

CC7220-1

LA WEB DE DATOS

PRIMAVERA 2023

LECTURE 3: RDF SCHEMA (RDFS) AND SEMANTICS


Aidan Hogan

aidhog@gmail.com

LAST TIME ...

SEMANTIC WEB: DATA

DATA:

Ireland 

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

Dublin 

(Ireland,capital,Dublin)
(Dublin,population,1000000)

LOGIC: $“(b, \text{capital}, a) \rightarrow (a, \text{partOf}, b)”$
 $“(a, \text{partOf}, b), (b, \text{partOf}, c) \rightarrow (a, \text{partOf}, c)”$

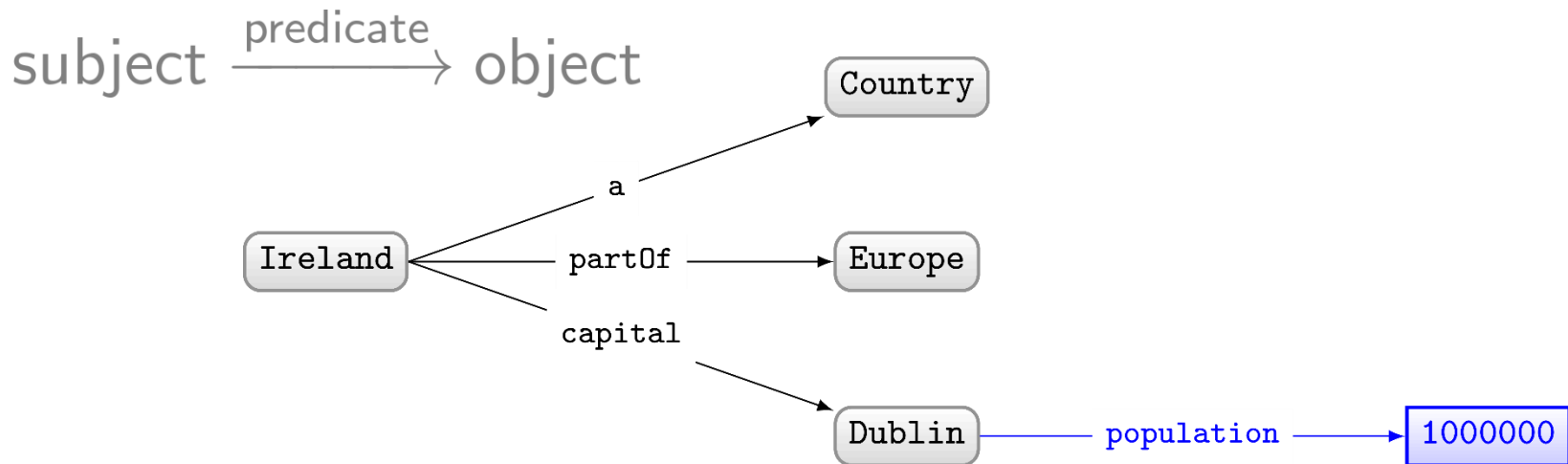
QUERY: $“(x, \text{partOf}, y)”$

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



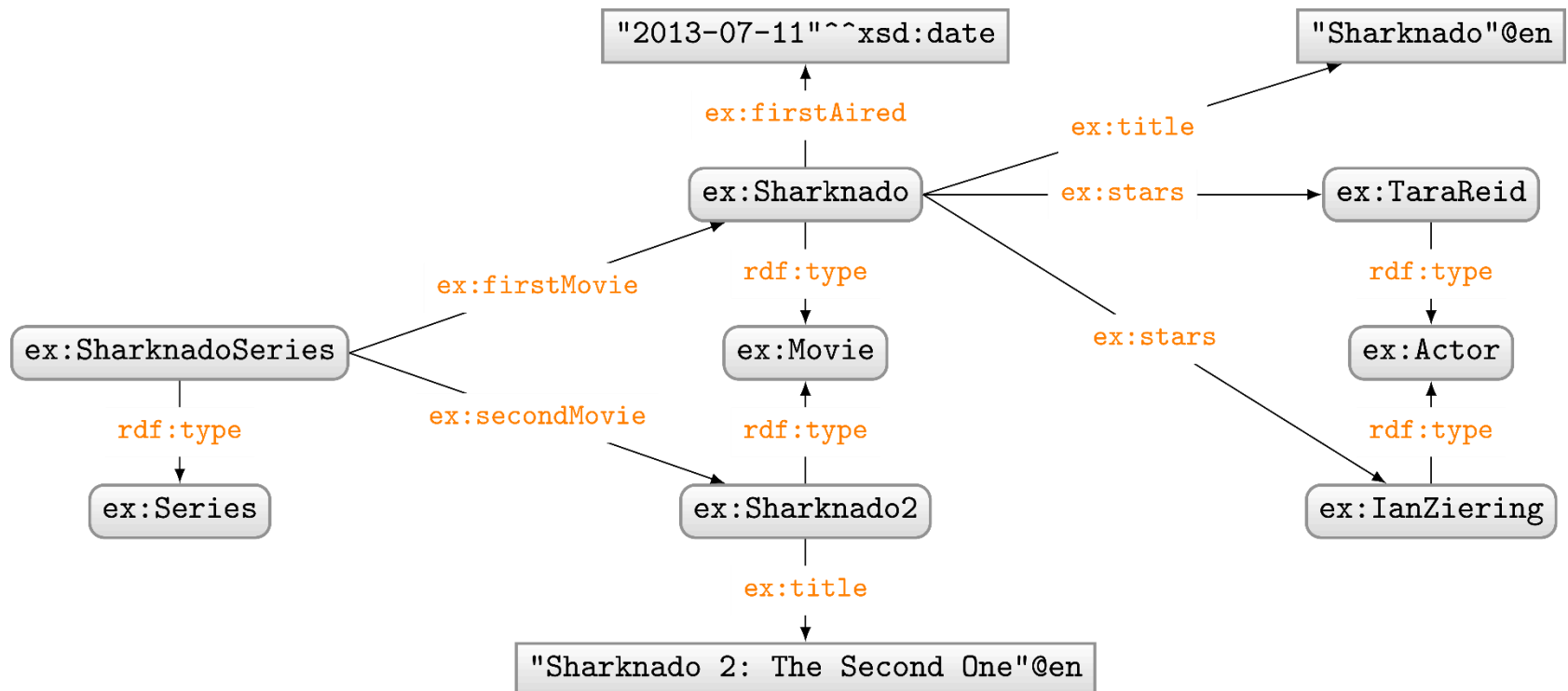
RDF OFTEN DRAWN AS A (DIRECTED, LABELLED) GRAPH

<i>subject</i>	<i>predicate</i>	<i>object</i>
Ireland	partOf	Europe
Ireland	a	Country
Ireland	capital	Dublin
Dublin	population	1,000,000



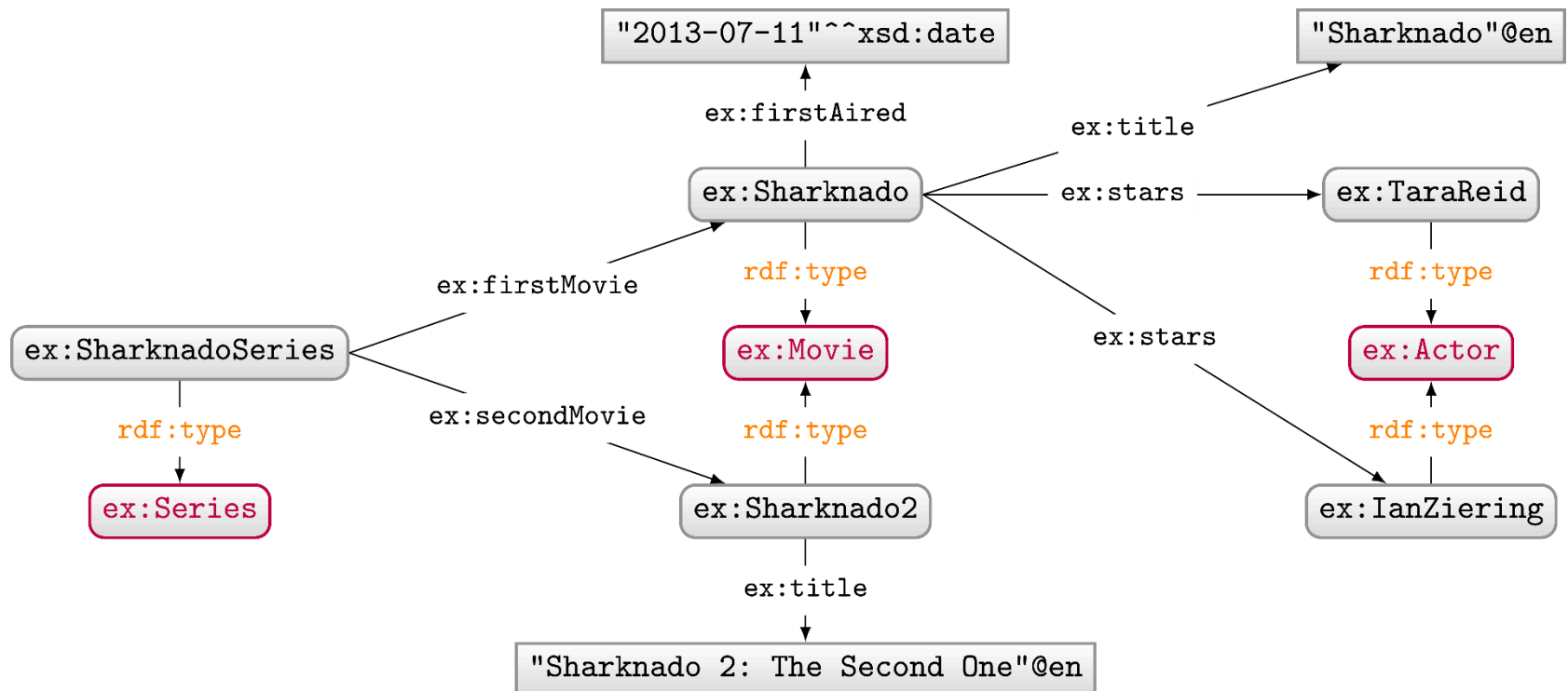
RDF PROPERTIES

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



RDF CLASSES


- Used to conceptually group resources
 - `ex:Movie`, `ex:Actor`, `ex:Series`, etc.
 - Uses predicate `rdf:type` to type a resource



TODAY'S TOPIC ...

SEMANTIC WEB: LOGIC

DATA:

Ireland 

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

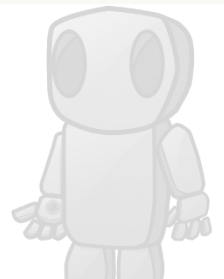
Dublin 

(Ireland,capital,Dublin)
 (Dublin,population,1000000)

LOGIC: $“(b, \text{capital}, a) \rightarrow (a, \text{partOf}, b)”$
 $“(a, \text{partOf}, b), (b, \text{partOf}, c) \rightarrow (a, \text{partOf}, c)”$

QUERY: $“(x, \text{partOf}, y)”$

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 CAPTURE LOGIC?

How should we capture logic on the Semantic Web?

LOGIC: “(b ,capital, a) \rightarrow (a ,partOf, b)”
 “(a ,partOf, b), (b ,partOf, c) \rightarrow (a ,partOf, c)”

SEMANTIC WEB ANSWER: SCHEMA/ONTOLOGIES

- Instead of rules, we can use RDF!
- Define relationships between classes and properties

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

Classes (in blue):
ex:Town
ex:City
ex:Country
ex:Place
foaf:Person
ex:CapitalCity

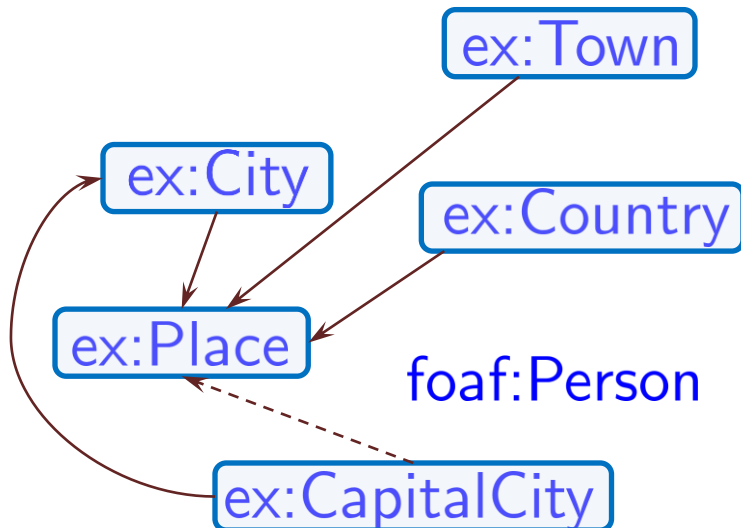
Properties (in orange):
ex:hasCapitalCity
ex:hasPart
foaf:familyName
ex:hasCity
ex:containsPlace

CLASS HIERARCHY

- Class **c** is a **sub-class** of Class **d**
 - If $(x, \text{rdf:type}, c)$ then $(x, \text{rdf:type}, d)$,

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

Which classes would be **sub-classes** of each other?

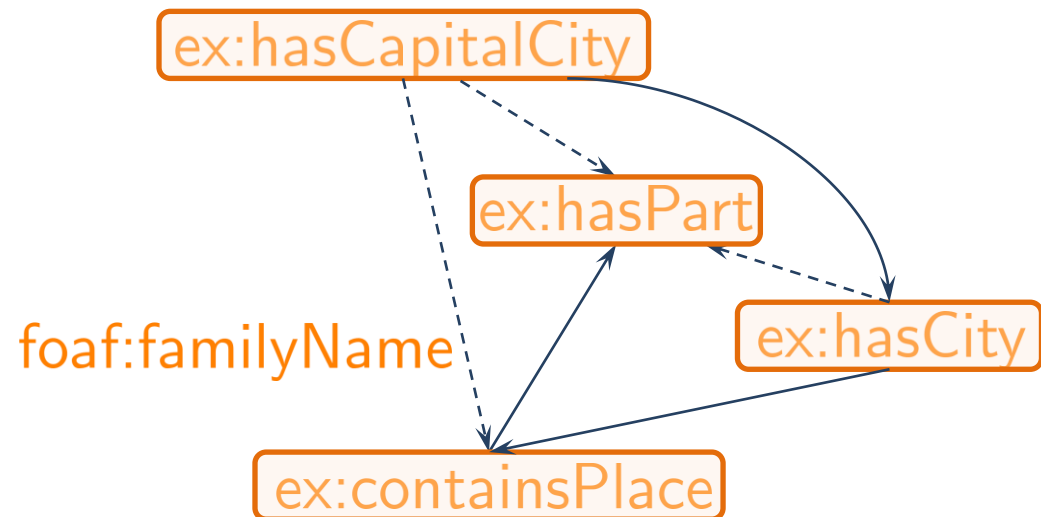


PROPERTY HIERARCHY

- Property **p** is a sub-property of **q**
 - If (x, p, y) then (x, q, y)

*Example: if `ex:hasCapitalCity` sub-property of `ex:hasCity`
and if $(\text{ex:Ireland}, \text{ex:hasCapitalCity}, \text{ex:Dublin})$
then $(\text{ex:Ireland}, \text{ex:hasCity}, \text{ex:Dublin})$*

Which properties would be sub-properties of each other?

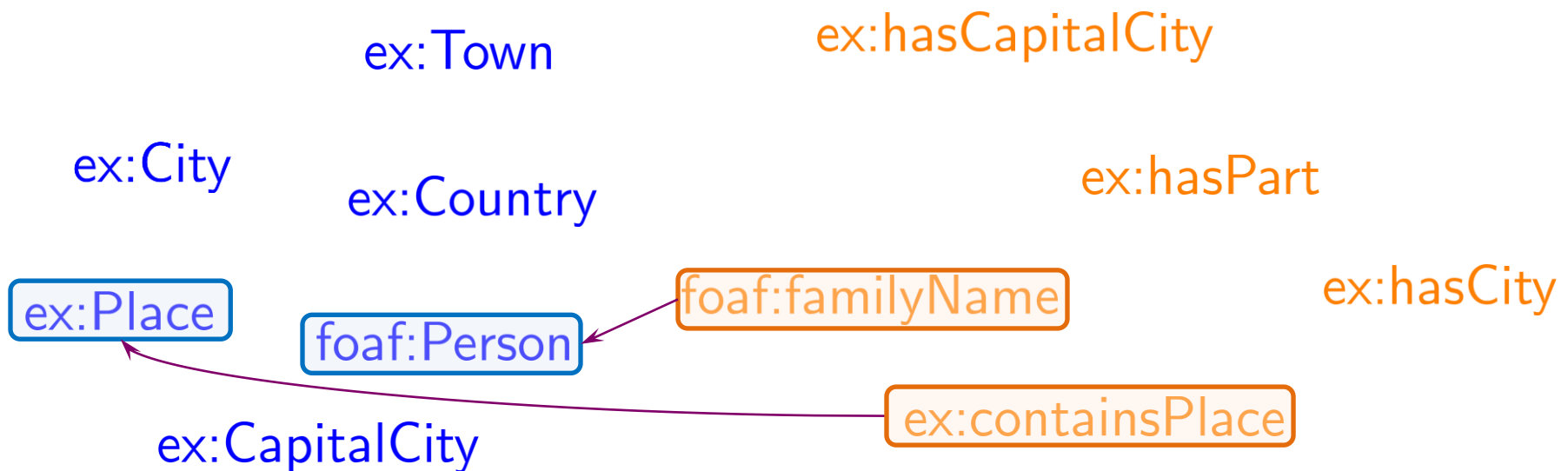


DOMAIN OF PROPERTIES

- Property p has domain class c
 - If (x, p, y) then $(x, \text{rdf:type}, c)$

*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?

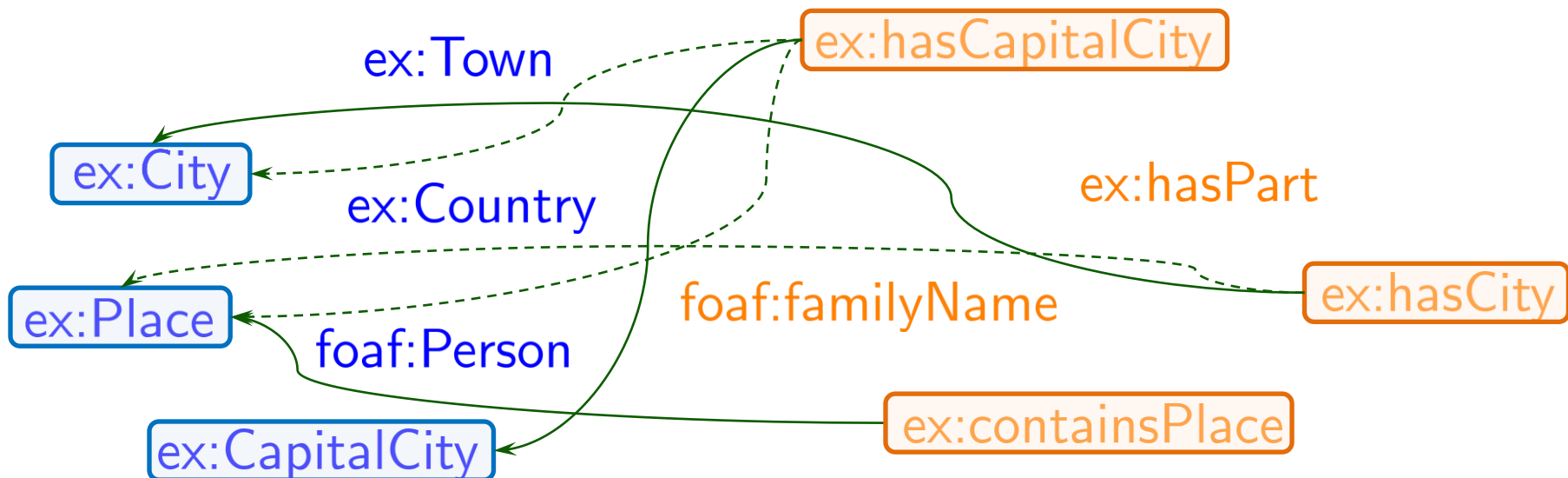


RANGE OF PROPERTIES

- Property p has range class c
 - If (x, p, y) then $(y, \text{rdf:type}, c)$

*Example: if ex:hasCity has range ex:City
and if $(\text{ex:Ireland}, \text{ex:hasCity}, \text{ex:Dublin})$
then $(\text{ex:Dublin}, \text{rdf:type}, \text{ex:City})$*

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 for capitals of states, 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.

TRADE-OFF: MORE SPECIFIC / LESS REUSABLE

- Another example:
 - ex:spouse has **domain/range** foaf:Person



Erika Eiffel

Married Eiffel Tower in 2007

Item Discussion

Erika Eiffel (Q509934)

American archer

spouse Eiffel tower

start time 2007

► 1 reference

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](#)

Any potential problems here?

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

Choose names of properties/classes carefully!

RDFS: RDF SCHEMA

RDFS (1.1): A WEB STANDARD

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



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](#)

RDFS: DESCRIBE “SCHEMA” IN RDF

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

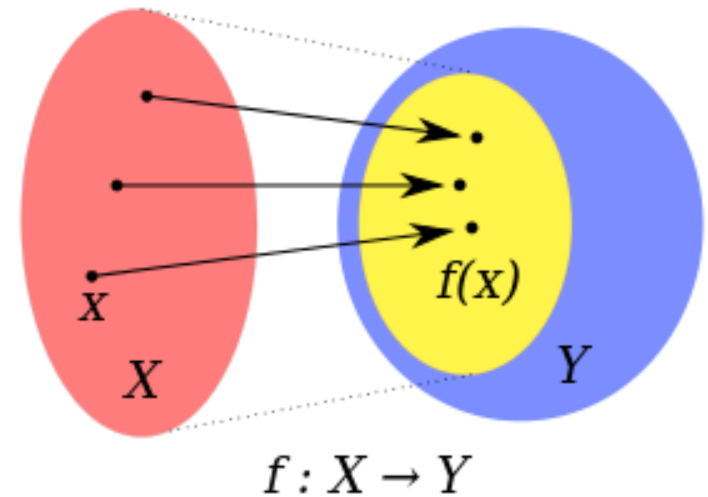
- Sub-class
 - `ex:CapitalCity rdfs:subClassOf ex:City .`
- Sub-property
 - `ex:hasCapitalCity rdfs:subPropertyOf ex:hasCity .`
- Domain
 - `foaf:familyName rdfs:domain foaf:Person .`
- Range
 - `ex:hasCapitalCity rdfs:range ex:CapitalCity .`
 - `foaf:familyName rdfs:range xsd:string .`

NOTE: WHY CALLED “DOMAIN” AND “RANGE”?

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

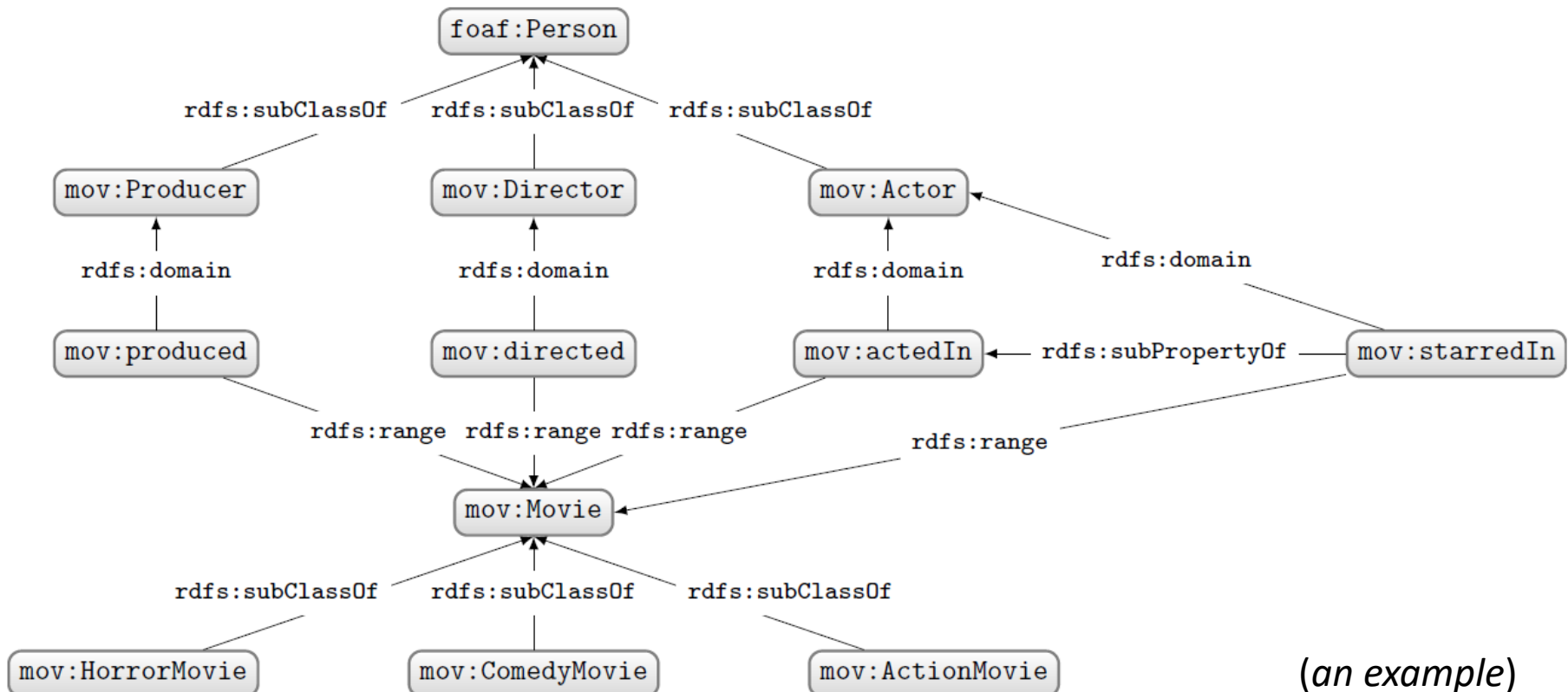
$$f: X \rightarrow Y$$

- 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 (horror, comedy, action), some different types of people involved (actor, producer, director), and how they are related.



(an example)

BUT WHAT, E.G., IS THE DOMAIN OF ... ?



BUT WHAT, E.G., IS THE DOMAIN OF ... ?



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

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

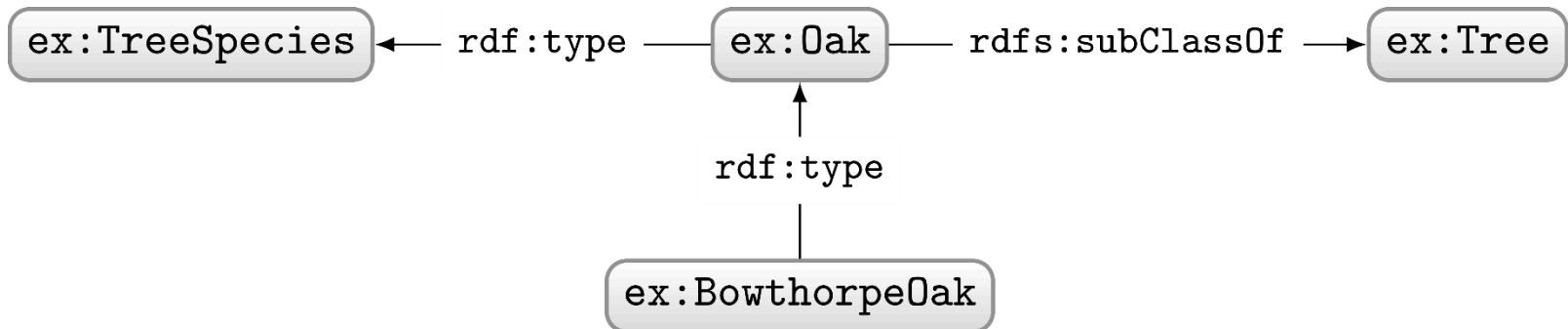
SOME META-CLASSES ...

- **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?



Classes can also "act" as instances: no strong distinction

`rdf:type` is akin to \in , `rdfs:subClassOf` is akin to \subseteq *

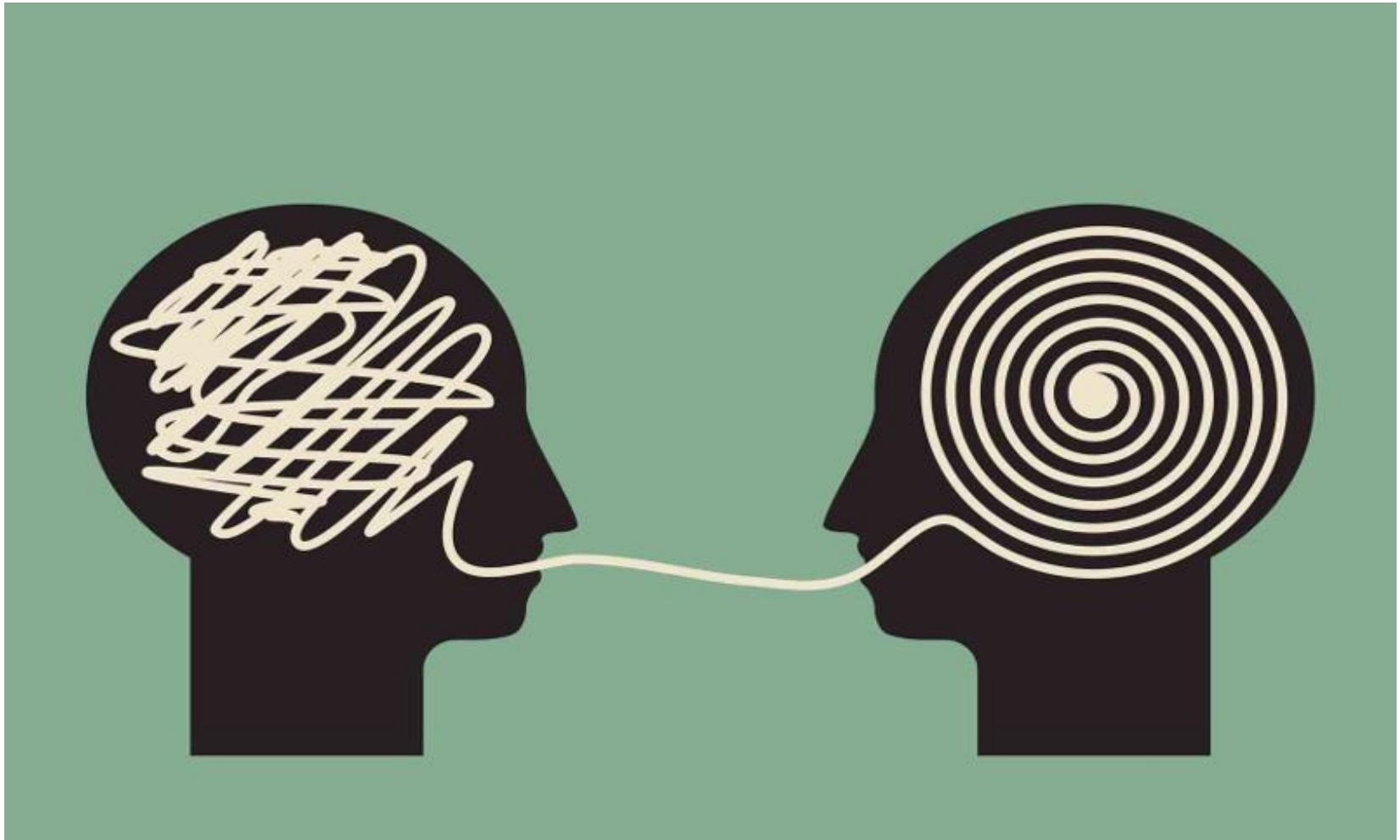
Which is transitive: `rdf:type` or `rdfs:subClassOf`?

`rdfs:subClassOf`

* Slight but useful simplification for now as classes are not quite sets; we will return to this topic later.

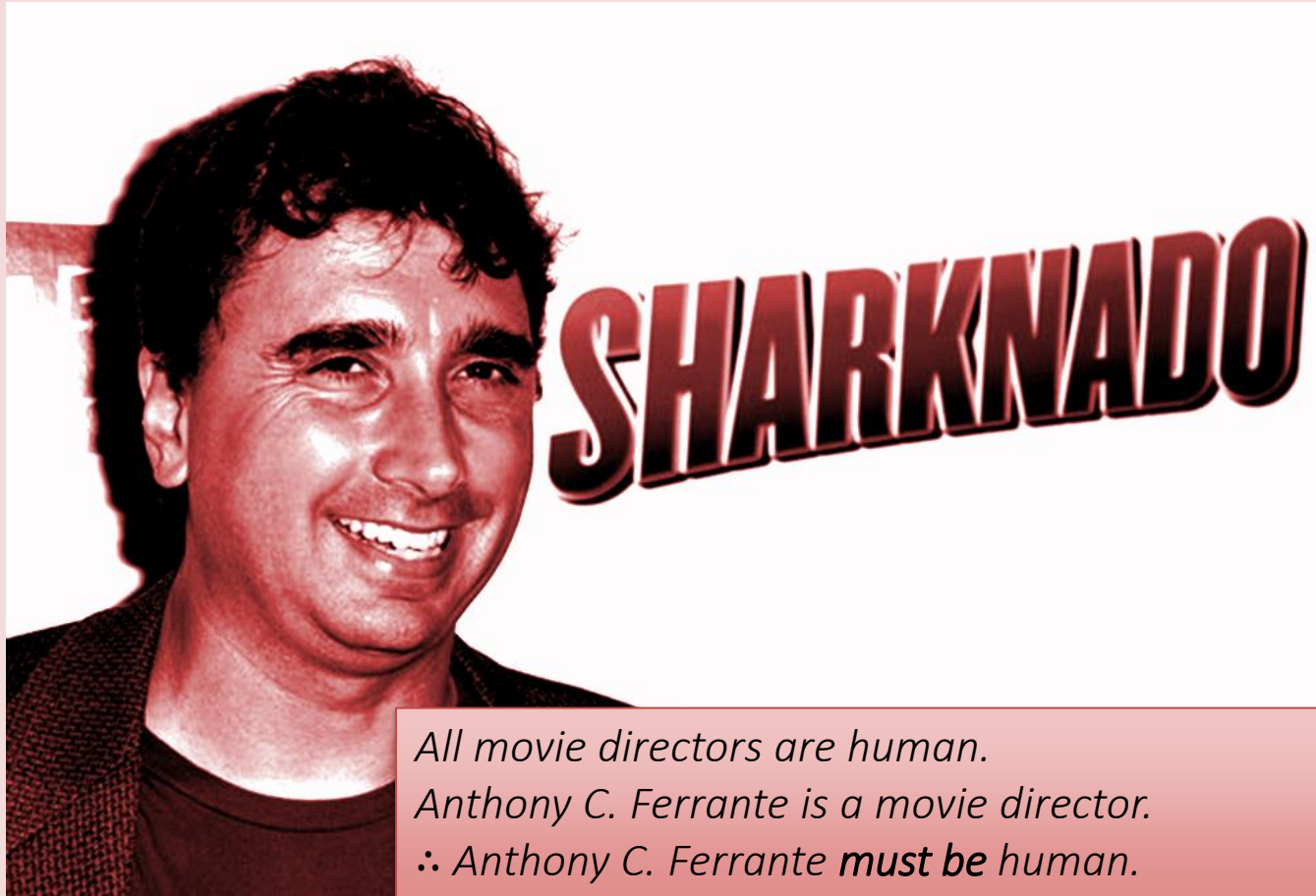
REASONING WITH RDFS

WHAT IS “REASONING”?



What general kinds of logical reasoning can we consider?

WHAT IS “REASONING”?



*All movie directors are human.
Anthony C. Ferrante is a movie director.
∴ Anthony C. Ferrante **must be** human.*

Deductive Reasoning: Make logical conclusion from rules/premises

WHAT IS “REASONING”?



Inductive Reasoning: Learn approximate rule(s) from premises

WHAT IS “REASONING”?



*Fred saw a movie with sharks in tornados.
Sharknado movies have sharks in tornados.
∴ Fred **may have** seen a Sharknado movie.*

Abductive Reasoning: Guess a premise/explanation

REASONING: SUMMARY

- (1) If $(x, \text{rdf:type}, \text{ex:SharknadoMovie})$ then $(x, \text{ex:depicts}, \text{ex:SharksInTornados})$
- (2) $(\text{ex:ItsAboutTime}, \text{rdf:type}, \text{ex:SharknadoMovie})$
- (3) $(\text{ex:ItsAboutTime}, \text{ex:depicts}, \text{ex:SharksInTornados})$

Deductive Reasoning: Given (1,2), conclude (3).

Inductive Reasoning:

???

Abductive Reasoning:

???

REASONING: SUMMARY

- (1) If $(x, \text{rdf:type}, \text{ex:SharknadoMovie})$ then $(x, \text{ex:depicts}, \text{ex:SharksInTornados})$
- (2) $(\text{ex:ItsAboutTime}, \text{rdf:type}, \text{ex:SharknadoMovie})$
- (3) $(\text{ex:ItsAboutTime}, \text{ex:depicts}, \text{ex:SharksInTornados})$

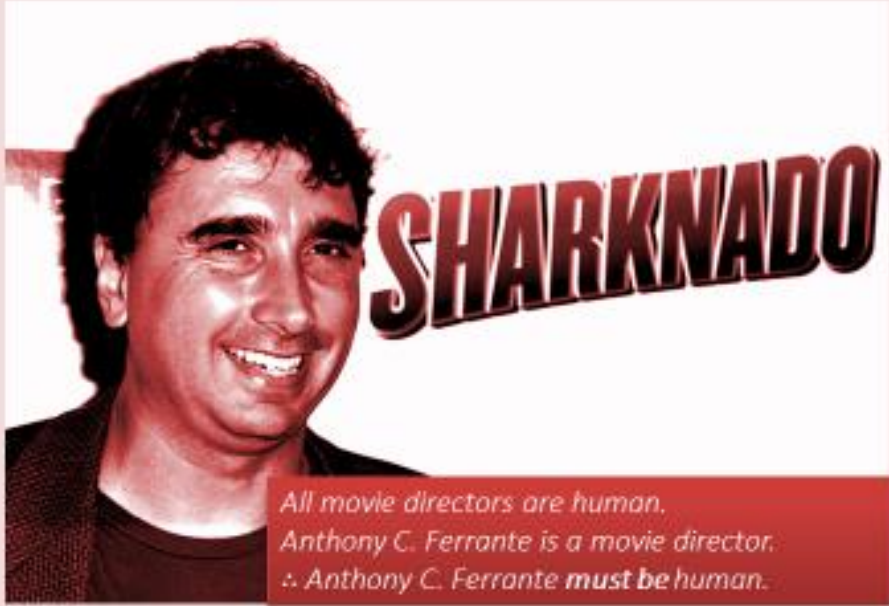
Deductive Reasoning: Given (1,2), conclude (3).

Inductive Reasoning: Given (2,3) (and similar such examples), propose (1).

Abductive Reasoning: Given (1,3), propose (2).

RDFS REASONING IS DEDUCTIVE ...

WHAT IS "REASONING"?

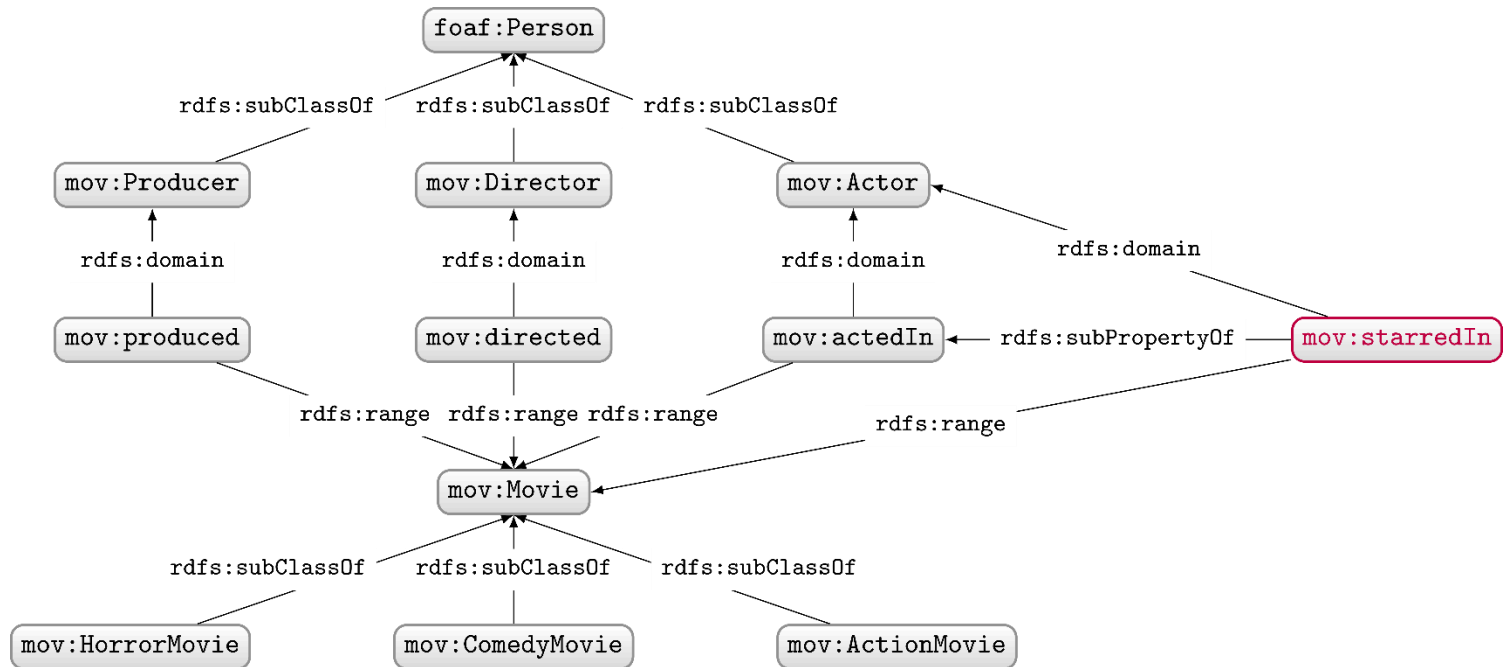


*All movie directors are human.
Anthony C. Ferrante is a movie director.
∴ Anthony C. Ferrante **must be** human.*

Deductive Reasoning: Make logical conclusion from rules/premises

... THE ONLY FORM OF REASONING THAT IS "CERTAIN"

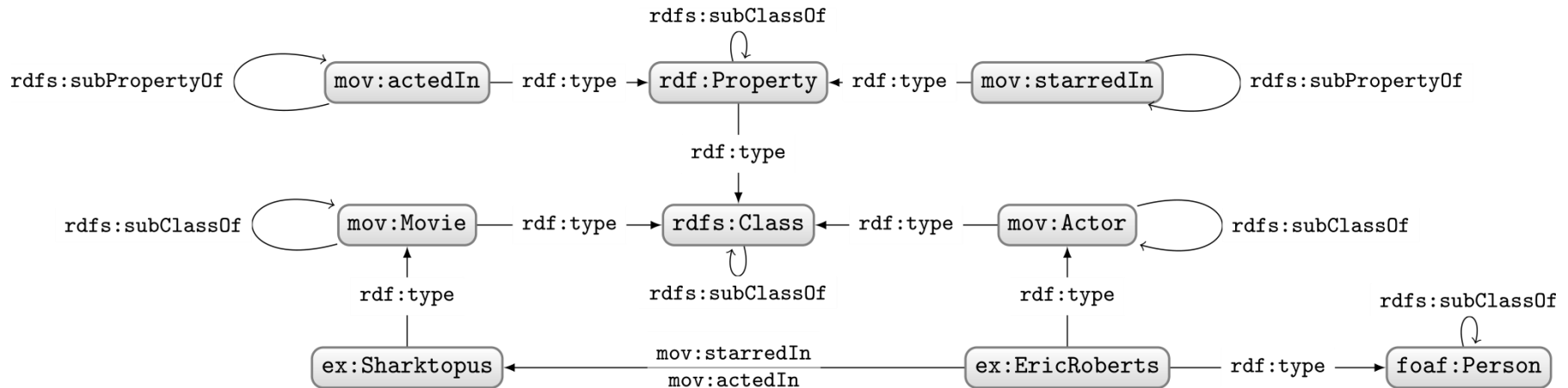
WHAT CONCLUSIONS CAN WE DEDUCE?



Given the above schema, what can we deduce from ...

`ex:EricRoberts` — `mov:starredIn` → `ex:Sharktopus`

SOME OF THE CONCLUSIONS ...



- **Not shown (for the sake of my/our sanity):**
 - Everything is of type **rdfs:Resource**
 - All classes are sub-class of **rdfs:Resource**
 - RDF/RDFS properties are of type **rdf:Property**

SHARKTOPUS JUST ONE MOVIE ...

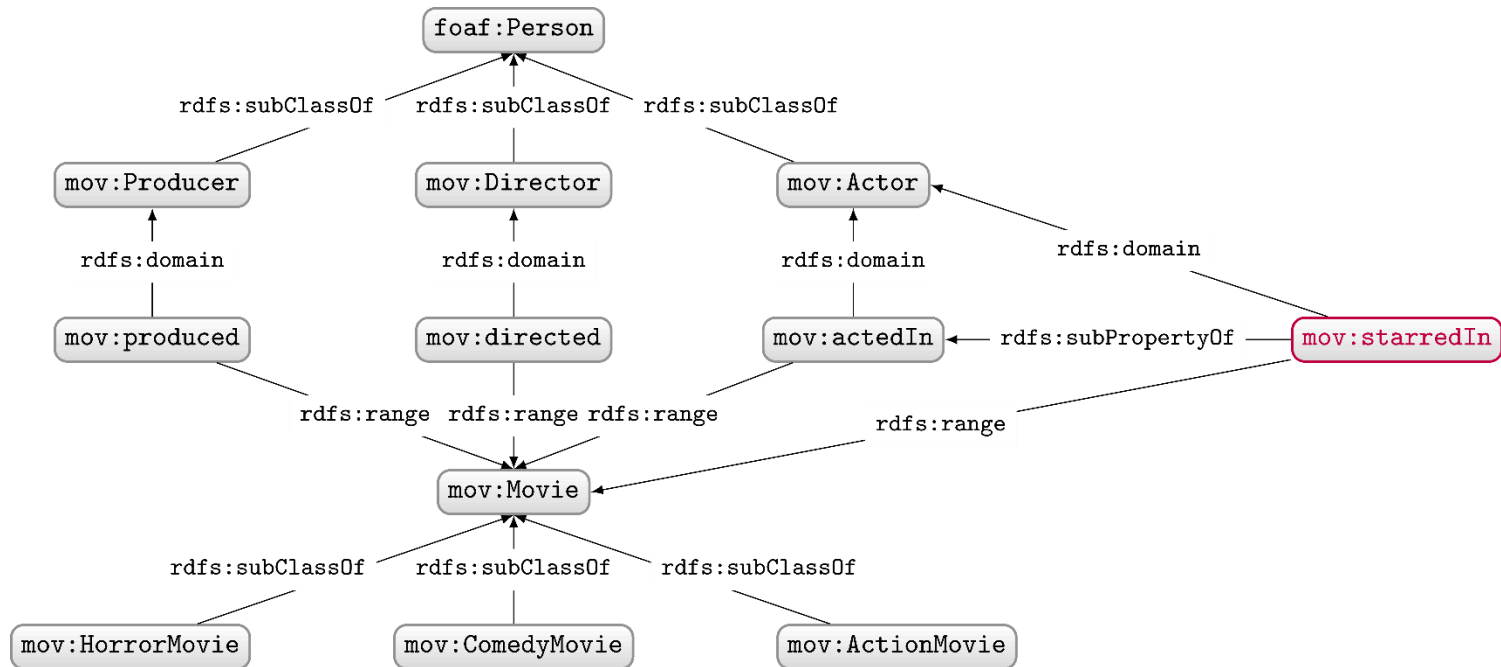


ex:EricRoberts

— mov:starredIn →

ex:Sharktopus

RDFS DEFINITIONS APPLY TO ANY MOVIE ...



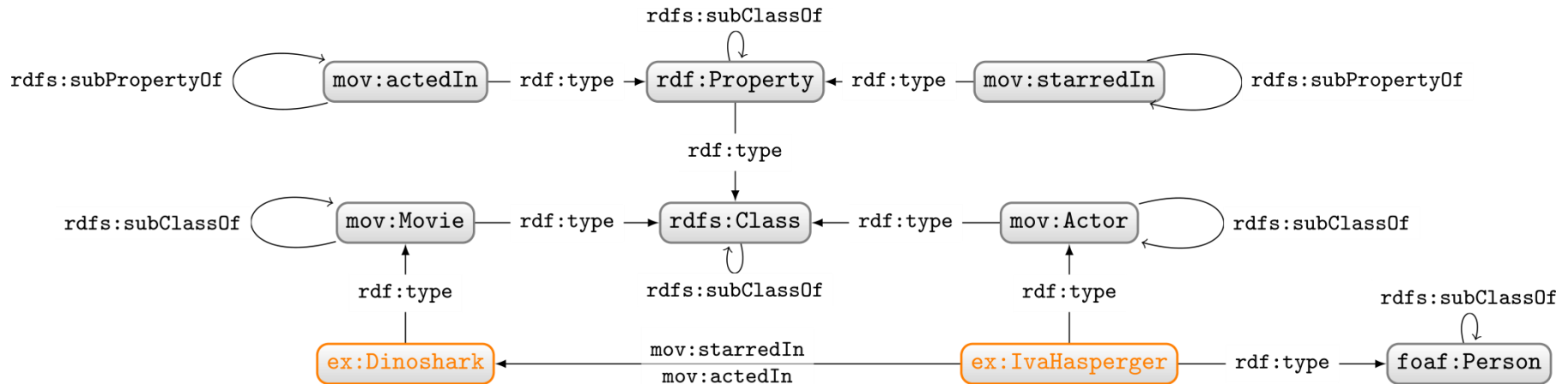
Given the above schema, what can we deduce from ...

`ex:IvaHasperger`

`mov:starredIn`

`ex:Dinoshark`

RDFS DEFINITIONS APPLY TO ANY MOVIE ...



- Not shown (for the sake of my/our sanity):
 - Everything is of type `rdfs:Resource`
 - All classes are sub-class of `rdfs:Resource`
 - RDF/RDFS properties are of type `rdf:Property`

APPLY RDFS REASONING USING “RULES”

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

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

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)

AXIOMATIC TRIPLES: ALWAYS TRUE IN RDFS

```
rdftype          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 .
```

(Don't worry about greyed-out triples)

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 always finish? Or can it run forever?

So long as we do not “invent” new terms, and axiomatic triples are finite, the process must end once G has all possible combinations of terms as triples (or before).

SEMANTIC WEB: LOGIC

DATA:

Ireland 

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

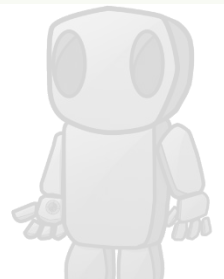
Dublin 

(Ireland,capital,Dublin)
 (Dublin,population,1000000)

LOGIC: $“(b, \text{capital}, a) \rightarrow (a, \text{partOf}, b)”$
 $“(a, \text{partOf}, b), (b, \text{partOf}, c) \rightarrow (a, \text{partOf}, c)”$

QUERY: $“(x, \text{partOf}, y)”$

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



RDFS (1.1): A WEB STANDARD

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



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?

