RDF Schema – Syntax and Intuition

Werner Nutt

Acknowledgment

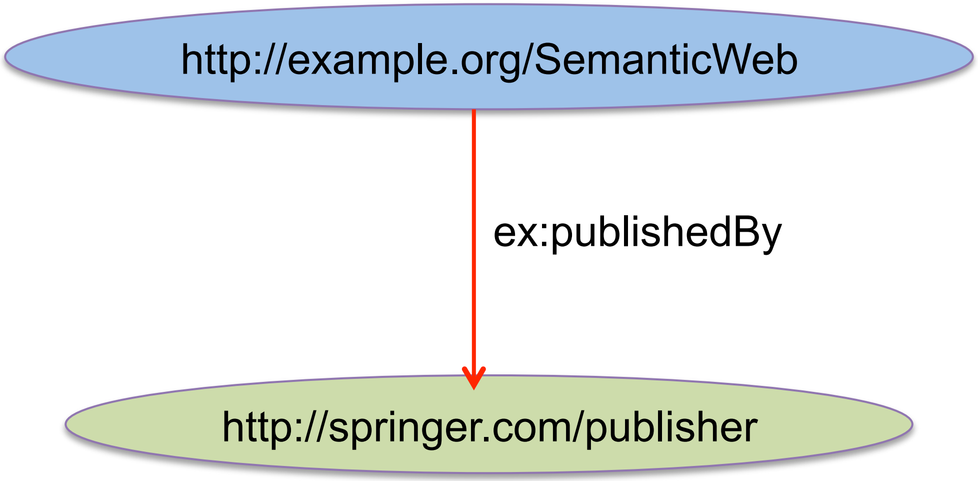
These slides are based on slide sets by Mariano Rodriguez and Sebastian Rudolph

- Motivation
- Classes and Class Hierarchies
- Properties and Property Hierarchies
- Property Restrictions
- Open Containers
- Additional Information in RDFS
- Simple Ontologies

- Motivation
- Classes and Class Hierarchies
- Properties and Property Hierarchies
- Property Restrictions
- Open Containers
- Additional Information in RDFS
- Simple Ontologies

Schema Knowledge with RDFS

RDF provides a universal possibility to encode data about facts on the Web



= Propositions about single resources (individuals), e.g. books and their relationships

Desirable: propositions about generic sets of individuals (classes), e.g. publishers, organizations, persons etc.

Schema Knowledge with RDFS/2

Also desirable: specification of logical interdependencies between individuals, classes and relationships, e.g.

- “Publishers are Organizations.”
- “Only persons write books.”

This would allow one to capture more of the semantics of the describe domain

In a database, we would collect such information in the “schema”

Schema Knowledge with RDFS/3

RDF Schema (RDFS):

- part of the W3C Recommendation of RDF
- allows for specifying schematic (also: terminological) knowledge
- uses dedicated RDF vocabulary
  (thus: every RDFS document is an RDF document)
- name space (usually abbreviated with rdfs):
  http://www.w3.org/2000/01/rdf-schema#

RDF Schema (RDFS)

- Vocabulary not domain-specific (like, e.g., with FOAF),
  but generic

- Allows for specifying (parts of) the semantics of arbitrary RDF vocabularies (could thus be called a “meta vocabulary”)
- Every RDFS-compliant software faithfully supports every vocabulary that has been defined through RDFS

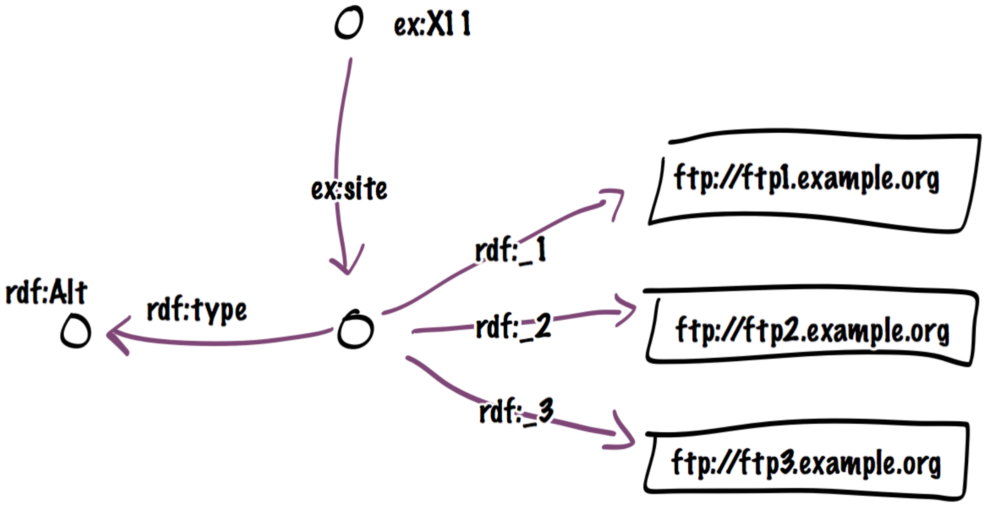
→ RDFS is a language for lightweight ontologies

“A little semantics goes a long way.” (Jim Hendler)

- Motivation
- Classes and Class Hierarchies
- Properties and Property Hierarchies
- Property Restrictions
- Open Containers
- Additional Information in RDFS
- Simple Ontologies

Classes and Instances

We have seen “typing” of resources in RDF when we discussed containers:



The predicate rdf:type

- endows the subject
- with the type denoted by the object

Or, equivalently,

- the subject is a member/instance of the class denoted by the subject

Classes and Instances/2

The triple

```
ex:SemanticWeb rdf:type ex:Textbook .
```

- characterizes “Foundations of Semantic Web Technologies” as an instance of the (newly defined) class “Textbook”.

A resource can be member of more than one class,
e.g. together with the above triple we may have:

```
ex:SemanticWeb rdf:type ex:Entertaining .
```

In general, individual and class names cannot be distinguished syntactically;

- this distinction is also difficult in reality, e.g. for

```
http://www.un.org/#URI
```

The Class of all Classes

One can also explicitly state that a URI denotes a class:

- a URI can be “typed” as class

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

- rdfs:Class is the “class of all classes”

  ➔ rdfs:Class is a member of itself

  ➔ the triple

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

is virtually present in every dataset that employs the RDFS vocabulary (according to the RDFS semantics)

Subclasses – Motivation

Suppose,

- our dataset contains `ex:SemanticWeb rdf:type ex:Textbook .`
- we are searching for instances of the class `ex:Book`

Result?

Solution Attempt 1:

- add the triple `ex:SemanticWeb rdf:type ex:Book .`
  - ➔ what happens if another `ex:Textbook` shows up?

Solution Attempt 2:

- whenever a triple *b* `rdf:type ex:Textbook .` is inserted automatically add the triple *b* `rdf:type ex:Book .`

Subclasses

Solution Attempt 3:

- introduce a statement saying that

  “*every textbook is a book*”

  that is, every instance of `ex:Textbook` is also an instance of `ex:Book`

This kind of statement can be expressed with the `rdfs:subClassOf` property:

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

means

“The class of all textbooks is a subclass of the class of all books.”

rdfs:subClassOf

- is a property
- is reflexive, thus

```
ex:Textbook rdfs:subClassOf ex:Textbook
```

- can be used to enforce that two URIs refer to the same class

```
ex:Haven rdfs:subClassOf ex:Port .
ex:Port  rdfs:subClassOf ex:Haven .
```

by declaring them subclasses of each other.

Class Hierarchies

Subclass relationships usually come in groups:

- class hierarchies (taxonomies)

E.g.,

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

RDFS semantics:

- the `rdfs:subClassOf` property is transitive

E.g., it follows from the above that

```
ex:Textbook rdfs:subClassOf ex:PrintMedium .
```

Class Hierarchies/2

Class hierarchies are often used for modeling,
e.g. in biology (Linnaean classification of living beings)

Example: zoological categorization of the modern human

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:ex=http://www.semantic-web-grundlagen.de/Beispiele#>
  <rdfs:Class rdf:about="&ex;Animalia"/>
  <rdfs:Class rdf:about="&ex;Chordata”>
    <rdfs:subClassOf rdfs:resource="&ex;Animalia"/>
  </rdfs:Class>
  <rdfs:Class rdf:about="&ex;Mammalia”>
    <rdfs:subClassOf rdfs:resource="&ex;Chordata"/>
  </rdfs:Class>
  <rdfs:Class rdf:about="&ex;Primates”>
    <rdfs:subClassOf rdfs:resource="&ex;Mammalia"/>
  </rdfs:Class>
  <rdfs:Class rdf:about="&ex;Hominidae”>
    <rdfs:subClassOf rdfs:resource="&ex;Primates"/>
  </rdfs:Class>
  ...
```

Classes

Intuitively, classes correspond to sets in set theory

`rdf:type`	corresponds to	$\in$
`rdfs:subClassOf`	corresponds to	$\subseteq$

This motivates

- reflexivity and transitivity of `rdfs:subClassOf` …
- and more inferences that we will see later on

However, as we will also see, the semantics of RDFS is much weaker than set theory

(otherwise inferences would be too difficult)

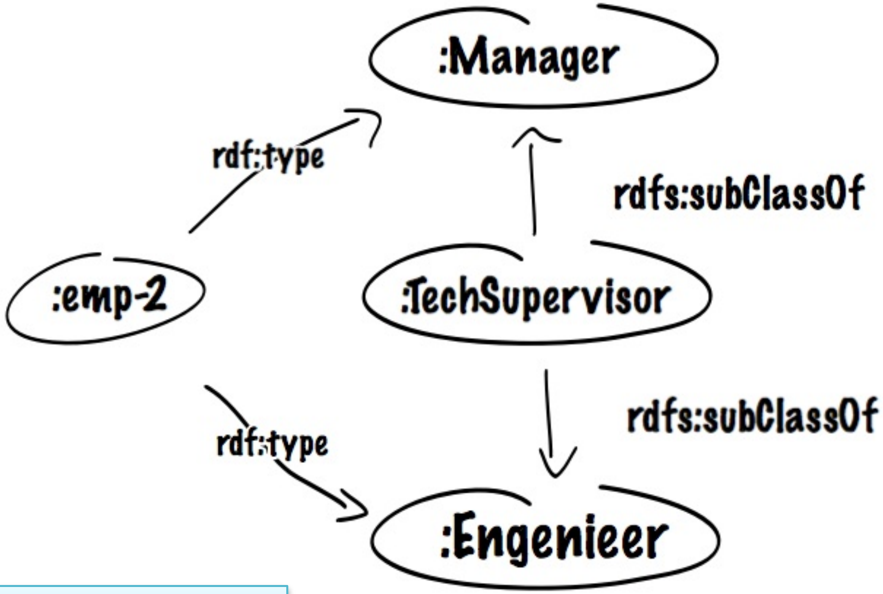
Set “intersection”

- Proper set intersection is not possible in RDFS
- However, expressing necessary membership to multiple classes is possible, i.e., A subset B AND C

```
A rdfs:subClassOf B
A rdfs:subClassOf C
```

consider

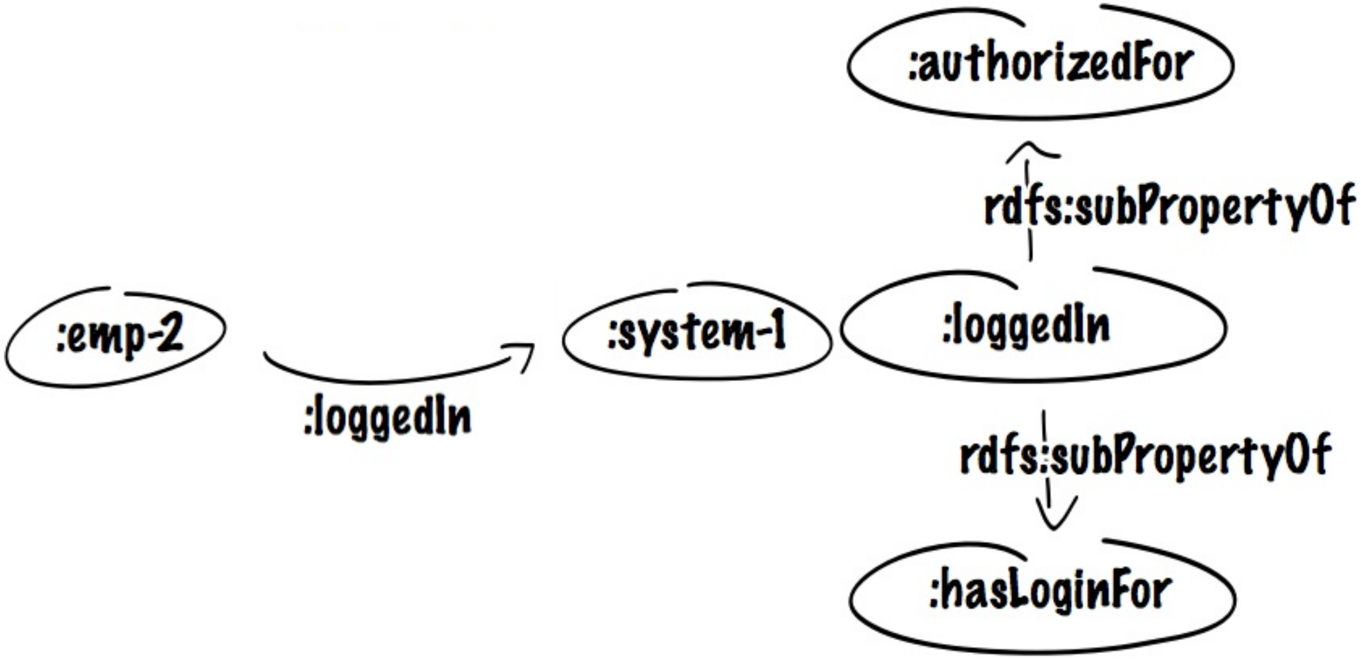
```
x rdf:type A
```



One direction only!

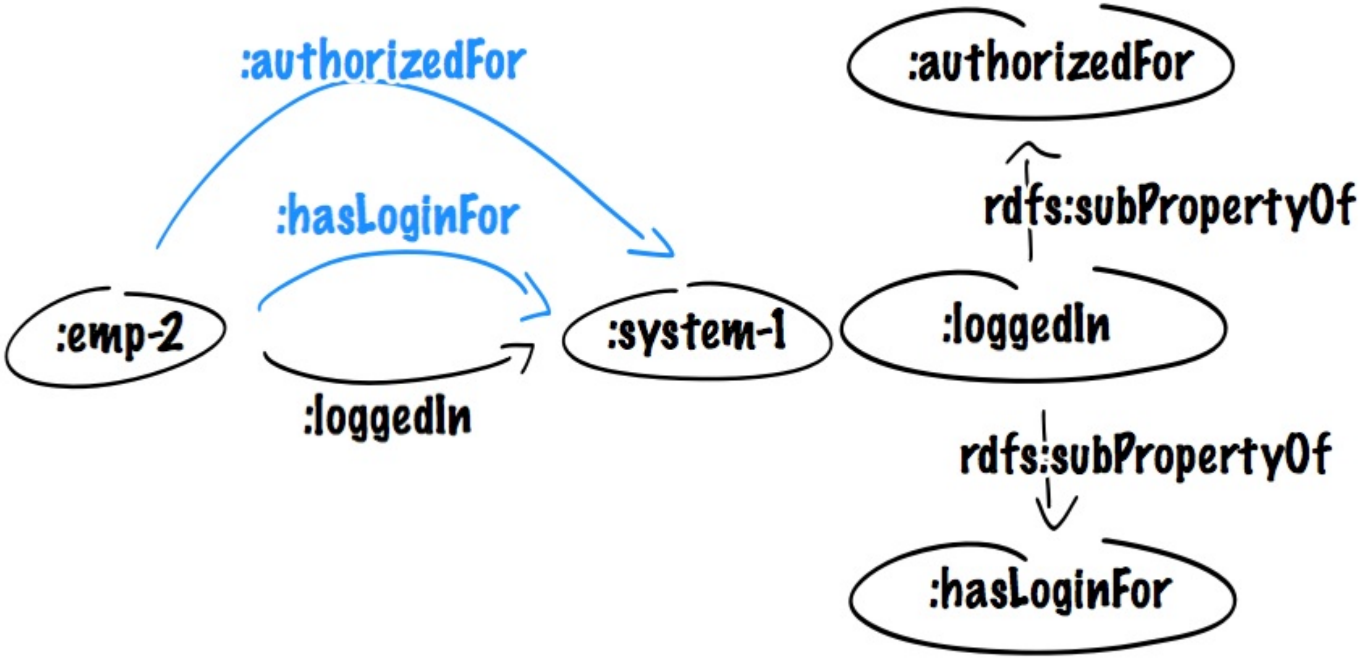
Set “intersection”

- Similar for roles



Set “intersection”

- Similar for roles



Set “union”

- Proper set union is not possible in RDFS
- However, A OR B subsetOf C

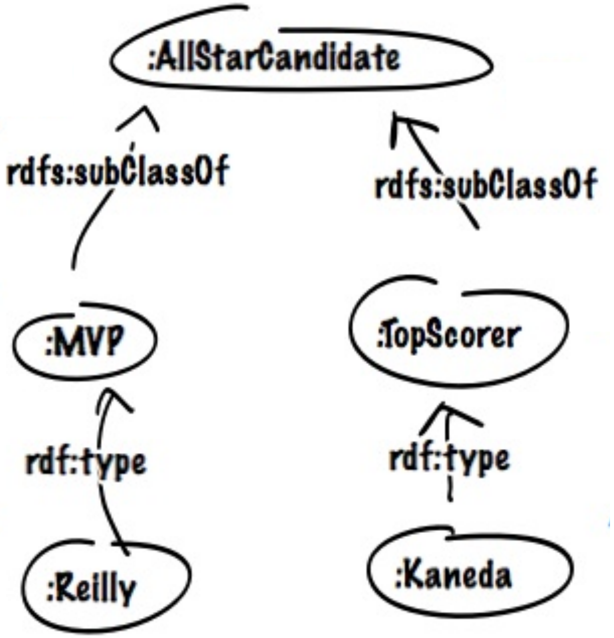
```
B rdfs:subClassOf A
C rdfs:subClassOf A
```

consider

```
x rdf:type B
```

or

```
x rdf:type C
```



Set “union”

- Proper set union is not possible in RDFS
- However, A OR B subsetOf C

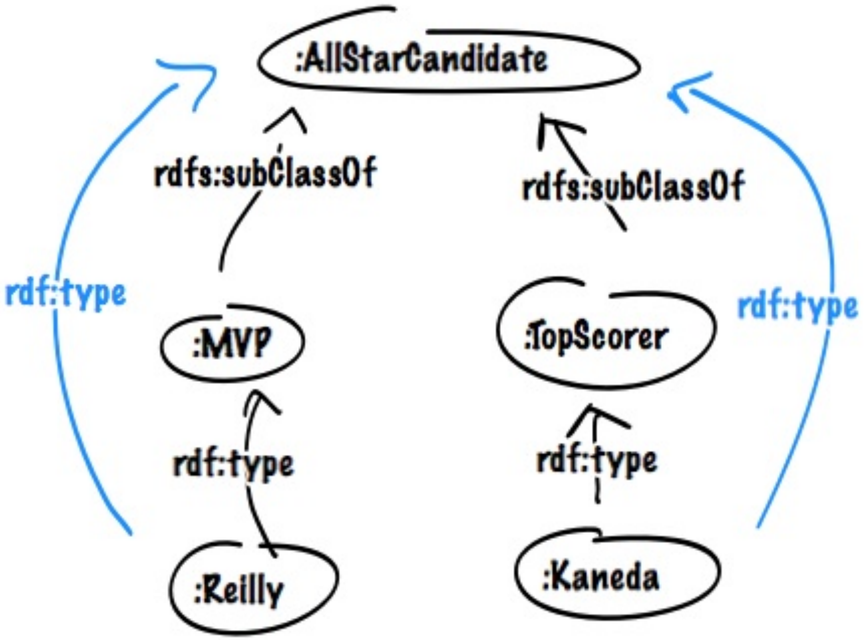
```
B rdfs:subClassOf A
C rdfs:subClassOf A
```

consider

```
x rdf:type B
```

or

```
x rdf:type C
```



Classes in RDF/XML Syntax

Abbreviated notation for specifying class instances:

```
<ex:HomoSapiens rdf:about="&ex;WernerNutt"/>
```

instead of

```
<rdf:Description rdf:about="&ex;WernerNutt">
     <rdf:type rdf:resource="&ex;HomoSapiens">
</rdf:Description>
```

Likewise:

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

Predefined Class URIs

- `rdfs:Resource`
  class of all resources (i.e., all elements of the domain)

- `rdf:Property`
  class of all relationships
  (= 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

Predefined Class URIs/2

- `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

- Motivation
- Classes and Class Hierarchies
- Properties and Property Hierarchies
- Property Restrictions
- Open Containers
- Additional Information in RDFS
- Simple Ontologies

Properties

- Properties characterize, in which way two resources are related to each other
  - also called: relations, relationships
- Beware: unlike in OOP, properties in RDF(S) are not assigned to classes
- Property URIs normally appear in the predicate position of a triple
- Mathematically, the content of a property/relation is sometimes represented as set of pairs:

$$\texttt{marriedWith} = \{(\texttt{Adam}, \texttt{Eve}), (\texttt{Brad}, \texttt{Angelina}), \ldots\}$$

- A URI can be marked as property name by typing it accordingly:

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

Subproperties

Like sub-/superclasses, also sub-/superproperties are possible and useful

- Specification in RDFS via `rdfs:subPropertyOf`, e.g.:

```
ex:happilyMarriedWith rdf:subPropertyOf
                      ex:marriedWith .
```

- Inference:
  Given

```
ex:mark ex:happilyMarriedWith ex:ann .
```

we can infer

```
ex:mark ex:marriedWith ex:ann .
```

- Motivation
- Classes and Class Hierarchies
- Properties and Property Hierarchies
- Property Restrictions
- Open Containers
- Additional Information in RDFS
- Simple Ontologies

Property Restrictions

Often: Usage of a property only makes sense for a certain kinds of resources, e.g.

`ex:publishedBy` only connects publications with publishers

➔ for all URIs *a*, *b*, the triple

```
a ex:publishedBy b .
```

intuitively entails:

```
a rdf:type ex:Publication .
b rdf:type ex:Publisher .
```

We can express this directly in RDFS:

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

domain restriction

range restriction

Can also be used to “prescribe” datatypes for literals:

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

Property Restrictions/2

- Property restrictions are the only way of specifying semantic interdependencies between properties and classes
- Attention: property restrictions are interpreted globally and conjunctively.
  E.g., what follows from

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

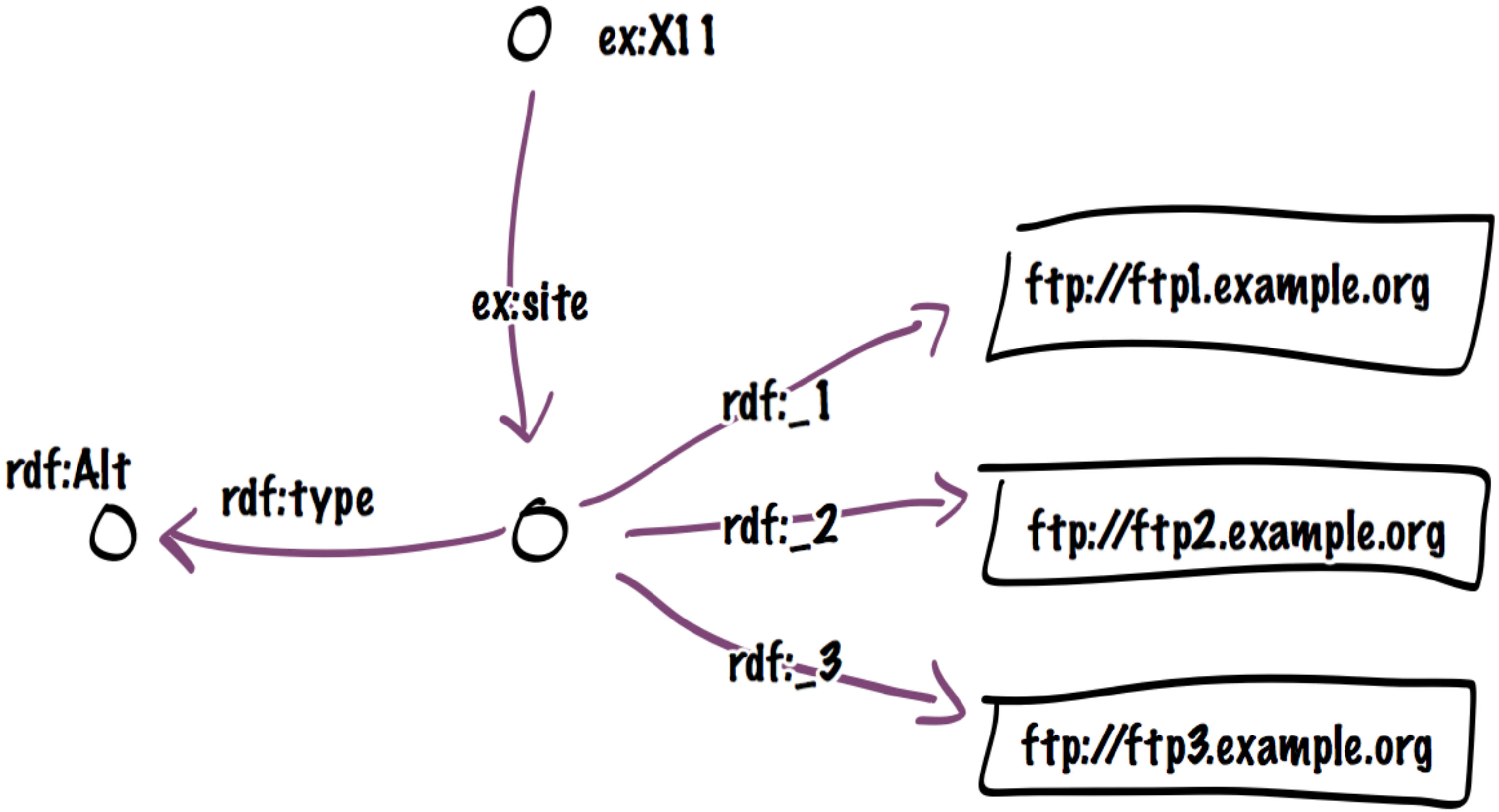
➔ This entails: `ex:fredsBook` is both a cookbook and a storybook

- Thus: When designing an RDFS schema,
  pick the most general possible class for domain/range specifications

- Motivation
- Classes and Class Hierarchies
- Properties and Property Hierarchies
- Property Restrictions
- Open Containers
- Additional Information in RDFS
- Simple Ontologies

Open Containers

Reminder: open collections in RDF



Open Containers/2

- New class: `rdfs:Container` as superclass of `rdf:Seq, rdf:Bag, rdf:Alt`
- New class: `rdfs:ContainerMembershipProperty`: instances of this class are no proper individuals, but themselves properties
- Intended semantics: every property encoding that

  the subject contains the object

  is an instance of

  `rdfs:ContainerMembershipProperty`

In particular, we have

```
rdf:_1 rdf:type rdfs:ContainerMembershipProperty .
rdf:_2 rdf:type rdfs:ContainerMembershipProperty .
```

etc.

Open Containers/3

- New property: `rdfs:member`

  superproperty of all properties that are instances of

  ```
  rdfs:ContainerMembershipProperty,
  ```

  could be called the “universal containedness relation”

- Part of the semantics of RDFS:
  whenever for a property *p* the triple

  ```
  p rdf:type rdfs:ContainerMembershipProperty .
  ```

  holds, then the triple

  ```
  a p b.
  ```

  gives rise to the triple

  ```
  a rdfs:member b .
  ```

- Motivation
- Classes and Class Hierarchies
- Properties and Property Hierarchies
- Property Restrictions
- Open Containers
- Additional Information in RDFS
- Simple Ontologies

Additional Information in RDFS

Like with programming languages, one sometimes wants to add comments (without changing the semantics)

- Purpose: increase understandability for human users

➔ Format of comments in graph?

- comments are nodes
- a comment node is attached to the commented node
- attachment is achieved by a suitable property

➔ Task: define a set of properties that serve this purpose

Additional Information/2

`rdfs:label`

- Property that assigns a name (Literal) to an arbitrary resource
- Often, URIs themselves are difficult to read, or “bulky” at best
- Names provided via `rdfs:label` are often used by tools that graphically represent the data

Example (also featuring language information):

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

Additional Information/3

`rdfs:comment`

- Property assigning an extensive comment (literal) to an arbitrary resource
- may e.g. contain the natural language description of a newly introduced class – this facilitates later usage

`rdfs:seeAlso, rdfs:definedBy`

- Properties giving resources (URIs!) where one can find further information or a definition of the subject resource

Additional Information/4

Example of usage

```
…
xmlns:wikipedia=http://en.wikipedia.org/wiki
…
<rdfs:Class rdf:about="&ex;Primates">
  <rdfs:label xml:lang="en">Primates</rdfs:label>
  <rdfs:comment>An order of mammals. Primates are
         characterized by a highly developed brain. Most
         primates live in tropical or subtropical regions.
  </rdfs:comment>
  <rdfs:seeAlso rdfs:resource="&wikipedia;Primate"/>
  <rdfs:subClassOf rdfs:resource="&ex;Mammalia"/>
</rdfs:Class>
```

- Motivation
- Classes and Class Hierarchies
- Properties and Property Hierarchies
- Property Restrictions
- Open Containers
- Additional Information in RDFS
- Simple Ontologies

Simple Ontologies

- 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.
- Consequently, RDFS can be seen as a (though not overly expressive) ontology language.

Exercise

Express in RDF and RDFS:

- Vegetable Thai curry is a Thai dish based on coconut milk.
- Fred is allergic to nuts.
- Fred eats vegetable Thai curry.
- Everyone allergic to nuts is pitiable.
- Everything having the property “Thai dish based on” is Thai.
- Everything satisfying the property “Thai dish based on” is nutty.
- The property “Thai dish based on” is a special case of the property “has ingredient”
- “Has ingredient” is a containedness relation.

Simple Ontology – Example

```
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 .
```

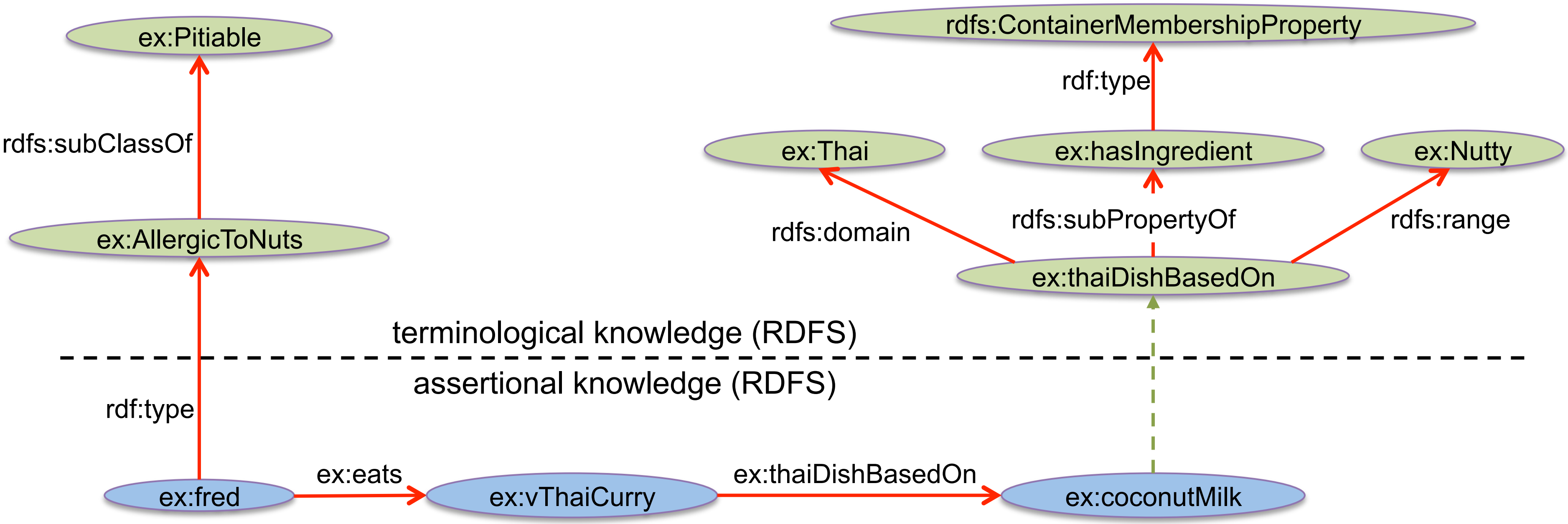
What would this look like graphically?

Distinguish between

- *assertional knowledge*
- *terminological knowledge*

Simple Ontology – Example

```
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 .
```



One Document – Three Interpretations

```
<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>
```

Interpretation 1: An XML tree

[Fill in a drawing!]

One Document – Three Interpretations

```
<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>
```

Interpretation 2: An RDF dataset

[Fill in a drawing!]

One Document – Three Interpretations

```
<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>
```

Interpretation 3: An RDFs schema

[Fill in a drawing!]

RDFS Exercise

Decide whether the following propositions can be satisfactorily modeled in RDFS and, if so, give the corresponding RDF(S) specification.

- Every pizza is a meal.
- Pizzas always have at least two toppings.
- Every pizza from the class PizzaMargarita has a Tomato topping.
- Everything having a topping is a pizza.
- No pizza from the class PizzaMargarita has a topping from the class Meat.
- “Having a topping” is a containedness relation.

RDFS Inferences

In RDFS, we can express statements about

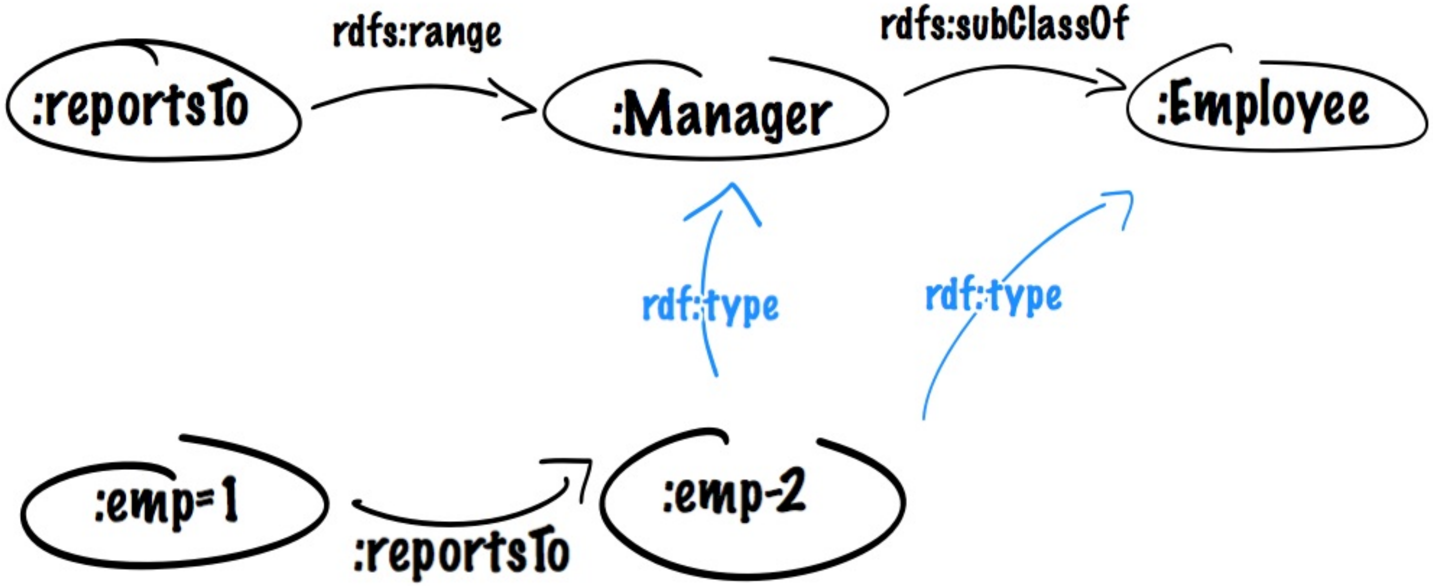
- resources/nodes being a member of a classes
- classes being subclasses of other classes
- properties being subproperties of other properties
- classes being domains and ranges of properties

What conclusions can we draw from such statements?

How do these statements interact?

Interactions

- All inferences interact


rdfs:range
rdfs:subClassOf
:reportsTo
:Manager
:Employee
rdf:type
rdf:type
:emp-1
:reportsTo
:emp-2