RDF SCHEMA

47

47

Schema

- Intended to give meaning to data
- RDF Schema does this based on *inference*

- Identify resources (sets and properties)
  - rdfs:Class
  - rdfs:subClassOf
  - rdfs:subPropertyOf
- Inferences based on use of these resources

48

48

Class and Subclass

- Specify sets:

```
:Faculty rdf:type rdfs:Class
:Researcher rdf:type rdfs:Class
```

- Specify subset inclusion

```
:Researcher rdfs:subClassOf :Faculty
```

- Semantics based on inference from subclassing
  - Given: `:Duggan rdf:type :Researcher`
  - Infer: `:Duggan rdf:type :Faculty`

49

49

Class and Subclass

- Nouns: subclassing

- Suppose $P_A(\ldots)$ is a unary predicate for class A
- Suppose $P_B(\ldots)$ is a unary predicate for class B

- If we specify `A rdfs:subClassOf B`, then the property P_A implies the property P_B
  - i.e., if $P_A(X)$ then we can infer $P_B(X)$
  - i.e., $P_A(X) \rightarrow P_B(X)$

50

50

Property and Subproperty

- Nouns: subclassing
- Verbs: subproperties

- If we specify `P rdfs:subPropertyOf Q`, then the property P implies the property Q
  - i.e., if P(X,Y) then we can infer Q(X,Y)
  - i.e., P(X,Y) $\rightarrow$ Q(X,Y)

51

51

Example: Ancestors

- Ancestor Relation in Logic

```
father(X,Y) → parent(X,Y)
mother(X,Y) → parent(X,Y)
parent(X,Y) → ancestor(X,Y)
ancestor(X,Y) & ancestor(Y,Z) → ancestor(X,Z)
father(Joe,Mary)
mother(Mary,Jane)
```

- Deductions

```
parent(Joe,Mary)          parent(Mary,Jane)
ancestor(Joe,Mary)        ancestor(Mary,Jane)
ancestor(Joe,Jane)
```

52

52

Example: Ancestors

- Ancestor Relation in RDFS Schema

  :father rdfs:subPropertyOf :parent.
  :mother rdfs:subPropertyOf :parent.
  :parent rdfs:subPropertyOf :ancestor.

  :Joe :father :Mary.
  :Mary :mother :Jane.
- Deductions

  :Joe :parent :Mary.
  :Joe :ancestor :Mary.
  ~~:Joe :ancestor :Jane.~~

  :Mary :parent :Jane.
  :Mary :ancestor :Jane.

53

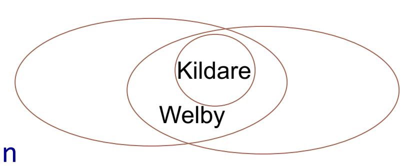
53

Design Pattern: Set Intersection

- We can define a set that is contained in both A and B, and therefore contained in their intersection
  - …but we cannot require it to be exactly the intersection
- Example:

  :Surgeon rdfs:subClassOf :Staff
  :Surgeon rdfs:subClassOf :Physician
  :Kildare rdf:type :Surgeon
- Infer:

  :Kildare rdf:type :Staff
  :Kildare rdf:type :Physician



54

54

Design Pattern: Property Intersection

- We can define a property that is contained in both P and Q, and therefore contained in their intersection
  - …but we cannot require it to be exactly the intersection
- Example:

```
:lodgedIn rdfs:subPropertyOf :billedFor.
:lodgedIn rdfs:subPropertyOf :assignedTo.
:Marcus :lodgedIn :Room101.
```

- Infer:

```
:Marcus :billedFor :Room101.
:Marcus :assignedTo :Room101.
```

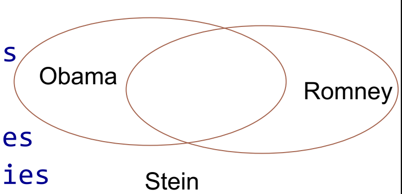
55

55

Design Pattern: Set Union

- We can define a set that contains A and B, and therefore contains their union
  - …but we cannot require it to be exactly the union
- Example:

```
:Democrats rdfs:subClassOf :BigTwoParties
:Republicans rdfs:subClassOf :BigTwoParties
:Obama rdf:type :Democrats
:Romney rdf:type :Republicans
```



- Infer:

```
:Obama rdf:type :BigTwoParties
:Romney rdf:type :BigTwoParties
```

56

56

Design Pattern: Property Union

- We can define a property that contains P and Q, and therefore contains their union
  - …but we cannot require it to be exactly the union
- Example:

```
:Library1:borrows
         rdfs:subPropertyOf has:Possession
:Library2:checkedOut
         rdfs:subPropertyOf has:Possession
```

- To make properties equivalent:

```
:Library1:borrows
         rdfs:subPropertyOf :Library2:checkedOut
:Library2:checkedOut
         rdfs:subPropertyOf :Library1:borrows
```


57

Typing Data by Usage

- RDF Schema (meta-)predicates
  - *Property* `rdfs:domain` *SubjectType*
  - *Property* `rdfs:range` *ObjectType*

- Then we can make inferences about types:
  - If P `rdfs:domain` D and x P y
    then x `rdf:type` D
  - If P `rdfs:range` R and x P y
    then y `rdf:type` R

58

Data Typing based on use

Table 6-1 Ships					
Name	**Maiden Voyage**	**Next Departure**	**Decommission Date**	**Destruction Date**	**Commander**
Berengaria	June 16, 1913		1938		Johnson
QEII	May 2, 1969	March 4, 2010			Warwick
Titanic	April 10, 1912			April 14, 1912	Smith
Constitution	July 22, 1798	January 12, 2009			Preble

Idea: Classify ships based on information known about them.

```
ship:DeployedVessel rdfs:subClassOf ship:Vessel .
ship:InServiceVessel rdfs:subClassOf ship:Vessel .
ship:OutOfServiceVessel rdfs:subClassOf ship:Vessel .
```

59

59

Data typing based on use (2)

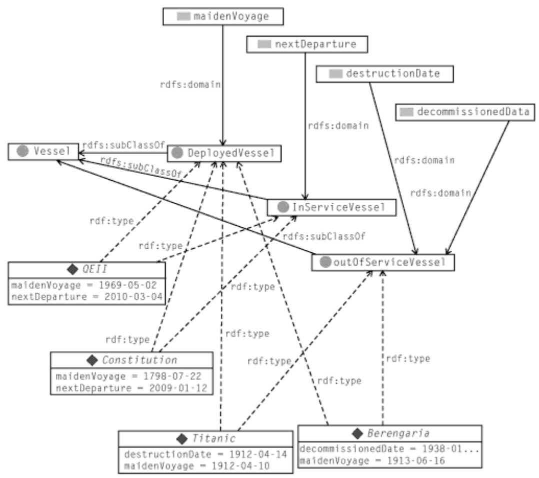
- A ship is deployed if it has had a maiden voyage
  `ship:maidenVoyage rdfs:domain ship:DeployedVessel .`
- A ship is still in service it it has a next departure date set
  `ship:nextDeparture rdfs:domain ship:InServiceVessel .`
- A ship is out of service if it has a decommissioned date or a destruction date

```
ship:decommissionedDate
   rdfs:domain ship:OutOfServiceVessel .
ship:destructionDate
   rdfs:domain ship:OutOfServiceVessel .
```

60

60

Data typing based on use (3)



61

61

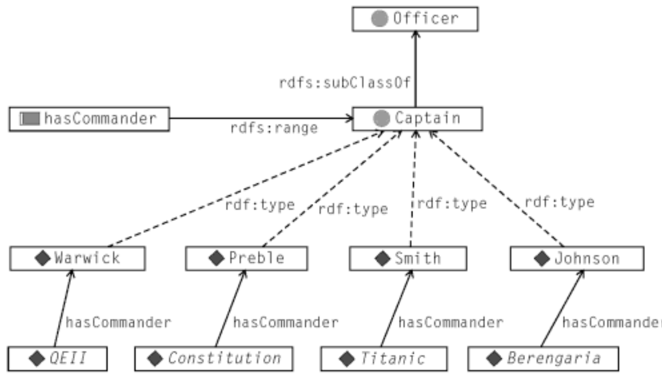
Data typing based on use (4)

- Define officer ranks:
  `ship:Captain rdfs:subClassOf ship:Officer`
  `ship:Commander rdfs:subClassOf ship:Officer`
  `ship:LieutenantCommander rdfs:subClassOf ship:Officer`
  `ship:Lieutenant rdfs:subClassOf ship:Officer`
  `ship:Ensign rdfs:subClassOf ship:Officer`
- A ship’s commander has rank Captain
  `ship:hasCommander rdfs:range ship:Captain`

62

62

Data typing based on use (5)



63

63

Data typing based on use (6)

- Filter set of ships based on properties known about them
- Ex: Define the class of departing ships
  `ship:DepartingVessel rdf:type rdfs:Class`
- Defined as those ships that have a departure date
  `ship:nextDeparture rdfs:domain`
  `ship:DepartingVessel`
- Note: rdfs:domain and rdfs:range are used for *knowledge discovery* rather than *knowledge description*

64

64

Multiple Domains/Ranges

- We had two definitions for nextDeparture domain:

```
ship:nextDeparture rdfs:domain DepartingVessel
ship:nextDeparture rdfs:domain InServiceVessel
```

- Consider the QEII:

```
ship:QEII ship:maidenVoyage “May 2, 1969” .
ship:QEII ship:nextDeparture “Mar 4, 2010” .
```

- Then we can conclude:

```
ship:QEII rdf:type ship:DepartingVessel
ship:QEII rdf:type ship:InServiceVessel
```

i.e. QEII is in the intersection of DepartingVessel and InServiceVessel

65

65

Limitations of RDF Schema

- Local scope of properties
  - rdfs:range cannot declare range restrictions that apply to some classes only
  - E.g. cows eat only plants, while other animals may eat meat, too
- Boolean combinations of classes
  - intersection, union, complement
- Special characteristics of properties
  - transitivity, uniqueness, inverse, …

66

66