Semantic Web 05: Semantic Schema with RDF Schema

Adila Krisnadhi – adila@cs.ui.ac.id Faculty of Computer Science, Universitas Indonesia

Outline

1. Syntax and Semantic

2. RDF Schema

Syntax vs. Semantics

- Syntax
  - Collection of symbols/terms accompanied by rules (i.e., grammar) that govern how sentences/statements in a language are formed of those symbols/terms.
  - Collection of terms = vocabulary
- Semantics (logic-based)
  - Meaning of those symbols and statements
  - Interpretation specifies what each symbol and statement stand for and when a statement becomes true or false.
  - Often expressed using mathematical sets, relations, etc. to make it unambiguous.

Entailment

- Entailment: Relationship between sentences that hold whenever a set of sentences logically follows from other sets of sentences.

Entailment

- Entailment: Relationship between sentences that hold whenever a set of sentences logically follows from other sets of sentences.
- A set of statements G entails another set of statements H iff the following holds: whenever all statements in G are true, then all statements in H are also true.

Entailment

- Entailment: Relationship between sentences that hold whenever a set of sentences logically follows from other sets of sentences.
- A set of statements G entails another set of statements H iff the following holds: whenever all statements in G are true, then all statements in H are also true.
- Reasoning/inference: Given two sets of statements G and H, if we assume that all statements in G are true, decide whether all statements in H are also true.

Entailment: Example in first-order logic

The set $\{Human(Socrates), \forall x.(Human(x) \rightarrow Mortal(x))\}$

Entailment: Example in first-order logic

The set $\{Human(Socrates), \forall x.(Human(x) \rightarrow Mortal(x))\}$ entails $Mortal(Socrates)$

Entailment: Example in first-order logic

The set $\{Human(Socrates), \forall x.(Human(x) \rightarrow Mortal(x))\}$ entails $Mortal(Socrates)$

- Terms: $Socrates$ (constant) and x (variable).
- Statements: $Human(Socrates)$, $Human(x)$, $Mortal(x)$, and $\forall x.(Human(x) \rightarrow Mortal(x))$
- Interpretation and entailment? (see whiteboard)

Outline

1. Syntax and Semantic

2. RDF Schema

IRI prefixes

RDF semantics specification: `https://www.w3.org/TR/rdf11-mt/`

We use the following IRI prefixes.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <http://www.example.org/data/> .
@prefix o: <http://www.example.org/vocab#>.
```

Semantics of RDF graph

- RDF graph is just a set of triples
  - no description about shape/structure governing the graph,
  - no specific meaning assigned to any IRIs — meaning of datatype IRIs are not defined by RDF semantics.

Semantics of RDF graph

- RDF graph is just a set of triples
  - no description about shape/structure governing the graph,
  - no specific meaning assigned to any IRIs — meaning of datatype IRIs are not defined by RDF semantics.
- When is an RDF graph true?

Semantics of RDF graph

- RDF graph is just a set of triples
  - no description about shape/structure governing the graph,
  - no specific meaning assigned to any IRIs — meaning of datatype IRIs are not defined by RDF semantics.
- When is an RDF graph true? Answer: when all of its triples are true.

Semantics of RDF graph

- RDF graph is just a set of triples
  - no description about shape/structure governing the graph,
  - no specific meaning assigned to any IRIs — meaning of datatype IRIs are not defined by RDF semantics.
- When is an RDF graph true? Answer: when all of its triples are true.
- When is a triple true?

RDF semantics (cont.)

- A triple (s, p, o) is true iff there is a binary relation identified by p connecting entity identified by s to an entity identifed by o.
  - Since one can almost always create such a binary relation, a triple is almost always true.
  - If o is a literal, then the lexical form must be compatible with its datatype for the triple to be true. For example, `"test"^^xsd:integer` is ill-typed.

RDF semantics (cont.)

- A triple (s, p, o) is true iff there is a binary relation identified by p connecting entity identified by s to an entity identifed by o.
  - Since one can almost always create such a binary relation, a triple is almost always true.
  - If o is a literal, then the lexical form must be compatible with its datatype for the triple to be true. For example, `"test"^^xsd:integer` is ill-typed.
- An RDF graph (i.e., a set of RDF triples) is **true** iff all triples in the graph are true.

Simple graph entailment

```
xxx ppp yyy .
```

entails

```
xxx ppp yyy .
```

Simple graph entailment

```
xxx ppp yyy .
```

entails

```
xxx ppp yyy .
```

```
xxx ppp yyy .
```

entails

```
xxx ppp _:nnn .
```

where `_:nnn` must be a new blank node in the graph.

Simple graph entailment

```
xxx ppp yyy .
```

entails

```
xxx ppp yyy .
```

```
xxx ppp yyy .
```

entails

```
xxx ppp _:nnn .
```

where `_:nnn` must be a new blank node in the graph.

```
xxx ppp yyy .
```

entails

```
_:nnn ppp yyy .
```

where `_:nnn` must be a new blank node in the graph.

RDF entailment

```
xxx ppp "aaa"^^ddd .
```

entails

```
xxx ppp _:nnn .
_:nnn rdf:type ddd .
```

where `_:nnn` must be a new blank node in the graph.

RDF entailment

```
xxx ppp "aaa"^^ddd .
```

entails

```
xxx ppp _:nnn .
_:nnn rdf:type ddd .
```

where `_:nnn` must be a new blank node in the graph.

```
xxx ppp yyy .
```

entails

```
ppp rdf:type rdf:Property .
```

RDF Schema

- RDF Schema (RDFS) defines meaning for some particular IRIs to allow us to describe structures in the graph.
- RDFS does not introduce new syntax — everything is syntactically described in RDF.

RDFS features

- Core vocabulary to model classes, instances, property constraints, class and property hierarchy.
- Inference rules for reasoning.
- Auxiliary vocabulary with partial formal meaning for modeling container classes and properties, collections, reification, and additional utility.

rdf:type and rdfs:Class

```
1 :Surabaya rdf:type o:City .
2 :Jakarta a o:City .
3 o:City rdf:type rdfs:Class .
```

- `rdf:type` = instance-of relation = set-membership relation.
- Line 1 reads ":Surabaya is a o:City." Line 2 reads ":Jakarta is a o:City."
- o:City is a class – set of individuals (in this case cities).
- RDFS semantic implies that o:City must be an instance of rdfs:Class. Thus, line 3 is actually implied by line 1 (or line 2).
- `rdfs:Class` is a class that contains all classes, including itself.

rdfs:subClassOf

```
1  :UI rdf:type o:University .
2  :Pertamina a o:OilCompany .
3  o:University rdfs:subClassOf o:EducationInstitution .
4  o:OilCompany rdfs:subClassOf o:ForProfitOrganization .
5  o:EducationInstitution rdf:subClassOf rdfs:Organization .
6  o:ForProfitOrganization rdf:subClassOf rdfs:Organization .
```

- `rdfs:subClassOf` = subset relation.
- Line 3 reads "every (instance of) o:University is a(n instance of) o:EducationInstitution.", i.e., if (x, rdf:type, o:University) is true, then (x, rdf:type o:EducationInstitution) must also be true.
- o:University, o:EducationInstitution, o:Company, o:ForProfitOrganization, and o:Organization are all classes.
- A class may have multiple subclasses and multiple superclasses.
- `rdfs:subClassOf` is transitive, hence can form a class hierarchy.

Properties

- RDF semantics says that every predicate of a triple:
  - semantically corresponds to a binary relation; and

Properties

- RDF semantics says that every predicate of a triple:
  - semantically corresponds to a binary relation; and
  - is called a property (recall the corresponding RDF entailment rule)

Properties

- RDF semantics says that every predicate of a triple:
  - semantically corresponds to a binary relation; and
  - is called a property (recall the corresponding RDF entailment rule)
- The IRI `rdf:Property` is in fact the class of all properties.

Properties

- RDF semantics says that every predicate of a triple:
  - semantically corresponds to a binary relation; and
  - is called a property (recall the corresponding RDF entailment rule)
- The IRI `rdf:Property` is in fact the class of all properties.
- Since it's a binary relation, a property can have domains and/or ranges, which are classes.

rdfs:domain and rdfs:range

```
1  o:taughtBy rdfs:domain o:Course .
2  o:taughtBy rdfs:range o:FacultyMember .
```

- Line 1 means/implies:

rdfs:domain and rdfs:range

```
1  o:taughtBy rdfs:domain o:Course .
2  o:taughtBy rdfs:range o:FacultyMember .
```

- Line 1 means/implies:
  - o:taughtBy is an instance of `rdf:Property`.

rdfs:domain and rdfs:range

```
1 o:taughtBy rdfs:domain o:Course .
2 o:taughtBy rdfs:range o:FacultyMember .
```

- Line 1 means/implies:
  - o:taughtBy is an instance of `rdf:Property`.
  - o:Course is an instance of `rdfs:Class`.

rdfs:domain and rdfs:range

```
1  o:taughtBy rdfs:domain o:Course .
2  o:taughtBy rdfs:range o:FacultyMember .
```

- Line 1 means/implies:
  - o:taughtBy is an instance of `rdf:Property`.
  - o:Course is an instance of `rdfs:Class`.
  - If (x, o:taughtBy, y) is true, then x is an instance of o:Course.

rdfs:domain and rdfs:range

```
1  o:taughtBy rdfs:domain o:Course .
2  o:taughtBy rdfs:range o:FacultyMember .
```

- Line 1 means/implies:
  - o:taughtBy is an instance of `rdf:Property`.
  - o:Course is an instance of `rdfs:Class`.
  - If (x, o:taughtBy, y) is true, then x is an instance of o:Course.
- Line 2 means/implies:

rdfs:domain and rdfs:range

```
1  o:taughtBy rdfs:domain o:Course .
2  o:taughtBy rdfs:range o:FacultyMember .
```

- Line 1 means/implies:
  - o:taughtBy is an instance of `rdf:Property`.
  - o:Course is an instance of `rdfs:Class`.
  - If (x, o:taughtBy, y) is true, then x is an instance of o:Course.
- Line 2 means/implies:
  - o:taughtBy is an instance of `rdf:Property`.

rdfs:domain and rdfs:range

```
1  o:taughtBy rdfs:domain o:Course .
2  o:taughtBy rdfs:range o:FacultyMember .
```

- Line 1 means/implies:
  - o:taughtBy is an instance of `rdf:Property`.
  - o:Course is an instance of `rdfs:Class`.
  - If (x, o:taughtBy, y) is true, then x is an instance of o:Course.
- Line 2 means/implies:
  - o:taughtBy is an instance of `rdf:Property`.
  - o:FacultyMember is an instance of `rdfs:Class`.

rdfs:domain and rdfs:range

```
1  o:taughtBy rdfs:domain o:Course .
2  o:taughtBy rdfs:range o:FacultyMember .
```

- Line 1 means/implies:
  - o:taughtBy is an instance of `rdf:Property`.
  - o:Course is an instance of `rdfs:Class`.
  - If (x, o:taughtBy, y) is true, then x is an instance of o:Course.
- Line 2 means/implies:
  - o:taughtBy is an instance of `rdf:Property`.
  - o:FacultyMember is an instance of `rdfs:Class`.
  - If (x, o:taughtBy, y) is true, then y is an instance of o:FacultyMember.

rdfs:subPropertyOf

```
o:taughtBy rdfs:subPropertyOf o:involves .
```

- `rdfs:subPropertyOf` = subset between two binary relations.

rdfs:subPropertyOf

```
1 o:taughtBy rdfs:subPropertyOf o:involves .
```

- `rdfs:subPropertyOf` = subset between two binary relations.
- Line 1 means/implies:

rdfs:subPropertyOf

```
1  o:taughtBy rdfs:subPropertyOf o:involves .
```

- `rdfs:subPropertyOf` = subset between two binary relations.
- Line 1 means/implies:
  - o:taughtBy is an instance of `rdf:Property`.

rdfs:subPropertyOf

```
1  o:taughtBy rdfs:subPropertyOf o:involves .
```

- `rdfs:subPropertyOf` = subset between two binary relations.
- Line 1 means/implies:
  - o:taughtBy is an instance of `rdf:Property`.
  - If (x, o:taughtBy, y) is true, then (x, o:involves, y) must also be true.

Example

```
:SemanticWeb o:taughtBy :Adila .
:Adila rdf:type o:Lecturer ; o:id "A234"; o:phone "+62-123-4567" .

o:Lecturer rdfs:subClassOf o:FacultyMember .
o:FacultyMember rdfs:subClassOf o:StaffMember .
o:taughtBy rdfs:domain o:Course ; rdfo:range o:FacultyMember ;
           rdfs:subPropertyOf o:involves .
o:id rdfs:domain o:StaffMember ;  rdfs:range rdfs:Literal .
o:phone rdfs:domain o:StaffMember ;  rdfs:range rdfs:Literal .
```

What triples can we infer? Enumerate the elements of each class and property.

RDFS core classes

- `rdfs:Class` - class of all classes.
- `rdf:Property` - class of all properties.
- `rdfs:Literal` - class of all literal values.
- `rdfs:Resource` - class of all resources.
  - Resources are everything, including instances, classes, properties, literals, and datatypes
- `rdfs:Datatype` - class of all datatypes (for literals)
- `rdf:Statement` - class of all RDF statements

RDFS entailment

```
xxx ppp yyy .
ppp rdfs:domain zzz .
```

entails

```
xxx rdf:type zzz .
```

RDFS entailment

```
xxx ppp yyy .
ppp rdfs:domain zzz .
```

entails

```
xxx rdf:type zzz .
```

```
xxx ppp yyy .
ppp rdfs:range zzz .
```

entails

```
yyy rdf:type zzz .
```

RDFS entailment (cont.)

```
ppp rdfs:subPropertyOf qqq .
qqq rdfs:subPropertyOf rrr .
```

entails

```
ppp rdfs:subPropertyOf rrr .
```

RDFS entailment (cont.)

```
ppp rdfs:subPropertyOf qqq .
qqq rdfs:subPropertyOf rrr .
```

entails

```
ppp rdfs:subPropertyOf rrr .
```

```
xxx ppp yyy .
ppp rdfs:subPropertyOf qqq .
```

entails

```
xxx qqq yyy .
```

RDFS entailment (cont.)

```
ppp rdf:type rdf:Property .
```

entails

```
ppp rdfs:subPropertyOf ppp .
```

RDFS entailment (cont.)

```
ppp rdf:type rdf:Property .
```

entails

```
ppp rdfs:subPropertyOf ppp .
```

```
xxx ppp yyy .
```

entails

```
xxx rdf:type rdfs:Resource .
yyy rdf:type rdfs:Resource .
```

RDFS entailment (cont.)

```
xxx rdf:type uuu .
uuu rdfs:subClassOf vvv .
```

entails

```
xxx rdf:type vvv .
```

RDFS entailment (cont.)

```
xxx rdf:type uuu .
uuu rdfs:subClassOf vvv .
```

entails

```
xxx rdf:type vvv .
```

```
uuu rdfs:subClassOf vvv .
vvv rdfs:subClassOf www .
```

entails

```
uuu rdfs:subClassOf www .
```

RDFS entailment (cont.)

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

entails

```
uuu rdfs:subClassOf
        rdfs:Resource .
uuu rdfs:subClassOf uuu .
```

RDFS entailment (cont.)

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

entails

```
uuu rdfs:subClassOf
        rdfs:Resource .
uuu rdfs:subClassOf uuu .
```

The following triples always hold for every datatype IRI `ddd` appearing in the graph:

```
ddd rdf:type rdfs:Datatype .
```

RDFS entailment (cont.)

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

entails

```
uuu rdfs:subClassOf
        rdfs:Resource .
uuu rdfs:subClassOf uuu .
```

The following triples always hold for every datatype IRI `ddd` appearing in the graph:

```
ddd rdf:type rdfs:Datatype .
```

Moreover,

```
ddd rdf:type rdfs:Datatype .
```

entails

```
ddd rdfs:subClassOf rdfs:Literal .
```

RDF/RDFS axiomatic triples

RDF/RDFS axiomatic triples are those that are set by the RDF/RDFS semantics to always be true.

```
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:nil rdf:type rdf:List .
rdf:_1 rdf:type rdf:Property .
rdf:_2 rdf:type rdf:Property .
...
```

```
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 .
```

RDF/RDFS axiomatic triples (cont.)

```
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 .
```

```
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 .

rdfs:Datatype rdfs:subClassOf rdfs:Class .

rdf:_1 rdf:type rdfs:ContainerMembershipProperty .
rdf:_1 rdfs:domain rdfs:Resource .
rdf:_1 rdfs:range rdfs:Resource .
rdf:_2 rdf:type rdfs:ContainerMembershipProperty .
rdf:_2 rdfs:domain rdfs:Resource .
rdf:_2 rdfs:range rdfs:Resource .
...
```

RDFS basic reasoning procedure

Given an RDF graph, RDFS reasoning computes all its entailed statements as follows:

1. Add all RDF/RDFS axiomatic triples except those containing the container membership property IRIs `rdf:_1`, `rdf:_2`, ...
2. For every container membership property IRI which occurs in the graph, add the axiomatic triples which contain that IRI.
3. Add triples to the graph according to the inference rules until no new triple can be added.

RDFS containers

- Vocabulary to describe containers (not constructing container like in programming languages).
  - Items in containers are enumerated
- No formal semantics: Intended just for human consumption
- Three types of containers: `rdf:Bag`, `rdf:Seq`, `rdf:Alt`.
- `rdf:_1`, `rdf:_2`, etc. are properties for enumerating They are all subproperty of `rdfs:member` and all of them are instances of `rdfs:ContainerMembershipProperty`.

Example

```
[] rdf:type rdf:Bag ;
   rdf:_1 :itemA ;
   rdf:_2 :itemB .

[] rdf:type rdf:Seq ;
   rdf:_1 :itemC ;
   rdf:_2 :itemD .

[] rdf:type rdf:Alt ;
   rdf:_1 :choice1 ;
   rdf:_2 :choice2 .
```

"A bag with two items" Indicates (informally) that the container is intended to be unordered

"A sequence with two items" Indicates (informally) that the numerical ordering of the container membership properties is significant.

"An 'Alternative' container with two choices" Indicates (informally) that a typical processing is to select one of the members of the container.

RDFS collections

- Vocabulary to describe “list structure”.
- Unlike containers, collections:
  - can have branching structure, and
  - has an explicit terminator.
- One type of collection: `rdf:List`.
- Use the properties `rdf:first` and `rdf:rest`, as well as the resource `rdf:nil` to form the list structure (akin to LISP/Haskell)
- Can be hidden using parentheses syntax in Turtle.

Example

A list with three elements whose third element is a list with two elements.

```
:somelist s:content (:x :y (:z :w) ) .
```

is equivalent to

Example

A list with three elements whose third element is a list with two elements.

```
:somelist s:content (:x :y (:z :w) ) .
```

is equivalent to

```
:somelist :content _:genid1 .
_:genid1 rdf:first :x ; rdf:rest _:genid2 .
_:genid2 rdf:first :y ; rdf:rest _:genid3 .
_:genid4 rdf:first :z ; rdf:rest _:genid5 .
_:genid5 rdf:first :w ; rdf:rest rdf:nil .
_:genid3 rdf:first _:genid4 ; rdf:rest rdf:nil .
```

RDFS container membership entailment

```
ppp rdf:type rdfs:ContainerMembershipProperty .
```

entails

```
ppp rdfs:subPropertyOf rdfs:member .
```

Other vocabulary terms

- None of these vocabulary terms have formal semantics!
- Utility properties
  - `rdfs:label` — human-readable label
  - `rdfs:comment` — for commenting
  - `rdfs:seeAlso` — pointing to other IRI that explains the resource
  - `rdfs:isDefinedBy` — subproperty of rdfs:seeAlso
- RDF reification (rarely used nowadays) — see the standards.
  - `rdf:Statement`
  - `rdf:subject`
  - `rdf:predicate`
  - `rdf:object`