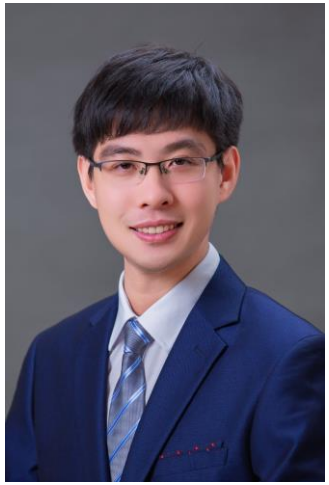
Web Science

Prof. Tianxing Wu
School of Computer Science and Engineering
Southeast University

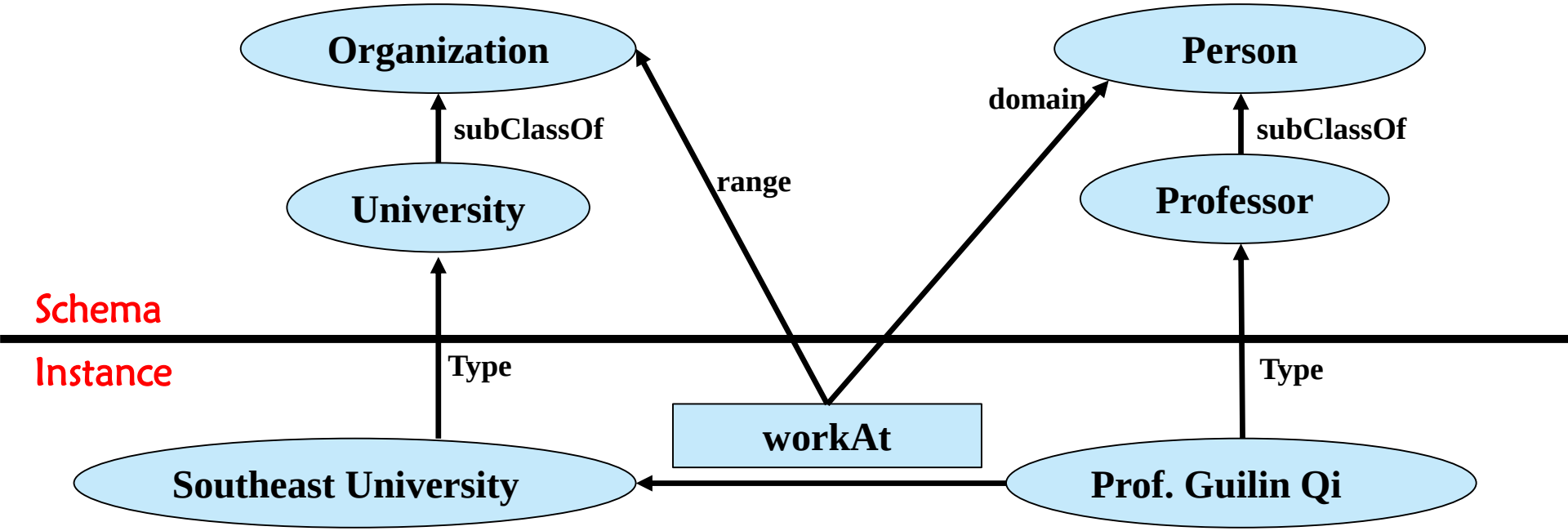
Email: tianxingwu@seu.edu.cn
Contact: 15077889931



Key Knowledge in the Last Lecture

• **Knowledge Engineering:** all technologies behind leveraging knowledge-based systems to solve domain-specific issues; an important branch of AI

• **Knowledge Graph:** the representative knowledge engineering technique under the background of the continuous development of World Wide Web



Key Knowledge in the Last Lecture

- **Knowledge Graph Projects:** DBpedia, YAGO, Wikidata, BabelNet, Zhishi.me…

- **Knowledge Graph Techniques:** knowledge representation, knowledge reasoning, knowledge extraction, knowledge fusion, knowledge storage and querying, knowledge-based question answering…

- **Knowledge Graph Applications:** Finance, Digital Library, Law, Publisher…

Introduction to Knowledge Graph Representation

Knowledge Representation

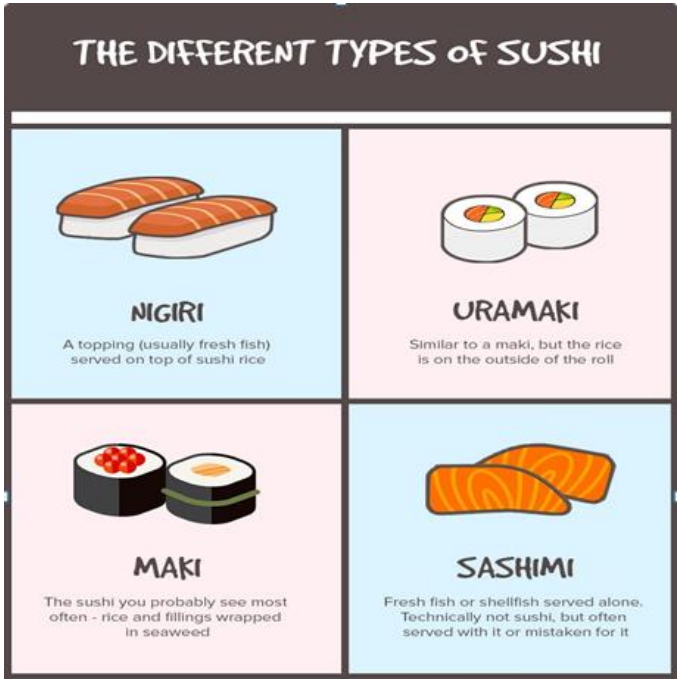
Knowledge representation is to use the representation that computer can deal with to express human knowledge.

- mathematical logic,
- production rules,
- semantic network,
- frames…

Knowledge Representation

Mathematical logic: also called form logic, symbolic logic,…

- Symbolic representations are important KR methods



Natural Language: Nigiri is a type of Sushi which has ingredient Rice and Fish

First Order Predicate Logic:

$\forall x.[\mathbf{Nigiri}(x) \rightarrow \mathbf{Sushi}(x) \wedge \exists y.[\mathbf{hasIngredient}(x,y) \wedge \mathbf{Rice}(y)] \wedge \exists z.[\mathbf{hasIngredient}(x,z) \wedge \mathbf{Fish}(z)]]$

Description Logic:

$\mathbf{Nigiri} \sqsubseteq \mathbf{Sushi} \sqcap \exists \mathbf{hasIngredient}.\mathbf{Rice} \sqcap \exists \mathbf{hasIngredient}.\mathbf{Fish}$

Example from Prof. Yizheng Zhao

Knowledge Representation

Production Rules: If A Then B(CF = [0, 1]), A→B;

- A: Premise, B: Conclusion;
- CF (Certainty Factor), i.e., rule confidence;
- Production rules can model uncertain knowledge with CF;

If 本微生物的染色斑是革兰氏阴性

本微生物的形状呈杆状

病人是中间宿主

Then 该微生物是绿脓杆菌，置信度为CF=0.6

Knowledge Representation

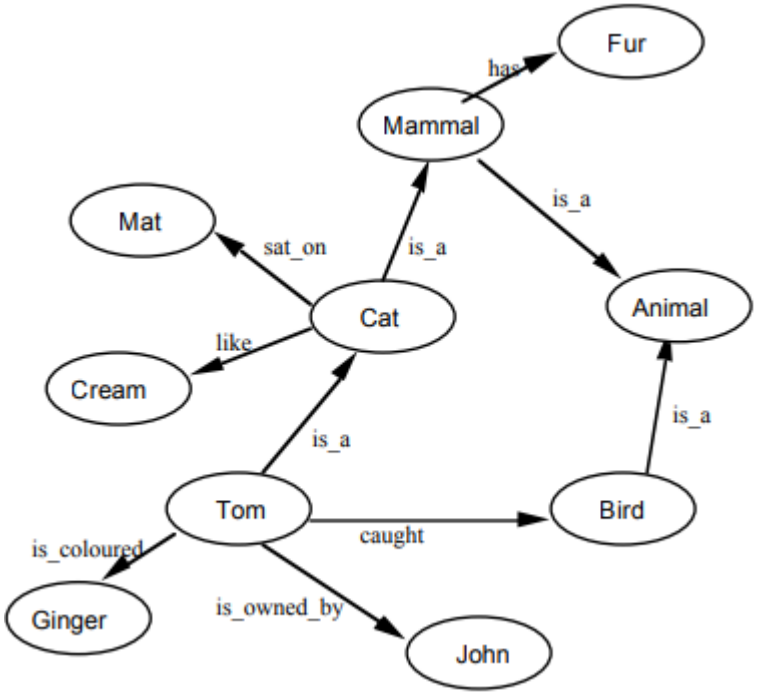
Semantic Network:

- a knowledge representation scheme involving nodes and links (arcs or arrows) between nodes, a directed graph;

- nodes: objects or concepts, links(directed): relations between nodes;

- use human language to label nodes and links.

- does have widely accepted formal representation.



Knowledge Representation

Frame:

- Each piece of information about a particular frame is held in a slot;
- The information can contain:
  - Facts or Data
    - Values (called facets)
  - Procedures (also called procedural attachments)
    - IF-NEEDED : deferred evaluation
    - IF-ADDED : updates linked information
  - Default Values
    - For Data
    - For Procedures
  - Other Frames or Subframes

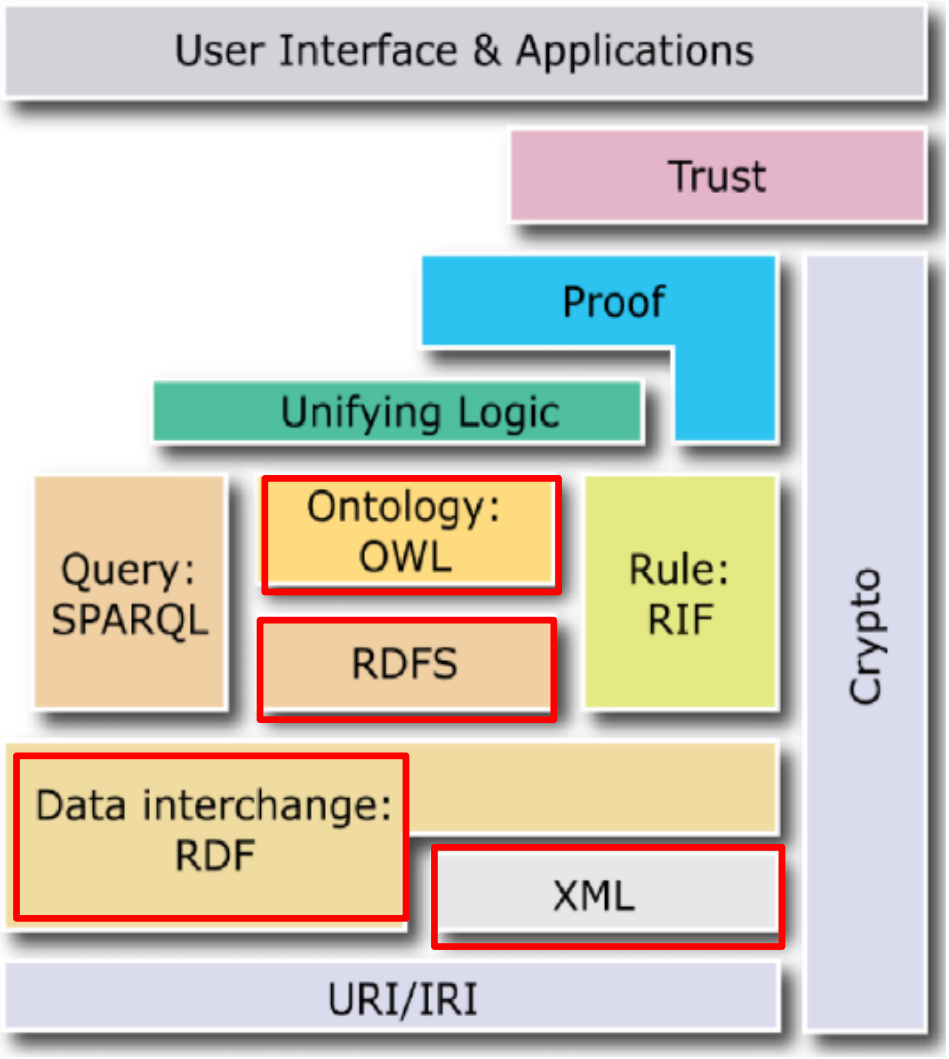
Slot	Value	Type
ALEX	_	(This Frame)
NAME	Alex	(key value)
ISA	Boy	(parent frame)
SEX	Male	(inheritance value)
AGE	IF-NEEDED: Subtract(current,BIRTHDATE);	(procedural attachment)
HOME	100 Main St.	(instance value)
BIRTHDATE	8/4/2000	(instance value)
FAVORITE_FOOD	Spaghetti	(instance value)
CLIMBS	Trees	(instance value)
BODY_TYPE	Wiry	(instance value)
NUM_LEGS	1	(exception)

Knowledge Representation

Knowledge representation is to use the representation that computer can deal with to express human knowledge.

Knowledge graph representation is to use the representation that computer can deal with to express the knowledge in knowledge graph.

Knowledge Graph Representation



W3C Stack

XML:
- Markup language for data exchange
- Surface syntax, no semantics
- XML Schema: describes structure of XML documents

RDF:
- Data model for "relations" between "things"

RDF Schema (RDFS):
- RDF vocabulary definition language

OWL:
- A more expressive vocabulary definition language

XML、RDF、RDFS

- **XML：**
  eXtensible Markup Language (XML) 1.0 (Fifth Edition)
  http://www.w3.org/TR/xml/

- **RDF**
- **RDFS**

XML

- **XML**：
  - A markup language for documents containing structured information;
  - Origins from SGML (Standard Generalized Markup Language)

HTML (HyperText Markup Language):
for documents designed to be displayed in a web browser

HTML document：

tag ← This book has the title KG.

Browser shows:

This book has the title **KG**.

XML

- **XML：**
  - Version 1.0 introduced by World Wide Web Consortium (W3C) in 1998

- **Comparison：**

XML

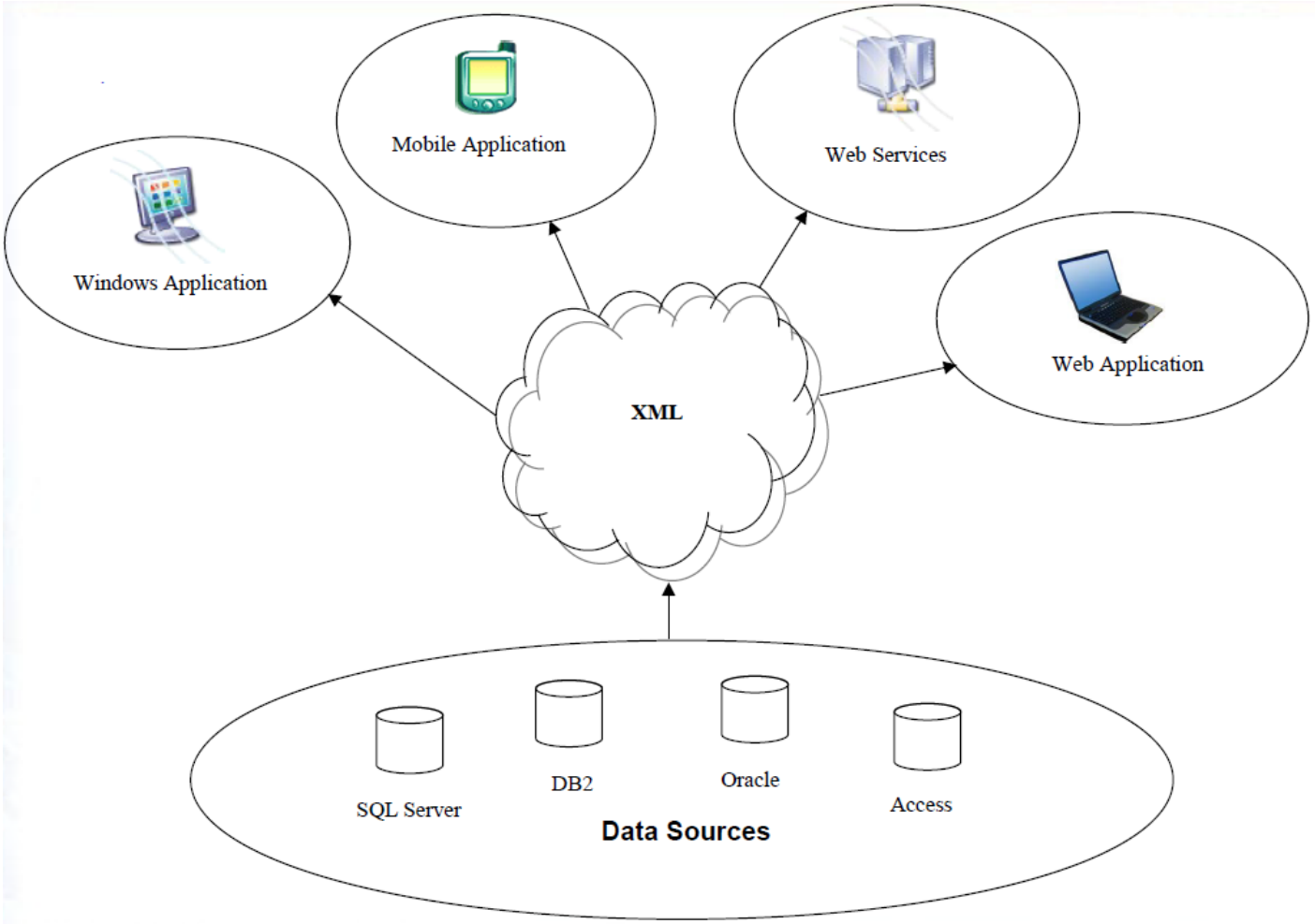
- Extensible set of tags
- Content orientated
- Standard Data infrastructure
- Allows multiple output forms

HTML

- Fixed set of tags
- Presentation oriented
- No data validation capabilities
- Single presentation

XML

XML is designed for data exchange.



XML Syntax

- Each XML document is a text document;

```
<bib name="IT">
  <book id="b001" year="1994">
    <title>TCP/IP Illustrated</title>
    <author><last>Stevens</last><first>W.</first></author>
    <publisher>Addison-Wesley</publisher>
  </book>
  <book id="b002" year="2000">
    <title>Data on the Web</title>
    <author><last>Abiteboul</last><first>Serge</first></author>
    <author><last>Buneman</last><first>Peter</first></author>
    <author><last>Suciu</last><first>Dan</first></author>
    <publisher>Morgan Kaufmann</publisher>
  </book>
  <journal id="j001" year="1998">
    <title>XML</title>
    <editor><last>Date</last><first>C.</first></editor>
    <editor><last>Gerbarg</last><first>M.</first></editor>
  </journal>
</bib>
```

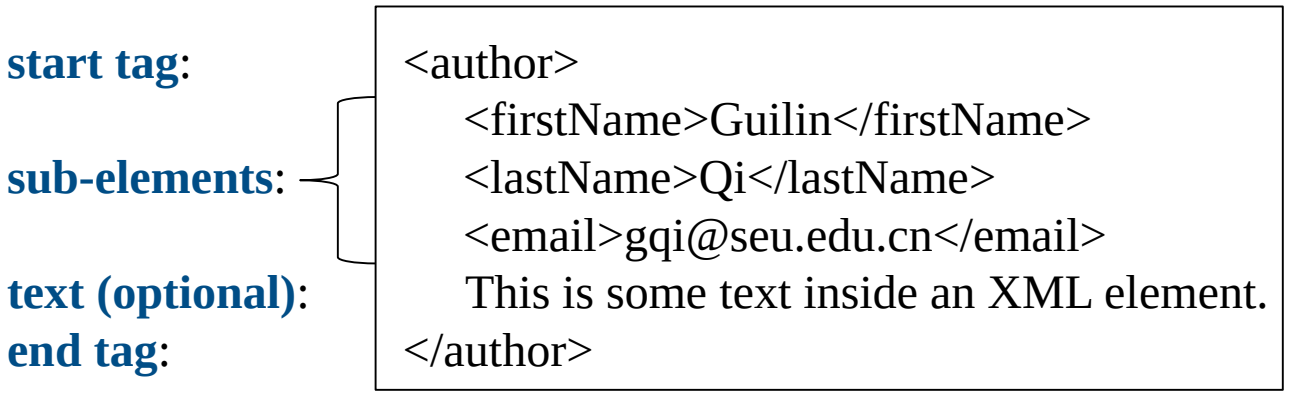
- Each XML document begins with a declaration containing
  - the **version number** of the used standard
  - and optionally, the **character encoding (default: Unicode)**

Example:

```
<?xml version="1.0" encoding="GB2312" ?>
```

XML Syntax

- XML Elements:
  - describe objects which are enclosed in matching tag-pairs,
    tags are case-sensitive: <CITY> <City> <city>
  - can contain text and/or further XML elements, arbitrarily nested;
  - empty elements can be abbreviated:
    e.g. <year></year> can be written as <year/>
  - the outermost element is called root element (there is only one)



XML Syntax

- **XML Attributes:**
  - an attribute is a name-value pair separated by an equal sign (=), example: <City ZIP= "210000" > Nanjing</City>

```
<author>
    <firstName>Guilin</firstName>
    <lastName>Qi</lastName>
    <email>gqi@seu.edu.cn</email>
     This is some text inside an XML element.
</author>
```

```
<author email="gqi@seu.edu.cn">
    <firstName>Guilin</firstName>
    <lastName>Qi</lastName>
     This is some text inside an XML element.
</author>
```

XML Syntax

Example: an XML document → **tree structure**

```
<?xml version="1.0" encoding="GB2312" ?>
<库存清单>
  <库存物品>
    <名称>毛巾</名称>
    <规格 单位="cm">25*50</规格>
    <数量 单位="条">400</数量>
    <生产厂家>中国棉纺厂</生产厂家>
    <生产日期>20010-09-01</生产日期>
    <进货价 单位="元">2.5</进货价>
    <零售价 单位="元">6.5</零售价>
  </库存物品>
  <库存物品> … </库存物品>
  …
</库存清单>
```

Authoring guidelines:

1. All elements must have an end tag.
2. All elements must be cleanly nested (overlapping elements are not allowed).
3. All attribute values must be enclosed in quotation marks.
4. Each document must have a unique first element, the root node.

Exercise

Please specify the errors in the following XML document.

```
<book>
 <title>Knowledge Graph</Title>
 <author>
        <firstName>Guilin</firstName>
        <lastName>Qi</lastName>
        <email>gqi@seu.edu.cn</email>
         This is some text inside an XML element.
  </author>
  <author>
        <firstName>Tianxing<lastName>
       </firstName>Wu</lastName>
        <email>tianxingwu@seu.edu.cn</email>
 </author>
```

XML Syntax

- XML Comment:
  example: <!-- **comment is written here**-->

- XML Pre-defined Entities:

```
<文章>
   <段落>
      <html>
        <head><title></title></head>
          <body>
             <h1>东南大学</h1>
          </body>
      </html>
  </段落>
</文章>
```

1. & → &
2. < → <
3. > → >
4. ' → ‘
5. " → “

problem: not user-friendly

XML Syntax

- XML Comment:
  example: <!-- **comment is written here**-->

- Solution: CDATA → <![CDATA[…]]>

```
<文章>
    <段落><![CDATA[
        <html>
            <head><title></title></head>
            <body>
                <h1>东南大学</h1>
            </body>
        </html>]]>
  </段落>
</文章>
```

XML Namespaces

- XML Namespaces provide a method to avoid element name conflicts.

XML document_1

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

XML document_2

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

- Solution : Adding prefixes

XML document_1

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

XML document_2

```
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

XML Namespaces

- XML Namespaces provide a method to avoid element name conflicts.

XML document_1

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

XML document_2

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

- Defining a namespace for each prefix: use an xmlns attribute

XML document_1

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

XML document_2

```
<f:table xmlns:f="https://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

The namespace can be defined by an xmlns attribute in the start tag of an element.

XML Namespaces

- XML Namespaces provide a method to avoid element name conflicts.
- The namespace can be defined by an xmlns attribute in the start tag of an element.

XML document_1

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

XML document_2

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

- Defining the default namespaces

XML document_1

```
<table xmlns="http://www.w3.org/TR/html4/">
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

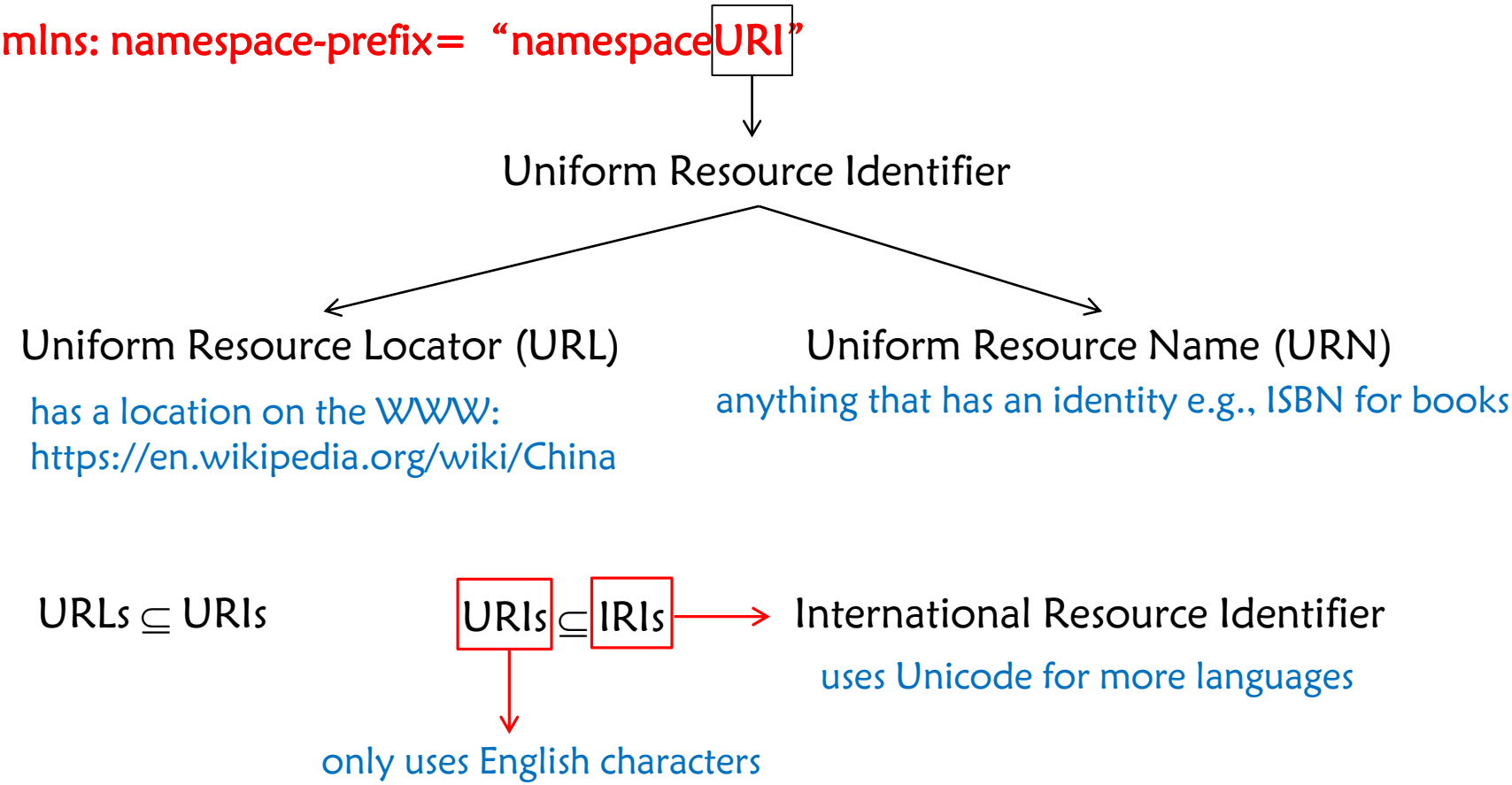
XML document_2

```
<table xmlns="https://www.w3schools.com/furniture">
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

XML Namespaces

- XML Namespace Syntax:

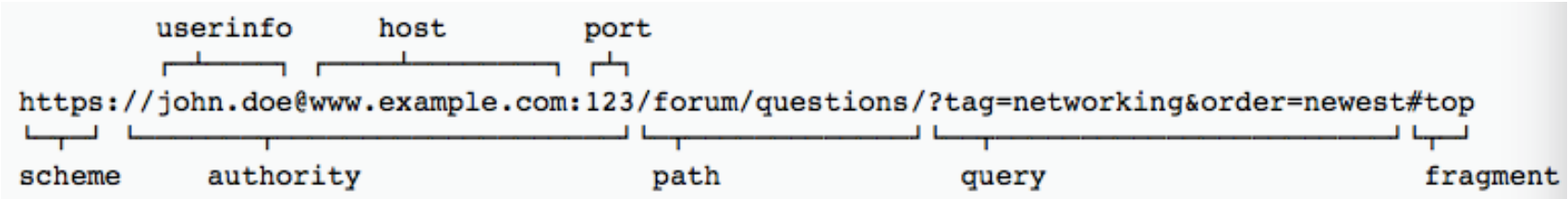
  xmlns: namespace-prefix= "namespaceURI"

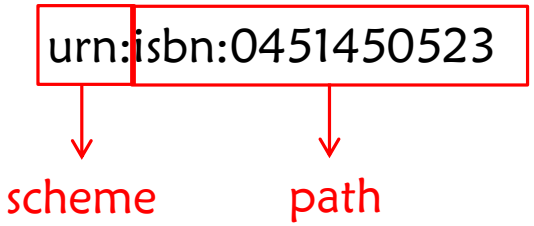


XML Namespaces

- **URI (IRI) format**

  URI = scheme:[//authority]path[?query][#fragment]
  -scheme: type of URI, e.g. http, ftp, mailto, file, urn
  -authority: optional; typically a domain name
  -path: e.g., file system path
  -query: optional; provides non-hierarchical information. Usually for parameters, e.g. for a web service
  -fragment: optional; often used to address part of a retrieved resource, e.g. section of a HTML file; the set of characters is limited to US-ASCII





XML Schema

- Why do we need XML Schema?

1) <author>Guilin Qi</author>

2) <author name= “Guilin Qi” />

3) <author><fullName>Guilin Qi</fullName></author>

4) <author><firstName>Guilin</firstName>
<secondName>Qi</secondName></author>

5) <author givenName= “Guilin” surname= “Qi” />

XML Schema

- Why do we need XML Schema?

  - These degrees of freedom get in the way when exchanging XML documents between applications!

  - It is necessary to come up with agreements about the structure of the information, including the names of tags and attributes, and whether certain sub-elements are required or not.

  - XML Schema is a W3C standard which provides for this.

  - XML schemas are themselves written in XML.

  - An XML document validated against an XML Schema is "Well Formed" and "Valid".

XML Schema

- **W3C Recommended XML Schema Language:**
  XML Schema Definition (XSD), https://www.w3.org/2001/XMLSchema#

```
<?xml version= “1.1” encoding= “utf-16” ?>
<xsd:schema xmlns:xsd= “http://www.w3.org/2001/XMLSchema” >
    <xsd:element name= “author” type= “xsd:string”
                 minOccurs= “1” maxOccurs= “unbounded” >
        <xsd:attribute name= “email” type= “xsd:string” use= “required” />
        <xsd:attribute name= “homepage” type= “xsd:anyURI” use= “optional” /
    </xsd:element>
</xsd:schema>
```

XML Schema

- According to the XML Schema, the XML document should write as:

```
…
<author email= “gqi@seu.edu.cn” homepage= “http://cse.seu..” >
      Guilin Qi
</author>
<author email= “tianxingwu@seu.edu.cn” >
      Tianxing Wu
</author>
…
```

XML: Summary

- **XML:**
  a markup language for data exchange,
  a serialization format,
  a syntax without semantics
  (lack term meaning/definition and clear relations between terms).

- **XML Schema:**
  specifies how to formally describe the elements in XML documents.

- **Others:**
  - Accessing and manipulating XML documents: DOM, SAX;
  - Styling and presenting XML documents: XSLT.

XML、RDF、RDFS

- **XML**
- **RDF：**
  Resource Description Framework (RDF)
  https://www.w3.org/TR/rdf-concepts/

- **RDFS**

RDF

- **RDF:**
  - the data model of Semantic Technologies and of the Semantic Web;
  - structures metadata about Web sites, pages, etc.:
    1. Page author, creator, publisher, editor, …
    2. Data about them: email, phone, job, …
  - can be used for machine-readable data exchange ;
  - is introduced in W3C Recommendation of 1999 (Version 1.0);
  - uses XML as main syntax for serialization.

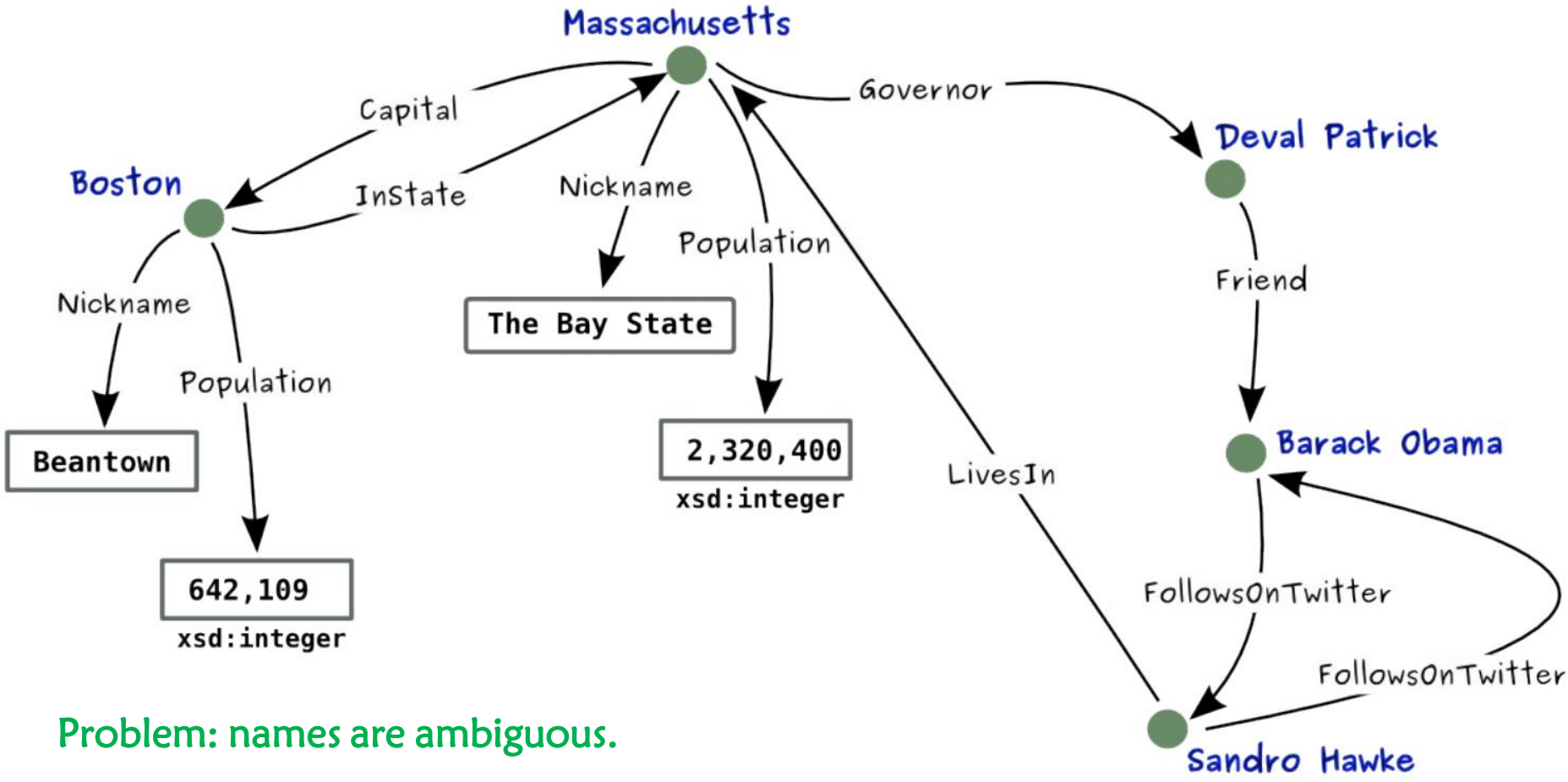
RDF

- **RDF:** a data model about triples: (subject , predicate , object)

(Southeast University, locateAt, Nanjing)

- **A RDF knowledge base:** a directed labeled graph, **i.e.,** Knowledge Graph

RDF Graph: Directed Labeled Graph

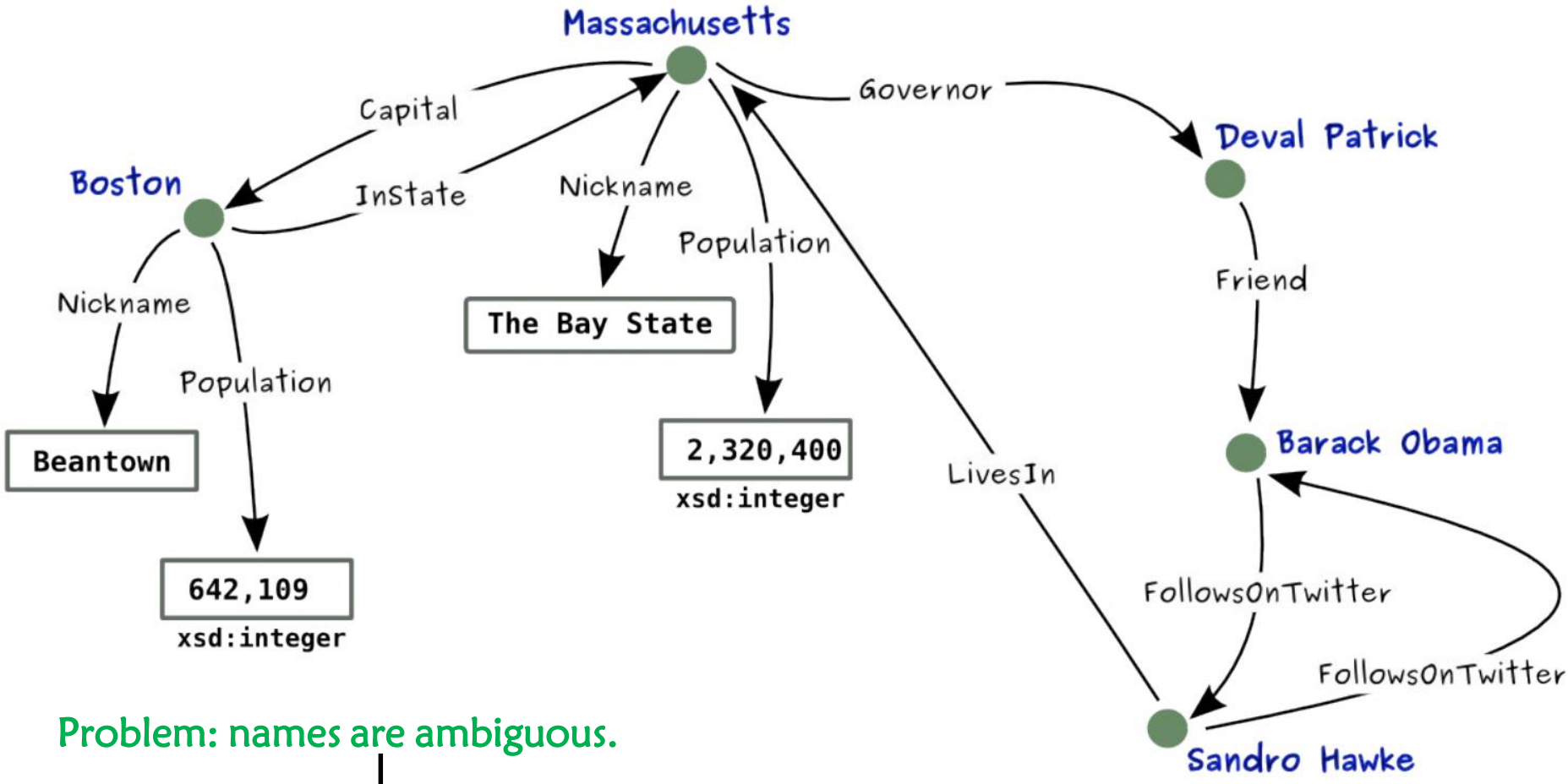


Problem: names are ambiguous.

How many things are named “Boston”?

What is meant by “Nickname”?

RDF Graph: Directed Labeled Graph



Problem: names are ambiguous.

↓

We need unambiguous identifiers, such as IRI, URI, URL on the Web.

RDF : QName

- **QName**: used in RDF as shorthand for long URIs (IRIs)

  Example:

  If
  prefix foo is bound to http://example.com/
  then
  foo:bar
  expands to
  http://example.com/bar

RDF : QName

- **QName**: used in RDF as shorthand for long URIs (IRIs)

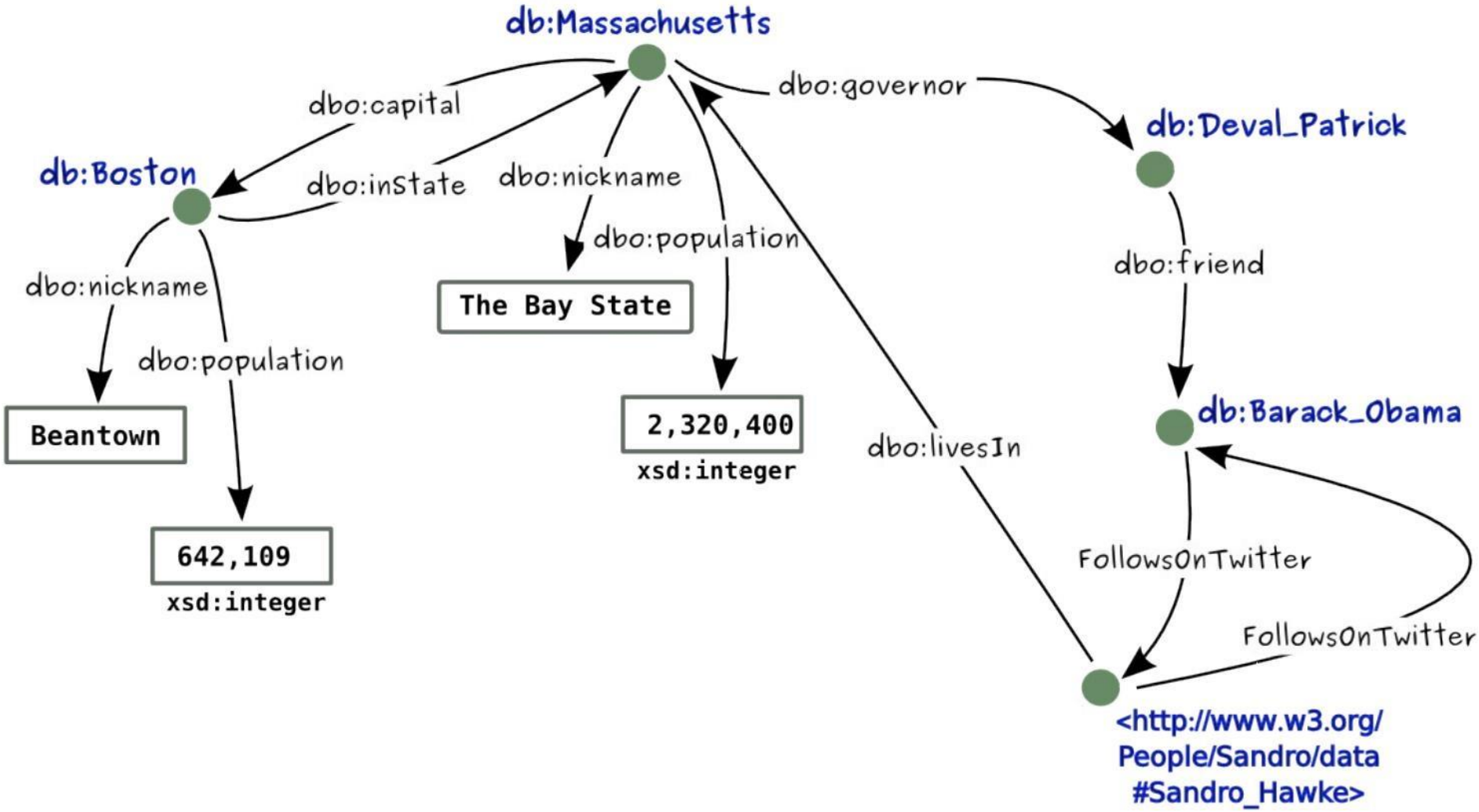
  How many things are named “Boston”?

  - **URI:** http://dbpedia.org/resource/Boston
  - **QName:** db:Boston

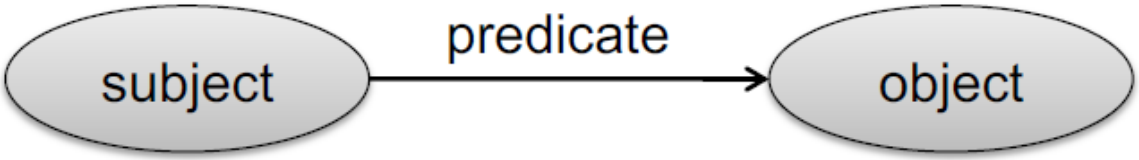
  What is meant by “Nickname”?

  - **URI:** http://example.org/terms/nickname
  - **QName:** dbo:nickname

RDF Graph: Use URIs and QNames



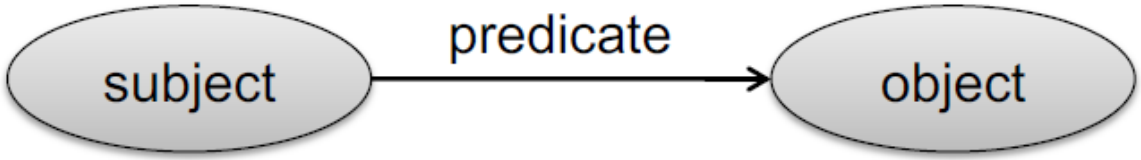
RDF Triple (Statement)



- **Subjects**: Resource or blank node
- **Predicates**: Resource
- **Object**: Resource, literal or blank node

Resources are: IRIs

RDF Triple (Statement)

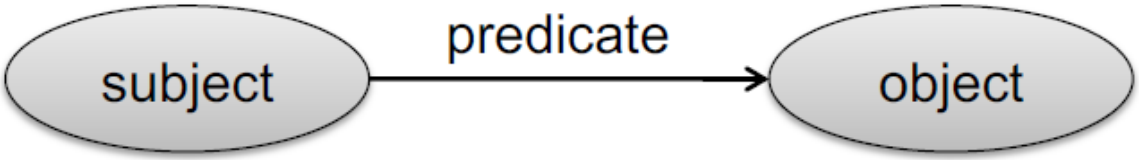


- **Subject**: Resource or blank node
- **Predicate**: Resource
- **Object**: Resource, literal or blank node

Literals are:

- data values;
- encoded as strings;
- interpreted by datatypes;
- treated the same as strings without datatypes, called plain literal;
  - A plain literal may have a language tag;
  - Datatypes are not defined by RDF, but usually from XML Schema.

RDF Triple (Statement)

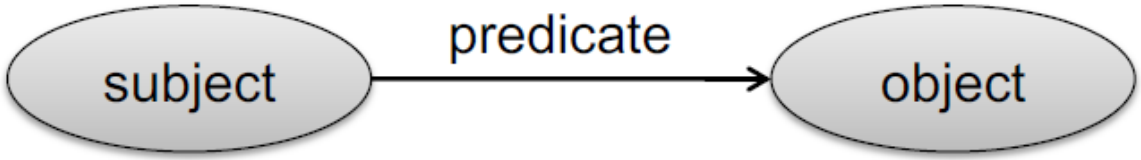


- **Subjects**: Resource or blank node
- **Predicates**: Resource
- **Object**: Resource, literal or blank node

Typed Literals:
"Beantown"^^xsd:string
"The Bay State" ^^xsd:string

Plain literal and literals with language tags:
"France" "France"@en "France"@fr
"法国"@zh "Frankreich"@de

RDF Triple (Statement)



- **Subjects**: Resource or blank node
- **Predicates**: Resource
- **Object**: Resource, literal or blank node

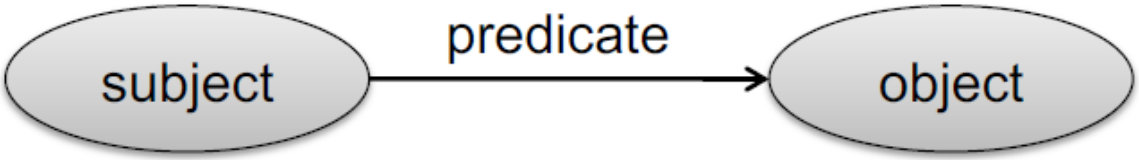
Equalities for Literals:
"001"^^xsd:integer = "1"^^xsd:integer
"123.0"^^xsd:decimal = "00123"^^xsd:integer (based on datatype hierarchy)

Built-in Datatype Hierarchy
anyType
all complex types
anySimpleType
ur-types
primitive types
duration
dateTime
time
date
gYearMonth
gYear
gMonthDay
gDay
gMonth
boolean
base64Binary
hexBinary
float
double
anyURI
QName
NOTATION
string
decimal
derived types
normalizedString
integer
token
nonPositiveInteger
long
nonNegativeInteger
language
Name
NMTOKEN
negativeInteger
int
unsignedLong
positiveInteger
NCName
NMTOKENS
short
unsignedInt
ID
IDREF
ENTITY
byte
unsignedShort
IDREFS
ENTITIES
unsignedByte
derived by restriction
derived by list
derived by extension or
ur types
built-in primitive types
built-in derived types

Exercise

Does the datatype “德 国” equals to “德 国” @ zh ?

RDF Triple (Statement)

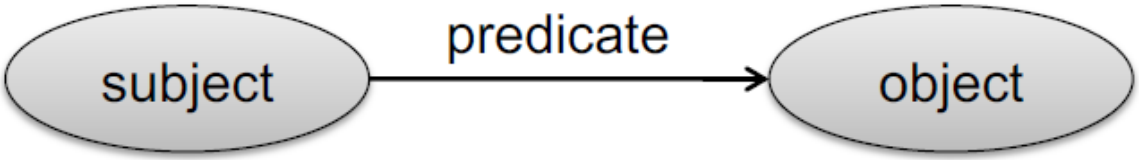


- **Subjects**: Resource or blank node
- **Predicates**: Resource
- **Object**: Resource, literal or blank node

Blank node: unnamed resource or complex node (later)

- The set of blank nodes, the set of all IRIs (named resources) and the set of all literals are pairwise disjoint;
- Representation of blank nodes is syntax-dependent: underline+colon+ID (Turtle syntax): _:xyz, _:bn;
- The scope of the ID of a blank node is only the document to which it belongs.

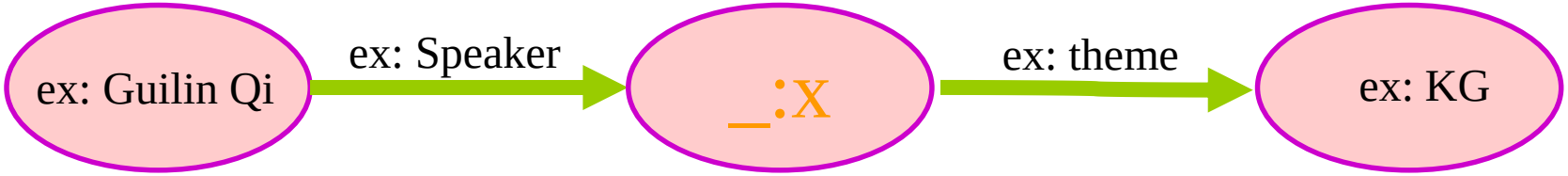
RDF Triple (Statement)



- **Subjects**: Resource or blank node
- **Predicates**: Resource
- **Object**: Resource, literal or blank node

Blank node: unnamed resource

Example: Guilin Qi is a speaker of a KG lecture.



@prefix ex: <http://example.org/> .

RDF Syntax

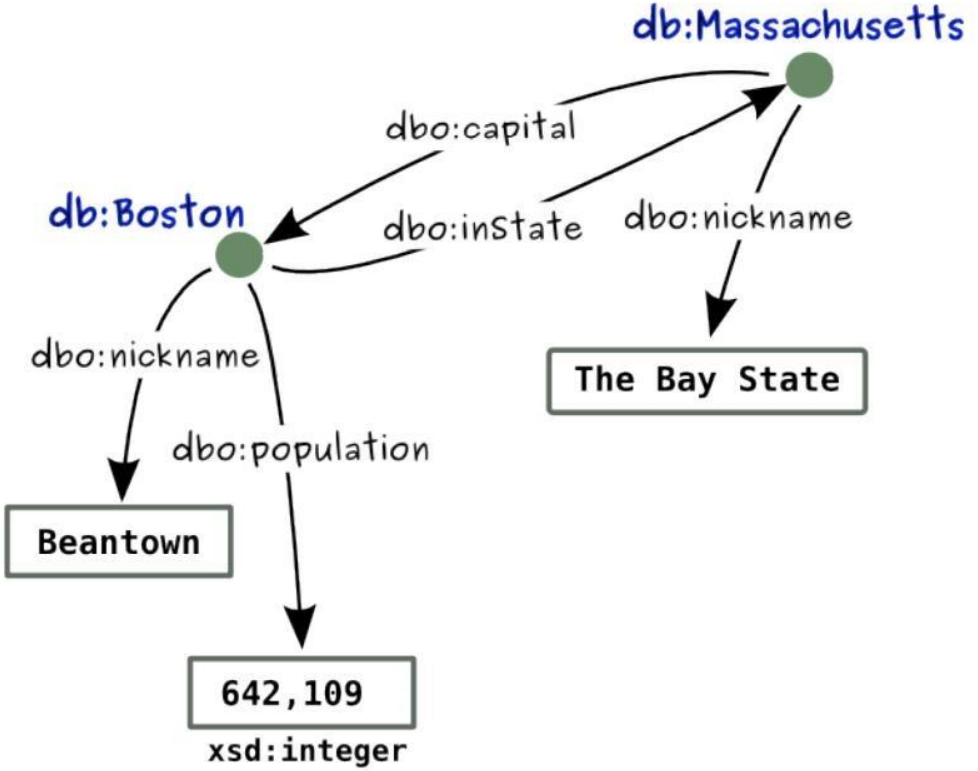
- Serialization formats (Syntaxes):
  - Turtle, N-Triples, N-Quads, JSON-LD, N3, RDF/XML, RDF/JSON…

RDF Syntax

- **Turtle (Terse RDF Triple Language):**
  - list as S P O triples (easy to read)
  - IRIs are in <angle brackets>
  - triples end with a full-stop .
  - whitespaces are ignored

RDF Syntax

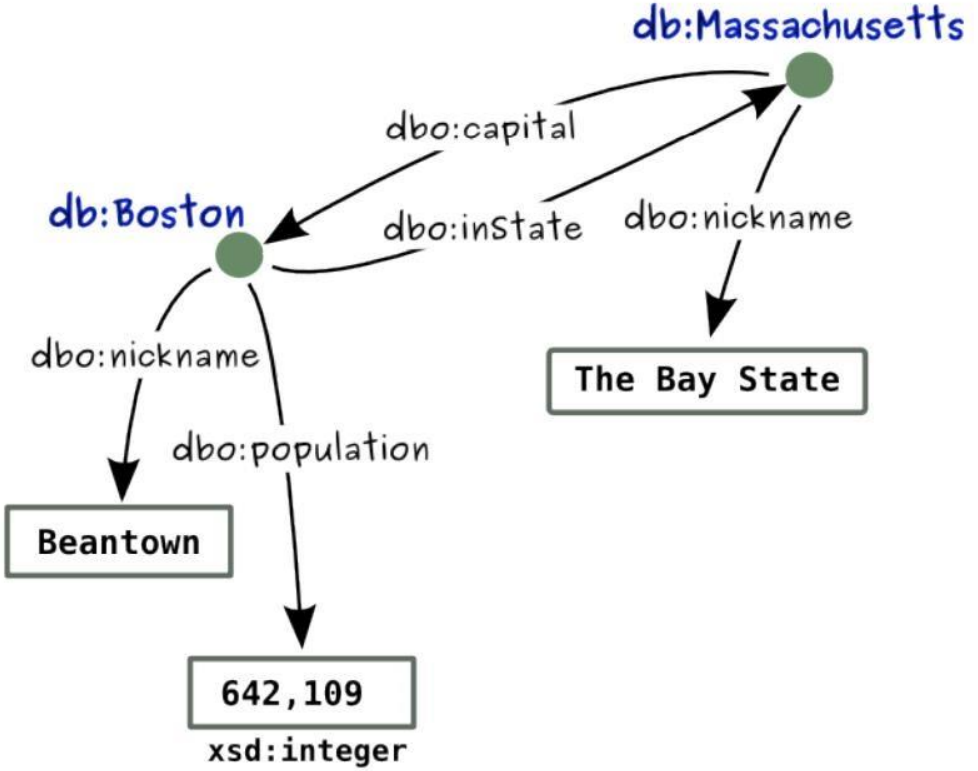
- **Turtle (Terse RDF Triple Language):**
  - list as S P O triples (easy to read)
  - IRIs are in <angle brackets>
  - triples end with a full-stop .
  - whitespaces are ignored



<http://dbpedia.org/resource/Massachusets> <http://example.org/terms/captial>
<http://dbpedia.org/resource/Boston> .
<http://dbpedia.org/resource/Massachusets> <http://example.org/terms/nickname>
"The Bay State" .
<http://dbpedia.org/resource/Boston> <http://example.org/terms/inState>
<http://dbpedia.org/resource/Massachusets> .
<http://dbpedia.org/resource/Boston> <http://example.org/terms/nickname>
"Beantown" .
<http://dbpedia.org/resource/Boston> <http://example.org/terms/population>
"642109"^^xsd:integer .

RDF Syntax

- **Turtle (Terse RDF Triple Language):**
  - list as S P O triples (easy to read)
  - IRIs are in <angle brackets>
  - triples end with a full-stop .
  - whitespaces are ignored

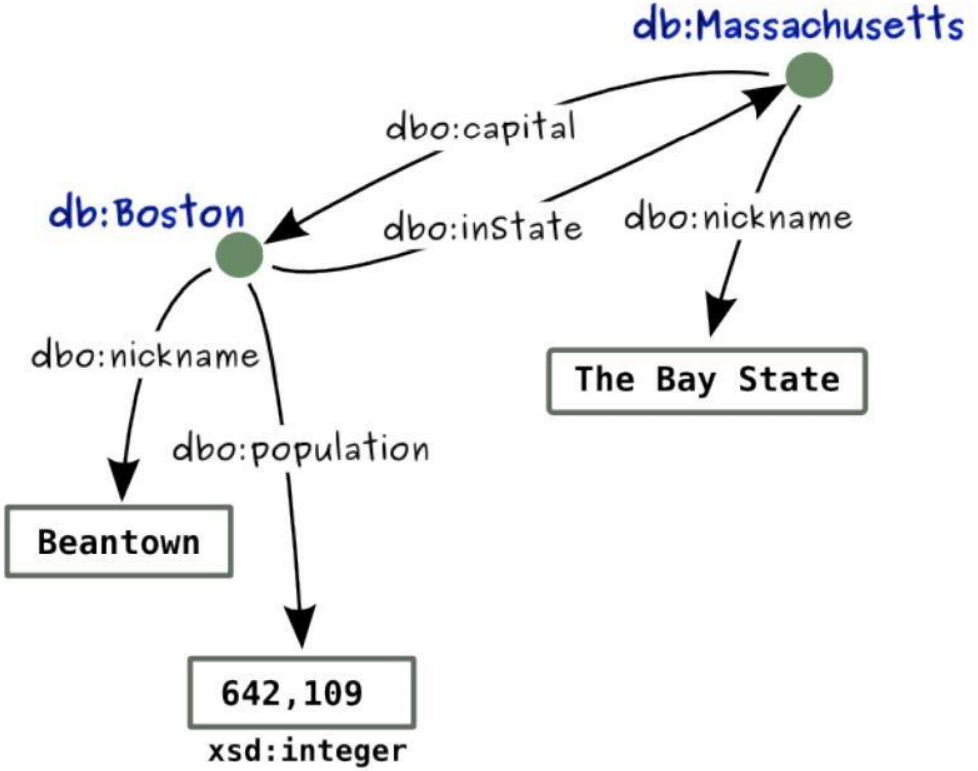


use QName

```
@prefix db: <http://dbpedia.org/resource/> .
@prefix dbo: <http://example.org/terms/> .

db:Massachusets  dbo:capital  db:Boston .
db:Massachusets  dbo:nickname  “The Bay State” .
db:Boston  dbo:inState  db:Massachusets .
db:Boston  dbo:nickname “Beantown” .
db:Boston  dbo:population “642109”^^xsd:integer .
```

RDF Syntax

- Turtle (Terse RDF Triple Language):
  - list as S P O triples (easy to read)
  - IRIs are in <angle brackets>
  - triples end with a full-stop .
  - whitespaces are ignored



use QName

1) Grouping of triples with the same subject using semi-colon ';' ;
2) Grouping of triples with the same subject and predicate using comma ',' .

```
@prefix db: <http://dbpedia.org/resource/> .
@prefix dbo: <http://example.org/terms/> .

db:Massachusets  dbo:captial   db:Boston ;
                               dbo:nickname  "The Bay State" .
db:Boston  dbo:inState  db:Massachusets ;
                 dbo:nickname  "Beantown" ;
                 dbo:population  "642109"^^xsd:integer .
```

RDF Syntax

- **RDF/XML:**
  - RDF is originally designed on basis of XML (data exchange format on the Web)
  - a lot of tools and libraries support XML

RDF Syntax

- **RDF/XML:**
  - Namespaces are used for disambiguating tags;
  - Tags belonging to the RDF language come with a fixed namespace, usually abbreviated "rdf".

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex ="http://example.org/">

  <rdf:Description rdf:about="http://semantic-web-book.org/uri">
    <ex:publishedBy>
      <rdf:Description rdf:about="http://crcpress.com/uri">
      </rdf:Description>
    </ex:publishedBy>
  </rdf:Description>

</rdf:RDF>
```

RDF Syntax

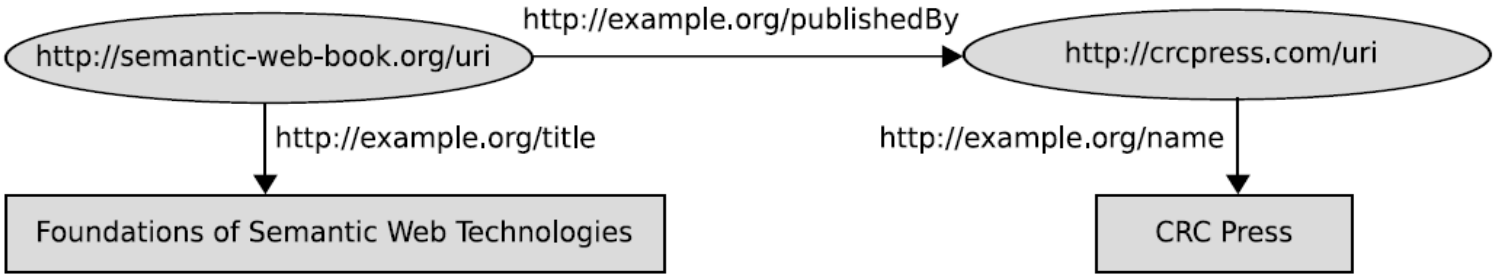
- **RDF/XML:**
  - Namespaces are used for disambiguating tags;
  - Tags belonging to the RDF language come with a fixed namespace, usually abbreviated "rdf".

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex ="http://example.org/">

  <rdf:Description rdf:about="http://semantic-web-book.org/uri">
    <ex:publishedBy>
      <rdf:Description rdf:about="http://crcpress.com/uri">
      </rdf:Description>
    </ex:publishedBy>
  </rdf:Description>

</rdf:RDF>
```

1) The XML declaration is followed by the root element of RDF documents: **<rdf:RDF>.**
2) The **<rdf:Description>** element contains the description of the resource identified by the **rdf:about** attribute.

RDF Syntax

- **RDF/XML:**
  - Untyped literals can be left as free text;
  - A subject can contain several property elements;
  - Object-descriptions can be used as subject-descriptions for further triples.




```
<rdf:Description rdf:about="http://semantic-web-book.org/uri">
  <ex:title>Foundations of Semantic Web Technologies</ex:title>
  <ex:publishedBy>
    <rdf:Description rdf:about="http://crcpress.com/uri">
      <ex:name>CRC Press</ex:name>
    </rdf:Description>
  </ex:publishedBy>
</rdf:Description>
```

RDF: N-ary Relations

Example:

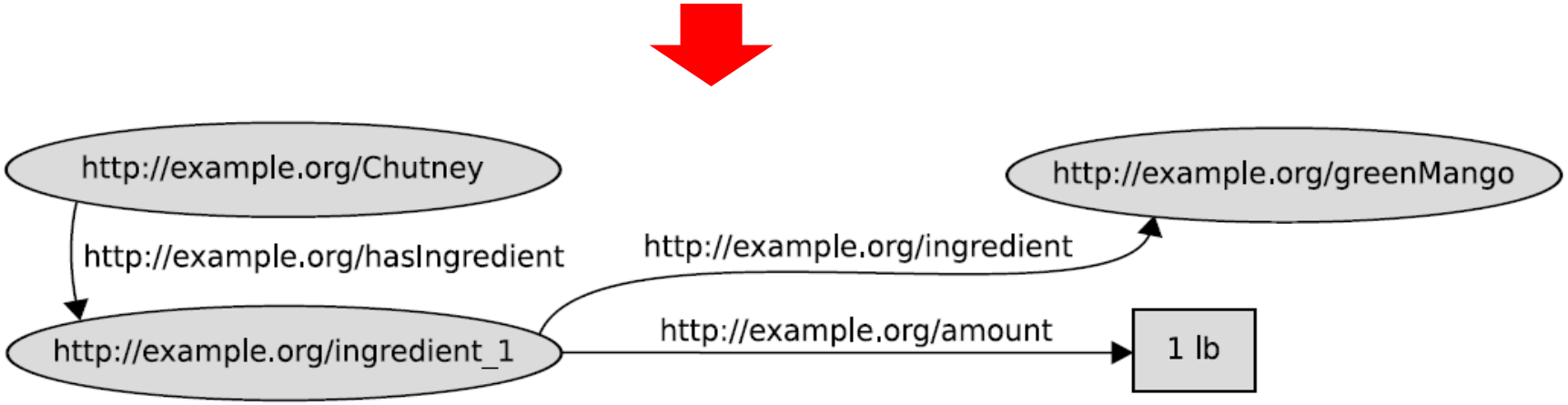
```
@prefix ex: <http://example.org/> .
ex:Chutney  ex:hasIngredient   "1lb green mango",
                               "1tsp. Cayenne pepper" .
```

```
@prefix ex: <http://example.org/> .
ex:Chutney  ex:ingredient   ex:greenMango;      ex:amount   "1lb" ;
            ex:ingredient   ex:CayennePepper;   ex:amount   "1tsp." .
```

RDF: N-ary Relations

Example:

```
@prefix ex: <http://example.org/> .
ex:Chutney  ex:ingredient   ex:greenMango;      ex:amount   "1lb" ;
            ex:ingredient   ex:CayennePepper;   ex:amount   "1tsp." .
```



RDF: N-ary Relations

Example:

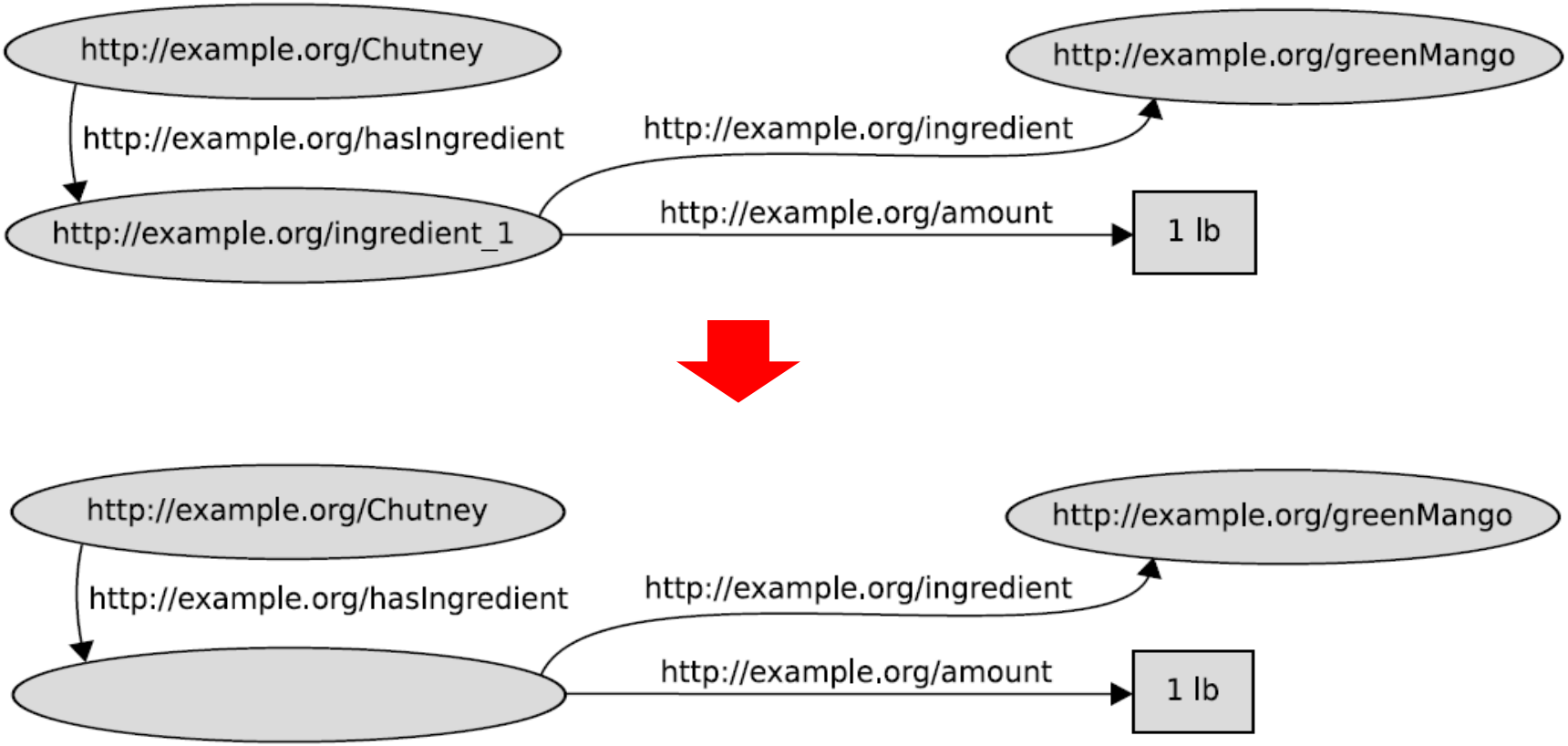

```
@prefix ex:  <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
ex:Chutney          ex:hasIngredient  ex:ingredient1 .
ex:ingredient1      rdf:value         ex:greenMango;
                    ex:amount         "1lb" .
```

RDF: N-ary Relations

- Not all relations are binary.
- All N-ary relations can be “encoded” as a set of binary relations using auxiliary nodes, and this process is called reification.

RDF: N-ary Relations and Blank Node

Example:



RDF: N-ary Relations and Blank Node

Example:


```
<rdf:Description rdf:about="http://example.org/Chutney">
   <ex:hasIngredient rdf:nodeID="id1" />
</rdf:Description>
<rdf:Description rdf:nodeID="id1">
   <ex:ingredient rdf:resource="http://example.org/greenMango" />
   <ex:amount>1lb</ex:amount>
</rdf:Description>
```

RDF: N-ary Relations and Blank Node

Example:

Shortcut:

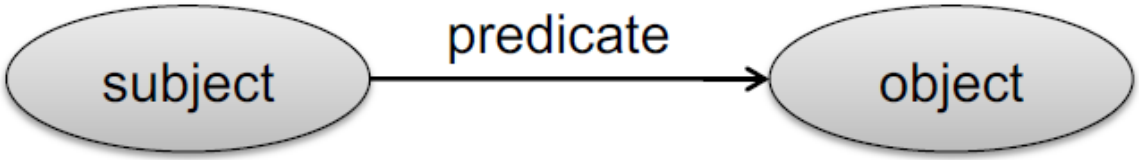
```
<rdf:Description rdf:about="http://example.org/Chutney">
   <ex:hasIngredient rdf:parseType="Resource">
     <ex:ingredient rdf:resource="http://example.org/greenMango" />
     <ex:amount>1lb</ex:amount>
   </ex:hasIngredient>
</rdf:Description>
```

RDF VS. XML

- IRIs solve the problem of term meaning.
- Triple-based data model describe relations or properties among terms.

RDF for Knowledge Graph

- RDF is one of the data models (most widely used) for knowledge graph.
- RDF provides a knowledge representation method for knowledge graph.



Semantic Web

- **Subjects**: Resource or blank-node
- **Predicates**: Resource
- **Object**: Resource, literal or blank-node

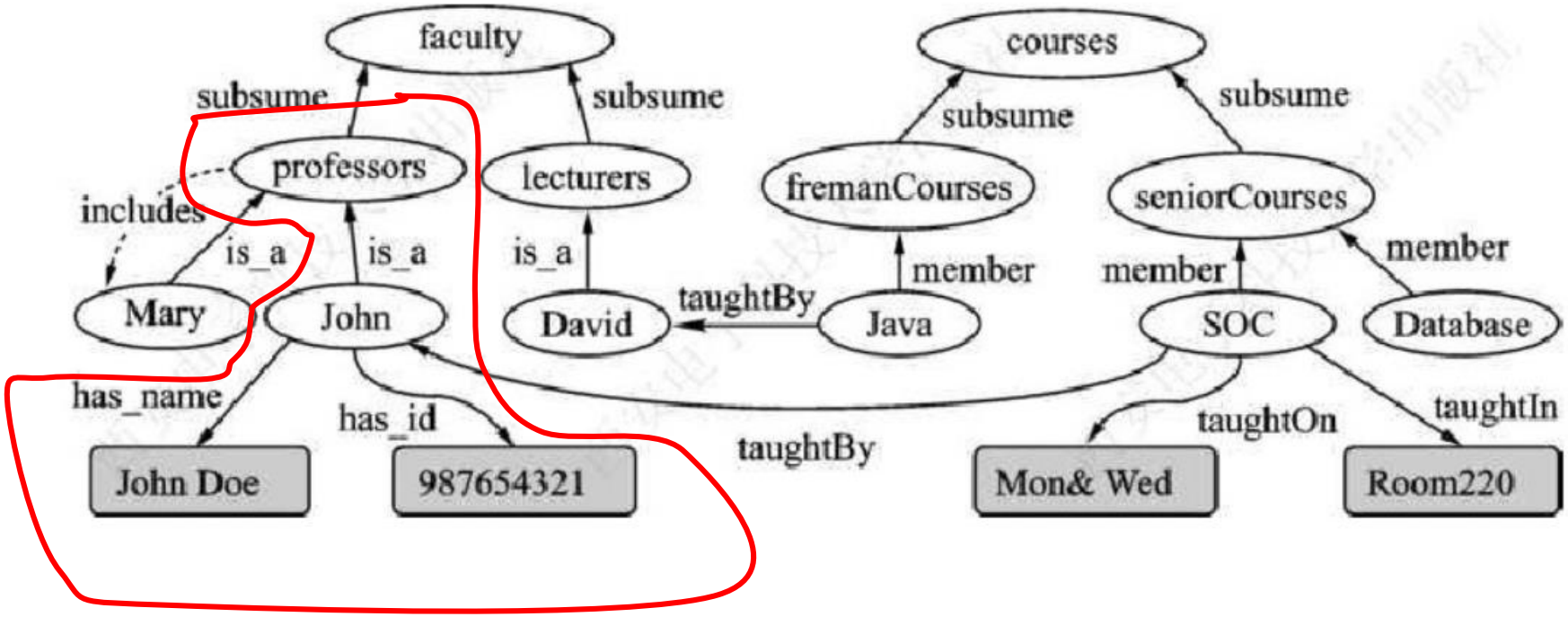
Knowledge Graph

- **Subjects**: Entity ID
- **Predicates**: Relation or Property ID
- **Object**: Entity ID or literal

Triple is good and easy to use, but cannot cover all kinds of knowledge!

Exercise

Write RDF in Turtle for the part circled by the red line. All resources are defined in the namespace <http://www.semanticweb.org/ontology-9/>, and the prefix of its QName is “sw”.



XML、RDF、RDFS

- **XML**
- **RDF**
- **RDFS:**
  RDF Vocabulary Description Language 1.0: RDF Schema
  http://www.w3.org/TR/rdf-schema

RDFS

- **RDFS:**
  - provides a data-modeling **vocabulary** for RDF data;
  - is an extension of the basic RDF vocabulary;
  - tries to provide consistent explanations for RDF data;
  - allows for specifying schema knowledge;
    - Mothers are female
    - Only persons write books
  - is a part of the W3C Recommendation.

RDFS

- **Why do we need RDFS when there already exists XML Schema?**
  - XML Schema only defines syntax without semantics;
  - XML Schema cannot reference "things" outside the document.

RDFS: Class and Instance

- Given a triple:

  ex:SemanticWeb rdf:type ex:Textbook .

  which characterizes "Foundations of Semantic Web Technologies" as an instance of the class "Textbook".

- A resource can be the member of more than one class

  ex:SemanticWeb rdf:type ex:Book .

- Instance and class names cannot be distinguished syntactically with IRIs.

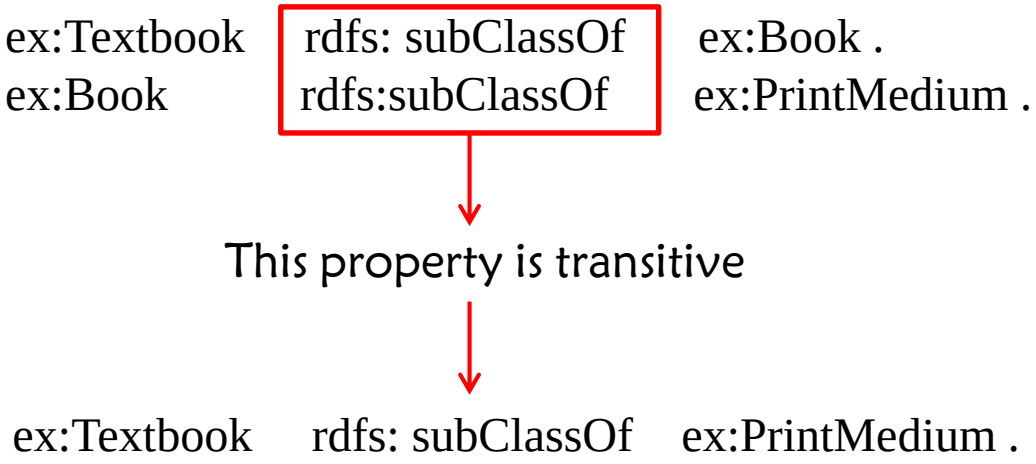
- RDFS helps explicitly state that a resource denotes a class:

  ex:book rdf:type **rdfs:Class** .

  rdfs:Class is the "class of all classes".

RDFS: Class Hierarchy (Taxonomy)

- Classes can be arranged in hierarchies: each textbook is a book



RDFS: Class Hierarchy (Taxonomy)

- rdfs: subClassOf is also reflexive:

  ex:Textbook rdfs: subClassOf ex:TextBook .
  ex:Book rdfs:subClassOf ex:Book .
  …

- rdfs: subClassOf can derive class equivalence:

  ex: Lecturer rdfs:subClassOf ex:Assistant_Professor .
  ex:Assistant_Professor rdfs:subClassOf ex: Lecturer .

ex: Lecturer = ex:Assistant_Professor

RDFS: Class Hierarchy (Taxonomy)

- Modeling the class hierarchy of living beings

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ex:  <http://www.semantic-web-grundlagen.de/Beispiele#> .

ex:Animalia  rdf:type  rdfs:Class .
ex:Chordata  rdf:type    rdfs:Class ;
       rdfs:subClassOf  ex:Animalia .
ex:Mammalia  rdf:type    rdfs:Class ;
       rdfs:subClassOf  ex:Chordata .
ex:Primates  rdf:type    rdfs:Class ;
       rdfs:subClassOf  ex:Mammalia .
 ex:Hominidae  rdf:type   rdfs:Class ;
       rdfs:subClassOf  ex:Primates .
```

Animalia:动物界
Chordata:脊索动物
Mammalia：哺乳动物
Primates：灵长类
Hominidae：人科

RDFS: Class in RDF/XML Syntax

Example:

```
<rdf:Description rdf:ID="horse">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="#animal"/>
</rdf:Description>
```

Example Abbreviated: use **rdfs:Class** instead of **rdf:Description**, and drop the **rdf:type** information

```
<rdfs:Class rdf:ID="horse">
  <rdfs:subClassOf rdf:resource="#animal"/>
</rdfs:Class>
```

RDFS: Predefined Classes

- **rdfs:Resource**
  class of all resources (i.e., all elements of the domain)
- **rdf:Property**
  class of all relationships (properties)
  (= those resources, that are referenced via predicate URIs)
- **rdf:List, rdf:Seq, rdf:Bag, rdf:Alt, rdfs:Container**
  diverse kinds of lists
- **rdfs:ContainerMembershipProperty**
  class of all relationships that represent a containedness relationship
- **rdf:XMLLiteral**
  class of all values of the predefined datatype XMLLiteral
- **rdfs:Literal**
  class of all literal values (every datatype is a subclass of this class)
- **rdfs:Datatype**
  class of all datatypes(therefore it is a class of classes, similar to rdfs:Class)
- **rdf:Statement**
  class of all reified propositions (triples)

RDFS: Property and Property Hierarchy

Given

ex:happilyMarriedWith rdfs:subPropertyOf ex:marriedWith .

ex:mark ex:happilyMarriedWith ex:ann .

ex:mark ex:marriedWith ex:ann .

RDFS: Property Restrictions

- Allow to state that a certain property can only be between things of a certain class:
  - e.g. when a is married to b, then both a and b are Persons

```
ex:isMarriedTo   rdfs:domain   ex:Person .
ex:isMarriedTo   rdfs:range    ex:Person .
```

- Datatypes can also be used for adding property restrictions:

```
ex:hasAge   rdfs:range   xsd:nonNegativeInteger .
```

RDFS: Property Restrictions

- Property restrictions are interpreted globally and conjunctively:

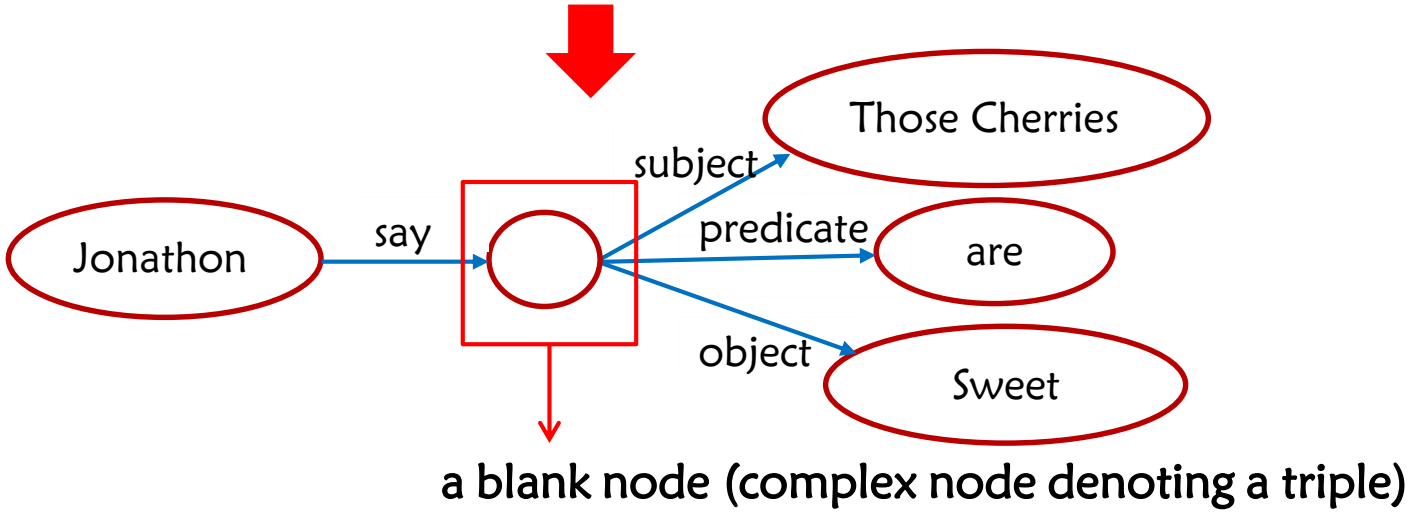
```
ex:authorOf    rdfs:range    ex:Cookbook .
ex:authorOf    rdfs:range    ex:Storybook .
ex:fred    ex:authorOf    ex:fredsBook .
```

ex:fredsBook is both a cookbook and a storybook.

RDFS: Reification

- How to graphically represent a sentence by means of the blank node?

Jonathon says those cherries are sweet.



a blank node (complex node denoting a triple)

Exercise

Represent the following sentence graphically by means of the blank node:

Wikipedia said that Tolkien wrote Lord of the Rings.

RDFS: Additional Information

- rdfs:label:
  - Property that assigns a name (Literal) to an arbitrary resource

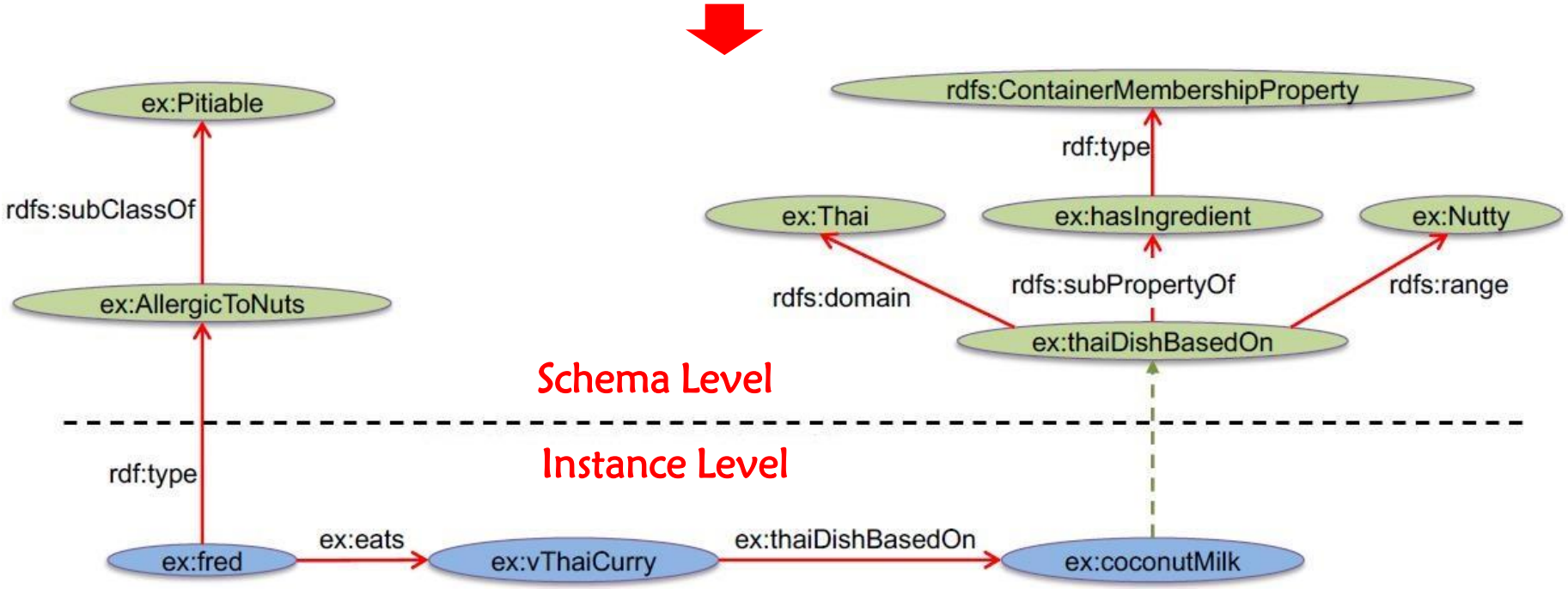
- rdfs:comment:
  - Property assigning an extensive comment (literal, usually natural language description) to an arbitrary resource

- rdfs:seeAlso, rdfs:definedBy
  - Properties giving resources where one can find further information or a definition of the subject resource

Example: Modeling with RDF(S)

Triples:

```
ex:vegetableThaiCurry   ex:thaiDishBasedOn   ex:coconutMilk .
ex:fred                 rdf:type             ex:AllergicToNuts .
ex:fred                 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 .
```



Example: Reasoning with RDFS

Given

ex:happilyMarriedWith rdfs:subPropertyOf ex:isMarriedTo .
ex:isMarriedTo rdfs:domain ex:Person .
ex:isMarriedTo rdfs:range ex:Person .
ex:pascal ex:happilyMarriedWith ex:lisa .

ex:pascal ex:isMarriedTo ex:lisa .
ex:pascal rdf:type ex:Person .
ex:lisa rdf:type ex:Person .

RDFS: Summary

- In RDFS, we can express statements about
  - resources/nodes being a member of a class;
  - classes being subclasses of other classes;
  - properties being subproperties of other properties;
  - classes being domains and ranges of properties.

- Vocabularies in RDFS are generic, not bound to a specific application area.

- By means of the modeling features of RDFS, important aspects of many domains can already be captured semantically.

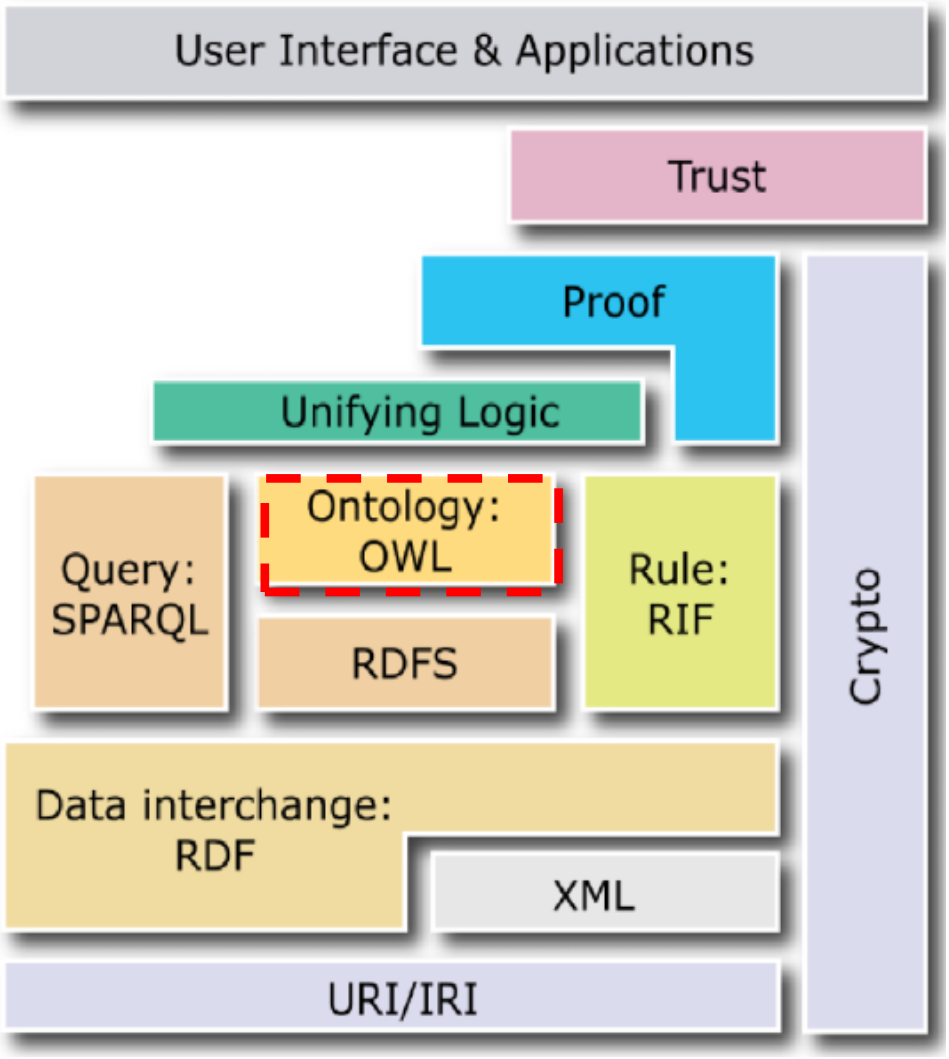
- Based on the RDFS semantics, a certain amount of implicit knowledge can be derived.

Exercise

What can be inferred from the following triples using RDFS semantics?

1) ex:Postgraduate_student rdfs:subClassOf ex:Student
2) ex:Professor rdfs:subClassOf ex:Academic_staff
3) ex:Supervise rdfs:domain ex:Professor
4) ex:Supervise rdfs:range ex:Postgraduate_student
5) ex:John ex:Supervise ex:Mary

Knowledge Graph Representation



W3C Stack

XML:
- Markup language for data exchange
- Surface syntax, no semantics
- XML Schema: describes structure of XML documents

RDF:
- Data model for "relations" between "things"

RDF Schema (RDFS):
- RDF vocabulary definition language

OWL:
- A more expressive vocabulary definition language

Why do we need OWL?

- **Problems in RDFS:**
  - The ability to describe resources is not so strong. In complex scenarios, the semantic expressive ability of RDFS is weak, since RDFS lacks many features:

1) RDFS does not have constraints on property domain and range:

For "**Human**", the **range** of "hasChild" should be "*Human*";
For "**Elephant**", the **range** of "hasChild" should be "*Elephant*".

2) RDFS does not have cardinality constraints:

The **number** of "**Father**" for one person should be **one**;
One person has at most **two parent**. (**property: "hasParent"**)

Why do we need OWL?

- **Problems in RDFS:**
  - The ability to describe resources is not so strong. In complex scenarios, the semantic expressive ability of RDFS is weak, since RDFS lacks many features:

3) RDFS does not have descriptions on property characteristics:

transitivity, symmetry, functionality,
relations to other properties (inverse, equivalence)…

4) RDFS lacks definitions on equivalence:

equivalence on individuals, classes, and properties

…

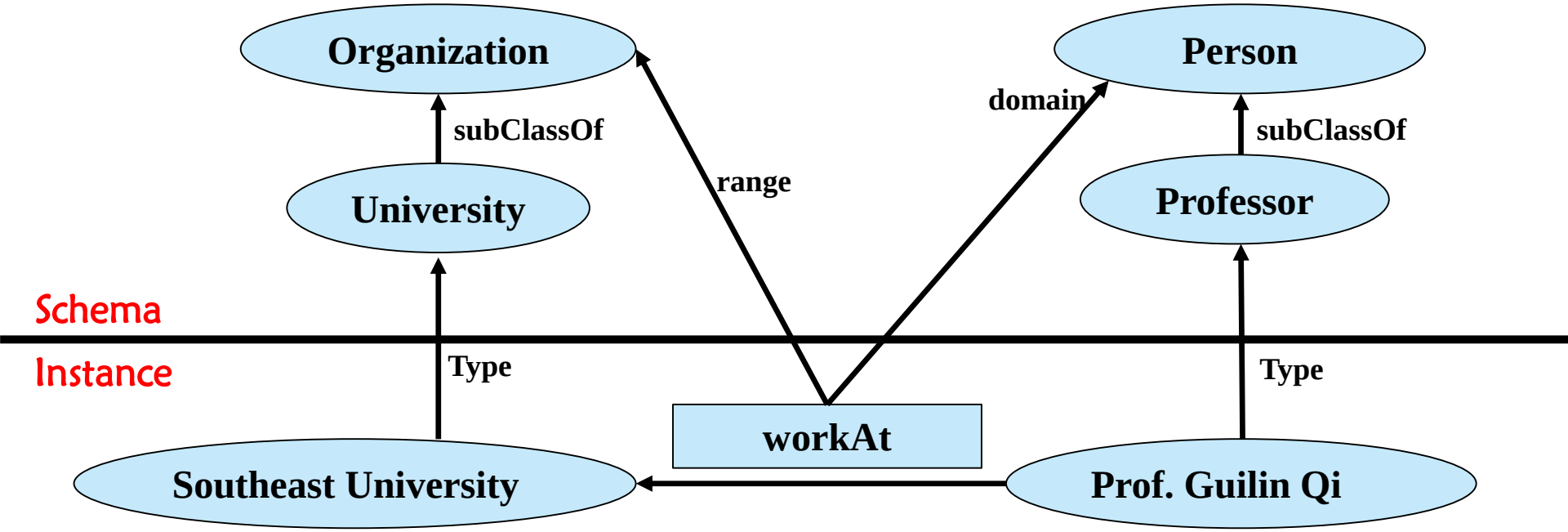
OWL

- **OWL：**
  Web Ontology Language (OWL)
  https://www.w3.org/TR/owl-semantics/
  https://www.w3.org/TR/owl-ref/

OWL

- **What is Ontology?**

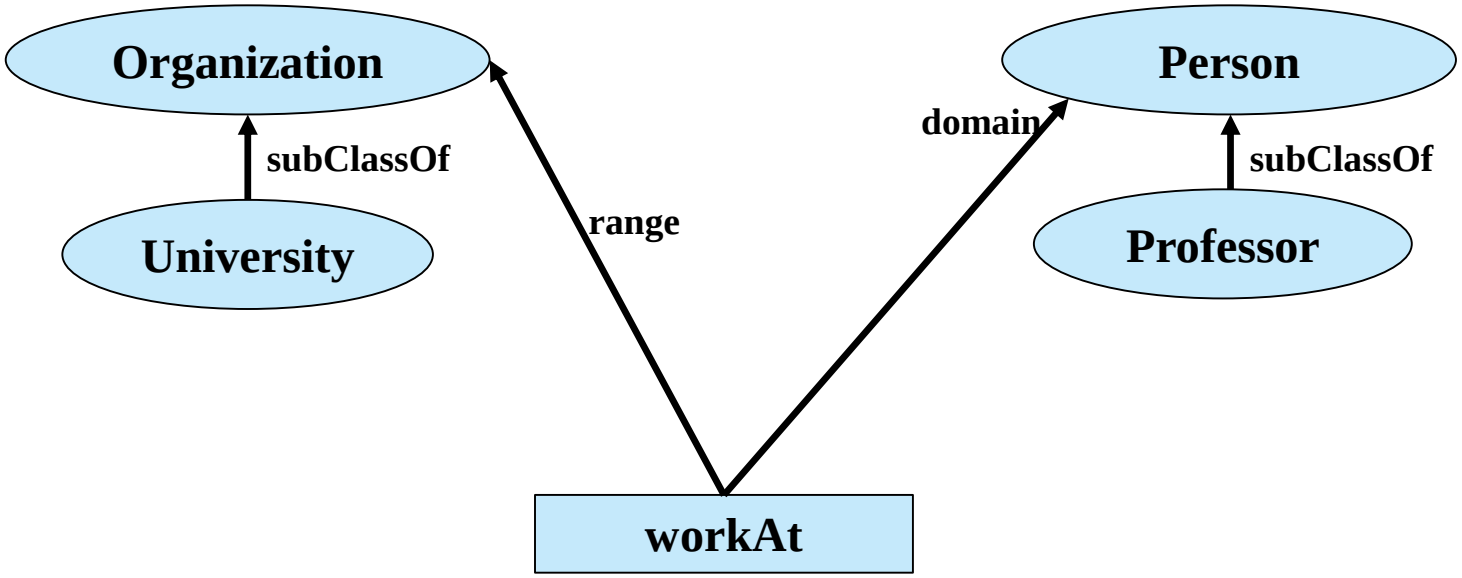
Knowledge Graph Example:



OWL

- **What is Ontology?**

Knowledge Graph Example:



Schema

≈

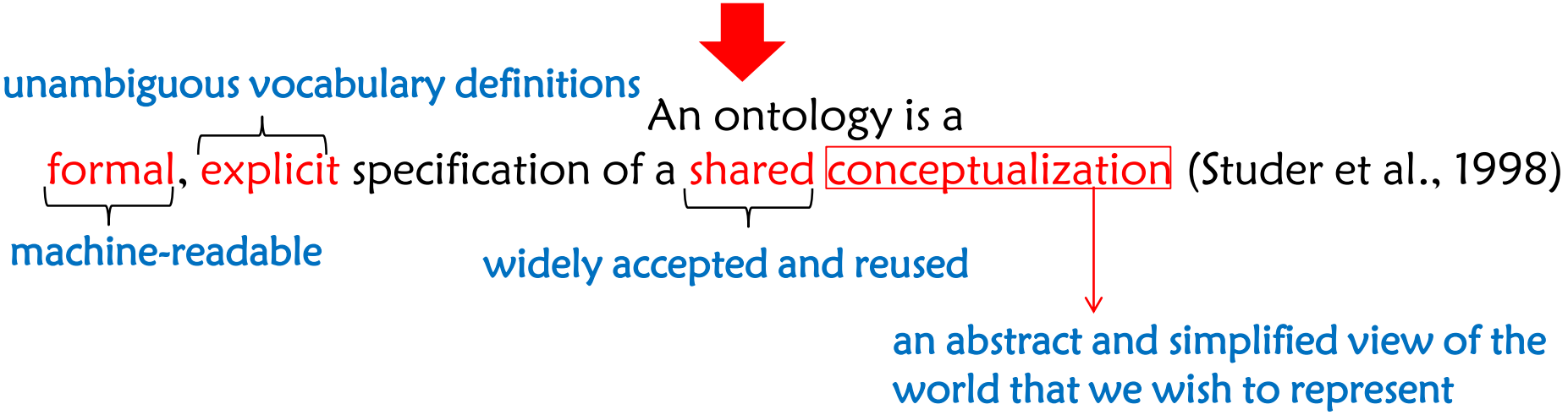
Ontology

OWL

- **What is Ontology?**

1) An ontology is an explicit specification of a conceptualization (Gruber, 1993)
   - a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents

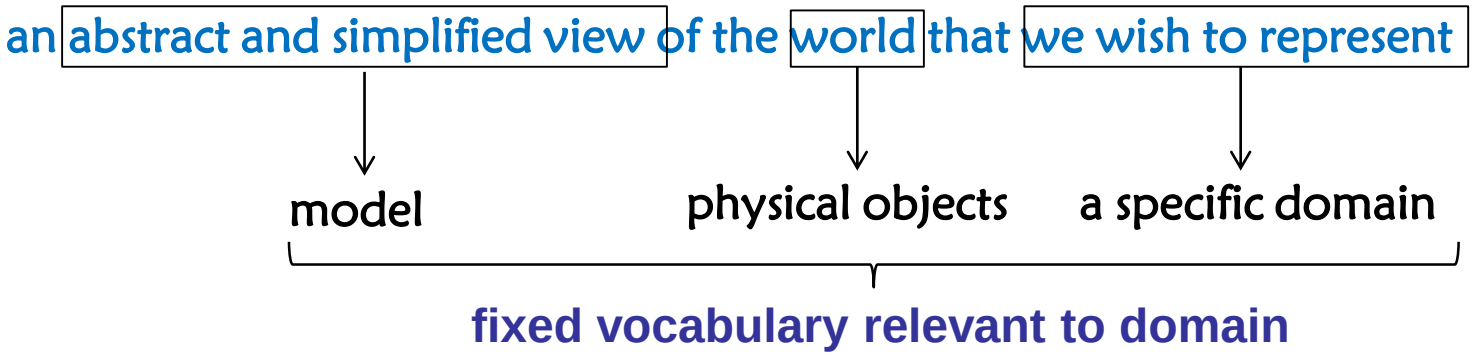
2) An ontology is a formal specification of a shared conceptualization (Borst, 1997)



OWL

- **What is Ontology?**

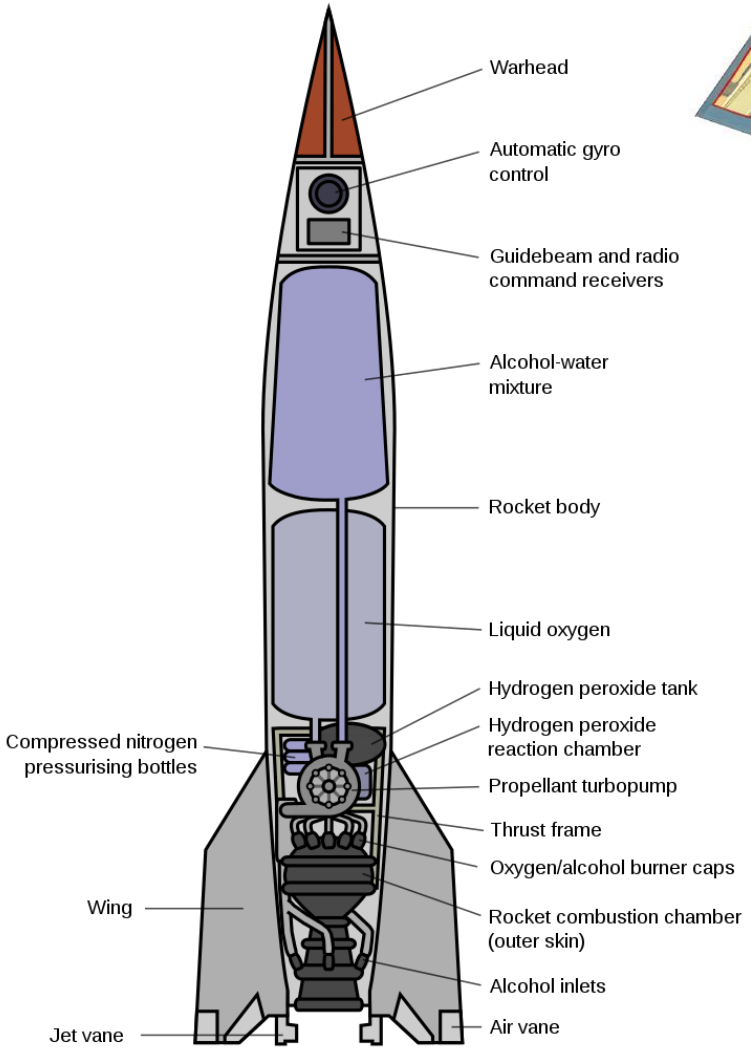
An ontology is a
formal, explicit specification of a shared conceptualization (Studer et al., 1998)

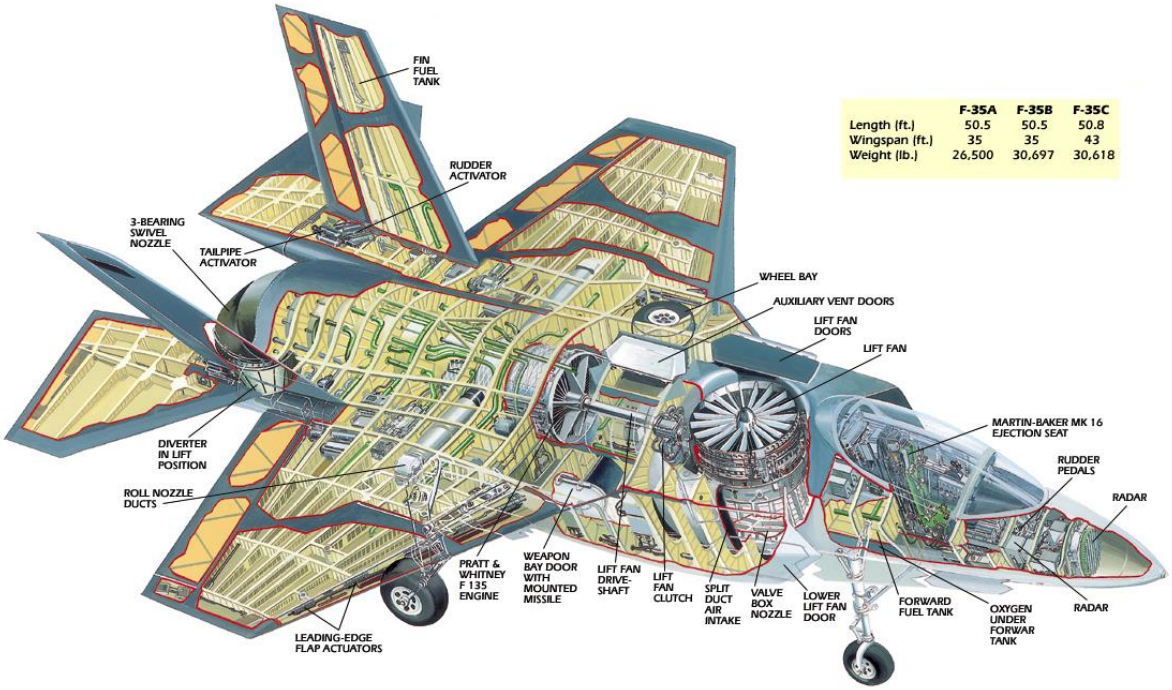


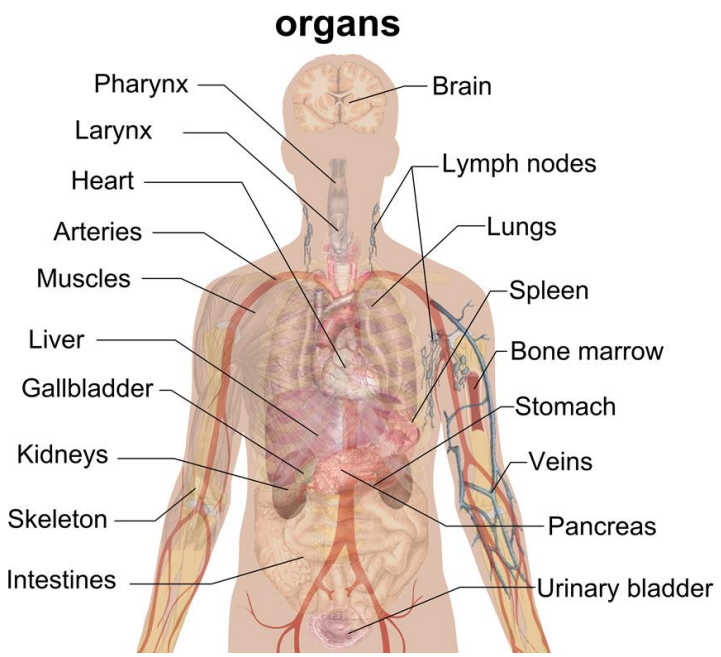
本体是一个共享概念化的一种形式化的显式规约。

OWL

- **What is Ontology?**







OWL

- **Ontology on the Web**
  - formal, explicit: use RDF data model
  - shared: fixed vocabulary definitions

vocabulary definition language: **RDFS, OWL**

OWL

- **OWL**
  - is introduced in W3C recommendation (2004);
  - is based on RDF Syntax, making OWL documents easy to understand by machines;
  - more expressive than RDFS;
  - allows logical reasoning (based on description logic).

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Equivalence between classes, individuals, and properties:

```
exp:Athlete owl:equivalentClass exp:SportsPlayer .

exp:obtain owl:equivalentProperty exp:acquire .

exp:SportsPlayerA owl:sameAs exp:Thomas .
```

```
@prefix   exp:  <http://example.org> .
```

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Disjointness between classes:

```
exp:Man owl:disjointWith exp:Woman .
```

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Intersection of classes:

```
exp:Mother rdf:type owl:class ;
           owl:intersectionOf (exp:Woman exp:Parent) .
```

can be multiple classes

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Union of classes:

```
exp:Parent rdf:type owl:class ;
           owl:unionOf (exp:Mother exp:Father) .
```

can be multiple classes

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Negation of a class:

```
exp:ABC rdf:type owl:class ;
        owl:complementOf exp:Meat .
```

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Property definitions: object property and data property.

```
ex:friendOf rdf:type owl:ObjectProperty .
ex:age rdf:type owl:DataProperty .
```

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Transitive property:

```
exp:ancestor rdf:type owl:TransitiveProperty .
```

Given triples

```
exp:Thmoas exp:ancestor exp:Jimmy  .
exp:Jimmy exp:ancestor exp:Michael  .
```

Then what can we infer from them?

```
exp:Thmoas exp:ancestor exp:Michael  .
```

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Functional property:

```
exp:hasMother rdf:type owl:FunctionalProperty .
```

Please explain using natural language:

Everyone has only one mother.

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Functional property:

```
exp:hasMother rdf:type owl:FunctionalProperty .
```

How about the inverse functional property?

exp:postgraduateSupervisorOf rdf:type owl:InverseFunctionalProperty .

Please explain using natural language:

Each post-graduate student has only one supervisor.

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Symmetric property:

```
exp:friendOf rdf:type owl:SymmetricProperty .
```

Given a triple

exp:Thmoas exp:friendOf exp:Lisa .

Then what can we infer from it?

exp:Lisa exp:friendOf exp:Thmoas .

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Inverse relations between properties:

```
exp:ancestor owl:inverseOf exp:descendant .
```

Given a triple

exp:Thmoas exp:ancestor exp:Jimmy .

Then what can we infer from it?

exp:Jimmy exp:descendant exp:Thmoas .

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Constraints on properties: universal quantifier ∀
  - for each instance of the class that is being described, every value for the property must fulfill the constraint

```
exp:Person rdf:type owl:Restriction ;
           owl:onProperty exp:hasMother ;
           owl:allValuesFrom exp:Women .
```

What do we know from the above triples?

If the subject of exp:hasMother comes from the class exp:Person, then the object can be only from the class exp:Women

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Constraints on properties: existential quantifier ∃
  - for each instance of the class that is being defined, there exists at least one value for the property that fulfills the constraint

```
exp:SemanticWebPapers rdf:type owl:Restriction ;
                      owl:onProperty exp:publishedIn ;
                      owl:someValuesFrom exp:AAAIPapers .
```

What do we know from the above triples?

If the subject of exp:publishedIn comes from the class exp:SemanticWebPapers, then the object may be (*at least one*) from the class exp:AAAIPapers.

A part of the Semantic Web papers were published in AAAI.

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Constraints on properties: cardinalities

```
exp:Person rdf:type owl:Restriction ;
           owl:onProperty exp:hasParent ;
           owl:cardinality  “2” ^ ^xsd:integer .
```

Please explain using natural language:

Each person should have two parents.

OWL

- **Important newly added vocabulary in OWL (in Turtle syntax):**

  Constraints on properties: cardinalities

```
exp:Person rdf:type owl:Restriction ;
           owl:onProperty exp:hasParent ;
           owl:cardinality  "2"  ^ ^xsd:integer .
```

owl:maxCardinality/ owl:minCardinality

Exercise

Please explain the following triples using natural language:

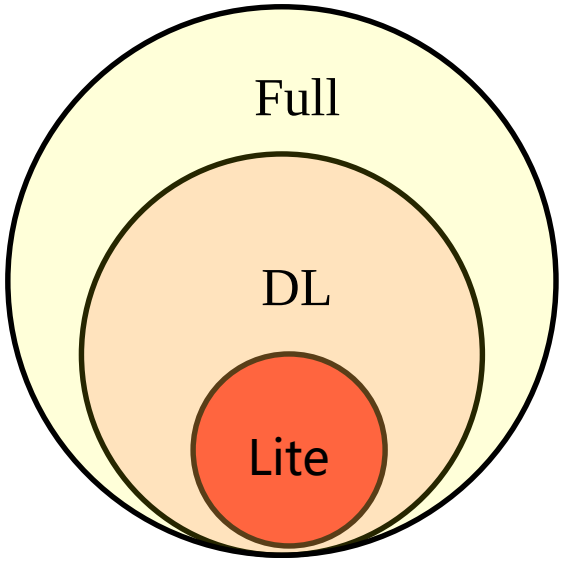
```
@prefix sw: <http://semanticweb.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

sw:Giraffe  rdfs:subClassOf   _:x .
_:x    rdfs:subClassOf   sw:Animal ;
       rdf:type   owl:Restriction ;
       owl:onProperty   sw:eat ;
       owl:allValuesFrom sw:Leaf .
```

Exercise

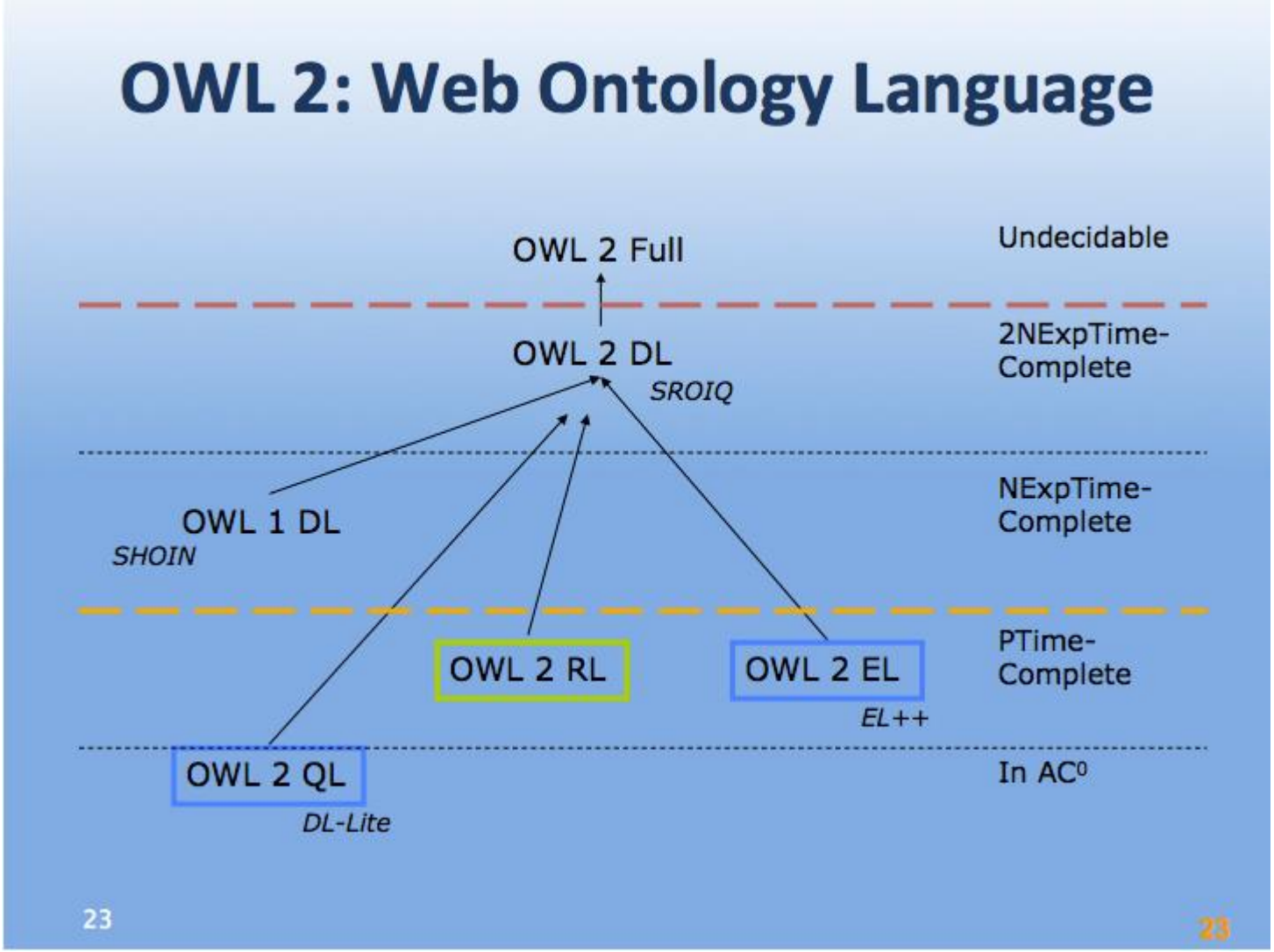
Please write the corresponding RDF triples (in Turtle) of the following sentence: “ChildlessPersons are the persons who are not parents”. Note that classes “ChildlessPerson”, “Person”, “Parent” can be defined in the namespace <http://semanticweb.org/> with the prefix “SW”.

OWL

- **OWL Sub-Languages**
  (http://www.ksl.stanford.edu/people/dlm/webont/OWLFeatureSynopsisJan22003.htm
  - OWL Lite
  - OWL DL
  - OWL Full



OWL


OWL 2: Web Ontology Language
OWL 2 Full
Undecidable
OWL 2 DL
SROIQ
2NExpTime-Complete
OWL 1 DL
SHOIN
NExpTime-Complete
OWL 2 RL
OWL 2 EL
EL++
PTime-Complete
OWL 2 QL
DL-Lite
In AC0
23
23

Thanks!