Knowledge Representation for the Semantic Web

Winter Quarter 2012

Slides 3 – 01/10/2012

Pascal Hitzler

Kno.e.sis Center

Wright State University, Dayton, OH
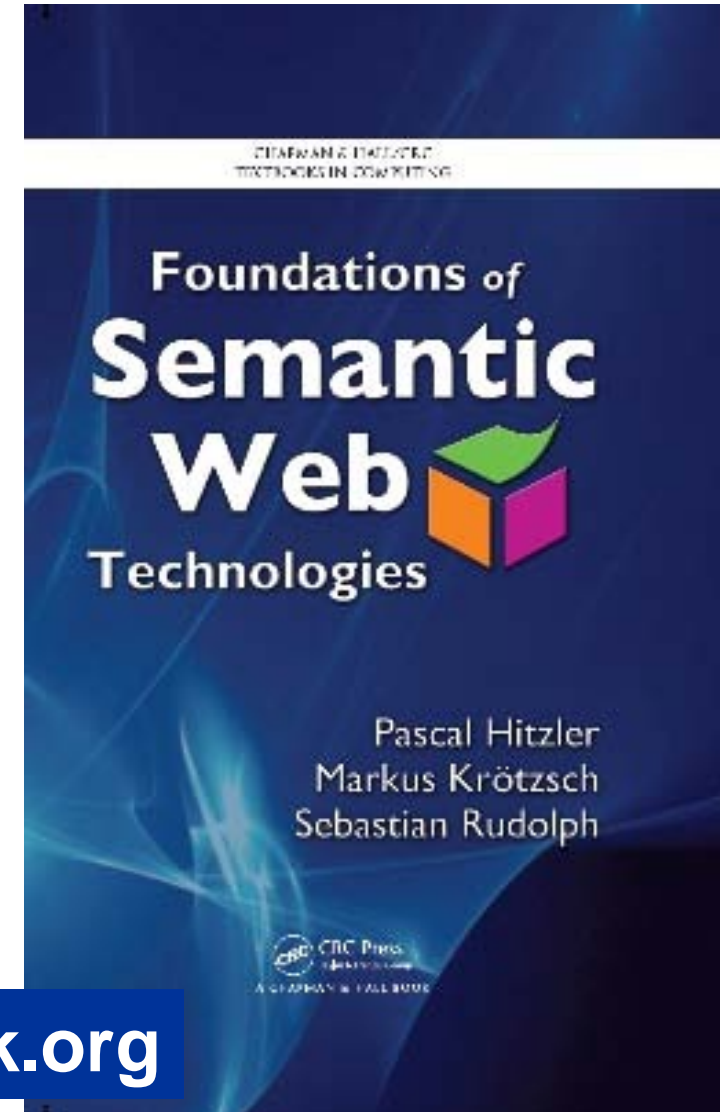
http://www.knoesis.org/pascal/

Textbook (required)

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph

Foundations of Semantic Web Technologies
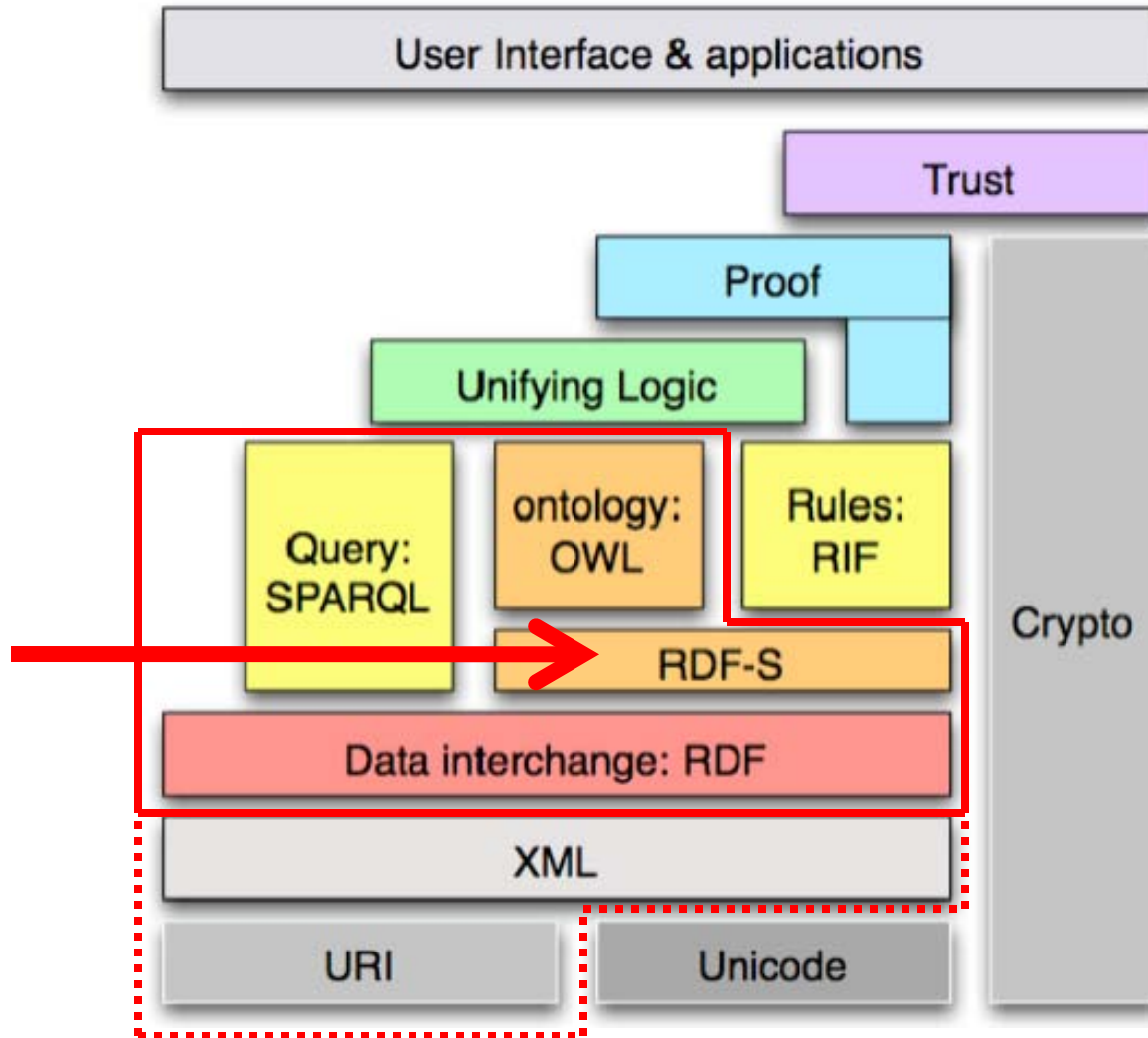
Chapman & Hall/CRC, 2010

Choice Magazine Outstanding Academic Title 2010 (one out of seven in Information & Computer Science)

http://www.semantic-web-book.org

Foundations of Semantic Web Technologies

Pascal Hitzler
Markus Krötzsch
Sebastian Rudolph

Today's Session: RDF Schema

1. **Motivation**
2. **Classes and Class Hierarchies**
3. **Properties and Property Hierarchies**
4. **Property Restrictions**
5. **Open Lists Revisited**
6. **Reification**
7. **Supplementary Information in RDFS**
8. **Simple Ontologies in RDFS**
9. **Class project**
10. **Class presentations**

Motivation

- **RDF allows to express facts**
  - **Anne is the mother of Merula**

- **But we'd like to be able to express more generic knowledge**
  - **Mothers are female**
  - **If somebody has a daughter then that person is a parent**

- **This kind of knowledge is often called *schema* knowledge or *terminological* knowledge.**

- **RDF Schema allows us to do some schema knowledge modeling. OWL (discussed later) gives even more expressivity.**

RDF Schema (RDFS)

- **part of the W3C Recommendation RDF**
- **for schema/terminological knowledge**
- **uses RDF vocabulary with pre-defined semantics**
- **every RDFS document is an RDF document**
- **Namespace: http://www.w3.org/2000/01/rdf-schema# - usually abbreviated by rdfs:**

- **vocabulary is generic, not bound to a specific application area**
  - **allows to (partially) specify the semantics of other/user-defined vocabularies (it's a kind of meta vocabulary)**
  - **hence, RDF software correctly interprets each vocabulary defined using RDF Schema**

Today's Session: RDF Schema

1. **Motivation**
2. **Classes and Class Hierarchies**
3. **Properties and Property Hierarchies**
4. **Property Restrictions**
5. **Open Lists Revisited**
6. **Reification**
7. **Supplementary Information in RDFS**
8. **Simple Ontologies in RDFS**
9. **Class project**
10. **Class presentations**

Classes and Instances

- **Classes stand for sets of things.
  In RDF: Sets of URIs.**

- **book:uri is a member of the class ex:Textbook**

```
book:uri     rdf:type     ex:Textbook .
```

- **a URI can belong to several classes**

```
book:uri     rdf:type     ex:Textbook .
book:uri     rdf:type     ex:WorthReading .
```

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

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

Pre-defined classes

- **every URI denoting a class is a member of rdfs:Class**

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

- **this also makes rdfs:Class a member of rdfs:Class (!)**

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

- **rdfs:Resource (class of all URIs)**
- **rdf:Property (class of all properties)**
- **rdf:XMLLiteral**
- **rdfs:Literal (each datatype is a subclass)**
- **rdf:Bag, rdf:Alt, rdf:Seq, rdfs:Container , rdf:List, rdf:nil, rdfs:ContainerMembershipProperty (see later)**
- **rdfs:Datatype (contains all datatypes – a class of classes)**
- **rdf:Statement (see later)**

Implicit knowledge

- **if an RDFS document contains**

```
u       rdf:type     ex:Textbook .
```

and

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

then

```
u       rdf:type     ex:Book .
```

is *implicitly* also the case: it's a *logical consequence*. (We can also say it is *deduced* (deduction) or *inferred* (inference). We do not have to state this explicitly. Which statements are logical consequences is governed by the formal semantics (covered in the next session).

- **From**

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

the following is a logical consequence:

```
ex:Textbook    rdfs:subClassOf    ex:PrintMedia .
```

I.e. rdfs:subClassOf is *transitive*.

Using implicit knowledge

Ontology (Knowledge Base)
e.g. RDF or OWL

online ⇕

Used like a database

Reasoner (accesses implicit knowledge)

⇕

Application

Using implicit knowledge

Ontology (Knowledge Base)
e.g. RDF or OWL

Reasoner (produces implicit knowledge)

offline

Completed (materialized) knowledge base

Used like a database

Application

Class equivalence
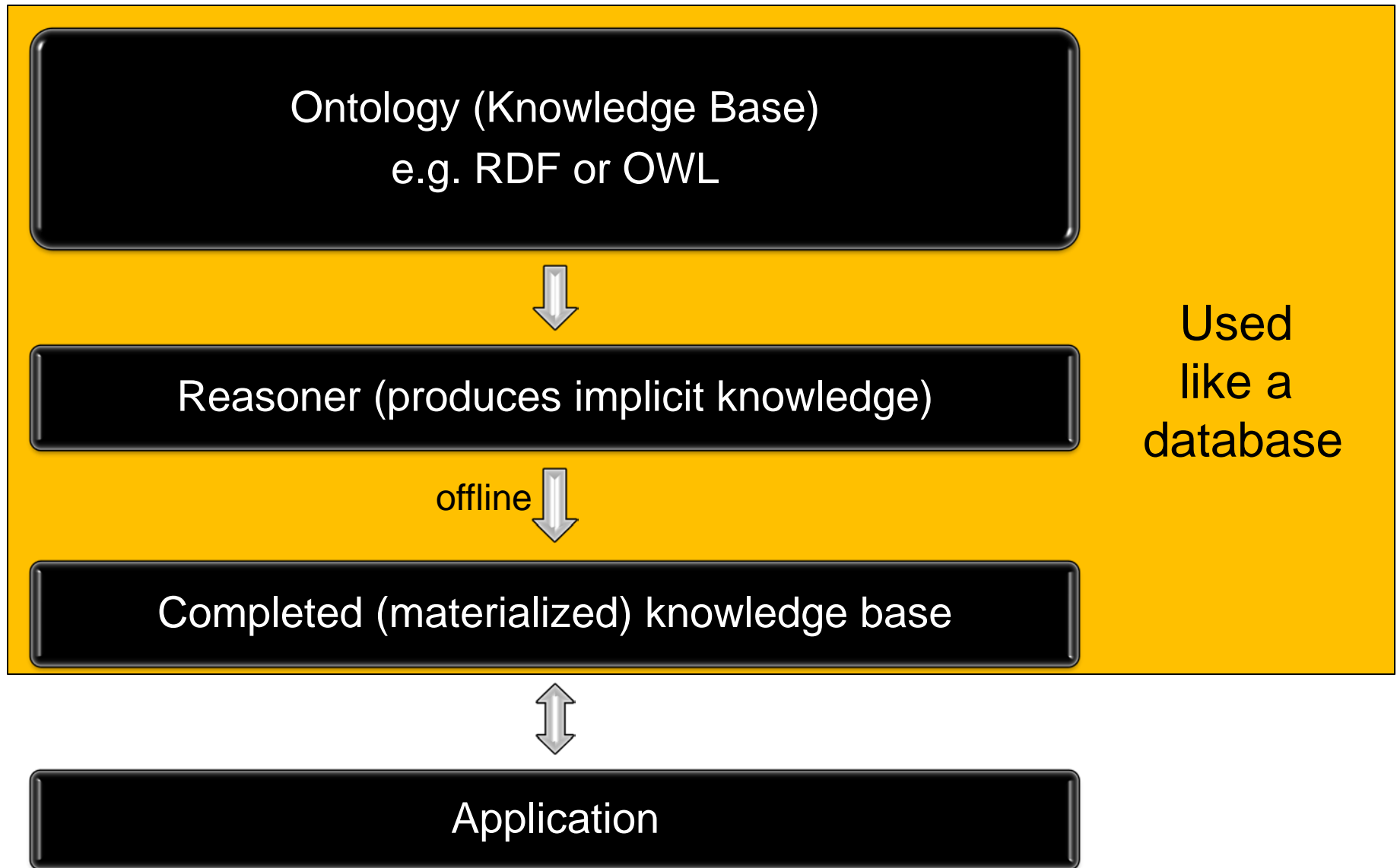
```
ex:MorningStar    rdfs:subClassOf    ex:EveningStar .
ex:EveningStar    rdfs:subClassOf    ex:MorningStar .
```

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

I.e. rdfs:subClassOf is *reflexive*.

Classes and RDF/XML syntax

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

is short for

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

Today's Session: RDF Schema

1. **Motivation**
2. **Classes and Class Hierarchies**
3. **Properties and Property Hierarchies**
4. **Property Restrictions**
5. **Open Lists Revisited**
6. **Reification**
7. **Supplementary Information in RDFS**
8. **Simple Ontologies in RDFS**
9. **Class project**
10. **Class presentations**

Property Hierarchies

From

ex:isHappilyMarriedTo   rdf:subPropertyOf ex:isMarriedTo.

and

ex:markus   ex:isHappilyMarriedTo   ex:anja .

we can infer that

ex:markus   ex:isMarriedTo   ex:anja .

Today's Session: RDF Schema

1. **Motivation**
2. **Classes and Class Hierarchies**
3. **Properties and Property Hierarchies**
4. **Property Restrictions**
5. **Open Lists Revisited**
6. **Reification**
7. **Supplementary Information in RDFS**
8. **Simple Ontologies in RDFS**
9. **Class project**
10. **Class presentations**

Property Restrictions

- **Allow to state that a certain property can only be between things of a certain rdf:type.**

- **E.g. when a is married to b, then both a and b are Persons.**

- **Expressed by rdfs:domain and rdfs:range:**

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

- **And similarly for datatypes:**

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

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

states that everything in the rdfs:range of ex:authorOf is both a ex:Textbook and a ex:Storybook!

```
ex:isMarriedTo        rdfs:domain      ex:Person .
ex:isMarriedTo        rdfs:range       ex:Person .
ex:instituteAIFB      rdf:type         ex:Institution .
```

```
ex:pascal             ex:isMarriedTo   ex:instituteAIFB .
```

A logical consequence of this is

```
ex:instituteAIFB    rdf:type          ex:Person .
```

Today's Session: RDF Schema

1. **Motivation**
2. **Classes and Class Hierarchies**
3. **Properties and Property Hierarchies**
4. **Property Restrictions**
5. **Open Lists Revisited**
6. **Reification**
7. **Supplementary Information in RDFS**
8. **Simple Ontologies in RDFS**
9. **Class project**
10. **Class presentations**

Open Lists revisited
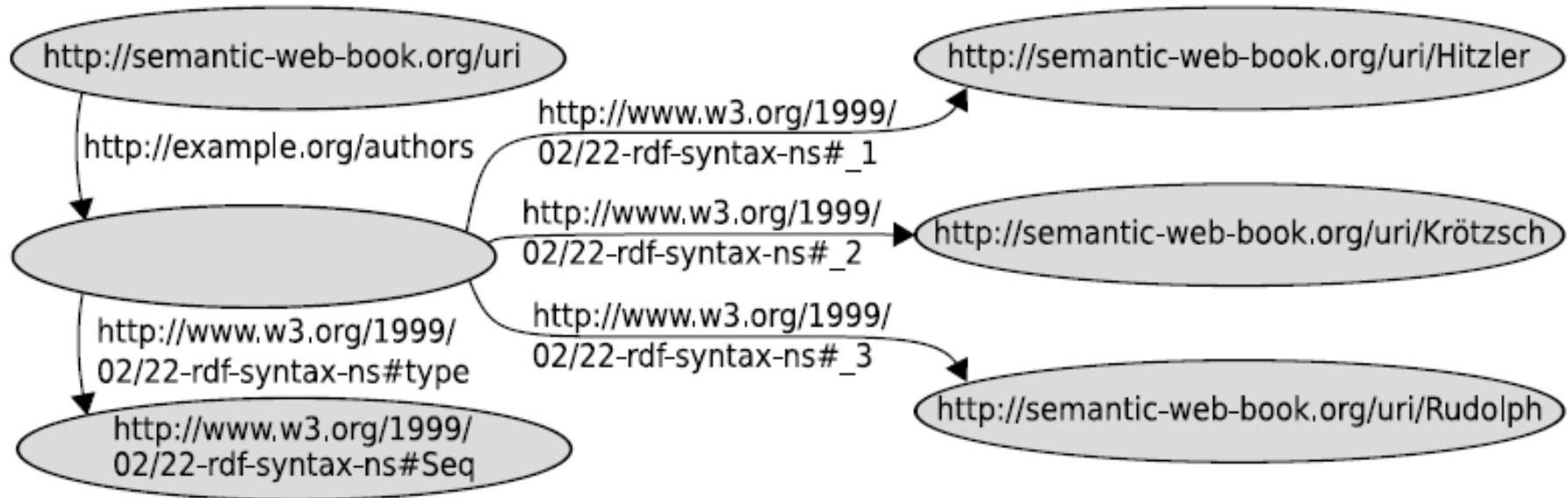
- **New class: rdfs:Container as superclass of rdf:Seq, rdf:Bag, rdf:Alt.**
- **New class: rdfs:ContainerMembershipProperty containing the properties used with containers, e.g.**

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

Open Lists revisited

- **New property rdfs:member
  Is superproperty of all properties contained in
  rdfs:ContainerMembershipProperty.**

- **The RDFS semantics specifies:**

  From

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

  and

  ```
  a  p  b  .
  ```

  the following is inferred:

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

Today's Session: RDF Schema

1. **Motivation**
2. **Classes and Class Hierarchies**
3. **Properties and Property Hierarchies**
4. **Property Restrictions**
5. **Open Lists Revisited**
6. **Reification**
7. **Supplementary Information in RDFS**
8. **Simple Ontologies in RDFS**
9. **Class project**
10. **Class presentations**

Talking about triples

- **How do you state in RDF:**
  "The detective supposes that the butler killed the gardener."

- **unsatisfactory:**

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

ex:detective    ex:supposes    ex:theButlerKilledTheGardener .
```

- **We would really like to talk about the triple**

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

- **How to do it properly in RDFS:**

```
ex:detective    ex:supposes      ex:theory .
ex:theory       rdf:subject      ex:butler .
ex:theory       rdf:predicate    ex:hasKilled .
ex:theory       rdf:object       ex:gardener .
```
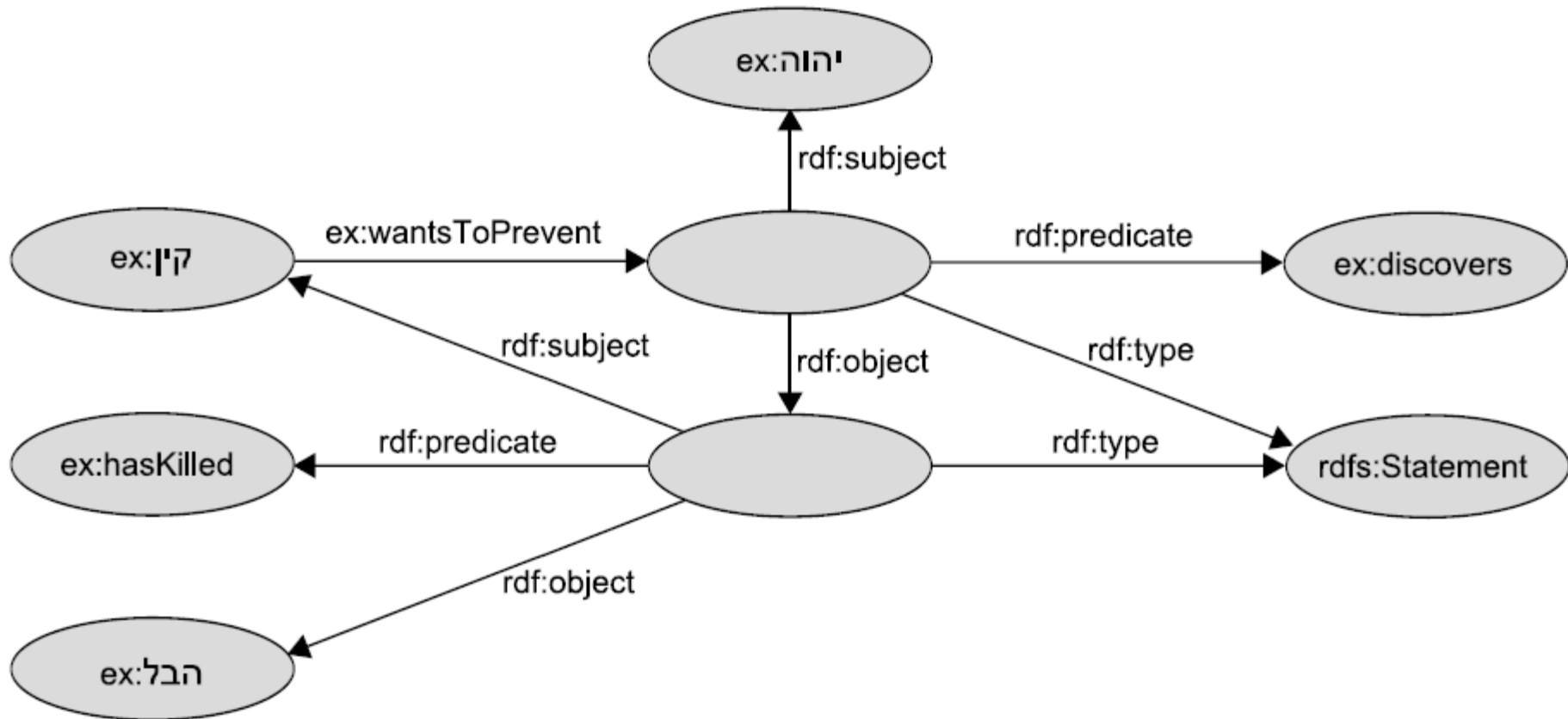
```
ex:theory       rdf:type         rdf:Statement .
```

- **Note however, that the following is not a logical consequence of this:**

```
ex:butler    ex:hasKilled    ex:gardener    .
```

- **One would usually use a blank node instead of ex:theory.**

A reification puzzle

You know that story? It's in the old testament :)

Today's Session: RDF Schema

1. **Motivation**
2. **Classes and Class Hierarchies**
3. **Properties and Property Hierarchies**
4. **Property Restrictions**
5. **Open Lists Revisited**
6. **Reification**
7. **Supplementary Information in RDFS**
8. **Simple Ontologies in RDFS**
9. **Class project**
10. **Class presentations**

Supplementary information

- **comments etc. which are not part of the actual ontology, but are for the human reader/user/developer**

- **in RDF, we also use triples to encode these**

- **i.e. we have a set of pre-defined properties which do this job**

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

Supplementary Information example

```
  :
 xmlns:wikipedia="http://en.wikipedia.org/wiki/"
  :
<rdfs:Class rdf:about="&ex;Primates">
  <rdfs:label xml:lang="en">primates</rdfs:label>
  <rdfs:comment>
     Order of mammals. Primates are characterized by an
     advanced brain. They mostly populate the tropical
     earth regions. The term 'Primates' was coined by
     Carl von Linné.
  </rdfs:comment>
  <rdfs:seeAlso rdf:resource="&wikipedia;Primates" />
  <rdfs:subClassOf rdfs:resource="&ex;Mammalia" />
</rdfs:Class>
```

Today's Session: RDF Schema

1. **Motivation**
2. **Classes and Class Hierarchies**
3. **Properties and Property Hierarchies**
4. **Property Restrictions**
5. **Open Lists Revisited**
6. **Reification**
7. **Supplementary Information in RDFS**
8. **Simple Ontologies in RDFS**
9. **Class project**
10. **Class presentations**

An example 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.
```
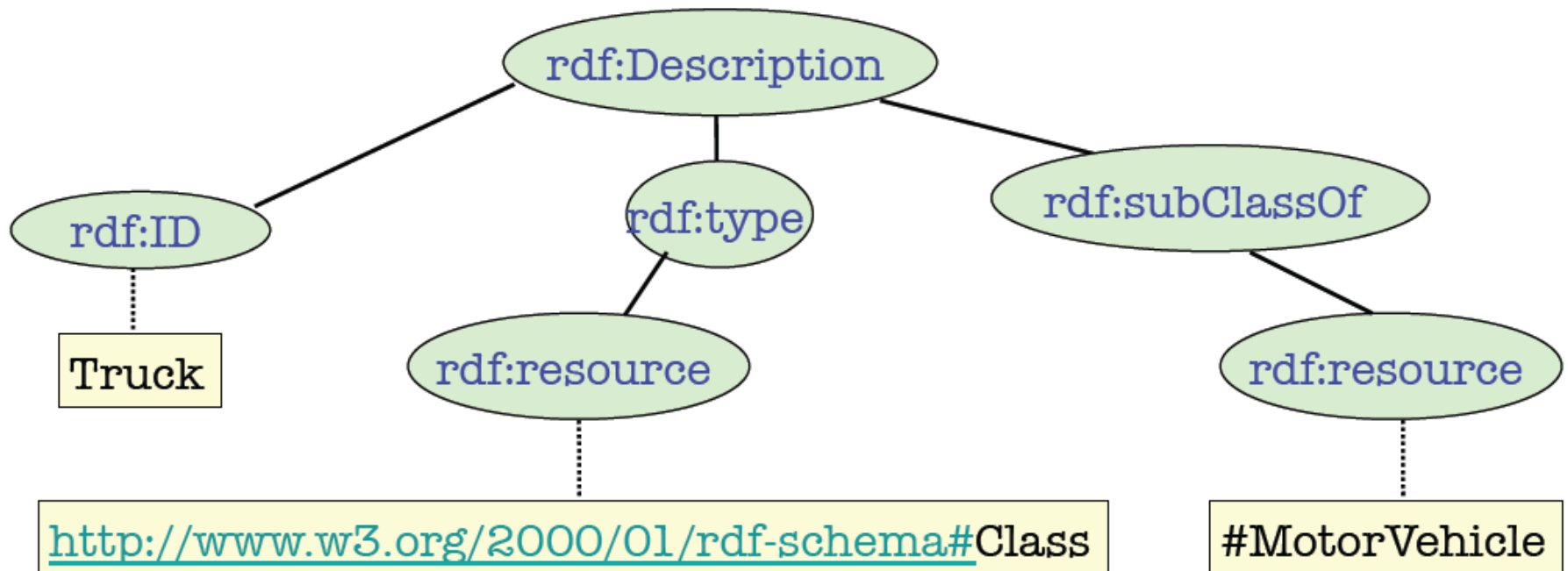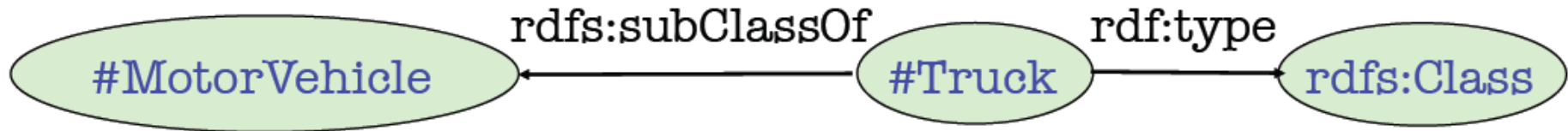
The same as graph
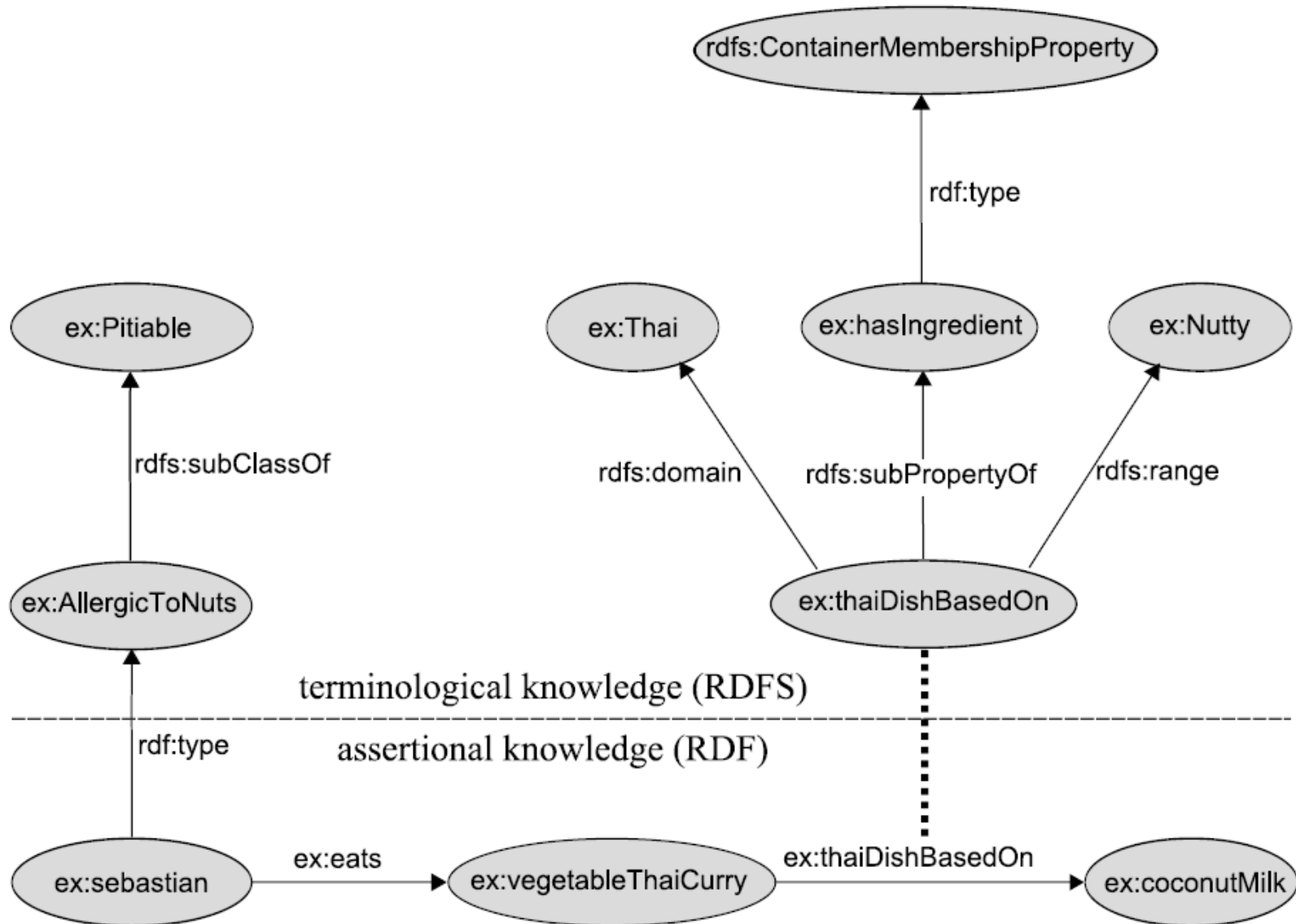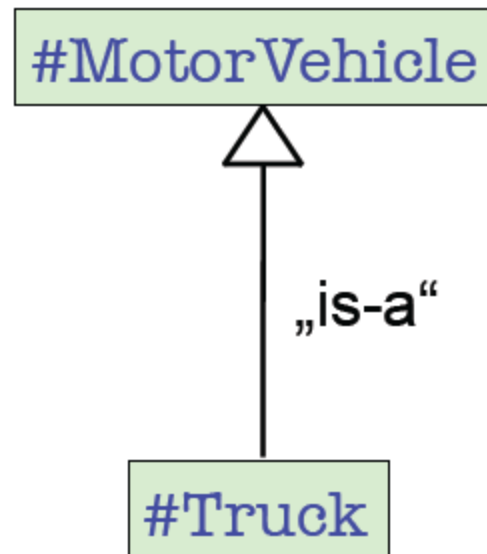
Note the multiple views: XML

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

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


#MotorVehicle ←—— rdfs:subClassOf —— #Truck —— rdf:type —→ rdfs:Class

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

#MotorVehicle

△

„is-a"

#Truck

Today's Session: RDF Schema

1. **Motivation**
2. **Classes and Class Hierarchies**
3. **Properties and Property Hierarchies**
4. **Property Restrictions**
5. **Open Lists Revisited**
6. **Reification**
7. **Supplementary Information in RDFS**
8. **Simple Ontologies in RDFS**
9. **Class project**
10. **Class presentations**

- **When is something an instance? When is something a class?**

Father	rdf:type	SocialRole .
Pascal	rdf:type	Father .

- **What about triples like the following?**

Parasite	hasHostOrganism	LivingThing .
LeapYear	isFollowedby	NonLeapYear .

- **These all are valid RDF triples, and it's also valid RDFS.**

- **But what does it mean?**

Type separation

- **It's usually good to clearly separate types (as long as it's feasible) and only break this if really needed.
Types: instances, properties, classes**
- **Reason: The semantics is clearer.**

- **<instance> rdf:type <class>**
- **<instance> someProperty <instance>**
- **<class> rdfs:subClassOf <class>**
- **<property> rdfs:subPropertyOf <property>**

- **In OWL 1 DL, type separation was strictly enforced.**
- **In OWL 2 DL, it's more relaxed, but the semantics is different.**

- **We'll talk more about this in the OWL sessions.**

Class project: next step

- **keep bugfixing**
- **extend, where necessary, your ontology so that it makes a correct use of each of the following (each at least once):**
  - **rdf:datatype**
  - **rdfs:subPropertyOf**
- **for each property in your ontology, add triples which give their rdfs:domain and rdfs:range.**
- **write up your ontology in RDF Turtle syntax and group axioms in such a way that it's easy to keep an overview of the contents.**

- **send to me by Saturday the 14th**
  - **the Turtle file as .txt file (validator: http://www.rdfabout.com/demo/validator/)**
  - **brief notes with lessons learned from this round of modeling (including the bugfixing)**

1. **Motivation**
2. **Classes and Class Hierarchies**
3. **Properties and Property Hierarchies**
4. **Property Restrictions**
5. **Open Lists Revisited**
6. **Reification**
7. **Supplementary Information in RDFS**
8. **Simple Ontologies in RDFS**
9. **Class project**
10. **Class presentations**

Class presentations – first topics

<nothing yet>

Class planning (tentative)

Tuesday 10th of January: RDF Schema
Thursday 12th of January: RDF and RDFS Semantics
Tuesday 17th of January: Exercise Session