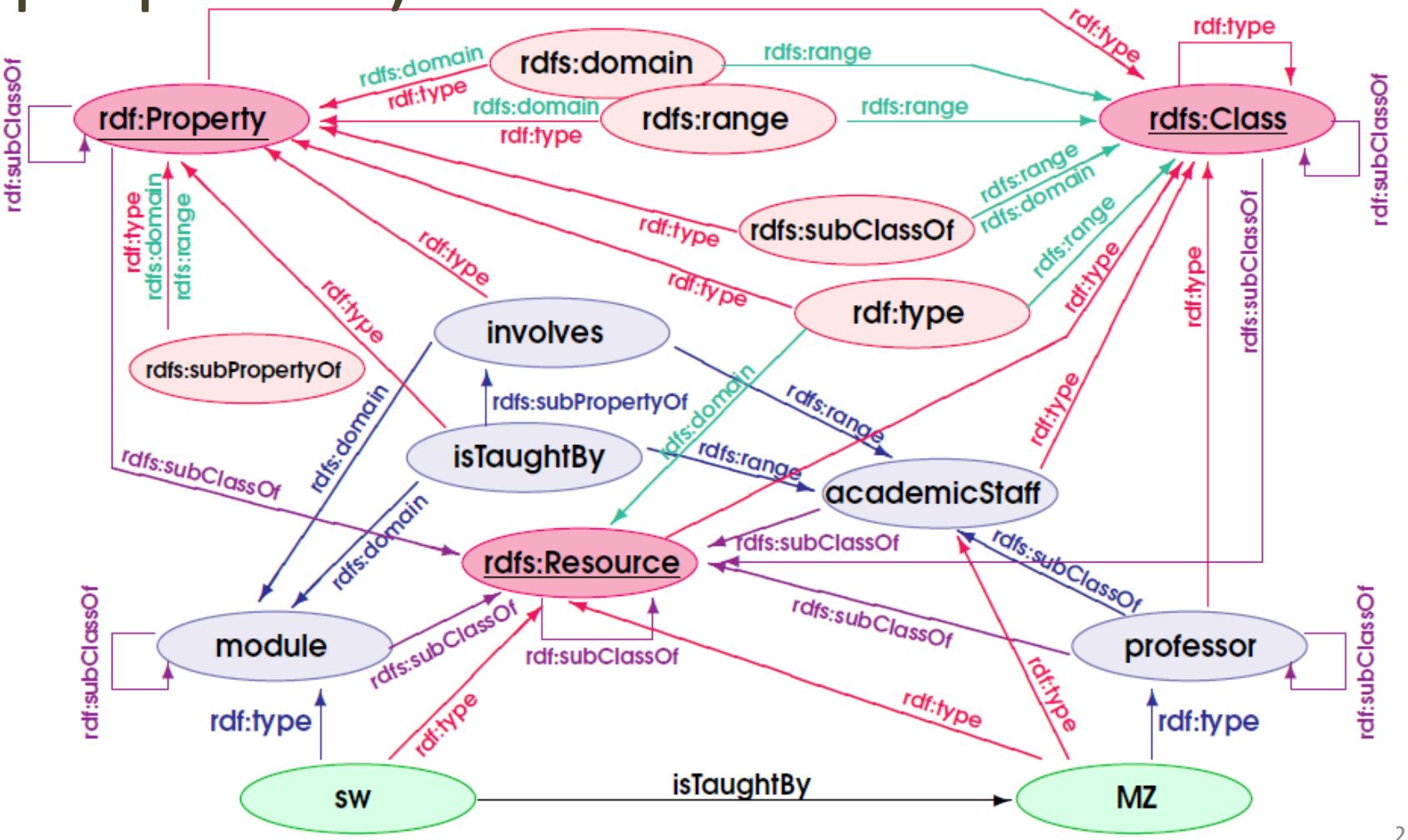
Example (properties, domains, ranges)



Example (reification of properties)



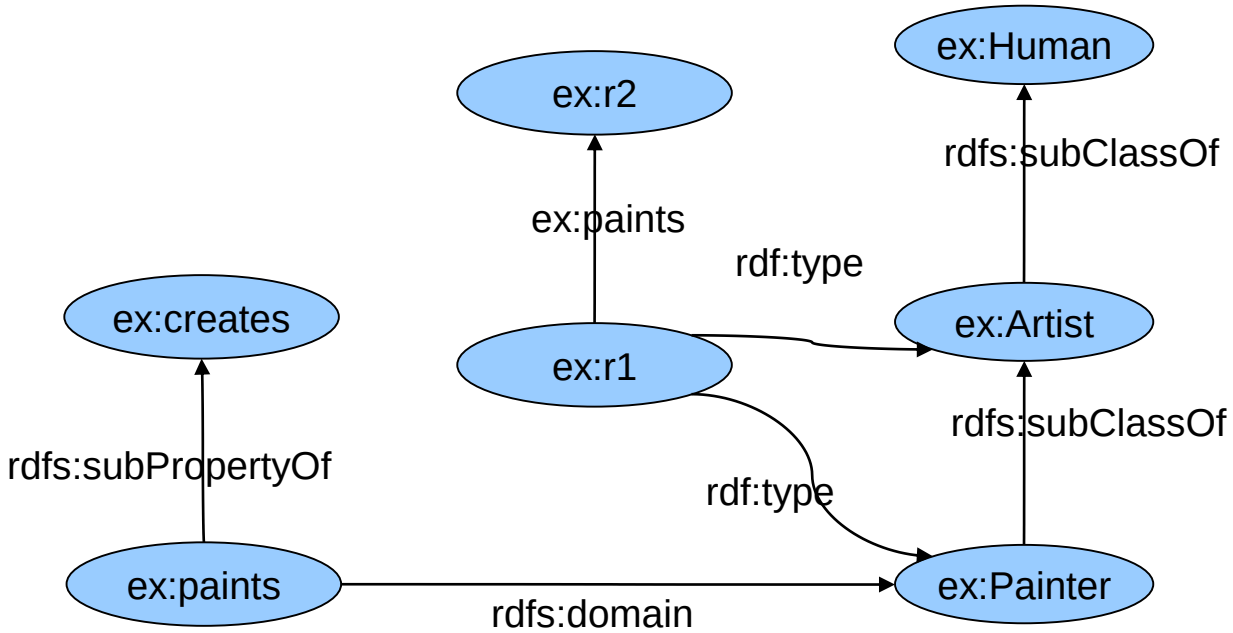
Example (reification of properties+axiomatic properties)



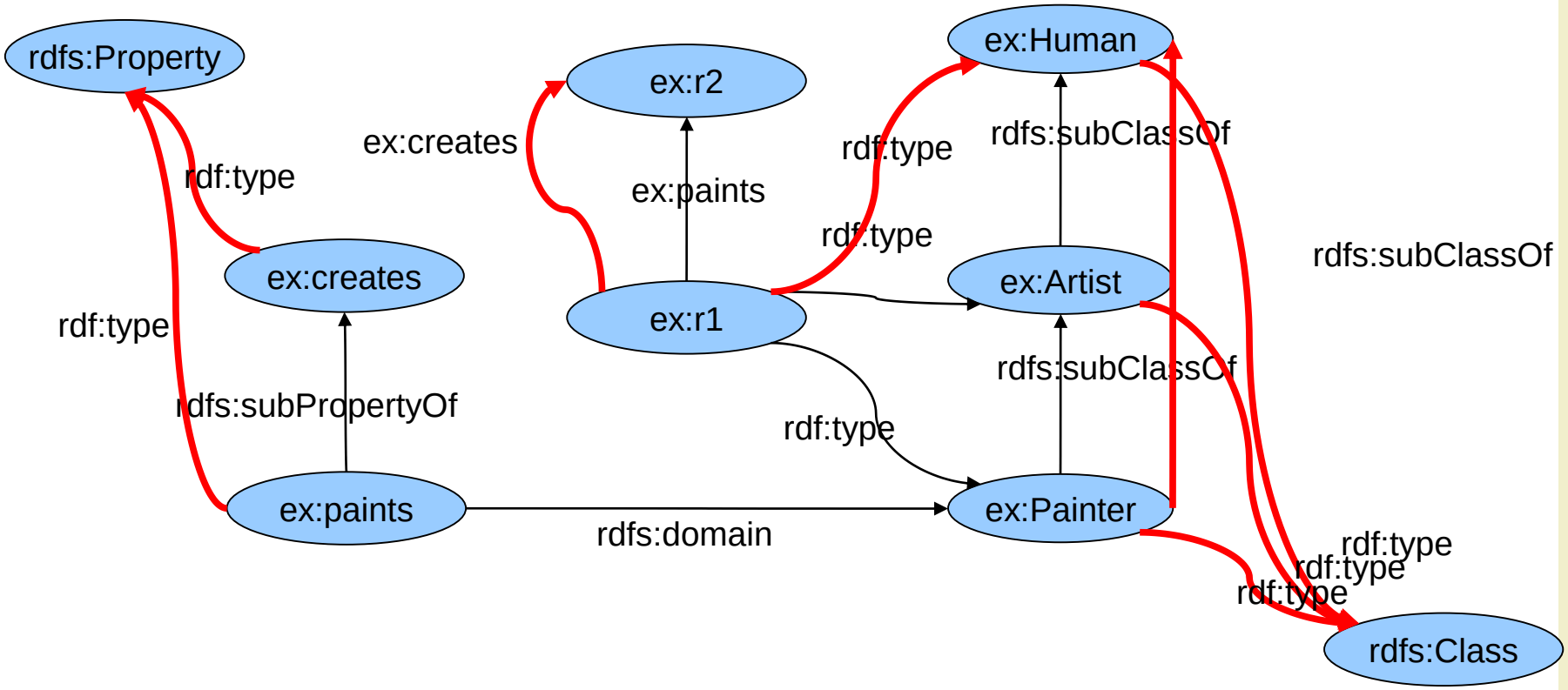
Closure and Reduction

- 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.

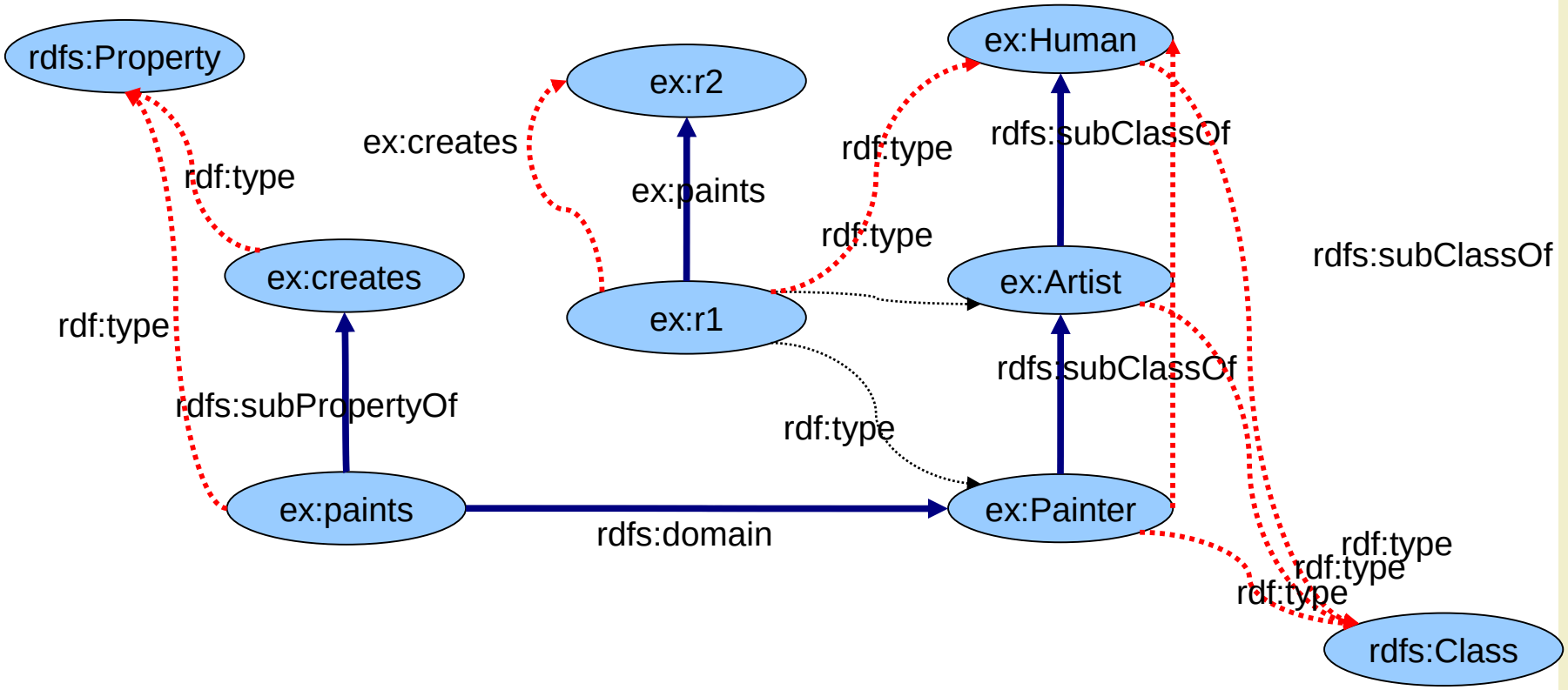
Example


ex:Human
ex:r2
rdfs:subClassOf
ex:paints
rdf:type
ex:creates
ex:Artist
ex:r1
rdfs:subClassOf
rdfs:subPropertyOf
rdf:type
ex:paints
ex:Painter
rdfs:domain

Example: closure



Example: reduction



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-semanic-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/