RDF and RDFS
axiomatic semantics

the meaning of the modeling primitives of RDF and RDF Schema is being translated into first-order logic

it provides a basis for reasoning support by automated reasoners

RDF and RDFS
axiomatic semantics - approach

language primitives are represented by constants: Resource, Class, Property, subClassOf, etc.

a few predefined predicates are used as for expressing relationships between the constants

variable names begin with “?”

all axioms are implicitly universally quantified

RDF and RDFS
axiomatic semantics - approach

basic predicates

PropVal(P, R, V)

• a predicate with 3 arguments, is used to represent an RDF statement with resource R, property P and value V
• an RDF statement (triple) (R, P, V) is represented as PropVal(P, R, V)

RDF
basic ideas – statements (from lecture 4)

a triple:

object-attribute-value

or in other words:

resource-property-value
(values are resources or literals)

or

subject-property-object
RDF and RDFS a. semantics

basic predicates

\[
\text{Type}(R, T)
\]

- short for \(\text{PropVal(type, R, T)}\)
- specifies that the resource \(R\) has the type \(T\)

\[
\text{Type(?r, ?t)} \leftrightarrow \text{PropVal(type, ?r, ?t)}
\]

RDF and RDFS a. semantics
classes

constants:
\[
\text{Class}, \text{Resource}, \text{Property}, \text{Literal}
\]

- all classes are instances of \(\text{Class}\)

\[
\begin{align*}
\text{Type}(\text{Class}, \text{Class}) \\
\text{Type}(\text{Resource}, \text{Class}) \\
\text{Type}(\text{Property}, \text{Class}) \\
\text{Type}(\text{Literal}, \text{Class})
\end{align*}
\]

RDF and RDFS a. semantics
classes (2)

\[
\text{Resource} \text{ is the most general class: every class and every property is a resource}
\]

\[
\begin{align*}
\text{Type(?p, Property)} & \rightarrow \text{Type(?p, Resource)} \\
\text{Type(?c, Class)} & \rightarrow \text{Type(?c, Resource)}
\end{align*}
\]

RDF and RDFS a. semantics
classes (3)

the predicate in an RDF statement must be a property

\[
\begin{align*}
\text{PropVal(?p,?r,?v)} & \rightarrow \text{Type(?p, Property)}
\end{align*}
\]

RDF and RDFS a. semantics
type property

\[
\text{type} \text{ is a property}
\]

\[
\begin{align*}
\text{Type(type, Property)}
\end{align*}
\]

is equivalent to

\[
\begin{align*}
\text{PropVal(type, type, Property)}
\end{align*}
\]

RDF and RDFS a. semantics
type property (2)

\[
\text{type} \text{ can be applied to resources (domain) and has a class as its value (range)}
\]

\[
\begin{align*}
\text{Type(?r, ?c)} & \rightarrow \\
(\text{Type(?r, Resource)} \land \text{Type(?c, Class)})
\end{align*}
\]
**RDF and RDFS a semantics subclass**

**subClassOf** is a property

\[
\text{Type}(\text{subClassOf}, \text{Property}) = \text{PropVal}(\text{type}, \text{subClassOf}, \text{Property})
\]

RDF and RDFS a semantics subclass

if a class \( C \) is a subclass of a class \( C' \), then all instances of \( C \) are also instances of \( C' \):

\[
\text{PropVal}(\text{subClassOf}, ?c, ?c') \leftrightarrow (\text{Type(?c, Class)} \land \text{Type(?c', Class)} \land \\
\forall ?x (\text{Type(?x, ?c) } \rightarrow \text{Type(?x, ?c'))})
\]

**RDF and RDFS a semantics subproperty**

**subPropertyOf** is a property

\[
\text{Type}(\text{subPropertyOf}, \text{Property}) = \text{PropVal}(\text{type}, \text{subPropertyOf}, \text{Property})
\]

RDF and RDFS a semantics subproperty

\( P \) is a subproperty of \( P' \), if \( P'(x,y) \) is true whenever \( P(x,y) \) is true:

\[
\text{PropVal}(\text{subPropertyOf}, ?p, ?p') \leftrightarrow (\text{Type(?p, Property)} \land \text{Type(?p', Property)} \land \\
\forall ?r \forall ?v (\text{PropVal(?p, ?r, ?v) } \rightarrow \text{PropVal(?p', ?r, ?v'))})
\]

RDF and RDFS a semantics domain and range

if the domain of \( P \) is \( D \), then for every \( P(x,y) \), \( x \in D \)

\[
\text{PropVal}(\text{domain}, ?p, ?d) \rightarrow \\
\forall ?x \forall ?y (\text{PropVal(?p, ?x, ?y) } \rightarrow \text{Type(?x, ?d)})
\]

RDF and RDFS a semantics domain and range

if the range of \( P \) is \( R \), then for every \( P(x,y) \), \( y \in R \)

\[
\text{PropVal}(\text{range}, ?p, ?r) \rightarrow \\
\forall ?x \forall ?y (\text{PropVal(?p, ?x, ?y) } \rightarrow \text{Type(?y, ?r)})
\]
RDF and RDFS a semantics

issues

can be used for automatic reasoning, but ...

it needs first-order logic proof system to do so

RDF and RDFS semantics

inference rules

semantics in terms of RDF triples instead of
restating RDF in terms of first-order logic
the inference system consists of inference rules:

IF E contains certain triples
THEN add to E certain additional triples

where E is an arbitrary set of RDF triples

RDF and RDFS semantics

inference rules – examples

IF E contains the triple (?x, ?p, ?y)
THEN E also contains (?p, rdf:type, rdf:Property)

any resource ?p used in the property position can be inferred to be
a member of the class rdf:Property

RDF and RDFS semantics

inference rules – examples

IF E contains the triples
(?u, rdfs:subClassOf, ?v) and (?v, rdfs:subClassOf, ?w)

THEN E also contains the triple (?u, rdfs:subClassOf, ?w)

encodes transitivity of the subclass relation

RDF and RDFS semantics

inference rules – examples

IF E contains the triples
(?x, rdf:type, ?u) and (?u, rdfs:subClassOf, ?v)

THEN E also contains the triple (?x, rdf:type, ?v)

RDF and RDFS semantics

inference rules – examples

IF E contains the triples
(?x, ?p, ?y) and (?p, rdfs:range, ?u)

THEN E also contains the triple (?y, rdf:type, ?u)

any resource ?y which appears as the value of a property ?p can be
inferred to be a member of the range of ?p
(range definitions in RDF Schema are not used to restrict the range
of a property, but rather to infer the membership of the range)
SPARQL
a query language for RDF

XML query language (XPath) not suitable

SPARQL
(SPARQL Protocol and RDF Query Language)
is based on matching graph patterns

SPARQL
example

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?c
WHERE {
  ?c rdf:type rdfs:Class .
}
this query retrieves all triple patterns, where:
-the property is rdf:type
-the object is rdfs:Class
-it retrieves all classes

SPARQL
example (2)

PREFIX uni: <http://www.mydomain.org/uni-ns#>
SELECT ?i
WHERE {
  ?i rdf:type uni:course .
}
to retrieve all instances of a particular class (e.g. course) :
(declaration of rdf, rdfs prefixes omitted for brevity)

SPARQL
select-from-where – query structure

SELECT
specifies the projection: the number and order of retrieved data

FROM
is used to specify the source being queried (optional)

WHERE
imposes constraints on possible solutions in the form of graph
pattern templates and boolean constraints

SPARQL
select-from-where

retrieve all phone numbers of staff members

SELECT ?x ?y
WHERE {
  ?x uni:phone ?y .
}
here ?x and ?y are variables, and ?x uni:phone ?y represents a
resource-property-value triple pattern
we restrict the second pattern only to those triples, the resource of which is in the variable \(?x\) (a semicolon indicates that the following triple shares its subject with the previous one)

retrieve all lecturers and their phone numbers:

```
SELECT ?x ?y
WHERE
{ ?x rdf:type uni:Lecturer ;
  uni:phone ?y . }
```

the previous query is equivalent to

```
SELECT ?x ?y
WHERE
{ ?x rdf:type uni:Lecturer .
  ?x uni:phone ?y . }
```

retrieve the name of all courses taught by the lecturer with ID 949352

```
SELECT ?n
WHERE
{ ?x rdf:type uni:Course ;
  uni:isTaughtBy :949352 .
  ?c uni:name ?n .
  FILTER (?c = ?x) .
}
```

as a solution we can adapt the query to use an optional pattern

```
SELECT ?name ?email
WHERE
{ ?x rdf:type uni:Lecturer ;
  uni:name ?name ;
  uni:email ?email .
}
```

the result: John Smith, js@ece.ualberta.ca

the result: Mike Knight, John Smith, js@ece.ualberta.ca
RDF summary

- provides a foundation for representing and processing metadata
- has a graph-based data model
- has an XML-based syntax
- allows incremental building of knowledge, and its sharing and reuse
- is domain independent

RDFS summary

- RDF Schema is a primitive ontology language (offers certain modelling primitives with fixed meaning)
- key concepts of RDF Schema are class, subclass relations, property, subproperty relations, and domain and range restrictions
- there are query languages for RDF and RDFS, including SPARQL