CC6202-1
La Web de Datos
Primavera 2015

Lecture 3: RDF Semantics and Schema

Aidan Hogan
aidhog@gmail.com

LAST TIME …

(1) **Data**, (2) Rules/Ontologies, (3) Query,

INPUT: "(x,partOf,y)"

DATA:

`http://ex.org/Ireland`

Ireland 

(Ireland,partOf,Europe)
(Ireland,a,Country)
(Ireland,capital,Dublin)

`http://ex.org/Dublin`

Dublin 

(Dublin,population,1000000)

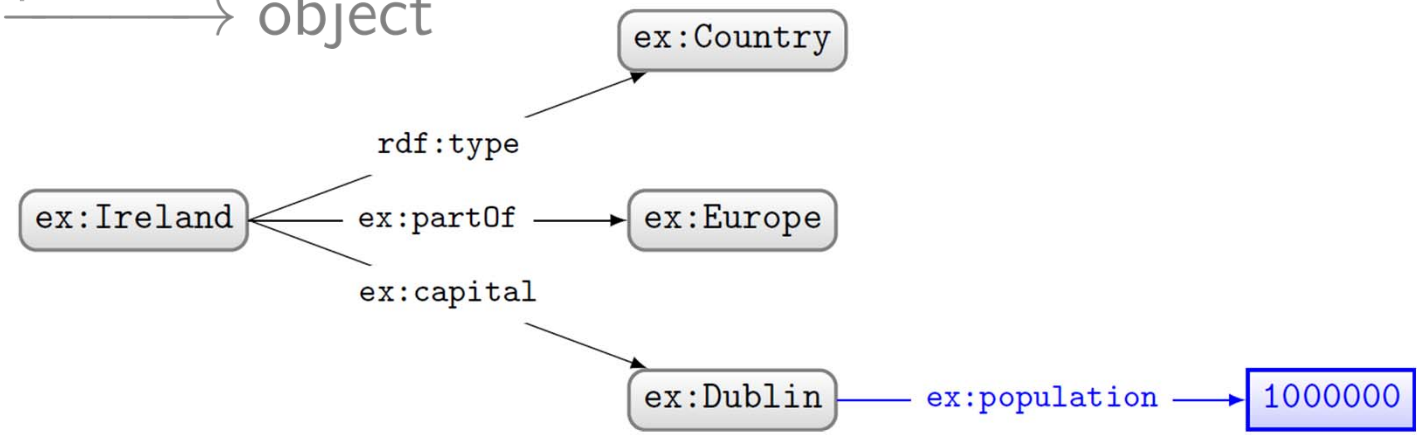
RULES: $(a, \text{capital}, b) \rightarrow (b, \text{partOf}, a)$
$(c, \text{partOf}, d), (d, \text{partOf}, e) \rightarrow (c, \text{partOf}, e)$

OUTPUT: $\{(x \mapsto \text{Ireland}, y \mapsto \text{Europe}), (x \mapsto \text{Dublin}, y \mapsto \text{Ireland})$
$(x \mapsto \text{Dublin}, y \mapsto \text{Europe})\}$

RDF: Resource Description Framework

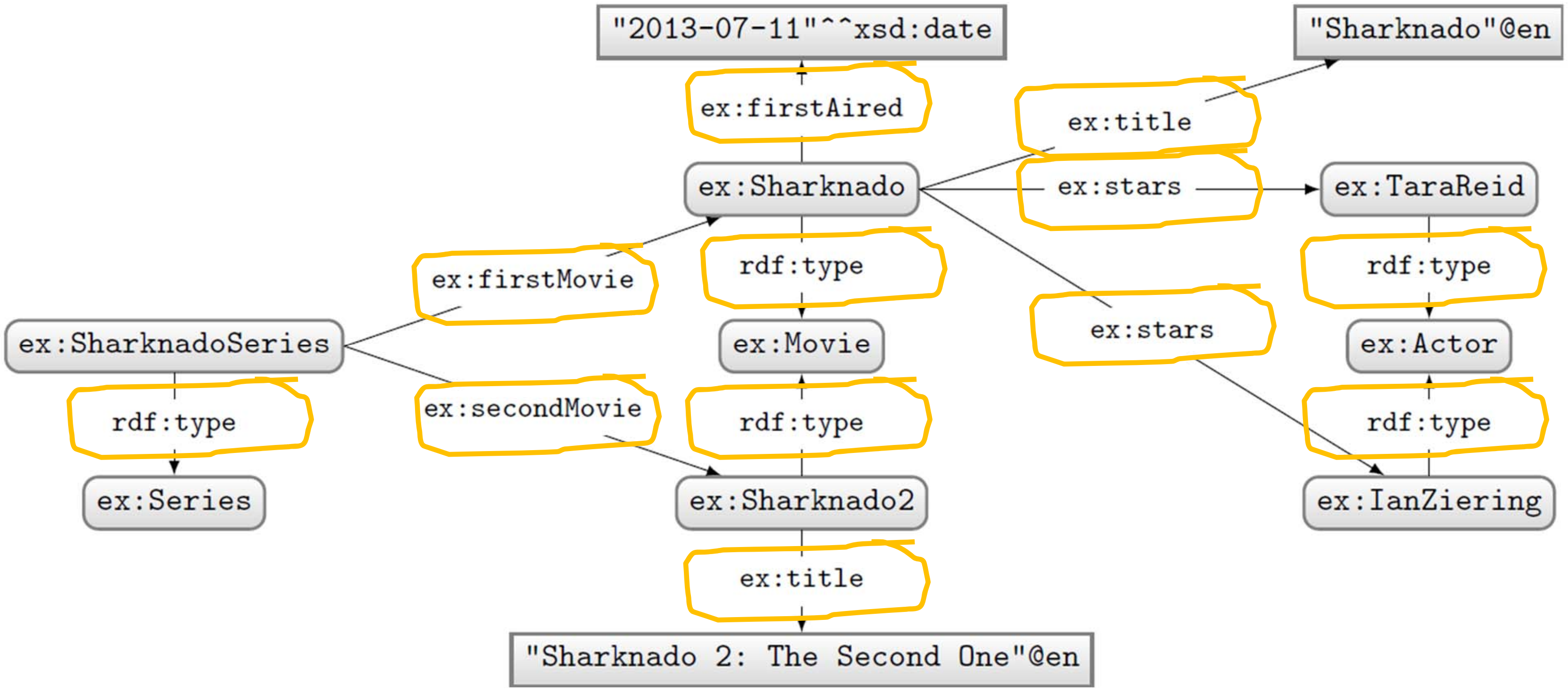
subject	*predicate*	*object*
ex:Ireland	ex:partOf	ex:Europe
ex:Ireland	rdf:type	ex:Country
ex:Ireland	ex:capital	ex:Dublin
ex:Dublin	ex:population	1,000,000

$\text{subject} \xrightarrow{\text{predicate}} \text{object}$



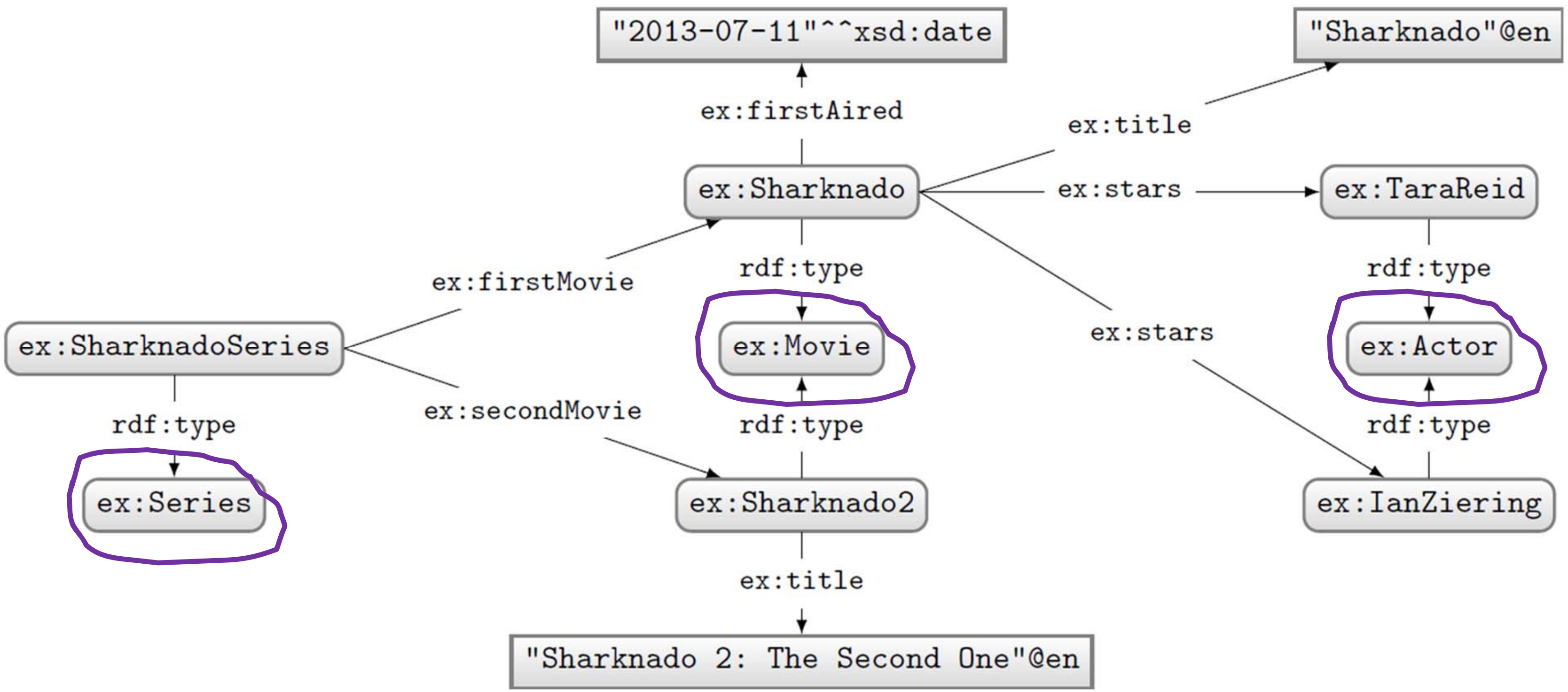
RDF Properties

- RDF Terms used as predicate
- `rdf:type`, `ex:firstMovie`, `ex:stars`, …



RDF Classes

- Used to conceptually group resources
- The predicate `rdf:type` is used to relate resources to their classes



LAB REVIEW …

Describe yourself in RDF

http://aidanhogan.com/foaf.rdf [view]

Q&D RDF Browser is powered by Graphite and ARC2 and hosted by ECS at the University of Southampton.

177 Triples

Aidan's FOAF profile a Document
http://aidanhogan.com/foaf.rdf
- → rdf:type → foaf:Document
- → dc:title → "Aidan's FOAF profile"@en
- → dc:creator → Aidan Hogan
- → dc:date → "2005-04-07"
- → foaf:primaryTopic → Aidan Hogan

Aidan Hogan a Person
http://aidanhogan.com/foaf.rdf#me

- → rdf:type → foaf:Person
- → owl:sameAs → http://www.deri.ie/about/team/member/Aidan_Hogan#me, http://sw.deri.org/~aidanh/foaf/foaf.rdf#Aidan_Hogan, http://aidanhogan.com/foaf/foaf.rdf#Aidan_Hogan, http://data.semanticweb.org/person/aidan-hogan, ...show 4 more...
- → foaf:name → "Aidan Hogan"@en, "Aodhán Ó hÓgáin"@ga
- → foaf:title → "Dr"
- → foaf:firstName → "Aidan"@en, "Aodhán"@ga-IE
- → foaf:surname → "Hogan"@en, "Ó hÓgáin"@ga
- → foaf:myersBriggs → "INTP"
- → foaf:gender → "male"@en

TODAY’S TOPIC …

(1) Data, **(2) Rules/Ontologies,** (3) Query

INPUT: "(x,partOf,y)"

DATA:

http://ex.org/Ireland

Ireland 

(Ireland,partOf,Europe)
(Ireland,a,Country)
(Ireland,capital,Dublin)

http://ex.org/Dublin

Dublin 

(Dublin,population,1000000)

RULES: $(a, \text{capital}, b) \to (b, \text{partOf}, a)$
$(c, \text{partOf}, d), (d, \text{partOf}, e) \to (c, \text{partOf}, e)$

OUTPUT: $\{(x \mapsto \text{Ireland}, y \mapsto \text{Europe}), (x \mapsto \text{Dublin}, y \mapsto \text{Ireland})$
$(x \mapsto \text{Dublin}, y \mapsto \text{Europe})\}$

How to structure "rules"?

- How to write them down?
- Where to publish them?

How should we structure
rules on the Semantic Web?

$$\textsc{Rules:} \quad \begin{array}{l} (a, \mathsf{capital}, b) \rightarrow (b, \mathsf{partOf}, a) \\ (c, \mathsf{partOf}, d), (d, \mathsf{partOf}, e) \rightarrow (c, \mathsf{partOf}, e) \end{array}$$

Semantic Web Answer: Schema/Ontologies

- Don't use rules: Use RDF!
- Define relationships between classes and properties

What sorts of relationships might be useful to define between the following classes and properties?

ex:Town

ex:hasCapitalCity

ex:hasCity

ex:City

ex:Country

foaf:familyName

ex:Place

foaf:Person

ex:geographicallyPartOf

ex:CapitalCity

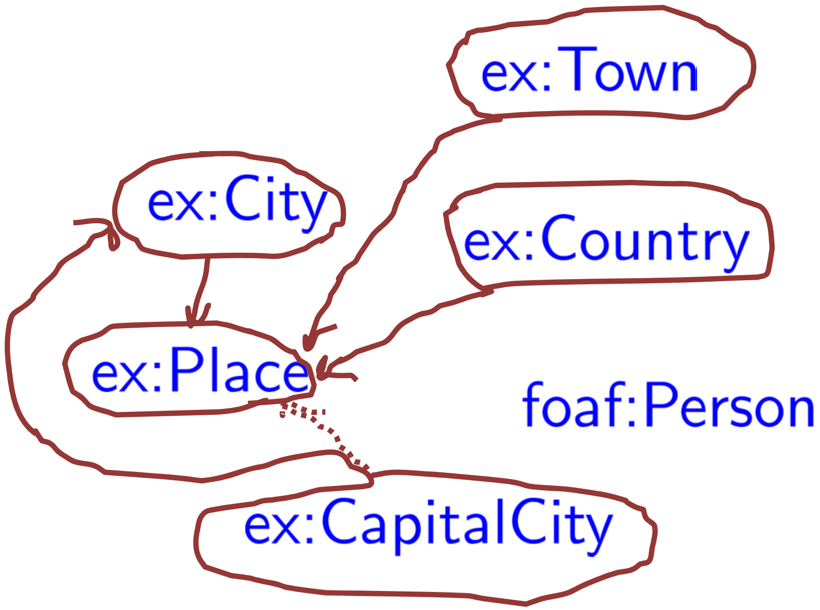
ex:partOf

Class hierarchy

- Class *c1* is a sub-class of Class *c2*
  - If (`x`,`rdf:type`,*c1*) then (`x`,`rdf:type`,*c2*),

Example: if `ex:CapitalCity` *sub-class of* `ex:City`
and if `(ex:Dublin,rdf:type,ex:CapitalCity)`
then `(ex:Dublin,rdf:type,ex:City)`

Which classes would be sub-classes of each other?

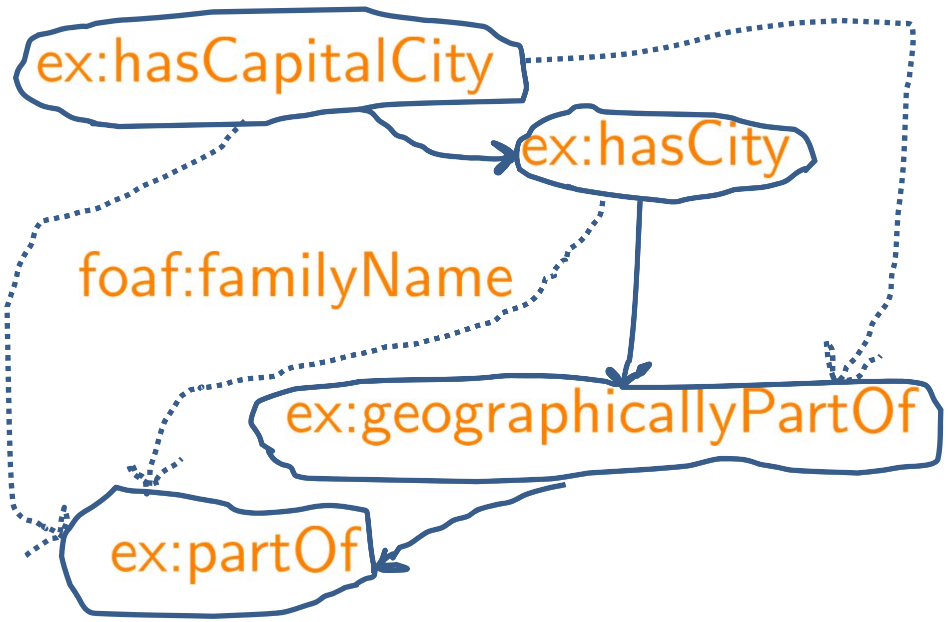


Property hierarchy

- Property $p1$ is a sub-property of $p2$
  - If `(x,`$p1$`,y)` then `(x,`$p2$`,y)`

Example: if `ex:hasCapitalCity` *sub-property of* `ex:hasCity`
and if `(ex:Ireland,ex:hasCapitalCity,ex:Dublin)`
then `(ex:Ireland,ex:hasCity,ex:Dublin)`

Which properties would be sub-properties of each other?



Domain of properties

- Property p has domain class c
  - If (x,p,y) then (x,rdf:type,c)

Example: if `foaf:familyName` *has domain* `foaf:Person`
and if (`ex:Aidan,foaf:familyName,"Hogan"`)
then (`ex:Aidan,rdf:type,foaf:Person`)

Which properties would have which classes as domain?

ex:Town

ex:hasCapitalCity

ex:hasCity

ex:City

ex:Country

foaf:familyName

ex:Place

foaf:Person

ex:geographicallyPartOf

ex:CapitalCity

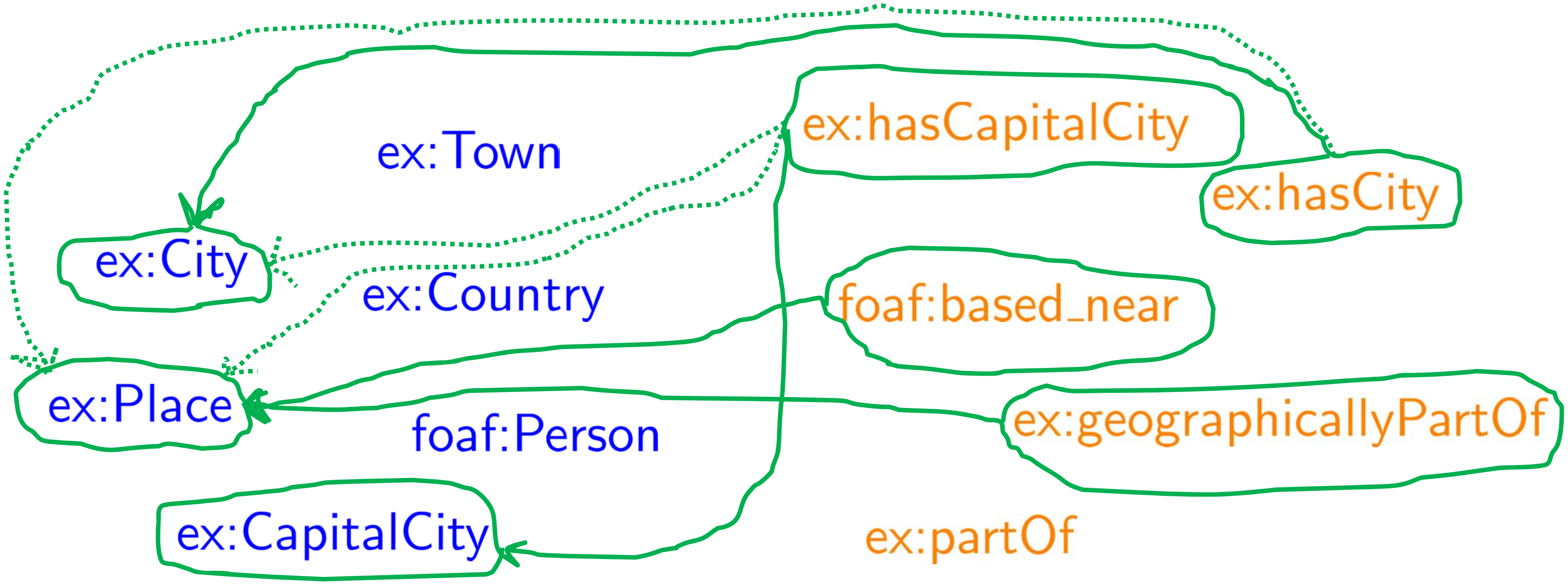
ex:partOf

Range of properties

- Property *p* has range class *c*
  - If `(x,p,y)` then `(y,rdf:type,c)`

Example: if `ex:hasCity` *has range* `ex:City`
and if `(ex:Ireland,ex:hasCity,ex:Dublin)`
then `(ex:Dublin,rdf:type,ex:City)`

Which properties would have which classes as range?



Trade-off: More Specific, Less Reusable

- More specific, more conclusions!
- Less specific, more reusable!

- Example:
  - `ex:hasCapitalCity` has domain `ex:Country`

PRO: Know that anything that has a capital city is a country

CON: Cannot use this property for capital cities of states/counties/regions, etc.

Trade-off: More Specific, Less Reusable

- Another example:
  - `ex:Mayor` sub-class of `foaf:Person`

Bosco the dog
Mayor of Sunol, California
1981–1994
R.I.P.

Beware of "hidden" definitions!

FOAF Vocabulary Specification 0.99

Namespace Document 14 January 2014 - *Paddington Edition*

Property: foaf:img

image - An image that can be used to represent some thing (ie. those depictions which are particularly representative of something, eg. one's photo on a homepage).

Status: testing

Domain: having this property implies being a Person

Range: every value of this property is a Image

Do you see any potential problems here?

`(ex:Dublin,foaf:img,ex:Dublin_night.jpg)`

- Choose names of properties/classes carefully!

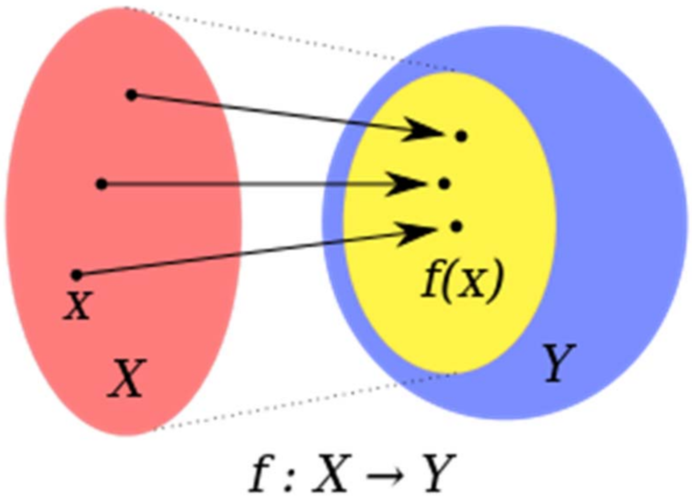
RDFS: RDF SCHEMA

RDFS: Describe “schema” in RDF

- Sub-class:
  - `ex:CapitalCity rdfs:subClassOf ex:City .`
- Sub-property:
  - `ex:hasCapitalCity rdfs:subPropertyOf ex:hasCity .`
- Domain:
  - `foaf:familyName rdfs:domain foaf:Person .`
- Range:
  - `ex:hasCapitalCity rdfs:range ex:CapitalCity .`
  - `foaf:familyName rdfs:range xsd:string .`

Note: Why called "domain" and "range"?

Any guesses why RDFS calls these "domain" and "range"

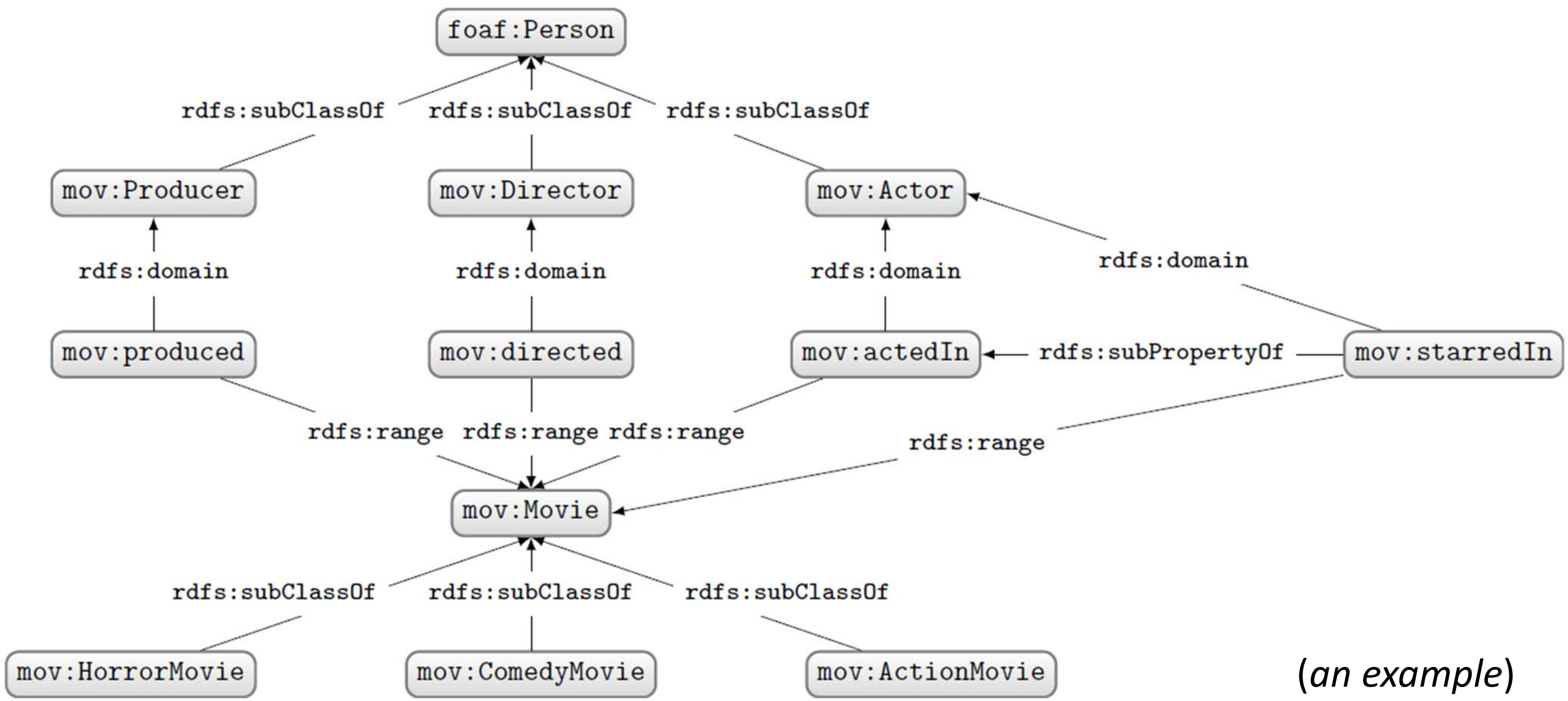


$f : X \longrightarrow Y$

- X: **domain** of the function
- Y: co-domain of the function
- $\{f(x) \mid x \in X\}$: image or **range** of the function

So let's build an RDF Schema …

Let's model an RDF Schema for Movies, including different types of movies, some different types of people involved, and how they are related.



(*an example*)

But what, e.g., is the domain of … ?

But what, e.g., is the domain of ... ?

ex:partOf $\xrightarrow{\text{rdfs:domain}}$ rdfs:Resource

- rdfs:Resource the class of everything!
  - Yes, even itself!
    - `(rdfs:Resource,rdf:type,rdfs:Resource)`

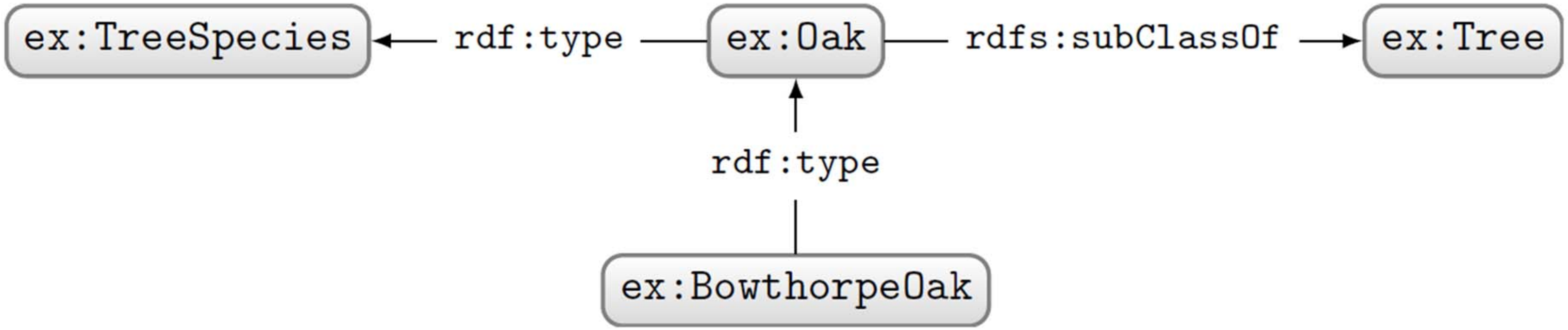
(Giving domain/range/sub-class rdfs:Resource says nothing new!)

Some meta-classes ...

- rdf:Property: class of all properties
  - `(ex:hasCity,rdf:type,rdf:Property)`
- rdfs:Class: class of all classes
  - `(ex:City,rdf:type,rdfs:Class)`

Note: Class or instance?

Would you define ex:Oak (“roble”@es)
as a class or an instance?



No clear distinction between classes and instances!

REASONING WITH RDFS

What is “Reasoning”?

- Three types of reasoning:
  1. Inductive reasoning (learning)

     (learn fuzzy rule from premises and conclusions)

     “The first three Sharknado movies were great, so the fourth will probably be great too.”
  2. Deductive reasoning (logic)

     (make logical conclusion from given rule and premise)

     “All movie directors are human. Anthony C. Ferrante is a movie director, therefore he/she/it must be human.”
  3. Abductive reasoning (learning/logic)

     (guess the likely premise from a given rule and conclusion)

     “All Sharknado movies involve sharks and tornados. Fred saw a movie with sharks and tornados. It was probably a Sharknado movie.”

Reasoning in RDFS is deductive

- Three types of reasoning:
  1. Inductive reasoning (learning)

     (learn fuzzy rule from premises and conclusions)

     "The first three Sharknado movies were great, so the fourth will probably be great too."

  2. **Deductive reasoning (logic)**

     (make logical conclusion from given rule and premise)

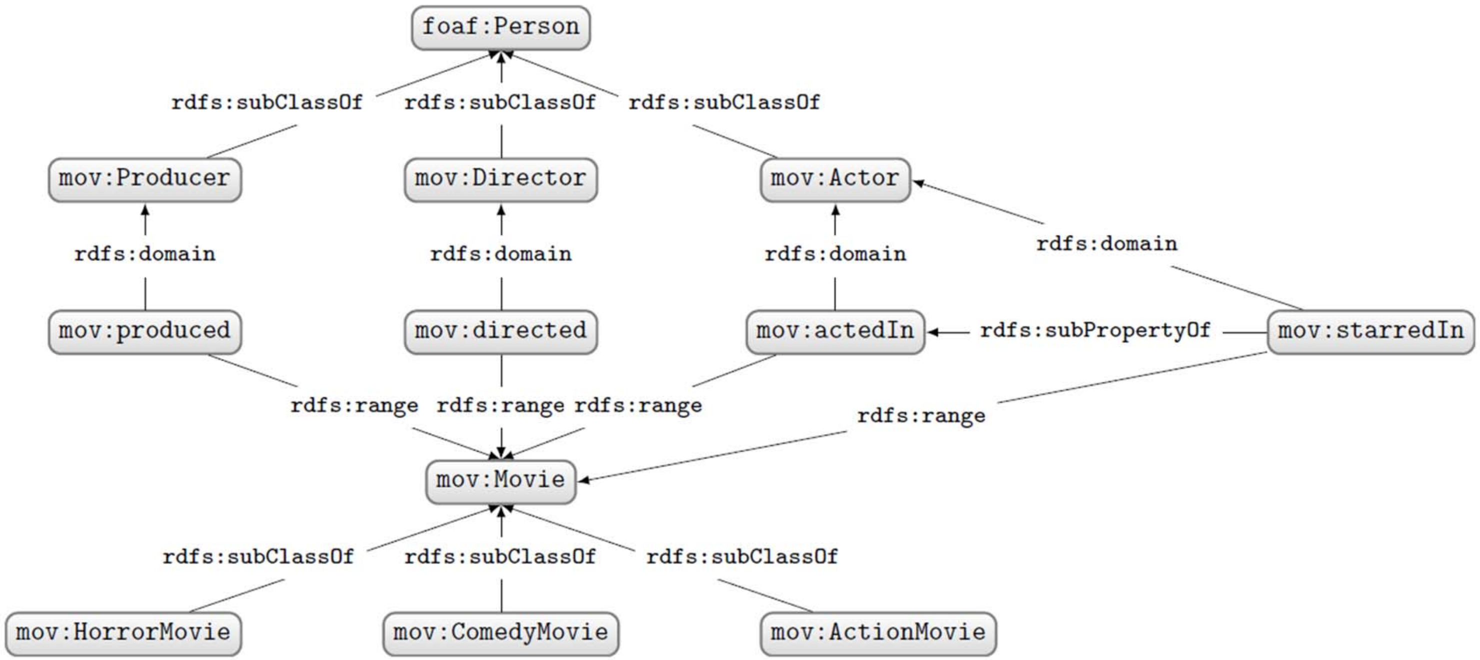
     "All movie directors are human. Anthony C. Ferrante is a movie director, therefore he/she/it must be human."

  3. Abductive reasoning (learning/logic)

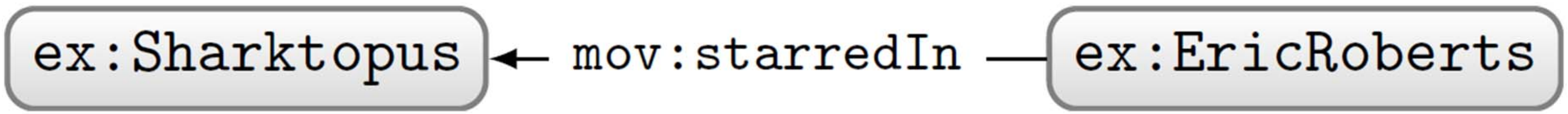
     (guess the likely premise from a given rule and conclusion)

     "All Sharknado movies involve sharks and tornados. Fred saw a movie with sharks and tornados. It was probably a Sharknado movie."

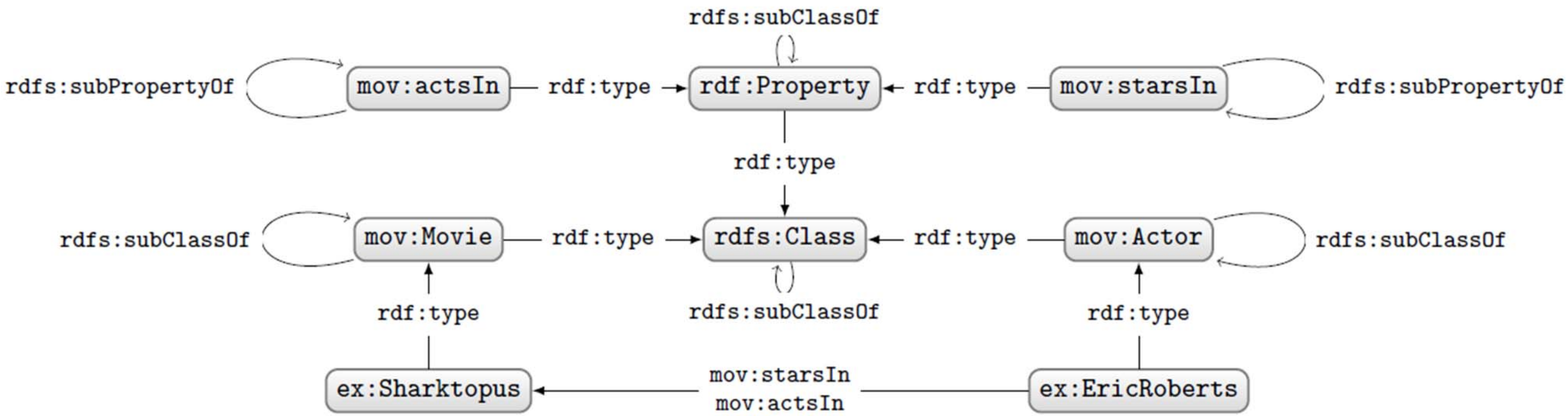
What conclusions can we deduce?



Given the above schema data, what can we deduce from:

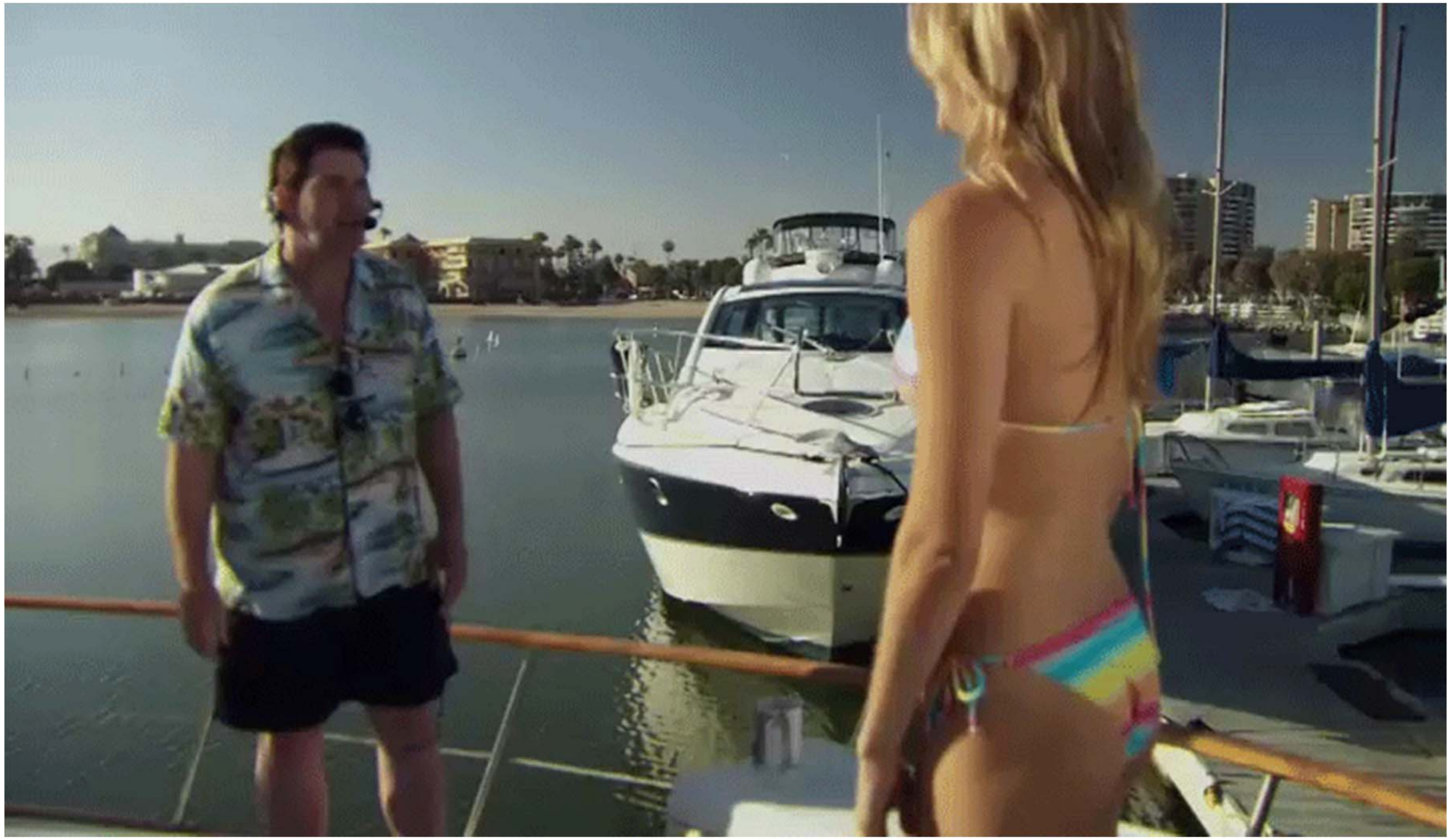


Some of the conclusions …



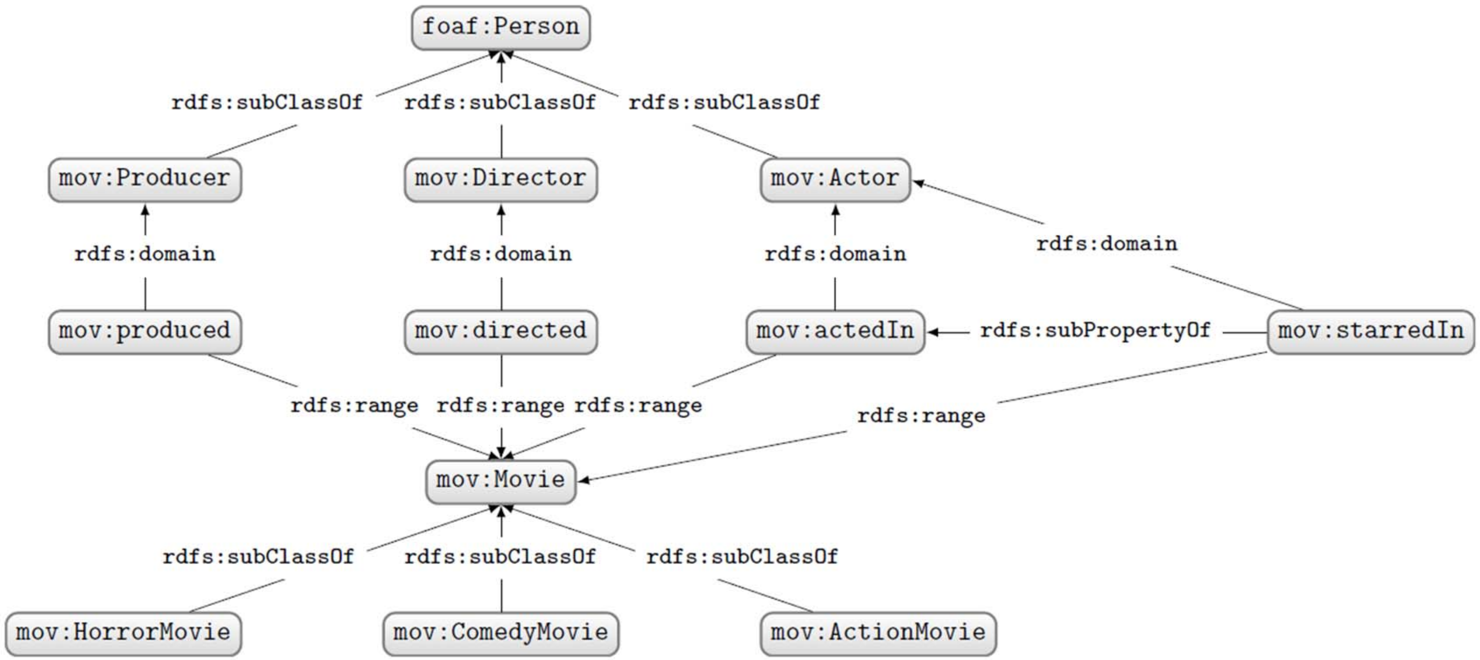
- Not shown (for sake of my/our sanity):
  - Everything of type rdfs:Resource
  - All classes sub-class of rdfs:Resource
  - RDF/RDFS properties of type rdf:Property
  - …
- A lot more data to answer queries with

Just in case you're curious ...

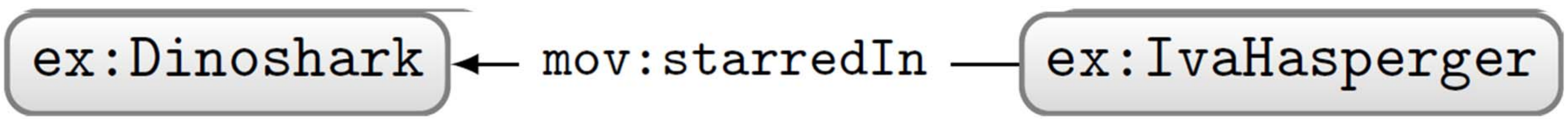


ex:Sharktopus ← mov:starredIn — ex:EricRoberts

Note: RDFS definitions apply to any movie!



Given the above schema data, what can we deduce from:



Apply RDFS reasoning using “rules”

ID	if G matches	then G RDFS$_D$-entails
rdfD1	?x ?p ?l . (?l a literal with datatype IRI dt(?l) $\in D$)	?x ?p _:b . _:b a dt(?l) .
rdfD2	?x ?p ?y .	?p a rdf:Property .
rdfs1	?u $\in D$	?u a rdfs:Datatype .
rdfs2	?p rdfs:domain ?c . ?x ?p ?y .	?x a ?c .
rdfs3	?p rdfs:range ?c . ?x ?p ?y .	?y a ?c .
rdfs4a	?x ?p ?y .	?x a rdfs:Resource .
rdfs4b	?x ?p ?y .	?y a rdfs:Resource .
rdfs5	?p rdfs:subPropertyOf ?q . ?x ?p ?y .	?x ?q ?y .
rdfs6	?p a rdf:Property .	?p rdfs:subPropertyOf ?p .
rdfs7	?p rdfs:subPropertyOf ?q . ?q rdfs:subPropertyOf ?r .	?p rdfs:subPropertyOf ?r .
rdfs8	?c a rdfs:Class .	?c rdfs:subClassOf rdfs:Resource .
rdfs9	?c rdfs:subClassOf ?d . ?x a ?c .	?x a ?d .
rdfs10	?c a rdfs:Class .	?c rdfs:subClassOf ?c .
rdfs11	?c rdfs:subClassOf ?d . ?d rdfs:subClassOf ?e .	?c rdfs:subClassOf ?e .
rdfs12	?p a rdfs:ContainerMembershipProperty .	?p rdfs:subPropertyOf rdfs:member .
rdfs13	?d a rdfs:Datatype .	?d rdfs:subClassOf rdf:Literal .

(Don’t worry about rdfD1, rdfs1, rdfs12, rdfs13)

Don't forget "axiomatic triples"

rdf:type	rdfs:domain	rdfs:Resource	; rdfs:range	rdfs:Class	.
rdfs:domain	rdfs:domain	rdf:Property	; rdfs:range	rdfs:Class	.
rdfs:range	rdfs:domain	rdf:Property	; rdfs:range	rdfs:Class	.
rdfs:subPropertyOf	rdfs:domain	rdf:Property	; rdfs:range	rdf:Property	.
rdfs:subClassOf	rdfs:domain	rdfs:Class	; rdfs:range	rdfs:Class	.
rdf:subject	rdfs:domain	rdf:Statement	; rdfs:range	rdfs:Resource	.
rdf:predicate	rdfs:domain	rdf:Statement	; rdfs:range	rdfs:Resource	.
rdf:object	rdfs:domain	rdf:Statement	; rdfs:range	rdfs:Resource	.
rdfs:member	rdfs:domain	rdfs:Resource	; rdfs:range	rdfs:Resource	.
rdf:first	rdfs:domain	rdf:List	; rdfs:range	rdfs:Resource	.
rdf:rest	rdfs:domain	rdf:List	; rdfs:range	rdfs:List	.
rdfs:seeAlso	rdfs:domain	rdfs:Resource	; rdfs:range	rdfs:Resource	.
rdfs:isDefinedBy	rdfs:domain	rdfs:Resource	; rdfs:range	rdfs:Resource	.
rdfs:comment	rdfs:domain	rdfs:Resource	; rdfs:range	rdfs:Literal	.
rdfs:label	rdfs:domain	rdfs:Resource	; rdfs:range	rdfs:Literal	.
rdf:value	rdfs:domain	rdfs:Resource	; rdfs:range	rdfs:Resource	.
rdf:_n	rdfs:domain	rdfs:Resource	; 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:Datatype	rdfs:subClassOf	rdfs:Class	.

rdfs:isDefinedBy rdfs:subPropertyOf rdfs:seeAlso .

rdf:_n rdf:type rdfs:ContainerMembershipProperty .

Reasoning in RDFS over RDF graph *G*

1. Add axiomatic triples to *G*
2. Apply rules exhaustively, adding conclusions to *G*, until nothing new found

Will this ever finish? Or will it run forever?

RECAP

(1) Data, **(2) Rules/Ontologies**, (3) Query

INPUT: "(x,partOf,y)"

DATA:

http://ex.org/Ireland

Ireland 

(Ireland,partOf,Europe)
(Ireland,a,Country)
(Ireland,capital,Dublin)

http://ex.org/Dublin

Dublin 

(Dublin,population,1000000)

RULES: $(a, \mathsf{capital}, b) \rightarrow (b, \mathsf{partOf}, a)$
$(c, \mathsf{partOf}, d), (d, \mathsf{partOf}, e) \rightarrow (c, \mathsf{partOf}, e)$

OUTPUT: $\{(x \mapsto \mathsf{Ireland}, y \mapsto \mathsf{Europe}), (x \mapsto \mathsf{Dublin}, y \mapsto \mathsf{Ireland})$
$(x \mapsto \mathsf{Dublin}, y \mapsto \mathsf{Europe})\}$

Don’t write rules: write triples

- RDFS: RDF Schema
  - Allows for defining classes and properties
  - Written down in triples
  - Can be used for reasoning with rules

RDFS: RDF Schema

- RDFS has four <u>main</u> features:
  - sub-class: an instance of a class *c1* is an instance of its sub-class *c2*
  - sub-property: if two things are related by *p1*, they are also related by its sub-property *p2*
  - domain: the subject of a relation with property *p* is the type of its domain *c*
  - range: the object of a relation with property *p* is the type of its range *c*
- A few other useful classes:
  - rdf:Property: the class of all properties
  - rdfs:Class: the class of all classes
  - rdfs:Resource: the class of everything!

RDFS: A Web Standard

http://www.w3.org/TR/rdf-schema/

W3C Recommendation

RDF Schema 1.1

W3C Recommendation 25 February 2014

This version:
    http://www.w3.org/TR/2014/REC-rdf-schema-20140225/
Latest published version:
    http://www.w3.org/TR/rdf-schema/
Previous version:
    http://www.w3.org/TR/2014/PER-rdf-schema-20140109/
Editors:
    Dan Brickley, Google
    R.V. Guha, Google
Previous Editors:
    Brian McBride

Please check the **errata** for any errors or issues reported since publication.

This document is also available in this non-normative format: diff w.r.t. 2004 Recommendation

Questions?