CS 7810 - Knowledge Representation and Reasoning (for the Semantic Web)

03 – RDF Schema (RDFS)

Adila Krisnadhi

Data Semantics Lab,
Wright State University, Dayton, OH

Outline

1. Motivation
2. Classes and Class Hierarchy
3. Properties and Property Hierarchy
4. Property Restrictions
5. Containers and Collections
6. Reification
7. Supplementary Information in RDFS
8. Simple RDFS Ontologies

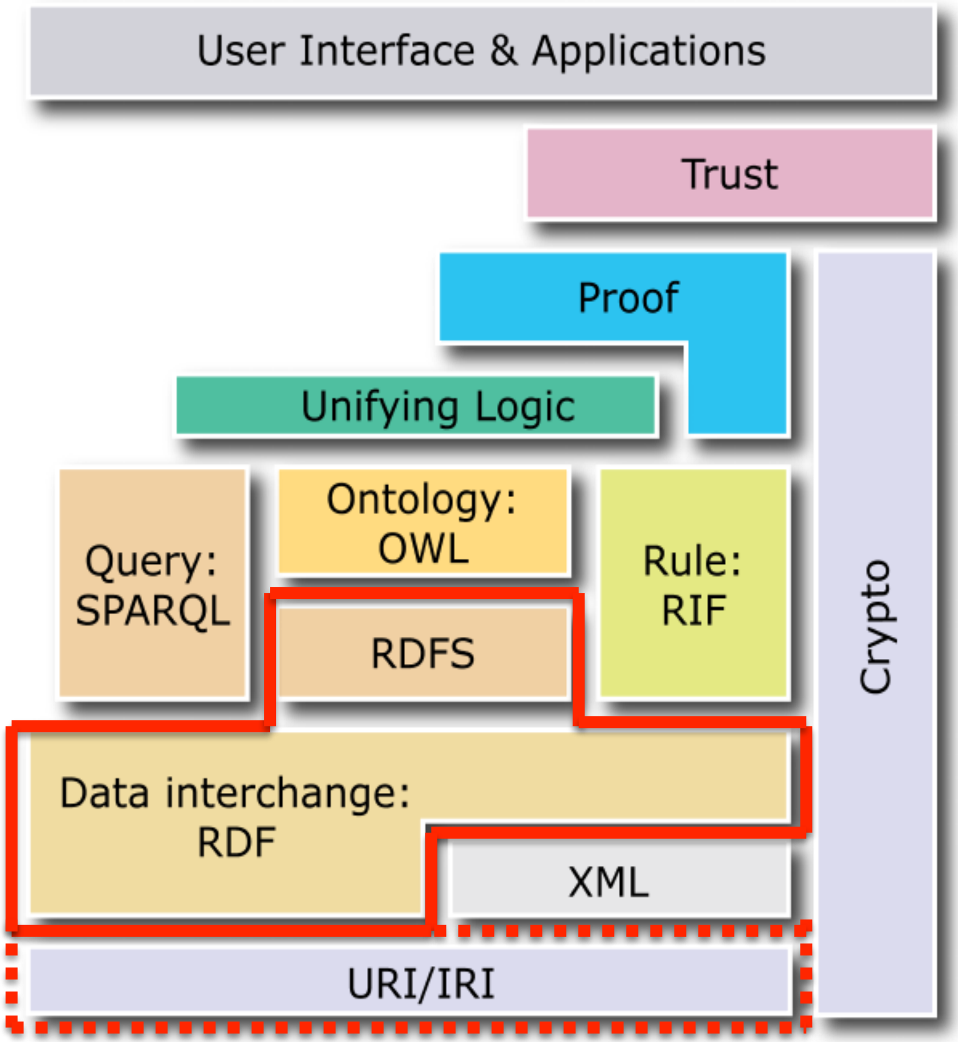
Acknowledgements

- Most of the slides in this presentation are adapted from:
  - Sebastian Rudolph, “RDF”, slides for Foundations of Semantic Web Technologies course, Dresden, April 11, 2014.
  - Sebastian Rudolph, “RDF Schema”, slides for Foundations of Semantic Web Technologies course, Dresden, April 11, 2014.
  - Pascal Hitzler, “Slides 3 – 01/10/2012”, slides for Knowledge Representation for the Semantic Web course, Winter quarter 2012.

Outline

1. **Motivation**
2. Classes and Class Hierarchy
3. Properties and Property Hierarchy
4. Property Restrictions
5. Containers and Collections
6. Reification
7. Supplementary Information in RDFS
8. Simple RDFS Ontologies

RDF(S) in the SW Stack



https://www.w3.org/2007/03/layerCake.png

Graph Vocabulary

- A *name/term* is a URI reference or a literal.
- A *typed literal* comprises two names: the literal itself and its type.
- *Vocabulary*: a set of names.
- *Graph vocabulary*: the set of names occurring in the graph as subject, predicate, or object.
- *RDF* (*RDFS*) *vocabulary*: names with predefined meaning/ semantics according to the RDF (RDFS) specs.
- Some names in a graph vocabulary may come from the vocabulary defined by standards.

Motivation

- RDF allows encoding of factual data on the Web → proposition about single resources (individuals) and their relationships.
  - “Foundations of Semantic Web Technologies was published by CRC Press”
- Desirable: expressing propositions about more generic knowledge (e.g., about sets of individuals).
  - Fathers are male.
  - An organization that publishes a book is a publisher.
- Such generic knowledge is often called *schema knowledge* or *terminological knowledge*.
- RDF Schema allow us to express schema knowledge.
  - So is OWL (discussed much later).

RDF Schema (RDFS)

- W3C recommendations:
  - RDF → https://www.w3.org/TR/rdf11-concepts/
  - RDFS → https://www.w3.org/TR/rdf-schema/
  - RDF semantics → https://www.w3.org/TR/rdf11-mt/
- RDFS defines the semantics of some names from the RDF namespace and introduce names in RDFS namespace (with predefined semantics).
  - Namespace `http://www.w3.org/1999/02/22-rdf-syntax-ns#` usually abbreviated with the prefix `rdf:`
  - Namespace `http://www.w3.org/2000/01/rdf-schema#` usually abbreviated with the prefix `rdfs:`
- Every RDFS document is a RDF document.
- RDF/RDFS vocabulary is generic and not domain-specific.
  - Allows one to partially specify the semantics of user-defined vocabularies → could thus be called meta-vocabulary.
  - Hence, every RDFS-compliant software correctly interprets the names from the RDFS vocabulary.
  - This means that RDFS is already an ontology language.

Outline

1. Motivation
2. **Classes and Class Hierarchy**
3. Properties and Property Hierarchy
4. Property Restrictions
5. Containers and Collections
6. Reification
7. Supplementary Information in RDFS
8. Simple RDFS Ontologies

Classes

- Class: a set of things. In RDF, a class is a set of URIs.
- Class membership → use rdf:type (corresponds to set-element relationship).

```
ex:fost rdf:type ex:Textbook .
```

- A URI can belong to several classes.

```
ex:fost rdf:type ex:Textbook ;
             rdf:type ex:Entertaining .
```

- In general, individuals cannot be distinguished syntactically (by only looking at their URIs) from classes
  - May need to explicitly declare that a URI stands for a class.
  - May also be inferred from triples involving `rdf:type` or `rdfs:subClassOf`
  - In RDF, a URI can be both a class and an individual simultaneously.

The Class of All Classes

- A URI can be declared as a class by typing it as `rdfs:Class`

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

- `rdfs:Class` is the class of all classes. Hence, the following is always true (regardless whether it’s declared explicitly or not).

```
rdfs:Class rdf:type rdfs:Class .
```

Class Hierarchy

- Classes can be arranged in a hierarchy of subclass-superclass → use `rdfs:subClassOf`

  ```
  ex:Textbook rdfs:subClassOf ex:Book .
  ```

  "Every textbook is a book"
- The above when together with the following triple

  ```
  ex:fost rdf:type ex:Textbook .
  ```

  we could (implicitly) infer/deduced the following triple:

  ```
  ex:fost rdf:type ex:Book .
  ```
- We say that the last triple is a *logical consequence* of the graph or *entailed* by the graph according to the RDFS semantics.
- Mathematically, rdfs:subClassOf corresponds to subset relationship in set theory.

Class Hierarchy

- `rdfs:subClassOf` is *reflexive*: every class is a subclass of itself, e.g., the following are always true.
  - `ex:Textbook rdfs:subClassOf ex:Textbook .`
  - `rdfs:Class rdfs:subClassOf rdfs:Class .`
- `rdfs:subClassOf` is *transitive*, i.e., we can create a taxonomy using it. Given the following two triples:

  ```
  ex:Textbook rdfs:subClassOf ex:Book .
  ex:Book rdfs:subClass ex:Publication .
  ```

  We could infer:

  ```
  ex:Textbook rdfs:subClassOf ex:Publication .
  ```
- To express that two different URIs refer to the same class, we use two `rdfs:subClassOf` triples:
  - ex:Airplane rdfs:subClassOf ex:Aircraft .
  - ex:Aircraft rdfs:subClassOf ex:Airplane .

Classes with RDF/XML syntax

```
<rdf:Description rdf:about="&ex;fost">
   <rdf:type rdf:resource="&ex;Textbook">
</rdf:Description>
```

- Or, abbreviated as:

```
<ex:Textbook rdf:about="&ex;fost"/>
```

- Thus, declaration of URIs as class could be written, e.g., as:

```
<rdfs:Class rdf:about="&ex;Textbook"/>
```

Predefined Class URIs

- `rdfs:Resource` – class of all resources (all elements in the domain of discourse)
- `rdfs:Class` – class of all classes
- `rdf:Property` – class of all RDF properties
- `rdfs:Container`, `rdf:Bag`, `rdf:Seq`, `rdf:Alt`, `rdf:List` – various kinds of lists
- `rdfs:ContainerMembershipProperty` – class of all properties that represent containedness relationship involving the lists above.
- `rdfs:Literal` – class of all literals (every datatype is a subclass of this)
- `rdfs:Datatype` – class of all of RDF datatypes (like rdfs:Class for classes)
- `rdf:langString` – class of language-tagged string literals
- `rdf:HTML` – class of all HTML literals/strings – not yet reached Recommendation status
- `rdf:XMLLiteral` – class of all XML literals/strings – not yet reached Recommendation status
- `rdf:Statement` – class of RDF statements

Outline

1. Motivation
2. Classes and Class Hierarchy
3. **Properties and Property Hierarchy**
4. Property Restrictions
5. Containers and Collections
6. Reification
7. Supplementary Information in RDFS
8. Simple RDFS Ontologies

Properties

- Also called: relation, relationships
  - characterize how two resources are related
  - Mathematically: a set of pairs of individuals.
- Note: properties are **not** assigned to classes (different from OOP)
- Every URI occurring as a predicate in a triple is a property.
  - May be declared explicitly as such by typing it as `rdf:Property`, e.g.,

```
 ex:publishedBy rdf:type rdf:Property .
```

Property Hierarchy

- Use `rdfs:subPropertyOf`
- Given

```
ex:worksFor rdfs:subPropertyOf ex:affiliatedWith .
ex:adila ex:worksFor ex:WSU .
```

- We can infer:

```
ex:adila ex:affiliatedWith ex:WSU .
```

- `rdfs:subPropertyOf` is *reflexive* and *transitive*.

Outline

1. Motivation
2. Classes and Class Hierarchy
3. Properties and Property Hierarchy
4. **Property Restrictions**
5. Containers and Collections
6. Reification
7. Supplementary Information in RDFS
8. Simple RDFS Ontologies

Property Restrictions

- Usually, the use of a property only makes sense for certain kinds of resources, e.g., `ex:publishedBy` only connects publications with publishers.
- For example,

```
ex:fost ex:publishedBy crc:uri.
```

intuitively implies the following:

```
ex:fost rdf:type ex:Publication .
crc:uri rdf:type ex:Publisher .
```

- The above can be achieved by asserting:

```
ex:publishedBy rdfs:domain ex:Publication .
ex:publishedBy rdfs:range ex:Publisher .
```

- And for datatypes:

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

Gotcha 1

- Given

```
ex:authorOf rdfs:range ex:Storybook .
ex:authorOf rdfs:range ex:Textbook .
ex:pascal ex:authorOf ex:fost .
```

- Which of the following is true?
  a) `ex:fost` is an instance of `ex:Storybook`.
  b) `ex:fost` is an instance of `ex:Textbook .`
  c) All of the above.
- Answer: a, b, and c ☺

Gotcha 2

- Given

```
ex:authorOf rdfs:domain ex:Person .
ex:authorOf rdfs:range ex:Book .
ex:UnitedNations ex:authorOf ex:UNResolution .
```

- What can you infer from them?

- Answer:

```
ex:UnitedNations rdf:type ex:Person .
ex:UNResolution rdf:type ex:Book .
```

Property Restrictions (contd.)

- In RDFS, property restrictions are the only way of specifying interdependencies between classes and properties.
  - In OWL, you could do much more.
- Property restrictions are interpreted **globally** and **conjunctively**.
- Be careful of not picking too specific class for domain/range assertions.

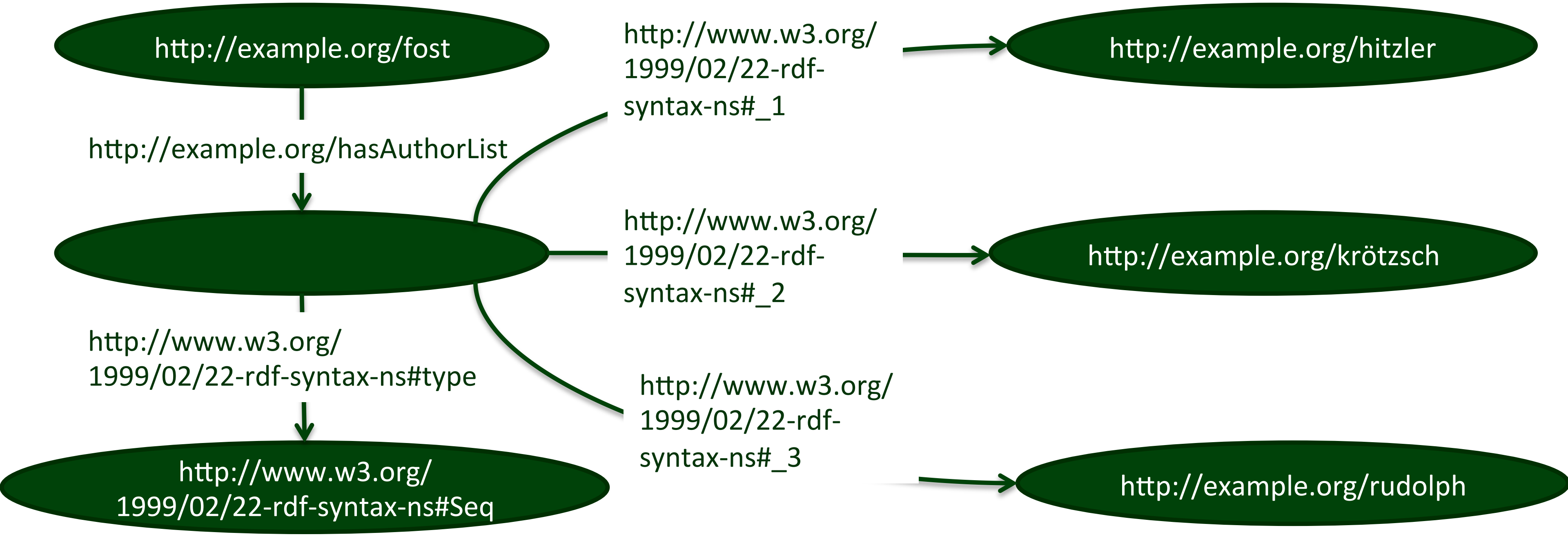
Outline

1. Motivation
2. Classes and Class Hierarchy
3. Properties and Property Hierarchy
4. Property Restrictions
5. **Containers and Collections**
6. Reification
7. Supplementary Information in RDFS
8. Simple RDFS Ontologies

RDF Lists

- RDF syntax features two ways of writing a list of objects.
  - Open list: no mechanism to state that there are no more members in the list;
  - Closed list: otherwise.
- Lists are modeled using triple representation (with the help of some vocabulary terms) – no additional expressivity.

Open Lists (Containers)


http://example.org/fost
http://example.org/hasAuthorList
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq
http://www.w3.org/1999/02/22-rdf-syntax-ns#_1
http://example.org/hitzler
http://www.w3.org/1999/02/22-rdf-syntax-ns#_2
http://example.org/krötzsch
http://www.w3.org/1999/02/22-rdf-syntax-ns#_3
http://example.org/rudolph

- in Turtle (if desired, the blank node above may also be labeled):

```
@prefix ex: <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
ex:fost    ex:hasAuthorList   [   rdf:type rdf:Seq ;
                                  rdf:_1 ex:hitzler ;
                                  rdf:_2 ex:krötzsch ;
                                  rdf:_3 ex:rudolph  ] .
```

Open Lists (Containers)

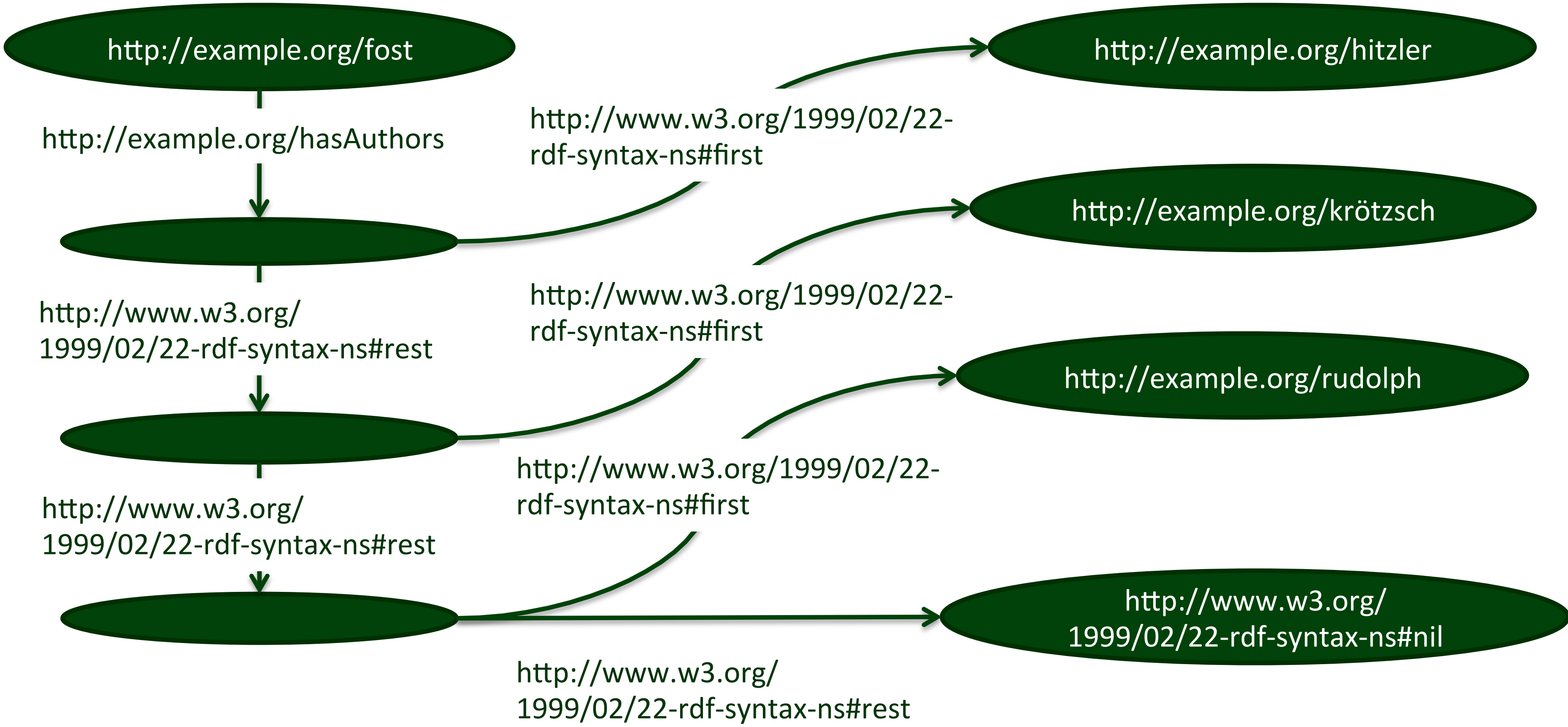
- In RDF/XML:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:ex="http://example.org/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdf="http://example.org/">
  <rdf:Description rdf:about="fost">
    <ex:hasAuthorList>
      <rdf:Seq>
        <rdf:_1 rdf:resource="hitzler"/>
        <rdf:_2 rdf:resource="krötzsch"/>
        <rdf:_3 rdf:resource="rudolph"/>
      </rdf:Seq>
    </ex:hasAuthorList>
  </rdf:Description>
</rdf:RDF>
```

Types of Open Lists

- Assigned to the root node of the list via `rdf:type` (see previous example):
  - rdf:Seq – ordered list
  - `rdf:Bag` – unordered set (encoded order is viewed as irrelevant)
  - `rdf:Alt` – set of alternatives (normally only one entry is relevant)
- Note: their RDF formal semantics are identical
  - Different classes above may be used to indicate **informally** further information.
- All the three classes are subclass of `rdfs:Container`.
- All of the properties `rdf:_1`, `rdf:_2`, `rdf:_3`, etc. are:
  - instances of `rdfs:ContainerMembershipProperty` class, itself a subclass of `rdf:Property`
  - subproperty of `rdfs:member`, itself an instance of `rdfs:ContainerMembershipProperty`.

Closed Lists (Collections)


http://example.org/fost
http://example.org/hasAuthors
http://www.w3.org/1999/02/22-rdf-syntax-ns#first
http://example.org/hitzler
http://example.org/krötzsch
http://www.w3.org/1999/02/22-rdf-syntax-ns#rest
http://www.w3.org/1999/02/22-rdf-syntax-ns#first
http://example.org/rudolph
http://www.w3.org/1999/02/22-rdf-syntax-ns#rest
http://www.w3.org/1999/02/22-rdf-syntax-ns#first
http://www.w3.org/1999/02/22-rdf-syntax-ns#nil
http://www.w3.org/1999/02/22-rdf-syntax-ns#rest

- Intuition: recursive deconstruction of a list into head element and (possibly empty) rest list.

Closed List Syntax

- Class of RDF lists is `rdf:List`.

- In Turtle (you could of course also write the triples more explicitly with `rdf:first`, `rdf:rest`, and `rdf:nil`):

```
@prefix ex: <http://example.org/> .
ex:fost ex:hasAuthors ( ex:hitzler ex:krötzsch ex:rudolph ) .
```

- In RDF/XML:

```
<rdf:RDF xmlns:ex="http://example.org/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="&ex;fost">
    <ex:hasAuthors rdf:parseType="Collection">
      <rdf:Description rdf:about="&ex;hitzler"/>
      <rdf:Description rdf:about="&ex;krötzsch"/>
      <rdf:Description rdf:about="&ex;rudolph"/>
    </ex:hasAuthors>
  </rdf:Description>
</rdf:RDF>
```

Outline

1. Motivation
2. Classes and Class Hierarchy
3. Properties and Property Hierarchy
4. Property Restrictions
5. Containers and Collections
6. **Reification**
7. Supplementary Information in RDFS
8. Simple RDFS Ontologies

Reification

- How do you express in RDF:
  “The detective suspects that the butler killed the gardener”.

- First try:

```
ex:detective ex:suspects
                "The butler killed the gardener" .
```

  - Unsatisfactory! Why?

- Second try:

```
ex:detective ex:suspects
                   ex:theButlerKilledtheGardener .
```

  - Unsatisfactory! Why?

Reification

- Without context, we can easily model that “The buttler killed the gardener”:

  ```
  ex:butler ex:killed ex:gardener .
  ```

- But the above triple may not necessarily be true.
  - Desirable: the above triple occurs as an object in another triple (however, this is impossible in RDF).
- Solution: reification.
  - Introduce an auxiliary node (possibly blank nodes) representing the nested proposition.

```
ex:detective ex:suspects ex:theory .
ex:theory   rdf:subject ex:butler ;
            rdf:predicate ex:killed ;
            rdf:object ex:gardener .
```

Reification

- The following triples:

```
ex:theory   rdf:subject ex:butler ;
            rdf:predicate ex:killed ;
            rdf:object ex:gardener .
```

Does **NOT** imply the following:

```
ex:butler ex:killed ex:gardener .
```

- If this is desired, the above un-reified triple must be added to the RDF document.
- Use the class `rdf:Statement` to mark nodes that represent reified statements.

Outline

1. Motivation
2. Classes and Class Hierarchy
3. Properties and Property Hierarchy
4. Property Restrictions
5. Containers and Collections
6. Reification
7. **Supplementary Information in RDFS**
8. Simple RDFS Ontologies

Supplementary Information in RDFS

- `rdfs:label`
  - property to assign a (human-readable) name (as literal) to any resource.
  - many RDF visualization tools use the information provided via this property when presenting the data graphically.

```
<rdfs:Class rdf:about="&ex;Hominidae">
<rdfs:label xml:lang="en">great apes</rdfs:label>
</rdfs:Class>
```

- `rdfs:comment`
  - property to assign an extensive comment (as literal) to any resource.
  - may, e.g., contain a natural language description of the resource (help later usage).
- `rdfs:seeAlso`, `rdfs:definedBy`
  - properties to provide any resource with a link to another resource (URI) where one can find further information or a definition of the subject resource.

Supplementary Information in RDFS

```
@prefix dbp: <http://dbpedia.org/resource/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
dbp:Mammal rdfs:comment
   "Mammals (class Mammalia /məˈmeɪli.ə/ from Latin mamma
   \"breast\") are any members of a clade of endothermic
   amniotes distinguished from reptiles and birds by the
   possession of hair, three middle ear bones, mammary
   glands, and a neocortex (a region of the brain)."@en .
dbp:Mammal rdfs:label "Mammal"@en .
dbp:Mammal rdfs:seeAlso dbp:Animal_cognition .
```

Outline

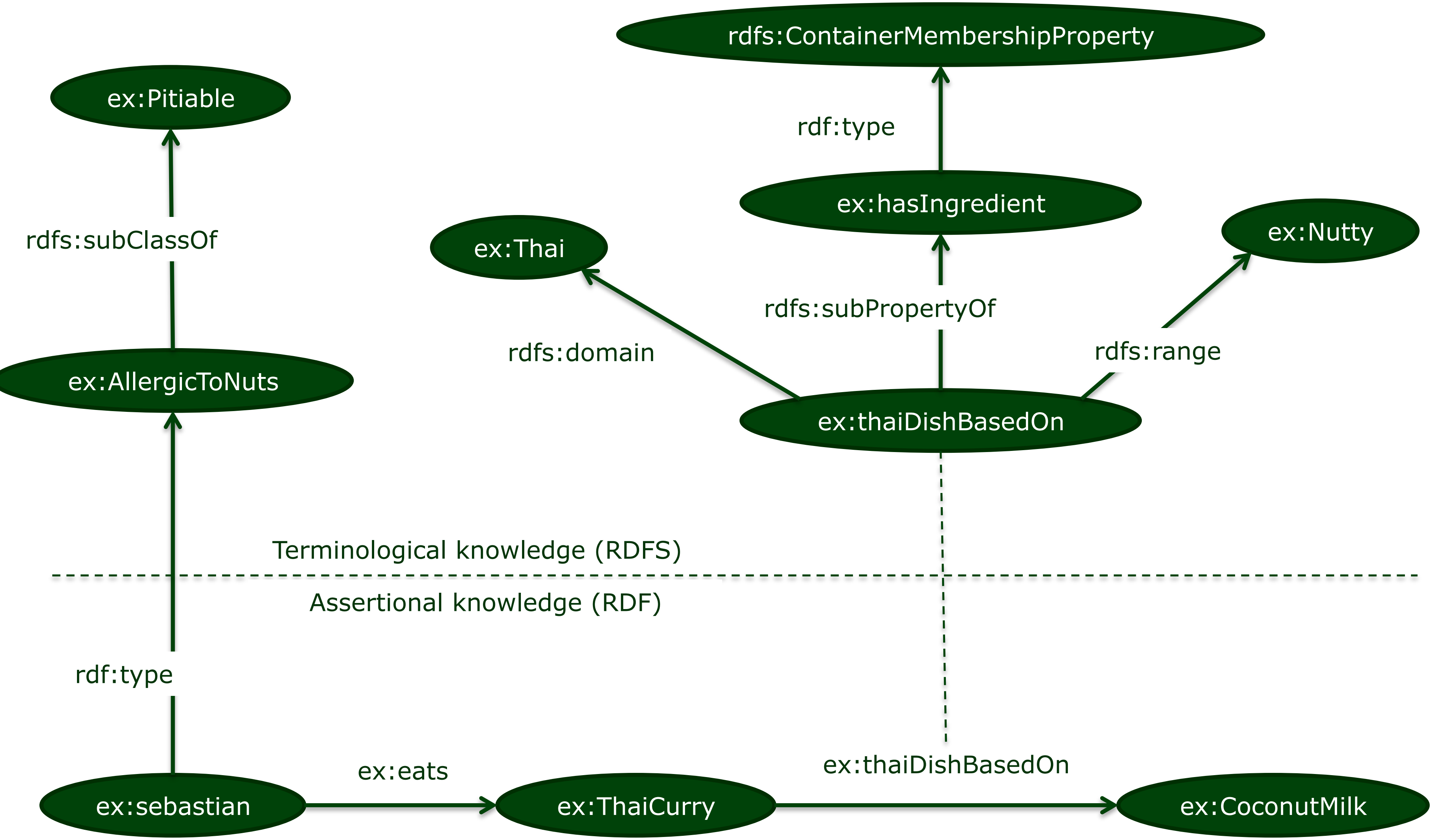
1. Motivation
2. Classes and Class Hierarchy
3. Properties and Property Hierarchy
4. Property Restrictions
5. Containers and Collections
6. Reification
7. Supplementary Information in RDFS
8. **Simple RDFS Ontologies**

Example RDFS Ontology

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

- Visualize it on the whiteboard!

Example RDFS Ontologies



Multiple Interpretations of one document

```
<rdf:Description rdf:ID="Truck">
   <rdf:type rdf:resource=
   "http://http://www.w3.org/2000/02/rdf-schema#Class"/>
   <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

- How is the above “read” as:
  - XML data?
  - RDF data?
  - RDFS knowledge?