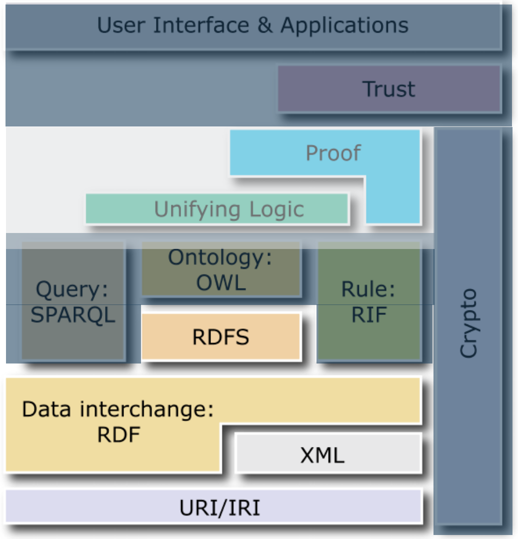
RDFS Semantics

where the *Web* starts to be *Semantic Web*



What can we conclude from a graph?

```
ex:jja ex:teaches ex:sw.

ex:jja rdf:type uni:Professor.

uni:Professor rdfs:subClassOf uni:AcademicStaff.
uni:Associate rdfs:subClassOf uni:AcademicStaff.
uni:Assistant rdfs:subClassOf uni:AcademicStaff
uni:AcademicStaff rdfs:subClassOf uni:Member.
uni:Student rdfs:subClassOf uni:Member.
uni:administrative rdfs:subClassOf uni:Member.
```

```
ex:teaches rdf:domain uni:AcademicStaff;
           rdf:range  uni:Course;
           rdf:subPropertyOf ex:isInvolvedIn.
ex:isInvolvedIn rdf:domain uni:Member;
           rdf:range  uni:Course;
ex:name rdf:range xsd:string.

uni:Professor rdfs:subClassOf ptuni:profCat.
ptuni:profCat rdfs:subClassOf uni:Professor.
```

- What else should we conclude from this?

`ex:jja rdf:type ptuni:profCat, uni:AcademicStaff, uni:Member.`

`ex:sw rdf:type uni:Course.` Why?

`_:x ex:teaches ex:sw.` There exists someone who teaches sw.

`ex:jja ex:teaches _:y.` jja teaches something.

`ex:jja ex:isInvolvedIn ex:sw.` Why?

`_:x rdf:type uni:Member.`

…

RDF(S) semantics

- Translate RDF(S) statements into logical sentences
  - vocabulary: URIs, bnodes, and Literals
  - triples give rise to sentences
    - (s, p, o) ∈ (URIs ∪ bnodes) × URIs × (URIs ∪ bnodes ∪ Lits)
- Model-theoretic semantics
  - Define interpretations, and set when an interpretation is a model of a graph
- Consequence relation
  - G ⊨ G' whenever all models of G are also models of G'
- Inference ⊢
  - Sound and complete procedure w.r.t entailment

Basic notions

- An **RDF graph** is a set of triples.
- A **subgraph of an RDF** graph is subset of the triples of the graph.
- A **ground RDF graph** is a graph without blank nodes
- A **name** is a URI Reference or a literal.
- A set of names is a **vocabulary**.
  - An interpretation assigns meaning to names by mapping them into a set plus some constraints upon the set and the mapping.

Instances of Graphs

- Let M be a mapping from blank nodes into literals, blank nodes or IRI references.
  - An *instance* of a graph G is obtained by substituting some or all blank nodes in G by the result of applying M to those blank nodes.
  - An *instance* with respect to a vocabulary V is an instance where all the names in the instance that were substituted for blank nodes in the original are names of V.
  - A *proper instance* of a graph is an instance in which a blank node has been replaced by a name, or two blank nodes in the graph have been mapped into the same node in the instance.
- An RDF graph is lean if it has no instance which is a proper subgraph of itself. Ground graphs are lean.

Simple interpretations (1.1)

- A **simple interpretation** I is defined by:
  - A non-empty set of resources I_R, the domain of I
  - A set I_P, designated the set of properties of I
  - A function I_{EXT} mapping I_P into the power set of $I_R \times I_R$, i.e. the set of sets of pairs $<x,y>$ with x and y in I_R.
  - A function I_S mapping IRIs into $(I_R \cup I_P)$
  - A partial mapping I_L from literals into I_R

- Note that all interpretations are infinite

Models of ground graphs

- If E is a literal then $I(E) = I_L(E)$
- If E is an IRI reference then $I(E) = I_S(E)$
- I is a simple model of a ground triple (s,p,o), denoted by $I \models (s,p,o)$, iff
  - $\{s,p,o\} \subseteq V$
  - $I(p) \in I_P$
  - $<I(s),I(o)> \in I_{EXT}(I(p))$
- I is a simple model of a ground RDF graph G,
  - $I \models G$ iff $I \models t$ for every triple $t \in G$

If IL(E) is undefined for some literal E then E has no semantic value, so any triple containing it will be false, so any graph containing that triple will also be false.

Models of graphs with bnodes

- Let *sk* be a partial mapping from a set of bnodes into the universe I_R of I
- Let I+*sk* be an extended interpretation identical to I except that *sk* is used for interpreting bnodes
  - If E is a bnode and $sk(E)$ is defined then $[I+sk](E) = sk(E)$
- I is a simple model of an RDF graph G, $I \models G$ iff
  - there exists a mapping *sk* such that $[I+sk] \models G$
- This definition assigns an existential semantics to blank nodes.

Exercise

- Construct a model for graph

  <ex:cd> <ex:teaches> <ex:md>
  <ex:cd> <ex:knows> _:xxx
  <ex:jja> <ex:knows> <ex:cd>

Simple Entailment

- An RDF graph G_1 simply entails another RDF graph (or triple) G_2, denoted as $G_1 \vDash G_2$, iff
  - For every simple interpretation I, if $I \vDash G_1$ then $I \vDash G_2$
- An inference procedure $\vdash$ constructing a graph G_1 from G_2 is
  - **sound** if $G_1 \vDash G_2$
  - **complete** if whenever $G_1 \vDash G_2$, then $\vdash$ is able to construct G_2 from G_1
- Inference procedures take a graph, and keep adding triples, according to some inference rules
  - Any obtained subgraph is inferred

Basic theoretic results

- **Empty Graph Lemma**:
  The empty set of triples is simply entailed by any graph, and it does not simply entail any graph except itself
- **Subgraph Lemma**:
  A graph simply entails all its subgraphs
- **Instance Lemma**:
  A graph is simply entailed by any of its instances
- **Merging Lemma**:
  The merge of set S of RDF graphs is simply entailed by S, and simply entails any member of S

More results

- **Monotonicity Lemma:**
  Let S be a subgraph of S' such that $S \models E$. Then $S' \models E$
- **Compactness Lemma:**
  If $S \models G$ and G is a finite graph, then $S' \models G$ for some finite $S' \subseteq S$
- **Skolemisation Lemma:**
  If *sk(E)* is a skolemization of E with respect to V, then $sk(E) \models E$.
  If $sk(E) \models F$ and the vocabulary of F is disjoint from the skolem vocabulary in V, then $E \models F$

Interpolation Lemma

$S \models E$ **iff** there is a subgraph of S which is an instance of E

- The interpolation lemma completely characterizes in syntactical terms simple entailment in RDF graps.

- Simple entailment is decidable but NP-complete.

FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA

Inference rules for simple entailment

Rule Name	If S contains	then add
se1	uuu aaa xxx .	uuu aaa _:nnn . where _:nnn designates a blank node allocated to xxx by rules se1 or se2.
se2	uuu aaa xxx .	_:nnn aaa xxx . where _:nnn designates a blank node allocated to uuu by rules se1 or se2.

uuu – blank node or URI

xxx – blank node, URI reference or literal

By using these rules the entailment problem is reduced to the existence of a subgraph

Exercise

- Verify whether the following subgraph S

  <ex:cd> <ex:teaches> <ex:md>
  <ex:cd> <ex:knows> _:xxx
  <ex:jja> <ex:knows> <ex:cd>

- Simply entails the RDF graph E
  _:yyy <ex:teaches> _:zzz
  _:yyy <ex:knows> _:www

- And, what if we add <ex:cd> <ex:knows> <ex:cd> to E ?

The RDF vocabulary

- The RDF vocabulary is the set of URI references in the rdf namespace, denoted by *rdfV* and is formed by:
  - `rdf:type` and `rdf:Property`
  - `rdf:XMLLiteral`
  - `rdf:List`, `rdf:first`, `rdf:rest`, and `rdf:nil`
  - `rdf:Seq`, `rdf:Bag`, `rdf:Alt`, `rdf:_1`, `rdf:_2` ...
  - `rdf:Statement`, `rdf:subject`, `rdf:predicate`, and `rdf:object`
  - `rdf:value`
- An rdf-interpretation imposes additional constraints in simple interpretations for specifying the meaning of `rdf:Property` as well its declaration via `rdf:type`.

Rationale of RDF entailment

- Any predicate which occurs in a triple must have type `rdf:Property`
- Supports the datatypes `rdf:langString` and `xsd:string`
- Any ill-typed literal results in an unsatisfiable graph

RDF 1.0 mandatorily supported rdf:XMLLiteral and no other datatype. In this presentation we adopt the more recent semantics of RDF 1.1

RDF-interpretations

- An RDF-interpretation recognizing D is a D-interpretation where D includes rdf:langString and xsd:string :
  - $x \in I_P$ iff $\langle x, I(rdf\text{:}Property)\rangle \in I_{EXT}(I(rdf\text{:}type))$
  - *For every IRI aaa* $\in$ *D, then* $\langle x, I(aaa)\rangle \in I_{EXT}(I(rdf\text{:}type))$ *iff x is in the value space of I(aaa)*
  - *RDF axiomatic triples are satisfied (see next slide)*

Note:

- RDF imposes no particular normative meanings on the rest of the RDF vocabulary.
- The datatype IRIs rdf:langString and xsd:string must be recognized by all RDF interpretations.
- Two other datatypes rdf:XMLLiteral and rdf:HTML are defined but RDF-D interpretations may fail to recognize these datatypes.

RDF axiomatic triples

- Moreover, the following triples must be satisfied by any RDF-interpretation, specifying the properties of the *rdfV* vocabulary:

```
rdf:type      rdf:type rdf:Property .
rdf:subject   rdf:type rdf:Property .
rdf:predicate rdf:type rdf:Property .
rdf:object    rdf:type rdf:Property .
rdf:first     rdf:type rdf:Property .
rdf:rest      rdf:type rdf:Property .
rdf:value     rdf:type rdf:Property .
rdf:_1        rdf:type rdf:Property .
rdf:_2        rdf:type rdf:Property .
...
rdf:nil       rdf:type rdf:Property .
```

RDF models and entailment

- Just like simple models and entailment, but with RDF-interpretation and a set of datatypes D. Formally,
  - S RDF entails E recognizing D when every RDF interpretation recognizing D which satisfies S also satisfies E.
  - When D is {rdf:langString, xsd:string} then we simply say that S RDF entails E.
  - E is RDF unsatisfiable (recognizing D) when it has no satisfying RDF interpretation
- To simplify discussion, we do not enter into details here about datatype support in the semantics.

Note: The properties of simple entailment described earlier do not all apply to RDF entailment. For example, all the RDF axioms are true in every RDF interpretation, and so are RDF entailed by the empty graph, contradicting interpolation for RDF entailment

Inference rules for rdf-entailment

Name	If S contains	then add
GrdfD1	xxx aaa "sss"^^ddd . for ddd in D	"sss"^^ddd rdf:type ddd .
rdfD2	uuu aaa yyy .	aaa rdf:type rdf:Property .

REMARK:

This requires generalized graphs since typed literals will be introduced as subjects of triples. In RDF 1.0 it was defined a different set of rules which did not require the use of generalized graphs, but that would lead to incompleteness for the case of RDFS-entailment (see next)

Generalized RDF closure of S towards E

1. Add to S all the RDF axiomatic triples which do not contain any container membership property IRI.
2. For each container membership property IRI which occurs in E, add the RDF axiomatic triples which contain that IRI.
3. If no triples were added in step 2., add the RDF axiomatic triples which contain rdf:_1.
4. Apply the rules GrdfD1 and rdfD2 with D={rdf:langString, xsd:string}, to the set in all possible ways, to exhaustion.

If S is RDF consistent, then S RDF entails E just when the generalized RDF closure of S towards E simply entails E.

FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA

Semantics of RDFS

- Can be defined similarly to that of RDF, as just seen
  - It is quite complex (see it at w3.org)

- It is easier to do it by translating models primitives into predicate logic with equality
  - This readily provides a precise meaning resorting to well understood 1st order logic

Translation into first-order logic

- The semantic conditions imposed on RDFS interpretations can be better understood via a natural translation into 1st order logic.

- An assertion s rdf:type o can be translated into the atom o(s).

  uni:Staff#n765 rdf:type uni:AssistantProfessor .
  ↕
  uni:AssistantProfessor(uni:Staff#n765)

- Any other triple s p o . is represented by atom p(s,o).

  uni:Staff#n765 uni:lecturer uni:Course#123
  ↕
  uni:lecturer(uni:Staff#n765 , uni:Course#123)

- Literals should also be translated (see LBase).

Fundamentals of rdfs-interpretations

- If property P has domain D, from s P o . it can be concluded that s has type D.

  P(s,o) → D(s)

- If property P has range C, from s P o . it can be concluded that o has type C.

  P(s,o) → C(o)

- Predicate rdfs:subPropertyOf is reflexive and transitve

  rdf:Property(P) → rdfs:subPropertyOf(P,P)
  rdfs:subPropertyOf(P,Q) /\ rdfs:subPropertyOf(Q,R) → rdfs:subPropertyOf(P,R)

- If P is a sub-property of Q then from s P o . it can be concluded the triple s Q o.

  P(s,o) → Q(s,o)

Fundamentals of rdfs-interpretations

- The subject and object of any triple have type rdfs:Resource
- If C is a class then C is a subclass of rdfs:Resource

  rdfs:Class(C) → rdfs:subClassOf(C,rdfs:Resource)

- Predicate rdfs:subClassOf is transitive and reflexive

  rdf:Class(C) → rdfs:subClassOf(C,C)
  rdfs:subClassOf(C,D) /\ rdfs:subClassOf(D,E) → rdfs:subClassOf(C,E)

- If C is a subclass of D then any entity of type C is of type D:

  C(s) → D(s)

Inference systems for RDFS

- With the first order logic semantics, a first order logic proof system can be used for inference in RDF and RDFS
- However this is, in general, quite heavy!
- Instead, one can defined a specialised inference system, acting directly at RDF and RDFS triples
  - With axiomatic triples, and
  - Inference rules

RDFS axiomatic triples

- Domain of properties

```
rdf:type              rdfs:domain rdfs:Resource .
rdfs:domain           rdfs:domain rdf:Property .
rdfs:range            rdfs:domain rdf:Property .
rdfs:subPropertyOf    rdfs:domain rdf:Property .
rdfs:subClassOf       rdfs:domain rdfs:Class .
rdf:subject           rdfs:domain rdf:Statement .
rdf:predicate         rdfs:domain rdf:Statement .
rdf:object            rdfs:domain rdf:Statement .
rdfs:member           rdfs:domain rdfs:Resource .
rdf:first             rdfs:domain rdf:List .
rdf:rest              rdfs:domain rdf:List .
rdfs:seeAlso          rdfs:domain rdfs:Resource .
rdfs:isDefinedBy      rdfs:domain rdfs:Resource .
rdfs:comment          rdfs:domain rdfs:Resource .
rdfs:label            rdfs:domain rdfs:Resource .
rdf:value             rdfs:domain rdfs:Resource .
```

RDFS axiomatic triples

- Range of properties

```
rdf:type              rdfs:range rdfs:Class .
rdfs:domain           rdfs:range rdfs:Class .
rdfs:range            rdfs:range rdfs:Class .
rdfs:subPropertyOf    rdfs:range rdf:Property .
rdfs:subClassOf       rdfs:range rdfs:Class .
rdf:subject           rdfs:range rdfs:Resource .
rdf:predicate         rdfs:range rdfs:Resource .
rdf:object            rdfs:range rdfs:Resource .
rdfs:member           rdfs:range rdfs:Resource .
rdf:first             rdfs:range rdfs:Resource .
rdf:rest              rdfs:range rdf:List .
rdfs:seeAlso          rdfs:range rdfs:Resource .
rdfs:isDefinedBy      rdfs:range rdfs:Resource .
rdfs:comment          rdfs:range rdfs:Literal .
rdfs:label            rdfs:range rdfs:Literal .
rdf:value             rdfs:range rdfs:Resource .
```

RDFS axiomatic triples

- Subclass and subproperty relations

```
rdf:Alt rdfs:subClassOf rdfs:Container .
rdf:Bag rdfs:subClassOf rdfs:Container .
rdf:Seq rdfs:subClassOf rdfs:Container .
rdfs:ContainerMembershipProperty rdfs:subClassOf rdf:Property.
rdfs:isDefinedBy rdfs:subPropertyOf rdfs:seeAlso .
```

- Datatypes

```
rdf:langString rdf:type rdfs:Datatype .
xsd:string rdf:type rdfs:Datatype .
rdf:langString rdfs:subClassOf rdfs:Literal .
xsd:string rdfs:subClassOf rdfs:Literal .
rdfs:Datatype  rdfs:subClassOf rdfs:Class .
```

- Containers

```
rdf:_1 rdf:type rdfs:ContainerMembershipProperty .
rdf:_1 rdfs:domain rdfs:Resource .
rdf:_1 rdfs:range rdfs:Resource .

…
```

Rules for rdfs-entailment

Name	If S contains:	then add
rdfs1	Any IRI aaa in D	aaa rdf:type rdfs:Datatype
rdfs2	aaa rdfs:domain xxx . uuu aaa yyy .	uuu rdf:type xxx .
rdfs3	aaa rdfs:range xxx . uuu aaa vvv .	vvv rdf:type xxx .
rdfs4a	uuu aaa xxx .	uuu rdf:type rdfs:Resource .
rdfs4b	uuu aaa vvv.	vvv rdf:type rdfs:Resource .

aaa, bbb, ... – URI

uuu, vvv, ... – blank node or URI

xxx, yyy, ... – blank node, URI reference or literal

Rules for rdfs-entailment

Name	If S contains:	then add
rdfs5	uuu rdfs:subPropertyOf vvv . vvv rdfs:subPropertyOf xxx .	uuu rdfs:subPropertyOf xxx .
rdfs6	uuu rdf:type rdf:Property .	uuu rdfs:subPropertyOf uuu .
rdfs7	aaa rdfs:subPropertyOf bbb . uuu aaa yyy .	uuu bbb yyy .
rdfs8	uuu rdf:type rdfs:Class .	uuu rdfs:subClassOf rdfs:Resource .
rdfs9	uuu rdfs:subClassOf xxx . vvv rdf:type uuu .	vvv rdf:type xxx .
rdfs10	uuu rdf:type rdfs:Class .	uuu rdfs:subClassOf uuu .
rdfs11	uuu rdfs:subClassOf vvv . vvv rdfs:subClassOf xxx .	uuu rdfs:subClassOf xxx .
rdfs12	uuu rdf:type rdfs:ContainerMembershipProperty .	uuu rdfs:subPropertyOf rdfs:member .
rdfs13	uuu rdf:type rdfs:Datatype .	uuu rdfs:subClassOf rdfs:Literal

FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA

rdfs-entailment (RDF 1.0)

Name	If S contains	then add
lg	uuu aaa lll .	uuu aaa _:nnn . where _:nnn identifies a blank node allocated to literal lll by this rule
gl	uuu aaa _:nnn . where _:nnn identifies a blank node allocated to literal lll by rule lg	uuu aaa lll .

RDFS-entailment lemma:

S rdfs-entails E if and only if there is a graph which can be derived from S plus the RDF and RDFS axiomatic triples by the application of rule lg, rule gl and the RDF and RDFS entailment rules and which either simply entails E or contains an XML clash.

WRONG !!!

FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA

Counterexample

- Consider the RDF graph

```
<http://p> rdfs:subPropertyOf <http://q> .
<http://q> rdfs:domain <http://u> .
<http://v> http://p "Example" .
```

- Can we conclude `<http://v> rdf:type <http://u> .` ? How ?
- Now consider the RDF graph

```
<http://p> rdfs:subPropertyOf _:q .
_:q rdfs:domain <http://u> .
<http://v> http://p "Example" .
```

- Can we conclude the same triple ?

Solution

- We have to consider generalized RDF graphs [Horst] where predicates can be blank nodes!
- The only change necessary to obtain a complete set of rules is to substitute rule rdfs7 by rdfs7x:

rdfs7x	aaa rdfs:subPropertyOf vvv . uuu aaa yyy .	uuu vvv yyy .

- Now the rule can be applied even to blank nodes in the object of rfs:subPropertyOf.

Generalized RDFS closure of S towards E

1. Add to S all the RDF and RDFS axiomatic triples which do not contain any container membership property IRI.
2. For each container membership property IRI which occurs in E, add the RDF and RDFS axiomatic triples which contain that IRI.
3. If no triples were added in step 2., add the RDF and RDFS axiomatic triples which contain rdf:_1.
4. Apply the rules GrdfD1, rdfD2, and the rules rdfs1 through rdfs13 with D={rdf:langString, xsd:string}, to the set in all possible ways, to exhaustion.

If S is RDFS consistent, then S RDFS entails E just when the generalized RDFS closure of S towards E simply entails E.

Minimal deductive systems

- Identifies a fragment of RDFS that covers the crucial vocabulary and preserves the original RDFS semantics.
- Efficient algorithms to check entailment
- The vocabulary ρdf contains only rdf:type, rdfs:domain, rdfs:range, rdfs:subClassOf, and rdfs:subPropertyOf.

Minimal Deductive Systems (blank node rule)

$$\frac{G}{H} \text{ if there is a homomorphism } \mu : H \to G$$

(note this is the interpolation lemma)

Minimal Deductive Systems (Core Rules)

- Subproperty (transitivity, definition)

$$\frac{(A,\mathtt{sp},B),(B,\mathtt{sp},C)}{(A,\mathtt{sp},C)} \qquad \frac{(A,\mathtt{sp},B),(X,A,Y)}{(X,B,Y)}$$

- Subclass (transitivity, definition)

$$\frac{(A,\mathtt{sc},B),(B,\mathtt{sc},C)}{(A,\mathtt{sc},C)} \qquad \frac{(A,\mathtt{sc},B),(X,\mathtt{type},A)}{(X,\mathtt{type},B)}$$

- Typing (domain, range)

$$\frac{(A,\mathtt{dom},B),(X,A,Y)}{(X,\mathtt{type},B)} \qquad \frac{(A,\mathtt{range},B),(X,A,Y)}{(Y,\mathtt{type},B)}$$

- Implicit Typing

$$\frac{(A,\mathtt{dom},B),(C,\mathtt{sp},A),(X,C,Y)}{(X,\mathtt{type},B)} \qquad \frac{(A,\mathtt{range},B),(C,\mathtt{sp},A),(X,C,Y)}{(Y,\mathtt{type},B)}$$

Minimal Deductive Systems (Reflexivity)

- Subproperty reflexivity

$$\frac{(X,A,Y)}{(A,\mathtt{sp},A)} \qquad \frac{}{(p,\mathtt{sp},p)} \text{ for } p \in \rho df$$

$$\frac{(A,\mathtt{sp},B)}{(A,\mathtt{sp},A),(B,\mathtt{sp},B)} \qquad \frac{(A,\mathtt{dom},X)}{(A,\mathtt{sp},A)} \quad \frac{(A,\mathtt{range},X)}{(A,\mathtt{sp},A)}$$

- Subclass reflexivity

$$\frac{(A,\mathtt{sc},B)}{(A,\mathtt{sc},A),(B,\mathtt{sc},B)} \qquad \frac{(X,\mathtt{range},A)}{(A,\mathtt{sc},A)}$$

$$\frac{(X,\mathtt{dom},A)}{(A,\mathtt{sc},A)} \qquad \frac{(X,\mathtt{type},A)}{(A,\mathtt{sc},A)}$$

Complexity of reasoning with minimal RDF(S)

- The size of closure of G is $|G|^2$
- Let H be a ground graph. Deciding if $G \models_{\rho df} H$ can be done in time $O(|H| \,.\, |G| \log |G|)$

Summary of RDF and RDFS

- RDF provides a schema-less data model, based on graphs, adequate for highly distributed and collaborative datasets.
- RDFS allows for the definition of terminology associated to the data, and provide for schema knowledge about the data
  - written in RDF itself, amalgamating data and meta-data
  - with a precise semantics and inference mechanism that provides a common understanding of data
- But as a schema language, RDFS is quite limited…

RDFS limitation

- Several features common in schema languages are not provided by RDFS, viz.:
  - cardinality restrictions on properties (e.g. a course has only one responsible professor)
  - keys on classes (e.g. student-number is a key for students)
  - negation and disjointness of classes (e.g. students and professors are disjoint)
  - restrictions in the range or cardinality of a property, depending of the type of class where it is used
  - combination of classes with set operations such as intersection, or union
  - …

Beyond RDFS

- We'll see how to define richer modelling languages later on in the course
- Before that, we will explore what can be done with just RDF and RDFS
    - Namely, we will study query languages