COMP318: RDFS entailment

`www.csc.liv.ac.uk/~valli/Comp318`

Dr Valentina Tamma
Room: Ashton 2.12
Dept of computer science
University of Liverpool
`V.Tamma@liverpool.ac.uk`

based on material by M. Rodriguez Muro, P. Hitzler and S. Rudolph

Where were we

- RDF entailment rules
- RDFS entailment rules

Reasoning engines

- Systems that perform inference are often called reasoning engines or reasoners.
  - ***Reasoner engine***: a system that infers new information based on the contents of a knowledgebase. This can be accomplished using rules and a rule engine, triggers on a database or RDF store, decision trees, tableau algorithms, or even programmatically using hard-coded business logic
  - A reasoner must be compliant to the semantics of the ontology language it supports
  - Hence, an ontology language must state its semantics in a formal way

- The RDFS reasoner uses ***entailment rules*** that are supposed to capture the intended semantics

Soundness and Completeness

- Theorem. A graph G1 RDFS-entails a graph G2 if there is a graph G1’ which has been derived from G1 via the rules lg, gl, rdfax, rdf1, rdf2, rdfsax and rdfs1 … rdfs13 such that:
  - G1’ simply entails G2 or
  - G1’ contains an XML clash.

- The inference rules for RDFS-entailment we presented previously are sound but not complete (*ter Horst, 2005*).

Example

- The following graph:

```
ex:isHappilyMarriedTo rdfs:subPropertyOf _:bnode.
_:bnode rdfs:domain ex:Person.
ex:markus ex:isHappilyMarriedTo ex:anja .
```

- •The triple **`ex:markus rdf:type ex:Person .`** is a semantic consequence of the graph above, but this cannot be derived from the inference rules

Decidability and complexity

- RDFS entailment is decidable, even though one has to deal with the infinite number of axiomatic triples:
  - due to the fact that the RDF vocabulary for encoding lists includes property names `rdf:_i` for all i $\geq$ 1, with several RDFS axiomatic triples for each `rdf:_i`
- The problem of deciding whether a graph G1 RDFS-entails another graph G2 is NP-complete. The problem becomes polynomial if G2 contains no blank nodes

RDFS entailment in state of the art systems

- Existing RDF stores (Jena, Sesame, Virtuoso, Oracle, etc) offer implementations of RDFS entailment together with ways of querying the stored graphs through SPARQL
- Implementations may be based on applying the rules in a backward chaining or a forward chaining fashion

RDFS entailment cheatsheet

RDFS entailment patterns.

	If S contains:	then S RDFS entails recognizing D:
rdfs1	any IRI aaa in D	aaa `rdf:type rdfs:Datatype .`
rdfs2	aaa `rdfs:domain` xxx . yyy aaa zzz .	yyy `rdf:type` xxx .
rdfs3	aaa `rdfs:range` xxx . yyy aaa zzz .	zzz `rdf:type` xxx .
rdfs4a	xxx aaa yyy .	xxx `rdf:type rdfs:Resource .`
rdfs4b	xxx aaa yyy.	yyy `rdf:type rdfs:Resource .`
rdfs5	xxx `rdfs:subPropertyOf` yyy . yyy `rdfs:subPropertyOf` zzz .	xxx `rdfs:subPropertyOf` zzz .
rdfs6	xxx `rdf:type rdf:Property .`	xxx `rdfs:subPropertyOf` xxx .
rdfs7	aaa `rdfs:subPropertyOf` bbb . xxx aaa yyy .	xxx bbb yyy .
rdfs8	xxx `rdf:type rdfs:Class .`	xxx `rdfs:subClassOf rdfs:Resource .`
rdfs9	xxx `rdfs:subClassOf` yyy . zzz `rdf:type` xxx .	zzz `rdf:type` yyy .
rdfs10	xxx `rdf:type rdfs:Class .`	xxx `rdfs:subClassOf` xxx .
rdfs11	xxx `rdfs:subClassOf` yyy . yyy `rdfs:subClassOf` zzz .	xxx `rdfs:subClassOf` zzz .
rdfs12	xxx `rdf:type rdfs:ContainerMembershipProperty .`	xxx `rdfs:subPropertyOf rdfs:member .`
rdfs13	xxx `rdf:type rdfs:Datatype .`	xxx `rdfs:subClassOf rdfs:Literal .`

Example

Given the RDF graph S:

:e rdfs:subClassOf :d .

:c rdf:type owl:Class .

:d rdfs:domain _:y .

:a rdfs:comment “string” .

:g :d :f.

Is the following graph **RDFS-entailed** by S? Explain the answer

:d rdf:type rdfs:Resource .

Example

Given the RDF graph S:

:e rdfs:subClassOf :d .

:c rdf:type owl:Class .

:d rdfs:domain _:y .

:a rdfs:comment “string” .

:g :d :f.

Is the following graph **RDFS-entailed** by S? Explain the answer

:d rdf:type rdfs:Resource .

Yes, this triple is entailed because it can be inferred from the axiomatic triples (remember in RDFS everything is a resource).

Exercise

Given the RDF graph S:

```
rdfs:range rdfs:range rdfs:Class .

:s   rdfs:domain   :t .

:u   rdfs:subPropertyOf   :s .

:a   :s   :b .

:a   rdf:type   :u .

:u   rdfs:subClassOf   :y .

:t   rdfs:subClassOf   :s .

:t   rdfs:comment   ‘‘bla’’ .
```

Is the following graph RDFS-entailed by S? Explain the answer

:a rdf:type :t .

Exercise

```
rdfs:range rdfs:range rdfs:Class .

:s   rdfs:domain   :t .

:u   rdfs:subPropertyOf   :s .

:a   :s   :b .

:a   rdf:type   :u .

:u   rdfs:subClassOf   :y .

:t   rdfs:subClassOf   :s .

:t   rdfs:comment   ''bla'' .
```

Is the following graph RDFS-entailed by S? Explain the answer

```
:a  rdf:type :t  .
```

Yes, because the following triples hold

:a   :s   :b .

:s   rdfs:domain :t and because of entailment rule rdfs2 (cheat sheet)

RDFS entailment

- Given the graph G below,

```
⟨d:Poe, o:wrote, d:TheGoldBug .⟩
⟨d:TheGoldBug, rdf:type, o:Novel .⟩
⟨d:Baudelaire, o:translated, d:TheGoldBug .⟩

⟨d:Poe, o:wrote, d:TheRaven .⟩
⟨d:TheRaven,rdf:type,o:Poem .⟩
⟨d:Mallarme´, o:translated, d:TheRaven .⟩

⟨d:Mallarme´,o:wrote,_:b .⟩
⟨_:b, rdf:type, o:Poem .⟩
<o:Poem rdfs:subClassOf ex:Literature .>
<o:Novel rdfs:subClassOf ex:Literature .>
```

- And the following graph S, determine if G entails (using simple and RDFS entailment) S, and explain why.

```
S= <d:Poe wrote _:c .> <_:c rdf:type ex:Literature .>
```

RDFS entailment

1. ⟨d:Poe, o:wrote, d:TheGoldBug .⟩
2. ⟨d:TheGoldBug, rdf:type, o:Novel .⟩
3. ⟨d:Baudelaire, o:translated, d:TheGoldBug .⟩

4. ⟨d:Poe, o:wrote, d:TheRaven .⟩
5. ⟨d:TheRaven,rdf:type,o:Poem .⟩
6. ⟨d:Mallarme´, o:translated, d:TheRaven .⟩

7. ⟨d:Mallarme´,o:wrote,_:b .⟩
⟨_:b, rdf:type, o:Poem .⟩

8. <o:Poem rdfs:subClassOf ex:Literature .>

9. <o:Novel rdfs:subClassOf ex:Literature .>

S= <d:Poe wrote _:c .>
<_:c rdf:type ex:Literature .>

From RDFS 9 applied to 9 and 2

10. ⟨d:TheGoldBug, rdf:type, o:Literature .⟩

Apply SE1 to 1 and we obtain
<d:Poe o:wrote _:c>
with -:c -> d;TheGoldBug

Apply SE2 to 10, and we obtain
11.⟨_:c, rdf:type, o:Literature .⟩

So, the graph S is RDFS entailed

Brief recap of RDF/RDFS

- RDF/RDFS describe subject predicate object triples and basic subsumption relations.
  - Very efficient deduction/querying
    - SPARQL based on simple entailment, but...
  - Very poor expressivity...
    - Describe data in terms of triples (RDF)
    - Describe the model behind the data (RDFS) by:
      - Declare the “types” and properties/relationships of the things we want to make assertions about
      - Allowing to infer new assertions implicitly stated from a set of given facts
  - But sometimes we need to express more advanced notions, e. g.:
    - A person has only one birth date
    - No person can be male and female at the same time

What can you represent with RDFS

- ***RDFS provides:***
  - Classes
    - `Lecturer rdf:type rdfs:Class`
  - Class hierarchies
    - `Lecturer rdfs:subClassOf AcademicStaff`
  - Properties
    - `teachesModule rdf:type rdf:Property`
  - Property hierarchies
    - `teachesModule rdfs:subPropertyOf coordinatesModule`
  - Domain and range declarations
    - `teachesModule rdfs:domain Lecturer`
    - `teachesModule rdfs:range Module`
      - They infer information rather than checking data

What can't you represent in RDFS

- ***RDFS does NOT provide:***
  - Disjointness of Classes
    - `Male` *and* `Female` *are disjoint*
      - *they cannot have any shared instaces*
  - Property characteristics (inverse, transitive, ...)
    - `Lecturer teachesModule Module` *and*
      `Module isTaughtByLecturer Lecturer`
      are not explicitly related
  - Local scope of properties
    - `Lecturer` has a property `hasTitle` whose values (range) is restricted to all values of the class `PhD`
      - *only people who hold a PhD can be lecturers*
      - but other `AcademicStaffMembers` can hold a BSc
  - Complex concept definitions (Boolean combination of classes)
    - `Person = Man ∪ Woman`
    - `Mother = Woman ∩ Parent`

What can't you represent in RDFS

- ***RDFS does NOT provide:***
  - Cardinality restrictions
    - `Person` may have at most 1 name
    - `Lecturer` *teaches exactly two modules*
  - A way to distinguish between classes and instances:
    - `Feline rdf:type rdfs:Class`
    - `Cat rdf:type Feline`
    - `felix rdf:type Cat`
      - `:felix` is an instance of an instance (`Cat`)
  - A way to distinguish between language constructors and ontology vocabulary:
    - `rdf:type rdfs:range rdfs:Class`
    - `rdfs:Property rdfs:type rdfs:Class`
    - `rdf:type rdfs:subPropertyOf rdfs:subClassOf`
  - Reasoning for these non-standard semantics

What can you infer in RDFS

Schema

```
ex:Lecturer
rdfs:subClassOf
ex: AcademicStaff
```

Assertion

```
staffUniv:john_smith
rdf:type
ex: Lecturer
```

RDFS Entailment ⇒

Inferred Fact

```
staffUniv:john_smith
rdf:type
ex: AcademicStaff
```

What can't you infer in RDFS

- However, not all types of inferences are possible in RDFS

Schema

```
:wife_of rdfs:subPropertyOf married_to.
:married_to rdfs:domain :Spouse;
            rdfs:range  :Spouse.
:wife_of rdfs:domain :Wife;
         rdfs:range  :Husband.
```

Assertion

```
:juliet :wife_of :romeo.
```

Facts Inferred

```
:juliet rdf:type   :Wife;
        rdf:type   :Spouse;
        married_to :romeo;
:romeo rdf:type :Spouse;
       rdf:type :Husband.
```

What can't you infer in RDFS

- What about if we want to model symmetry, *i.e.* `:x :married_to :y` implies `:y :married_to :x`?

Schema

```
:wife_of rdfs:subPropertyOf married_to.
:married_to rdfs:domain :Spouse;
            rdfs:range  :Spouse.
:wife_of rdfs:domain :Wife;
         rdfs:range  :Husband.
:husband_of rdfs:domain :Husband;
            rdfs:range  :Wife.
```

Assertion

```
:juliet :wife_of :romeo.
```

Facts Inferred

```
:juliet rdf:type    :Wife;
        rdf:type    :Spouse;
        married_to :romeo;
:romeo rdf:type :Spouse;
       rdf:type :Husband.
```

Facts ***NOT Inferred***

```
:romeo :married to :juliet.
:romeo :husband_of :juliet.
```

Too much representational freedom is not good!

- We might want to be able to detect what might seem inconsistent facts, but RDFS is not able to constrain models through consistency and axioms:

Schema

```
:wife_of rdfs:subPropertyOf married_to.
:married_to rdfs:domain :Spouse;
            rdfs:range  :Spouse.
:wife_of rdfs:domain :Wife;
         rdfs:range  :Husband.
:husband_of rdfs:domain :Husband;
            rdfs:range  :Wife.
```

Assertions

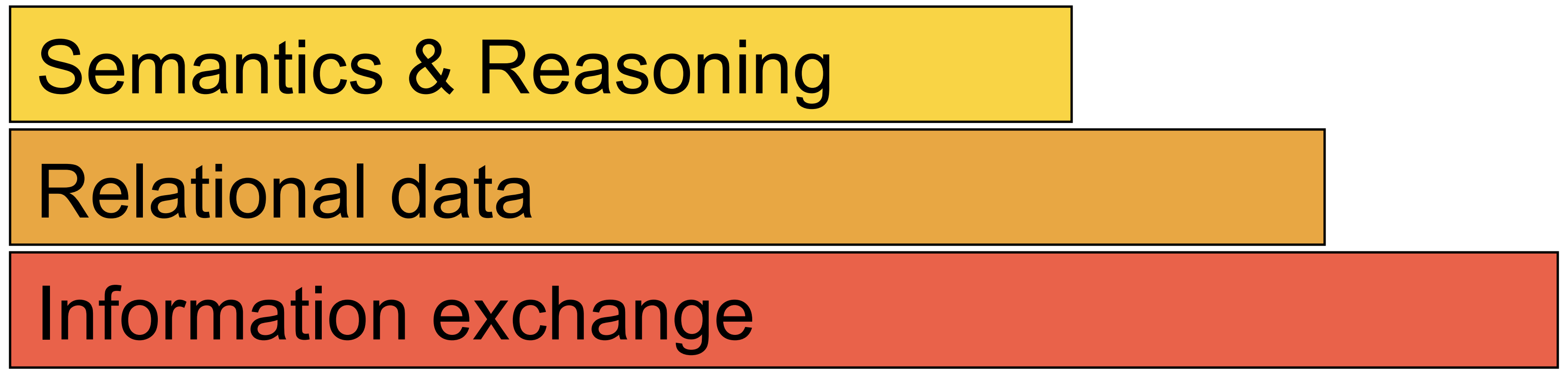
```
:romeo rdf:type :Husband.
:romeo :wife_of : juliet.
```

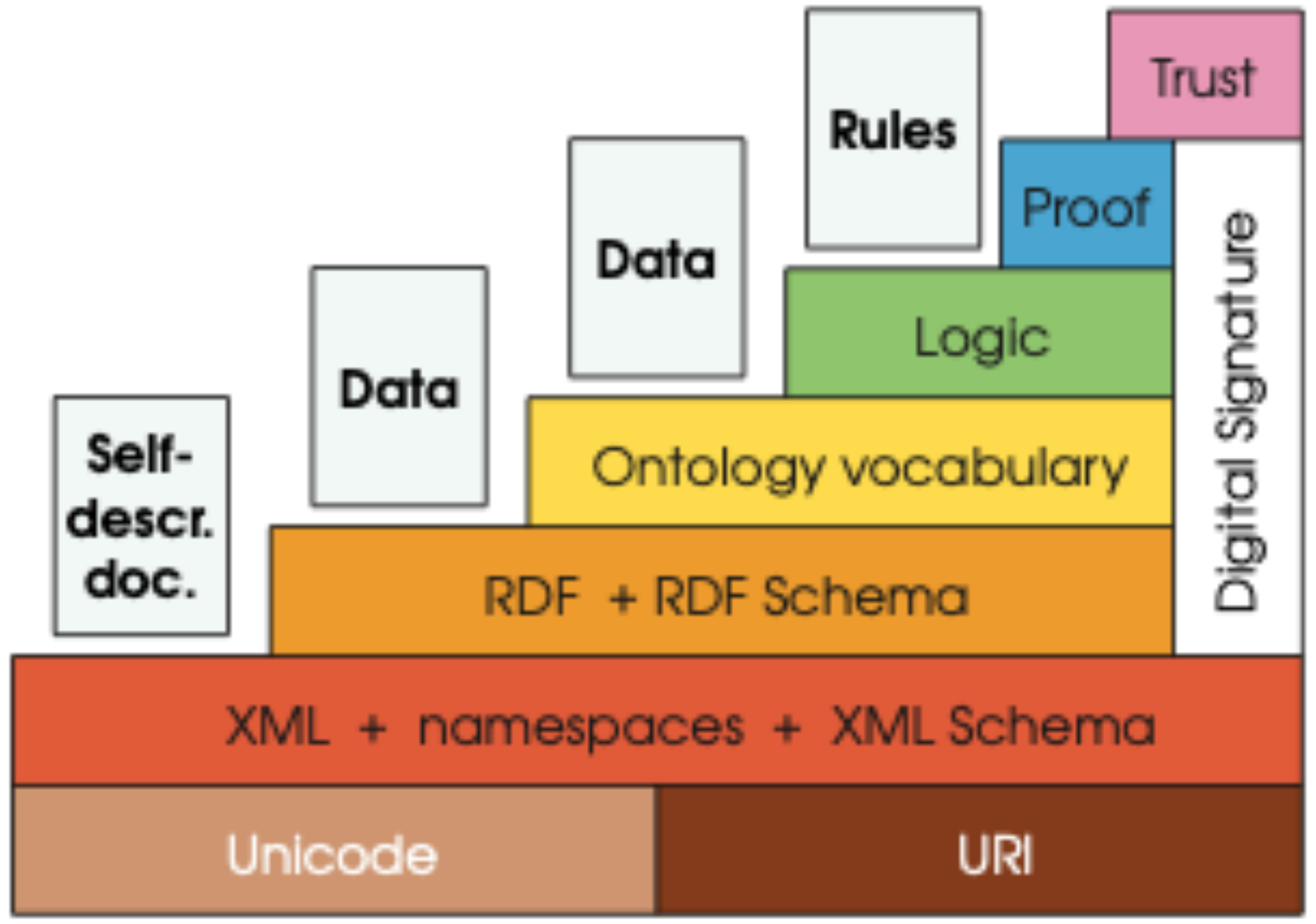
Facts *INCORRECTLY Inferred*

```
:romeo rdf:type :Wife.
```

There is no contradiction, and the mis-modelling is not diagnosed automatically!

Layering of SW languages





T. Berners-Lee

Semantic + Web = Semantic Web

Represent Web content in a form that is more easily machine-processable:

describe meta-data about resources on the Web

i.e. descriptions about the data being represented, the model and constraints used to represent them.

Use intelligent techniques to take advantage of these representations:

process meta-data in a way that is similar to human reasoning and inference

thus information gathering can be done by a machine in a similar way to how humans currently gather information on the web..

Recap

- RDFS entailment
- Limitation of RDFS