TECHNISCHE UNIVERSITÄT DRESDEN

FOUNDATIONS OF SEMANTIC WEB TECHNOLOGIES

RDF Schema

Sebastian Rudolph

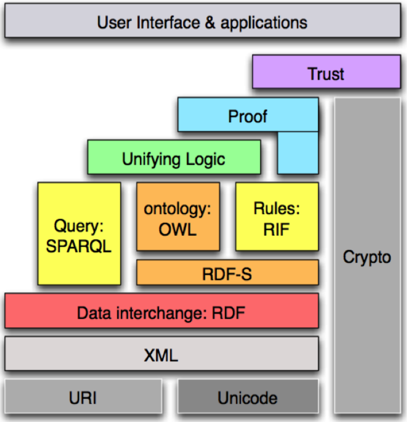
Dresden, 11 Apr 2014

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

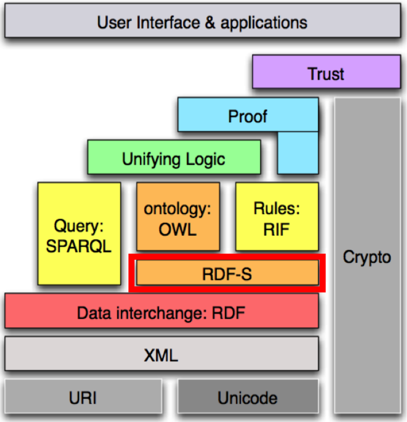
TECHNISCHE UNIVERSITÄT DRESDEN

Content

RDF Schema


User Interface & applications
Trust
Proof
Unifying Logic
Query: SPARQL
ontology: OWL
Rules: RIF
RDF-S
Crypto
Data interchange: RDF
XML
URI
Unicode

RDF Schema


User Interface & applications
Trust
Proof
Unifying Logic
Query: SPARQL
ontology: OWL
Rules: RIF
RDF-S
Crypto
Data interchange: RDF
XML
URI
Unicode

Agenda

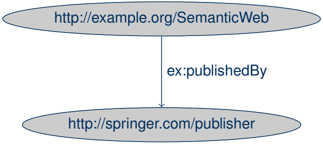
- Motivation
- Classes and Class Hierarchies
- Properties and Property Hierarchies
- Property Restrictions
- Open Lists
- Reification
- Additional Information in RDFS
- Simple Ontologies

Agenda

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

Schema Knowledge with RDFS

- RDF provides universal possibility to encode factual data on the Web



- = proposition about single resources (individuals) and their relationships
- desirable: propositions about generic sets of individuals (classes), e.g. publishers, organizations, persons etc.

Schema Knowledge with RDFS

- also desirable: specification of logical interdependencies between individuals, classes and relationships, in order to capture as much of the semantics of the described domain as possible, e.g.:
  "Publishers are Organizations."
  "Only persons write books."
- in database speak: schema knowledge

Schema Knowledge with RDFS

RDF Schema (RDFS):

- part of the W3C Recommendation of RDF
- allows for specifying schematic (also: terminological) Knowledge
- use of 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#`

Schema Knowledge with RDFS

RDF Schema (RDFS):

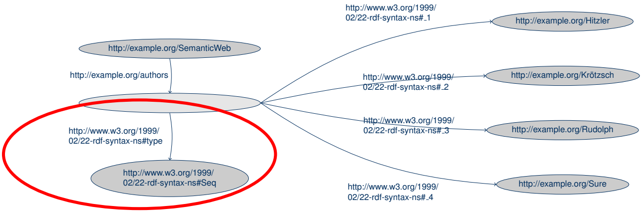
- yet: vocabulary not domain-specific (like, e.g., with FOAF), but generic
- allows for specifying (parts of) the semantics of arbitrary RDF vocabularies (could thus be called a “meta vocabulary”)
- advantage: every RDFS-compliant software faithfully supports every vocabulary that has been defined through RDFS
- this functionality makes RDFS an ontology language for lightweight ontologies
- “A little semantics goes a long way.”

Agenda

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

Classes and Instances

- We have already seen “typing” of resources in RDF when we discussed lists:



- the predicate `rdf:type` endows the subject with the type denoted by the object
- the object is seen as the identifier of a class, of which the resource denoted by the subject is a member (also called an instance of that class)

Classes and Instances

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

- characterizes “Semantic Web - Grundlagen” as instance of the (newly defined) class “Textbook”
- class membership is not exclusive, e.g. together with the above triple we may have:
  `ex:SemanticWeb rdf:type ex:Entertaining .`
- in general: a priori individual and class names cannot be distinguished syntactically
- also in reality, this distinction is sometimes difficult: e.g. for
  `http://www.un.org/#URI`

The Class of all Classes

- however, sometimes one wants to state that a URI denotes a class
- can be done by “typing” that URI as `rdfs:Class`

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

- `rdfs:Class` is the “class of all classes” and therefore also contains itself, thus the following triple is always valid:

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

Subclasses – Motivation

- given the triple
  `ex:SemanticWeb rdf:type ex:Textbook .`
- we do not get a result when searching for instances of the class ex:Book
- option: add the triple
  `ex:SemanticWeb rdf:type ex:Book .`
- this just solves the problem only for the specific resource
  `ex:SemanticWeb`
- automatically adding it for all instances would blow up the RDF document

Subclasses

- better: one statement telling that every textbook is also a book, i.e., every instance of `ex:Textbook` is automatically also an instance of `ex:Book`
- realized via the `rdfs:subClassOf` property:

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

"The class of all textbooks is a subclass of the class of all books."

Subclasses

- the `rdfs:subClassOf` property is reflexive, i.e., every class is its own subclass, thus:

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

- on the contrary, we can enforce that two URIs refer to the same class by declaring them as mutual subclasses, like:

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

Class Hierarchies

- common: not just singular subclass relationships but whole class hierarchies (aka: taxonomies) e.g.:

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

- “built in” in RDFS semantics: transitivity of the `rdfs:subClassOf` property, i.e., it follows

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

Class Hierarchies

- class hierarchies particularly often used for modeling, e.g. in biology (e.g. 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

- intuitive conection to set theory:

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

- this also justifies the reflexivity and transitivity of `rdfs:subClassOf`

Classes in RDF/XML Syntax

- abbreviated notation for specifying class instances:

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

  instead of

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

- `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 (discussed later)

Agenda

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

Properties

- also called: relations, relationships
- beware: unlike in OOP, properties in RDF(S) are not assigned to classes
- property URIs normally in predicate position of a triple
- properties characterize, in which way two resources are related to each other
- mathematically often represented as set of pairs:
  marriedWith = $\{(\text{Adam}, \text{Eve}), (\text{Brad}, \text{Angelina}), \ldots\}$
- URI can be marked as property name by typing it accordingly:
  `ex:publishedBy rdf:type rdf:Property .`

Subproperties

- like sub-/superclasses also sub-/superproperties possible and useful
- specification in RDFS via `rdfs:subPropertyOf` e.g.:

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

- Then, given the triple

```
ex:markus ex:happilyMarriedWith ex:anja .
```

  we can infer

```
ex:markus ex:marriedWith ex:anja .
```

Agenda

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

Property Restrictions

- common: usage of property only makes sense for certain kinds of resources, e.g. `ex:publishedBy` only connects publications with publishers
- thus, 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 .
  ```
- Can also be used to “prescribe” datatypes for literals:
  ```
  ex:hasAge rdfs:range xsd:nonNegativeInteger .
  ```

Property restrictions

- property restrictions are the only way of specifying semantic interdependencies between properties and classes
- beware: property restrictions are interpreted globally and conjunctively: z.B.

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

  means: every entity having an author is both a cookbook and a storybook
- thus: always pick the most general possible class for domain/range specifications

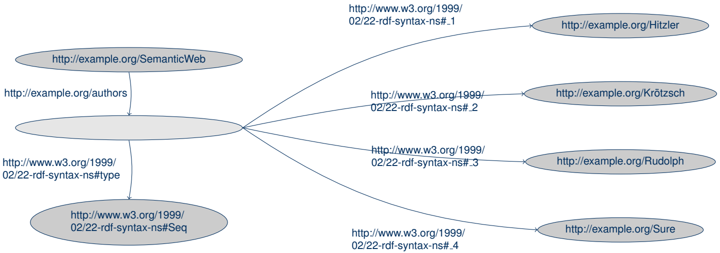
TECHNISCHE UNIVERSITÄT DRESDEN

Agenda

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

Working with open lists

Zur Erinnerung: offene Listen in RDF:



Working with Open Lists

- 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.

Working with Open Lists

- new property: `rdfs:member`
  superproperty of all properties that are instances of `rdfs:ContainerMembershipProperty`, could be called the "universal containedness relation"
- Hard-wired in 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 .
  ```

TECHNISCHE UNIVERSITÄT DRESDEN

Agenda

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

Reification

- problematic in RDF(S): model propositions about proposition (in natural language, such propositions can be identified by a leading “that”), e.g.: “The detective suspects that the butler killed the gardener.”

Reification

- problematic in RDF(S): model propositions about proposition (in natural language, such propositions can be identified by a leading “that”), e.g.: “The detective suspects that the butler killed the gardener.”
- first modeling attempt:

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

  – Suboptimal: the literal object cannot be easily referenced in other triples.

Reification

- problematic in RDF(S): model propositions about proposition (in natural language, such propositions can be identified by a leading "that"), e.g.: "The detective suspects that the butler killed the gardener."
- first modeling attempt:

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

  - Suboptimal: the literal object cannot be easily referenced in other triples.

- second modeling attempt:

```
ex:detektiv ex:suspects ex:theButlerKilledTheGardener
.
```

  - Suboptimal: we lose the inner structure of the talked about proposition

Reification

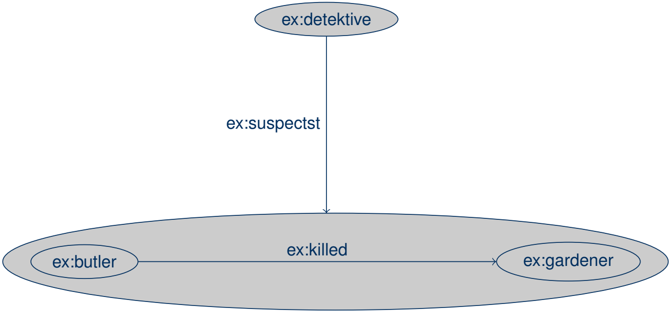
- problematic in RDF(S): model propositions about proposition (in natural language, such propositions can be identified by a leading “that”), e.g.: “The detective suspects that the butler killed the gardener.”
- Out of context, proposition can be easily modeled in RDF:

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

- desirable: this whole triple should occur as an object of another triple, however, this is not valid RDF

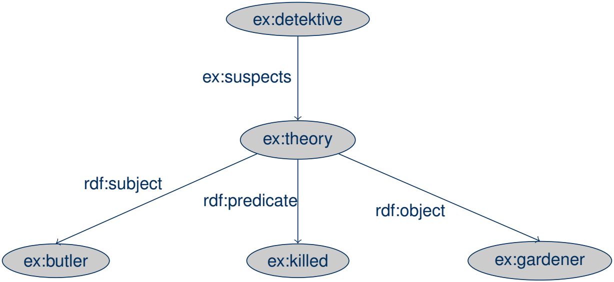
Reification

solution (similar to multi-valued relationships): introduce auxiliary nodes representing the nested proposition:



Reification

solution (similar to multi-valued relationships): introduce auxiliary nodes representing the nested proposition:

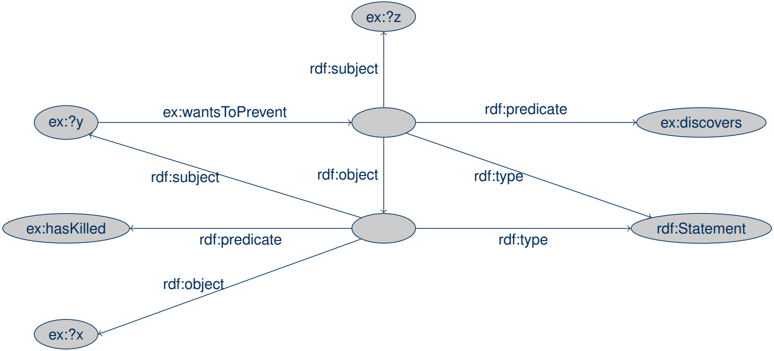


Reification

- caution: reified triple does not need to hold (would not be always sensible either, cf. propositions like: “The detective has doubts that the butler killed the gardener.”)
- if this is wanted, the original (un-reified) triple has to be added to the RDF document
- the class `rdf:Statement` is used to mark nodes which represent reified propositions
- in case this proposition is not referred to from the “outside”, the auxiliary node may be a bnode

Reification

A small reification riddle: another criminal story...



TECHNISCHE UNIVERSITÄT DRESDEN

Agenda

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

Additional Information

- like with programming languages, one sometimes wants to add comments (without changing the semantics)
- purpose: increase understandability for human users
- it is to be preferred to model this knowledge in a graph-based way (e.g., due to compatibility reasons)
- thus: defined set of properties that serve this purpose

Additional Information

`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 feat. language information):

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

Additional Information

`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

Example of usage

```
:
xmlns:wikipedia="http://en.wikipedia.org/wiki" : <rdfs:Class
rdf:about="\&ex;Primates">
  <rdfs:label xml:lang="de">Primaten</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>
```

TECHNISCHE UNIVERSITÄT DRESDEN

Agenda

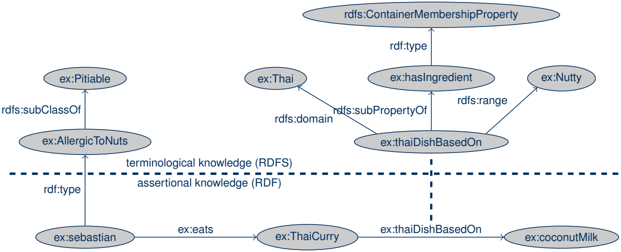
- Motivation
- Classes and Class Hierarchies
- Properties and Property Hierarchies
- Property Restrictions
- Open Lists
- Reification
- Additional Information in RDFS
- Simple Ontologies

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.

Simple Ontologies - Example

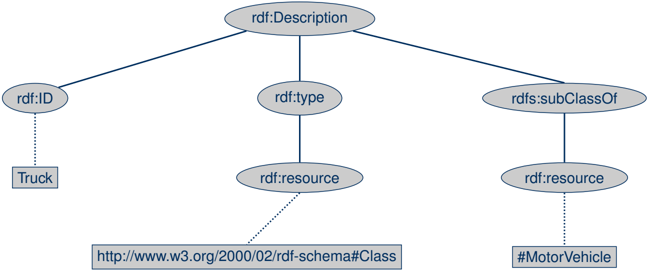
```
ex:vegetableThaiCurry  ex:thaiDishBasedOn    ex:coconutMilk .
ex:sebastian           rdf:type              ex:AllergicToNuts .
ex:sebastian           ex:eats               ex:vegetableThaiCurry .
ex:AllergicToNuts      rdfs:subClassOf       ex:Pitiable .
ex:thaiDishBasedOn     rdfs:domain       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



1 Document - 3 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 as XML:



TECHNISCHE UNIVERSITÄT DRESDEN

1 Document - 3 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 as RDF:

- another data model
- `rdf:Description`, `rdf:ID` and `rdf:resource` have a fixed meaning

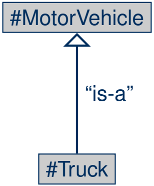
subject	predicate	object
`#Truck`	`rdf:type`	`rdfs:Class`
`#Truck`	`rdfs:subClassOf`	`#Motorvehicle`

1 Document - 3 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 as RDF Schema:

- yet another data model
- `rdf:type` and `rdf:subClassOf` have a specific interpretation



Agenda

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