COMP318

RDFS Semantics

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

Dr Valentina Tamma

Room: Ashton 2.12

Dept of computer science

University of Liverpool

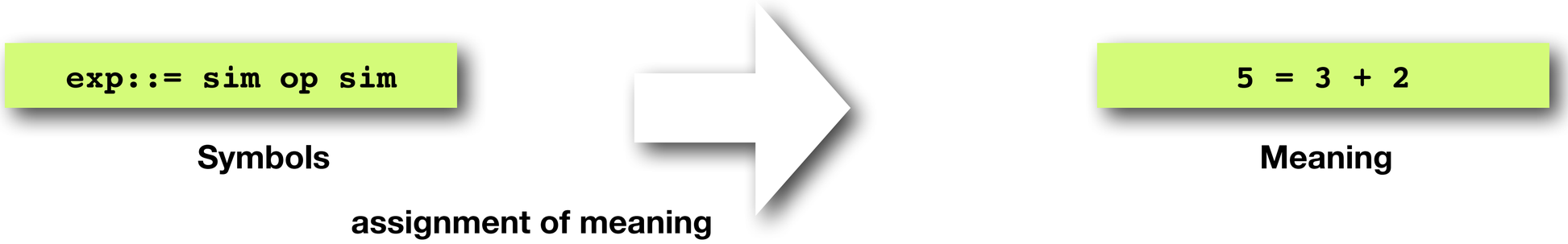
V.Tamma@liverpool.ac.uk

Where were we

- RDF and RDFS
- Vocabulary and model

Syntax vs semantics

- **Syntax**: set of symbols without meaning
- **Semantics**: the meaning associated to the symbols



Semantics

- RDF(S) vocabulary has built-in “meaning”
- RDF(S) Semantics
  - Makes meaning explicit
  - Defines what follows from an RDF graph
- Semantic notions
  - Subgraph
  - Entailment

Example of logical consequence

```
mus: Shoegazing  rdfs:subClassOf mus: Alternative_rock
mus: Alternative_rock rdfs: subClassOf mus: Music_genre
```

- What can we derive???

Example of logical consequence

```
mus: Shoegazing  rdfs:subClassOf mus: Alternative rock
mus: Alternative_rock rdfs: subClassOf mus: Music_genre
```


- What can we derive???

```
mus: Shoegazing  rdfs: subClassOf mus: Music_genre
```

Example of logical consequence

- In deriving the triple

```
mus: Shoegazing  rdfs: subClassOf mus: Music_genre
```

we used the transitive property holding for `subclassOf`

- we used properties of the RDFS vocabulary

RDF(S) entailment

- An RDF(S) graph entails implicit triples
  - triples not explicitly contained in the graph, but that can be derived from an RDF(S) graph
    - using the special semantics of the vocabulary of the graph
      - vocabulary of the graph: set of names which occurs as the subject, predicate, object
  - Interpretations assign special meaning to the symbols in a particular vocabulary

Semantics

- RDF(S) vocabulary has built-in “meaning”
- RDF(S) Semantics
  - Makes meaning explicit
  - Defines what follows from an RDF graph
- Semantic notions
  - Subgraph
  - Entailment

Subgraph

- G2 is a subgraph of G1 if and only if the triples in G2 are a subset of the triples in G1

```
ex1:johnURI       ex1:hasName     ex1:johnfullname
     ex1:johnfullname ex1:firstName   "John"
     ex1:johnfullname ex1:surname     "John"
```

- Each of the following set of triples is a subgraph:

```
ex1:johnURI       ex1:hasName     ex1:johnfullname
```

```
ex1:johnURI       ex1:hasName     ex1:johnfullname
     ex1:johnfullname ex1:firstName   "John"
```

```
     ex1:johnfullname ex1:firstName   "John"
     ex1:johnfullname ex1:surname     "John"
```

RDF(S) entailment

- An RDF(S) graph entails implicit triples
  - triples not explicitly contained in the graph, but that can be derived from an RDF(S) graph
    - using the special semantics of the vocabulary of the graph
      - vocabulary of the graph: set of names which occurs as the subject, predicate, object
  - Interpretations assign special meaning to the symbols in a particular vocabulary

Entailment regimes

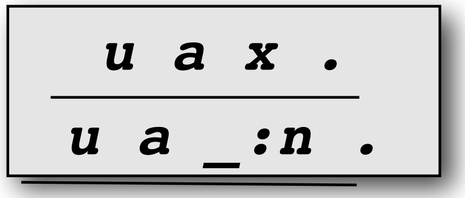
- Three entailment regimes
  - **simple entailment**: no particular extra conditions are posed on a vocabulary, including the RDF vocabulary itself;
    - it involves only graph transformations.
  - RDF entailment;
  - RDFS entailment: some extra conditions are posed by in the form of axiomatic triples and semantic conditions

Let’s state the formalism

- a, b,
  - refer to any arbitrary URI
    - (i.e. anything that can appear in the predicate of a triple)
- u, v,
  - refer to any arbitrary URI or blank node ID
    - (i.e. anything that can appear in the subject of a triple)
- x, y,
  - refer to an arbitrary URI, blank node ID or literal
    - (i.e. anything that can appear in the object of a triple)
- _:n,
  - refer to the ID of a blank node
    - (i.e. appearing as a subject or object)
- l,
  - refers to a literal
    - (i.e. a string that is sometimes found in the object)

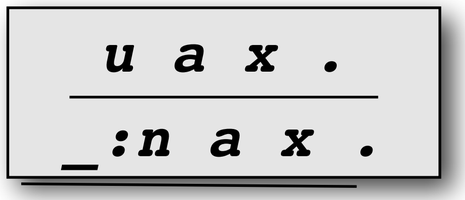
Deduction rules

- If the triple <u, a, x> is valid, then we can entail that the triple <u, a, _:n> is valid



se1

- If the triple <u, a, x> is valid, then we can entail that the triple <_:n, a, x> is valid



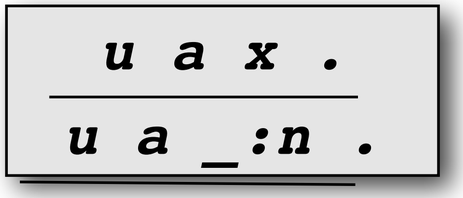
se2

Formalism

a, b, refer to any arbitrary URI
u, v, refer to any arbitrary URI or blank node ID
x, y, refer to any arbitrary URI, blank node ID or literal
_:n, refer to the ID of a blank node
l, refers to a literal

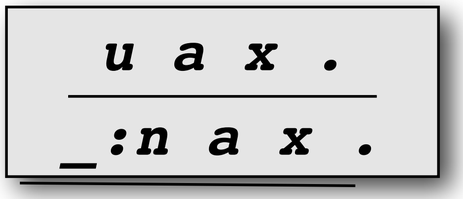
Simple entailment deduction rules

- URIs are all treated equally
  - we can decide whether a graph entails by applying the following rules se1 and se2 and adding the resulting triples to the original graph



se1

*I.e. if you have an **object**, you can derive a **blank node for that object**, as long as the **subject and predicate** stay the same.*



se2

*I.e. if you have a **subject**, you can derive a **blank node for that subject**, as long as the **object and predicate** stay the same.*

Simple entailment deduction rules

- A graph G1 simply entails a graph G2, if G1 can be extended to a graph G’1 by virtue of the rules se1 and se2 such that G2 is contained in G’1
  - G2 ⊆ G’1

Note

- se1 and se2 effectively “weaken” the subject and the object of the triples they are applied
  - by applying se2 to the triple

```
ex1:john    ex1:hasWife    ex1:mary
```

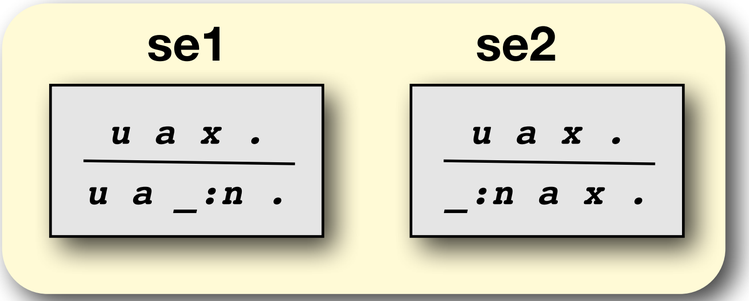
we derive that

```
_:n    ex1:hasWife    ex1:mary
```

  - john hasWife mary is weakened into the statement someone hasWife mary (i.e. mary is a wife)

- se1and se2 can be safely applied only if the blank node _:n that is being introduced was not already present in the graph. If _:n was already introduced, then the rules can be used only to assign _:n with the same resource that which it was originally assigned (see example next)

Example



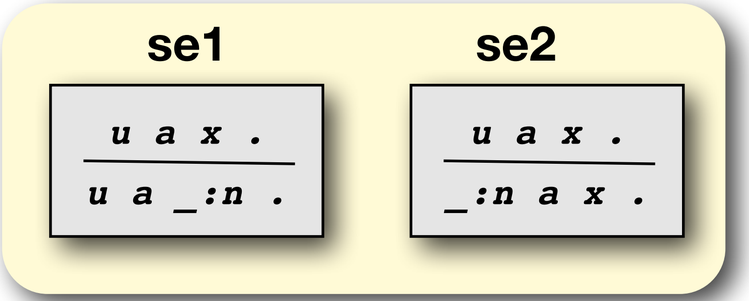
- Let's consider the graph G_1

```
book:uri  ex:publishedBy  crc:uri .
book:uri  ex:title        "SW Technologies" .
crc:uri   ex:name         "CRC Press" .
```

- Let's see if G_1 entails the graph G_2 below

```
book:uri  ex:publishedBy _:blank1 .
_:blank1       ex:name   _:blank2 .
_:blank1       ex:name   "CRC Press" .
```

Example



- Let's consider the graph G1

```
book:uri  ex:publishedBy  crc:uri .
book:uri  ex:title        "SW Technologies" .
crc:uri   ex:name         "CRC Press" .
```

- We need to find out whether (and if so, how) the deduction rules se1 and se2 can be applied to G1 to produce a graph G1' that contains G2

```
book:uri  ex:publishedBy _:blank1 .
_:blank1        ex:name   _:blank2 .
_:blank1        ex:name   "CRC Press" .
```

Example

- By applying se1 to the first triple in G1 we add a triple with a blank node to the graph

```
book:uri  ex:publishedBy  _:blank1 .
```

- By applying se2 to crc:uri ex:name “CRC Press” we can add the triple

```
_:blank1        ex:name    “CRC Press” .
```

  - note that the empty node referenced by _:blank1 has been introduced by rule se1 exactly for crc:uri (and no other URI)

Example

- Finally, by applying se1 to the triple just generated

```
_:blank1        ex:name    “CRC Press” .
```

- we obtain the triple

```
_:blank1   ex:name   _:blank2 .
```

- so now we have

Original Triples

Derived Triples

```
book:uri   ex:publishedBy   crc:uri .
book:uri   ex:title         “SW Technologies” .
crc:uri    ex:name          “CRC Press” .
book:uri   ex:publishedBy   _:blank1 .
_:blank1   ex:name          _:blank2 .
_:blank1   ex:name          “CRC Press” .
```

Example

- G_2

```
book:uri   ex:publishedBy _:blank1 .
_:blank1       ex:name    _:blank2 .
_:blank1       ex:name    “CRC Press” .
```

- is contained in G'_1 and therefore G_1 entails G_2

```
book:uri   ex:publishedBy   crc:uri .
book:uri   ex:title         “SW Technologies” .
crc:uri    ex:name          “CRC Press” .
book:uri   ex:publishedBy   _:blank1 .
_:blank1   ex:name          _:blank2 .
_:blank1   ex:name          “CRC Press” .
```

Exercise

Given the RDF graph G:

```
rdfs:range rdfs:range            rdfs:Class .
:s         rdfs:domain           :b .
_:m        rdfs:subPropertyOf    :s .
:v         :s                    :x .
:v         rdf:type              _:m .
_:m        rdfs:subClassOf       _:n .
:b         rdfs:subClassOf       :s .
:b         rdfs:comment          “blah” .
```

Is the following graph **simple**-entailed by G? Explain the answer

```
:b rdf:type rdfs:Literal .
```

Exercise

Given the RDF graph G:

```
rdfs:range rdfs:range           rdfs:Class .
:s         rdfs:domain          :b .
_:m        rdfs:subPropertyOf   :s .
:v         :s                   :x .
:v         rdf:type             _:m .
_:m        rdfs:subClassOf      _:n .
:b         rdfs:subClassOf      :s .
:b         rdfs:comment         “blah” .
```

Is the following graph **simple**-entailed by G? Explain the answer

```
:b rdf:type rdfs:Literal .
```

No, since there is no triple in the graph that has :b as subject and rdf:type as predicate

Exercise

Given the RDF graph G:

```
rdfs:range rdfs:range           rdfs:Class .
:s         rdfs:domain          :t .
:u         rdfs:subPropertyOf   :s .
:a         :c                   :b .
:a         rdf:type             :u .
:u         rdfs:subClassOf      :y .
:t         rdfs:subClassOf      :s .
:t         rdfs:comment         “blah” .
```

Is the following graph **simple**-entailed by G? Explain the answer

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

Exercise

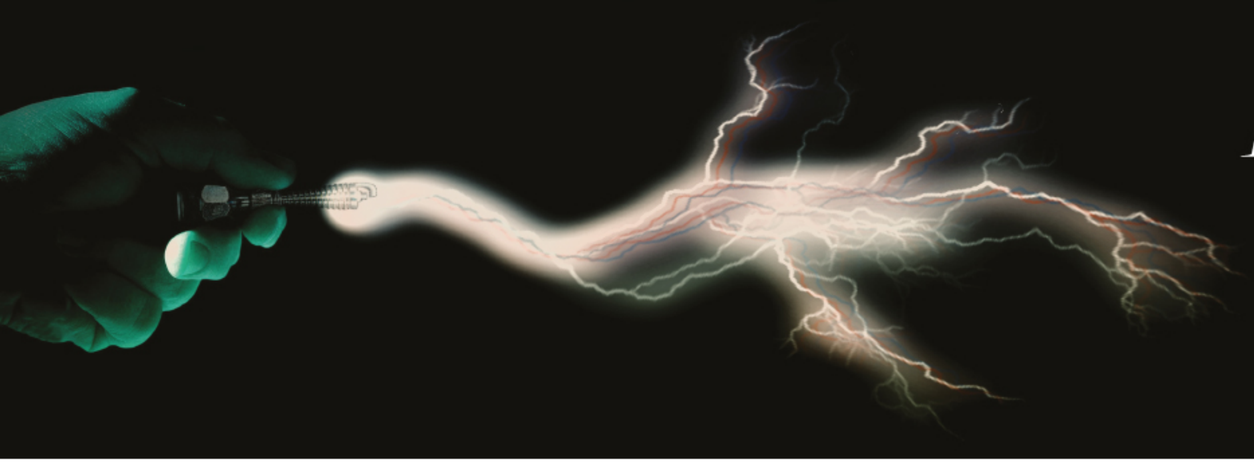
Given the RDF graph G:

```
rdfs:range rdfs:range            rdfs:Class .
:s         rdfs:domain           :t .
:u         rdfs:subPropertyOf    :s .
:a         :c                    :b .
:a         rdf:type              :u .
:u         rdfs:subClassOf       :y .
:t         rdfs:subClassOf       :s .
:t         rdfs:comment          “blah” .
```

Is the following graph **simple**-entailed by G? Explain the answer

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

Yes, it is entailed as the triple is already in the graph S. Therefore, whatever other triple we might add through the rules se1 and se2 the triple would still be contained in S



IGNITE YOUR FUTURE

Preparing for your future, today

8-13 February 2020
Department of Computer Science
University of Liverpool

Our Ignite Your Future series of events this year is even bigger and better!

Throughout the week events run by students for students brings you employers you need, and want, to know.

There will be tutorials where you can increase your knowledge and networking which will enable you to meet new people, build your confidence and get to know about careers in Computer Science.

Come to one or all of the events throughout the week, find out about opportunities and get up to speed with what is happening in the tech sector around the North West and beyond.

Build your network of contacts as you go, bringing you closer to your next internship, placement or job.

Find out more and book via CareerHub.

UNIVERSITY OF LIVERPOOL | Careers and Employability

@livunicareers #IYF2020

SCAN ME

Computer Science Hackathon
Saturday 8 February, 10am - 7pm

How to network
Monday 10 February, 1pm - 3pm

Exploring alternative careers
Monday 10 February, 5pm - 7pm

Now you SME me!
Tuesday 11 February, 2pm - 3pm

Careers in cyber security
Tuesday 11 February, 5pm - 8pm

Computer Science skills session
Wednesday 12 February, from 1.30pm

ISE Careers Fair - Amsterdam
Thursday 13 February, all day event

Making a successful application
Thursday 13 February, 1pm - 2pm

Alumni & friends networking event
Thursday 13 February, 5pm - 8pm

Recap

- Simple entailment