CC6202-1
La Web de Datos
Primavera 2016

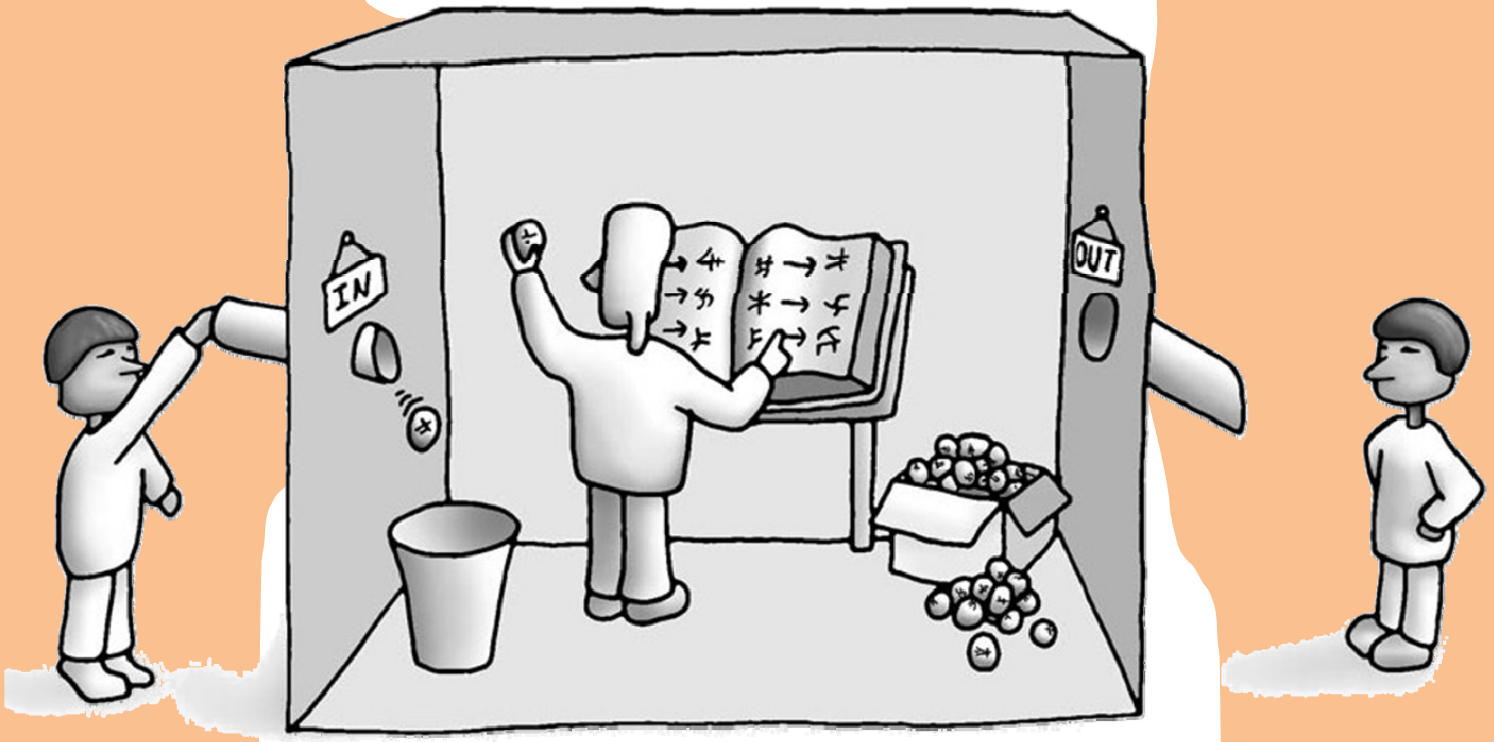
Lecture 2: RDF Model & Syntax

Aidan Hogan
aidhog@gmail.com

LAST TIME …

The “Semantic Web”

What if we could “structure” everything …



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

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

STRUCTURING DATA WITH RDF: RESOURCE DESCRIPTION FRAMEWORK

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

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

DATA:

`http://ex.org/Ireland`

Ireland

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

`http://ex.org/Dublin`

Dublin

(Dublin,population,1000000)

RDF is based on triples:

(Ireland,capital,Dublin)

(subject,predicate,object)

Modelling the world with triples

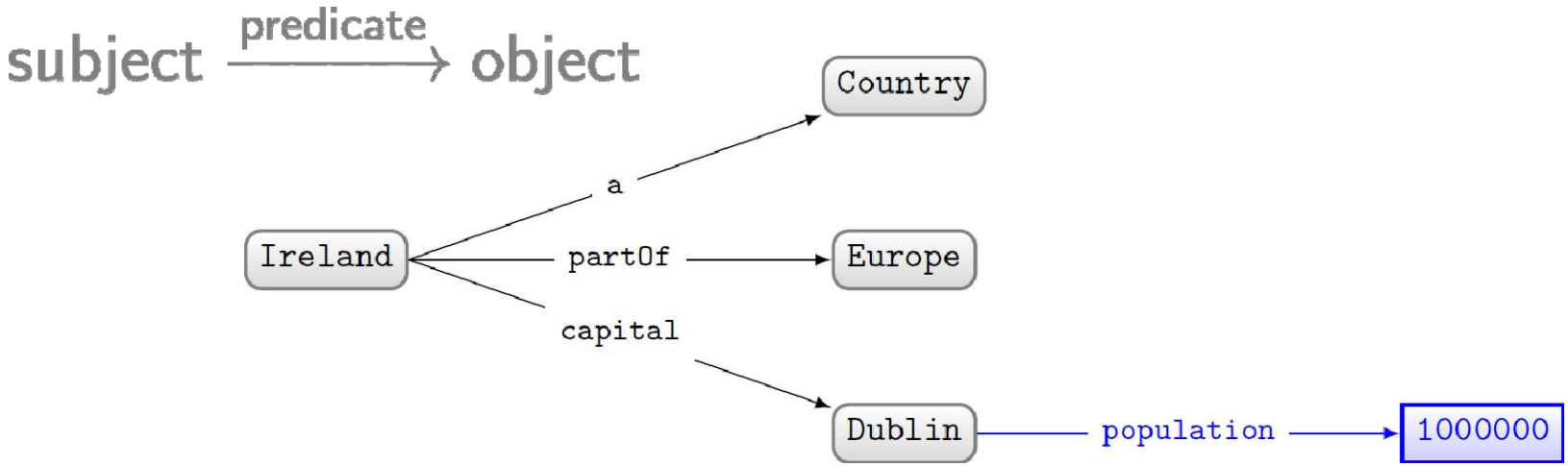
subject	*predicate*	*object*
Ireland	partOf	Europe
Ireland	a	Country
Ireland	capital	Dublin

Concatenate to “integrate” new data

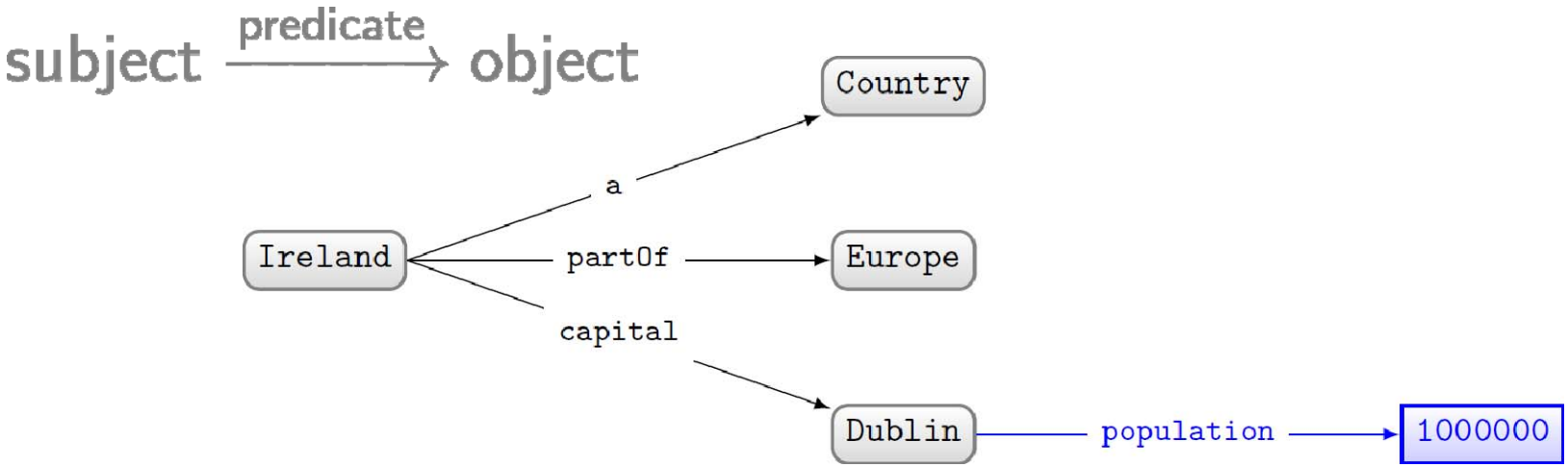
subject	*predicate*	*object*
Ireland	partOf	Europe
Ireland	a	Country
Ireland	capital	Dublin
Dublin	population	1,000,000

RDF often drawn as a (directed, labelled) graph

subject	*predicate*	*object*
Ireland	partOf	Europe
Ireland	a	Country
Ireland	capital	Dublin
Dublin	population	1,000,000



Set of triples thus called an “RDF Graph”



But why triples / graphs?

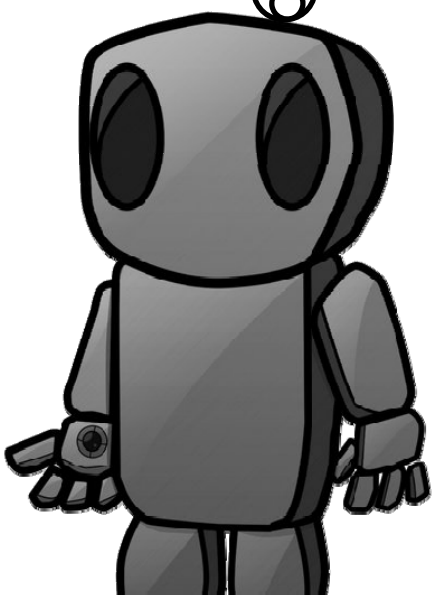
subject	*predicate*	*object*
Ireland	partOf	Europe
Ireland	a	Country
Ireland	capital	Dublin
Dublin	population	1,000,000

What is the benefit of triples / graphs?

NAMING THINGS IN RDF: IRIS

One symbol, *one* meaning …

Dublín ≠



Need unambiguous symbols/identifiers

- Since we're on the Web … use Web identifiers
- URL: Uniform Resource Location
  - The location of a resource on the Web
  - `http://ex.org/Dubl%C3%ADn.html`
- URI: Uniform Resource Identifier (RDF 1.0)
  - Need not be a location, can also be a name
  - `http://ex.org/Dubl%C3%ADn`
- IRI: Internationalised Resource Identifier (RDF 1.1)
  - A URI that allows Unicode characters
  - `http://ex.org/Dublín`

We will use IRIs with prefixes

- `http://ex.org/Dublín` ↔ `ex:Dublín`

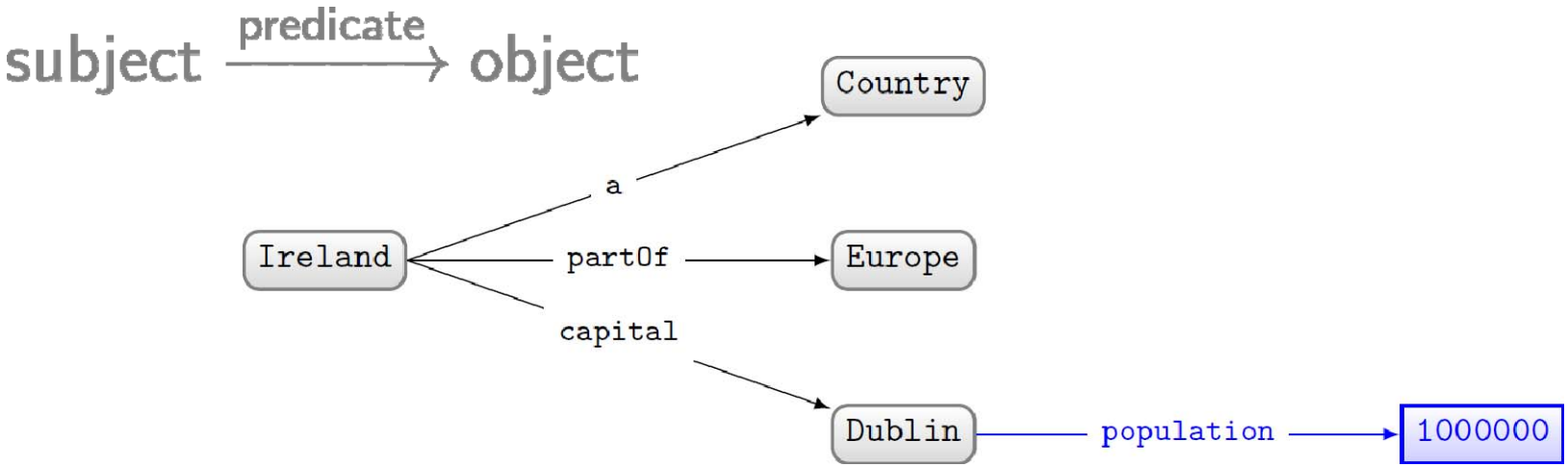
- "`ex:`" denotes a prefix for `http://ex.org/`
- "`Dublín`" is the local name

Frequently used prefixes

Prefix	Value
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
xsd:	http://www.w3.org/2001/XMLSchema#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
owl:	http://www.w3.org/2002/07/owl#

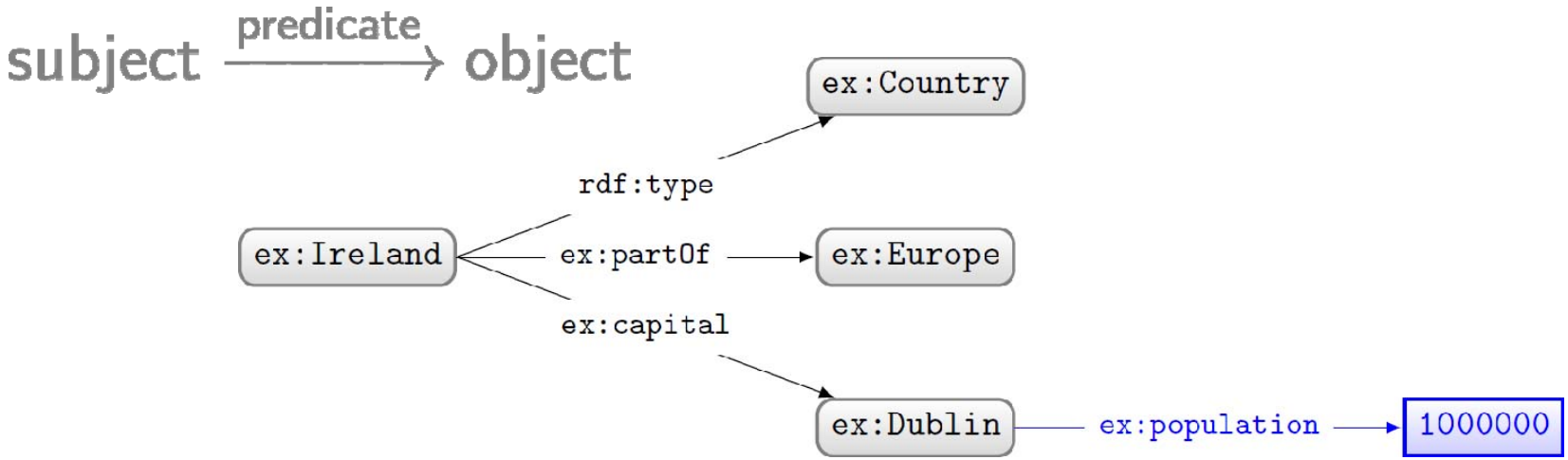
From strings …

subject	*predicate*	*object*
Ireland	partOf	Europe
Ireland	a	Country
Ireland	capital	Dublin
Dublin	population	1,000,000



… to IRIs …

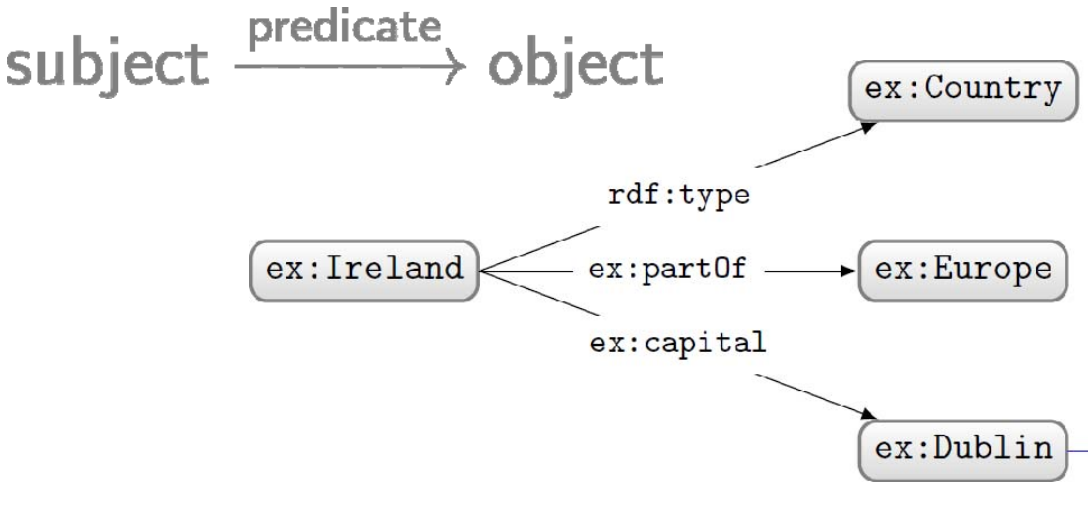
subject	*predicate*	*object*
ex:Ireland	ex:partOf	ex:Europe
ex:Ireland	rdf:type	ex:Country
ex:Ireland	ex:capital	ex:Dublin
ex:Dublin	ex:population	1,000,000



NAMING THINGS IN RDF: LITERALS

What about numbers?

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



Should we assign IRIs to numbers, etc.?

RDF allows “literals” in object position

- Literals are for datatype values, like strings, numbers, booleans, dates, times
- Only allowed in object position

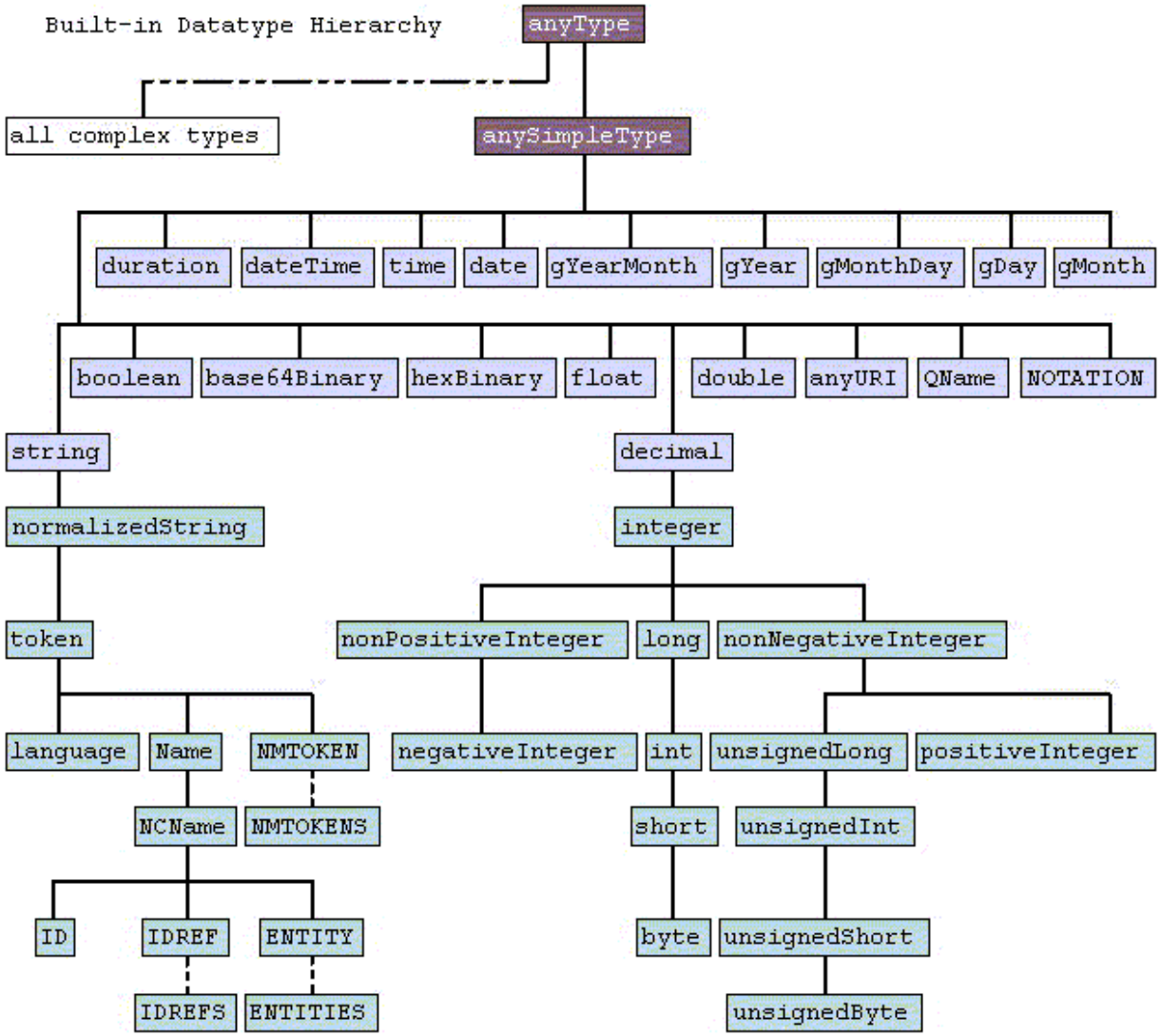
subject	*predicate*	*object*	
ex:Dublin	ex:population	1,000,000	CORRECT
1,000,000	ex:populationOf	ex:Dublin	INCORRECT
ex:Dublin	1,000,000	ex:population	INCORRECT

Datatype literals

- `“lexical-value”^^ex:datatype`
  - `“200”^^xsd:int`
  - `“2014-12-13”^^xsd:date`
  - `“true”^^xsd:boolean`
  - `“this is a string”^^xsd:string`

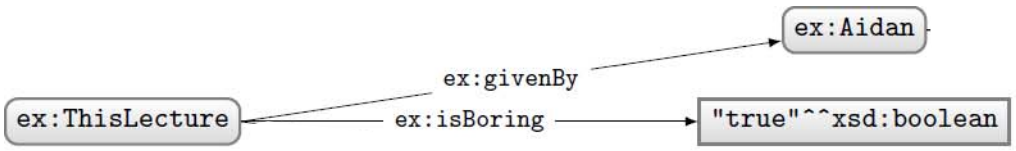
- If the datatype is omitted, it’s a string
  - `“this is a string”`
  - `“200”` is a string, not a number!

Many datatypes borrowed from XML Schema



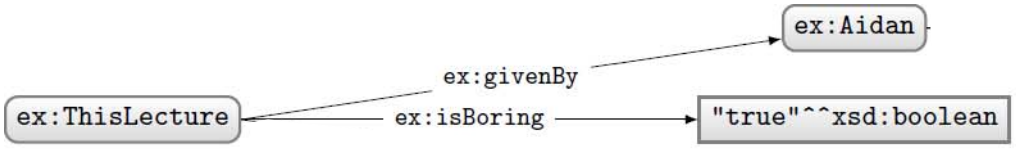
Boolean datatype

Boolean		
xsd:boolean	"true", "false", "1", "0"	Case sensitive



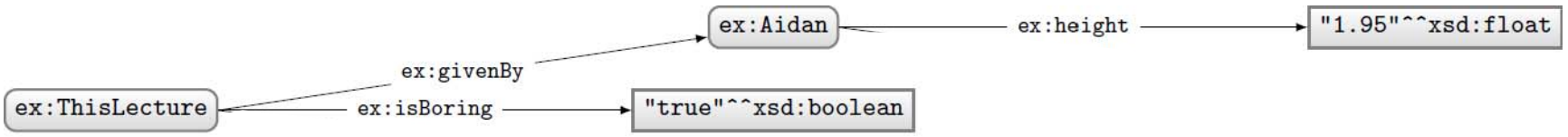
Numeric datatypes

Numeric		
xsd:decimal	"-2.320"	Any precision
└ xsd:integer	"-3"	Any precision, $x \in \mathbb{Z}$
└ xsd:long	"-9223372036854775808"	$-2^{63} \leq x < 2^{63}$
└ xsd:int	"+2147483647"	$-2^{31} \leq x < 2^{31}$
└ xsd:short	"-32768"	$-2^{15} \leq x < 2^{15}$
└ xsd:byte	"127"	$-2^{7} \leq x < 2^{7}$
└ xsd:nonNegativeInteger	"0"	$0 \leq x < \infty$
└ xsd:positiveInteger	"3152"	$1 \leq x < \infty$
└ xsd:unsignedLong	"18446744073709551615"	$0 \leq x < 2^{64}$
└ xsd:unsignedInt	"+4294967295"	$0 \leq x < 2^{32}$
└ xsd:unsignedShort	"65535"	$0 \leq x < 2^{16}$
└ xsd:unsignedByte	"+255"	$0 \leq x < 2^{8}$
└ xsd:nonPositiveInteger	"0"	$x \leq 0$
└ xsd:negativeInteger	"-3152"	$x < 0$
xsd:double	"1.7e308" "-4.9E-324", "NaN", "INF", "-INF"	IEEE 64-bit floating point
xsd:float	"3.4E38", "-1.4e-45", "NaN", "INF", "-INF"	IEEE 32-bit floating point



Temporal datatypes

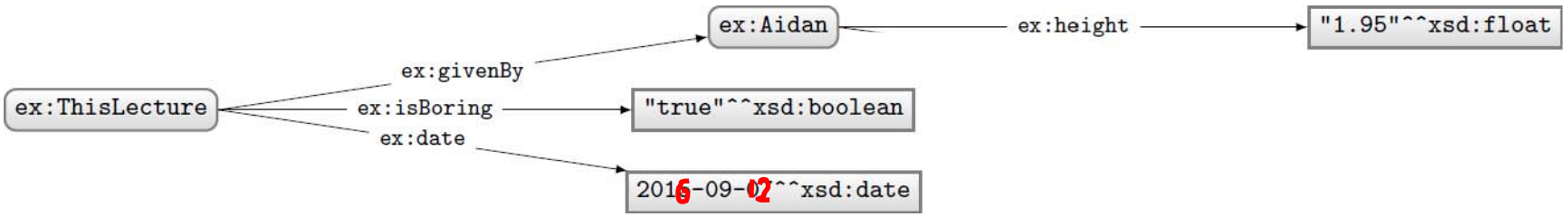
Temporal		
xsd:time	"05:04:12", "05:04:12Z", "05:04:12.00-10:00"	Z indicates +00:00 timezone
xsd:date	"2012-02-29", "2012-12-31+04:00"	Timezone optional
xsd:dateTime	"2012-12-31T00:01:02.034"	Timezone optional
└ xsd:dateTimeStamp	"2012-12-31T00:01:02+04:00"	Timezone required
xsd:duration	"P6Y9M15DT25H61M4.2S"	6 Years ... 4.2 Seconds
├ xsd:dayTimeDuration	"P2DT8H14S"	No month or year
└ xsd:yearMonthDuration	"-P89Y13M"	No days or time
xsd:gDay	"---15", "---01-13:59"	Day recurring every month
xsd:gMonth	"--12", "--01+14:00"	Month recurring every year
xsd:gMonthDay	"--02-29", "--03-01Z"	Date recurring every year
xsd:gYear	"1985", "-0005"	A year ($-y$ indicates B.C.)



6 12

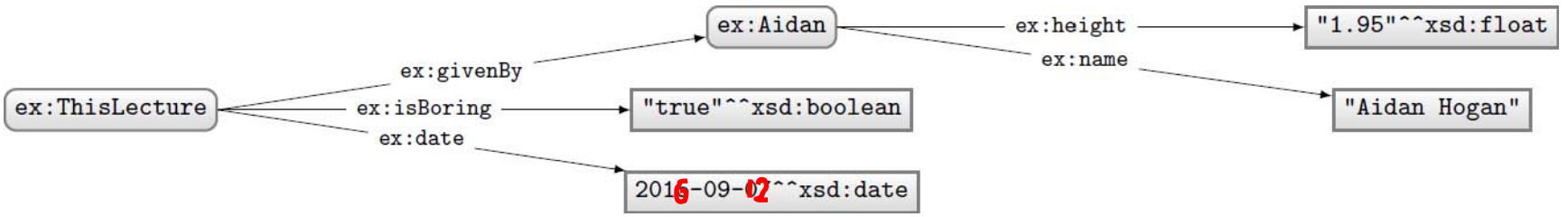
Text/string datatypes

Text		
xsd:string	" tab-> <-tab "	Most Unicode characters
└ xsd:normalizedString	" multiple-> <-spaces "	No \r, \n, \t
└ xsd:token	"one-> <-space"	No leading or double spaces
├ xsd:language	"en", "en-UK", "en-uk", "zh-yue-Hant"	Generalises BCP47 [57]
├ xsd:name	"ns:some_name"	XML names
│ └ xsd:NCName	"some_name"	XML names: no colons
└ xsd:NMTOKEN	"1some_name"	XML names: 1st char relaxed
xsd:base64Binary	"QS5ILiBuZWVkcyBhIHNtb2tlLg=="	Base-64 encoded strings
xsd:hexBinary	"2e2e2e20616e6420616c636f686f6c2e"	Hexadecimal strings
xsd:anyURI	"http://example.com/",	Full IRI strings
rdf:HTML	" some data "	Well-formed HTML content
rdf:XMLLiteral	"<flavours><fruit>apple</fruit></flavours>"	Well-formed XML content



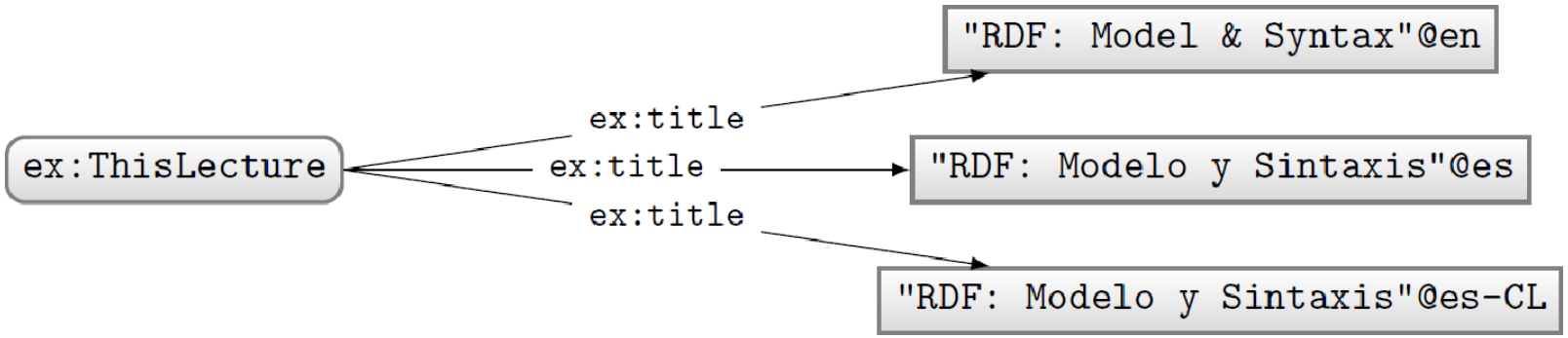
Text/string datatypes

Text		
xsd:string	" tab-> <-tab "	Most Unicode characters
└ xsd:normalizedString	" multiple-> <-spaces "	No \r, \n, \t
└ xsd:token	"one-> <-space"	No leading or double spaces
├ xsd:language	"en", "en-UK", "en-uk", "zh-yue-Hant"	Generalises BCP47 [57]
├ xsd:name	"ns:some_name"	XML names
│ └ xsd:NCName	"some_name"	XML names: no colons
└ xsd:NMTOKEN	"1some_name"	XML names: 1st char relaxed
xsd:base64Binary	"QS5ILiBuZWVkcyBhIHNtb2tlLg=="	Base-64 encoded strings
xsd:hexBinary	"2e2e2e20616e6420616c636f686f6c2e"	Hexadecimal strings
xsd:anyURI	"http://example.com/",	Full IRI strings
rdf:HTML	" some data "	Well-formed HTML content
rdf:XMLLiteral	"<flavours><fruit>apple</fruit></flavours>"	Well-formed XML content



Language-Tagged Strings

- Specify that a string is in a given language
- `“string”@lang-tag`
- No datatype!



(NOT) NAMING THINGS IN RDF: BLANK NODES

Having to name everything is hard work

"Now! ... *That* should clear up a few things around here!"

For this reason, RDF gives blank nodes

- Syntax: `_:blankNode`
- Represents existence of something
  - Often used to avoid giving an IRI (e.g., shortcuts)
- Can only appear in subject or object position

subject	*predicate*	*object*	
ex:Ireland	ex:capital	_:b1	CORRECT
_:b2	ex:capital	ex:Dublin	CORRECT
ex:Ireland	_:b3	ex:Dublin	INCORRECT

- (More later)

RDF TERMS: SUMMARY

A Summary of RDF Terms

1. IRIs (Internationalised Resource Identifiers)
   – Used to name generic things
2. Literals
   – Used to refer to datatype values
   – Strings may have a language tag
3. Blank Nodes
   – Used to avoid naming things
   – A little mysterious right now

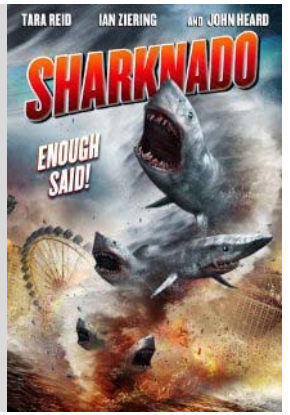
subject	*predicate*	*object*
[IRI, Blank Node]	[IRI]	[IRI, Blank Node, Literal]

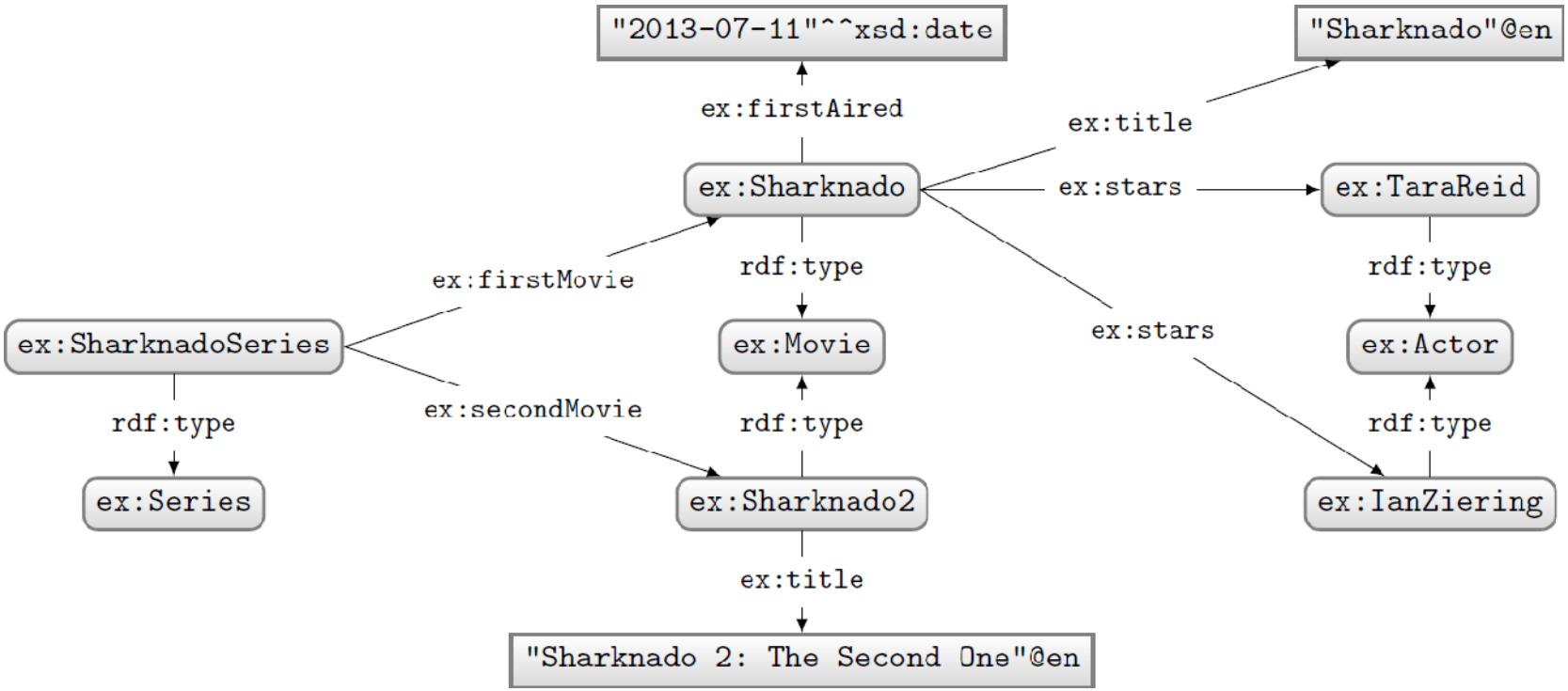
MODELLING DATA IN RDF

Let's model something in RDF …

Model the following in RDF:

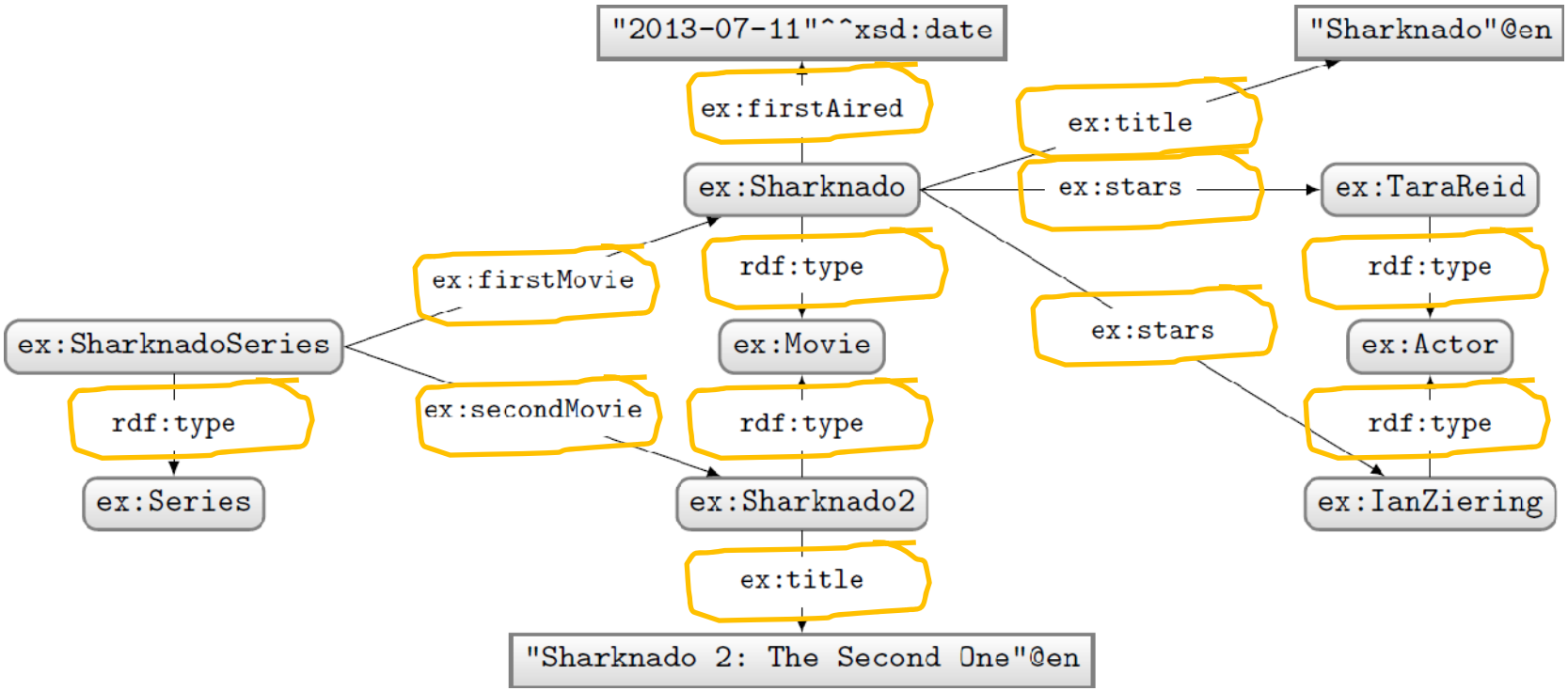
"Sharknado is the first movie of the Sharknado series. It first aired on July 11, 2013. The movie stars Tara Reid and Ian Ziering. The movie was followed by 'Sharknado 2: The Second One'."





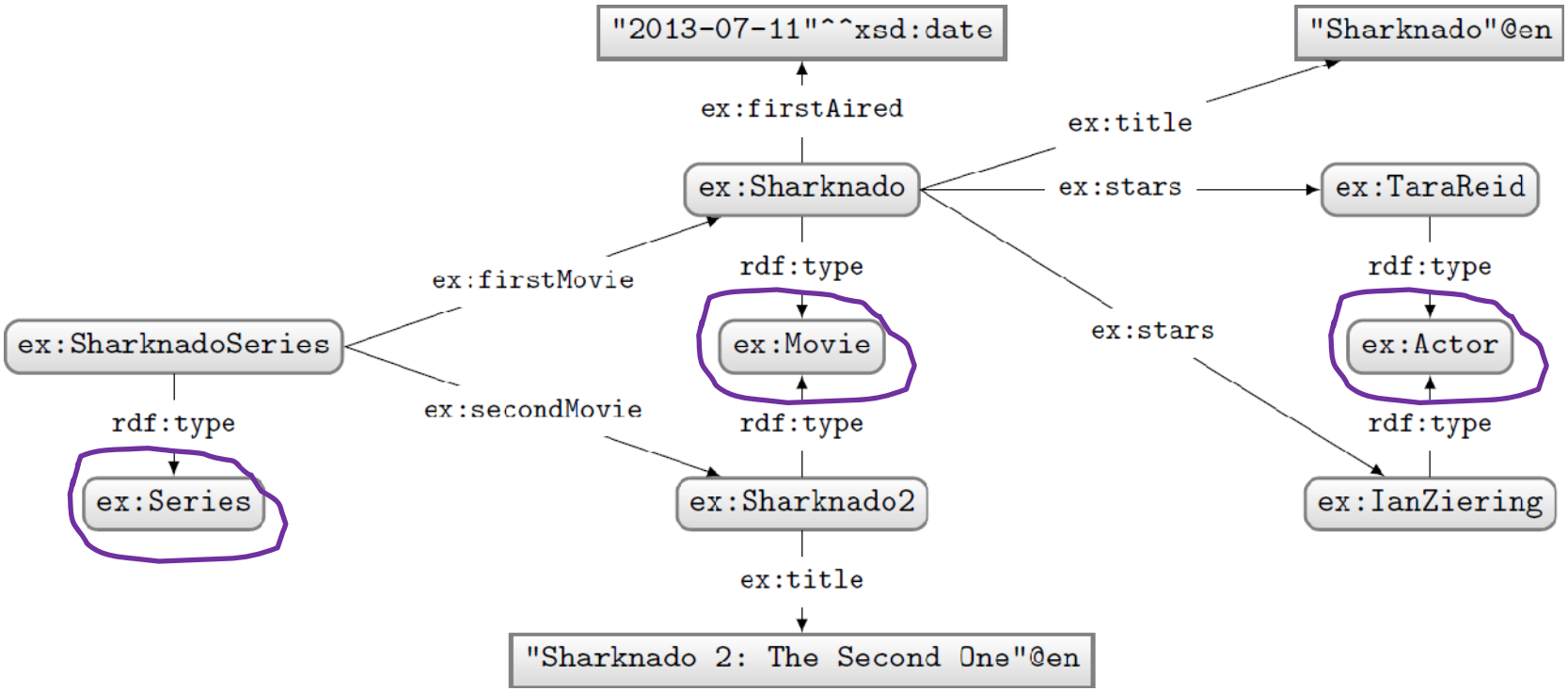
RDF Properties

- 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



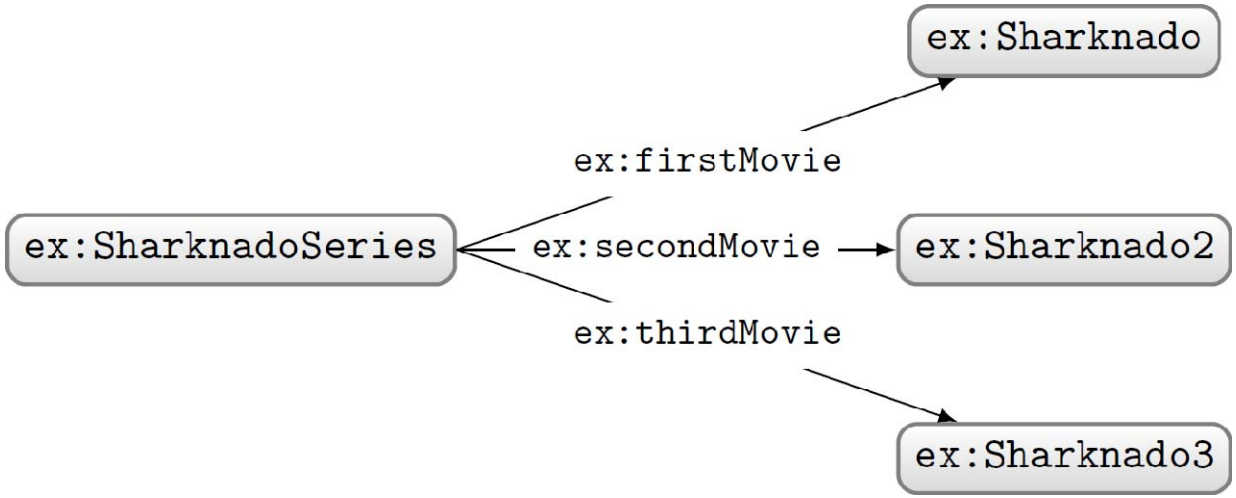
Modelling in RDF not always so simple

Model the following in RDF:
"Sharknado stars Tara Reid in the role of 'April Wexler'.

Modelling in RDF not always so simple

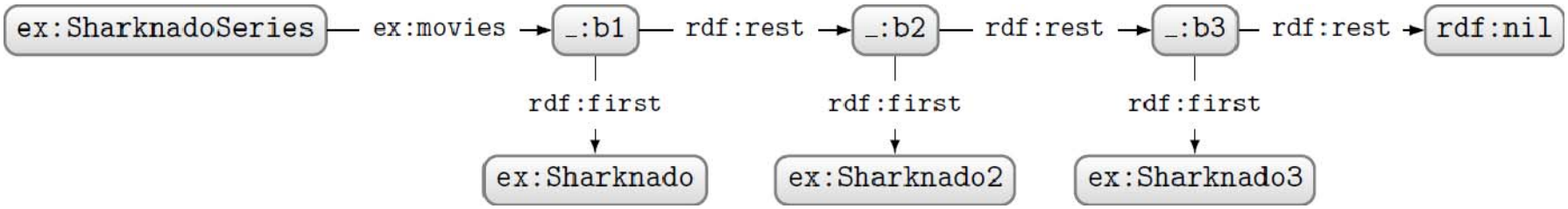
Model the following in RDF:

"The first movie in the Sharknado series is 'Sharknado'. The second movie is 'Sharknado 2: The Second One'. The third movie is 'Sharknado 3: Oh Hell No!'.



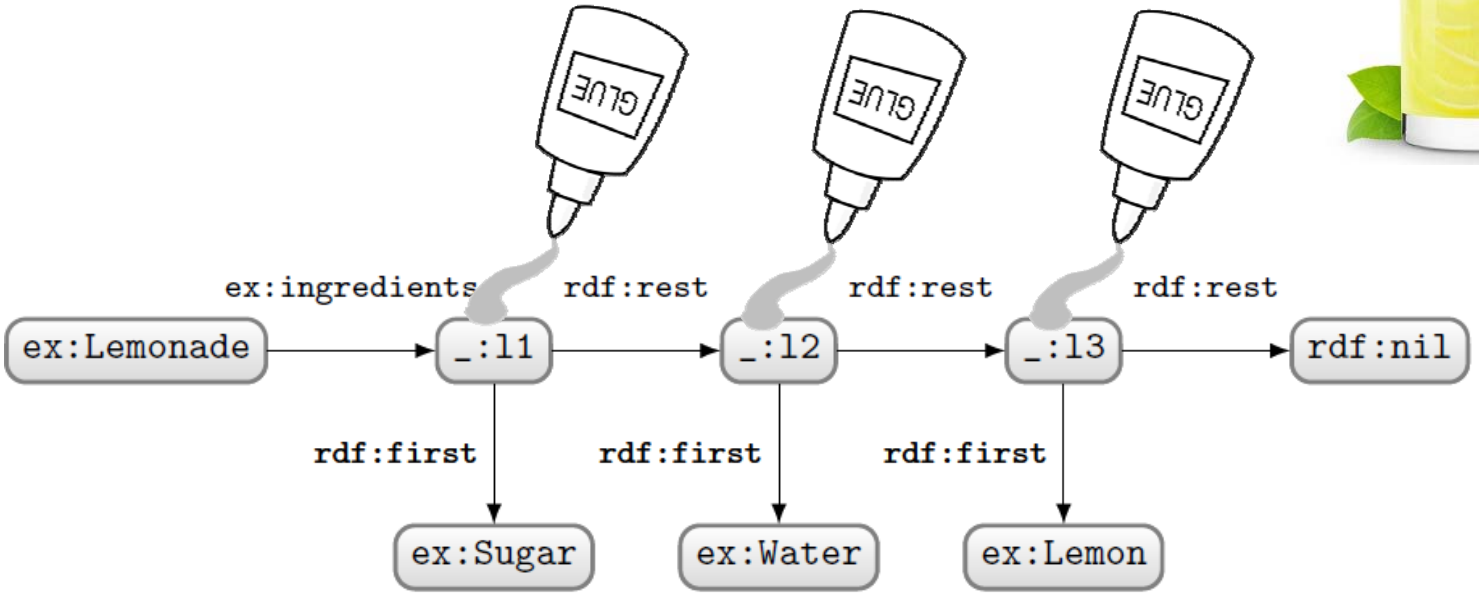
RDF Collections: Model Ordered Lists

- Standard way to model linked lists in RDF
- Use `rdf:rest` to link to rest of list
- Use `rdf:first` to link to current member
- Use `rdf:nil` to end the list



RDF Collections: Generic Modelling

- Not just for Sharknado series



ANOTHER MODELLING EXAMPLE: FROM TABLES TO RDF

Course data as an RDF graph?

professors.csv

ID	Name	Course
...		
24.482.054-9	Aidan Hogan	CC3201
24.482.054-9	Aidan Hogan	CC5212
24.482.054-9	Aidan Hogan	CC6202
...		

students.csv

ID	Name	Course
...		
10.323.634-4	Pia García	CC3201
12.323.792-8	Juan Ramírez	CC6202
12.323.792-8	Juan Ramírez	CC5212
...		

courses.csv

Code	Name
...	
CC3201	Databases
CC5212	Big Data Processing
...	

How might we model these relational data as an RDF graph?

tas.csv

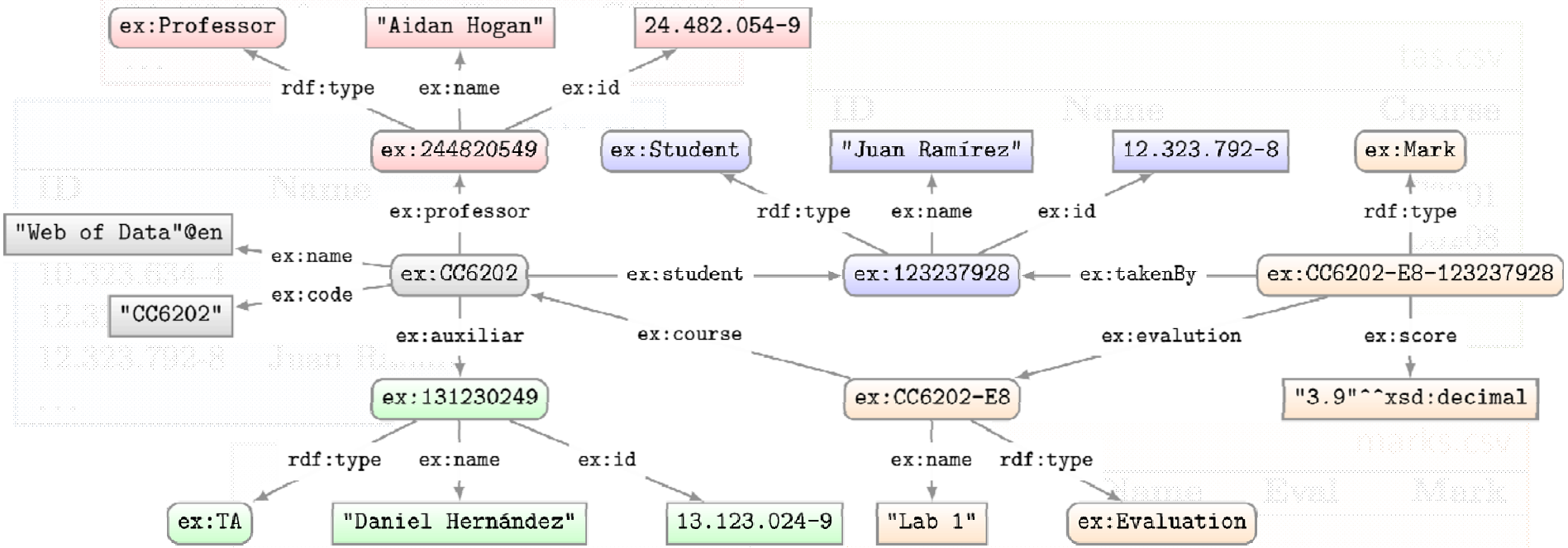
ID	Name	Course
...		
12.412.412-4	Sebastián Ferrada	CC3201
12.412.412-4	Sebastián Ferrada	CC5208
13.123.024-9	Daniel Hernández	CC6202
...		

marks.csv

ID	Name	Eval	Mark
...			
10.323.634-4	CC3201	Lab1	6.0
10.323.634-4	CC3201	Lab2	4.5
12.323.792-8	CC6202	Exam1	3.9
...			

Course data as an RDF graph?

How might we model these relational data as an RDF graph?



… for example

Graphs vs. Relational Tables

- Graphs more flexible
  - Just add more nodes and edges
    - (possibly with new labels!)
- May be more concise

- Graphs less “structured”
  - No longer have a clear schema
- May be more verbose

Can we model any relational table(s) as an (RDF) graph?
Can we model any (RDF) graph as a relational table?

Yes.

RDF SYNTAXES:
WRITING RDF DOWN

N-Triples

- Line delimited format
- No shortcuts

ex1:Jen	rdf:type	ex1:Person
ex1:Jen	rdf:type	ex1:Female
ex1:Jen	rdfs:label	"Jen"@en
ex1:Jen	ex1:allergy	ex1:Citrus
ex1:Jen	ex1:location	_:loc
_:loc	ex1:lat	"53.3"^^xsd:decimal
_:loc	ex1:long	-9.0^^xsd:decimal

```
<http://ex1.org/#Jen> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://ex1.org/#Person> .
<http://ex1.org/#Jen> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://ex1.org/#Female> .
<http://ex1.org/#Jen> <http://www.w3.org/2000/01/rdf-schema#label> "Jen"@en .
<http://ex1.org/#Jen> <http://ex1.org/#allergy> <http://ex1.org/#Citrus> .
<http://ex1.org/#Jen> <http://ex1.org/#location> _:loc .
_:loc <http://ex1.org/#lat> "53.3"^^ <http://www.w3.org/2001/XMLSchema#decimal> .
_:loc <http://ex1.org/#long> -9.0^^ <http://www.w3.org/2001/XMLSchema#decimal> .
```

RDF/XML

ex1:Jen	rdf:type	ex1:Person
ex1:Jen	rdf:type	ex1:Female
ex1:Jen	rdfs:label	"Jen"@en
ex1:Jen	ex1:allergy	ex1:Citrus
ex1:Jen	ex1:location	_:loc
_:loc	ex1:lat	"53.3"^^xsd:decimal
_:loc	ex1:long	-9.0^^xsd:decimal

- Legacy format
- Just horrible

```
<?xml version="1.0"?>
<!DOCTYPE img [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#"> ]>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:ex1="http://example1.org/#">
 <ex1:Person rdf:about="http://example1.org/#Jen">
  <rdf:type rdf:resource="http://example1.org/#Female" />
  <rdfs:label xml:lang="en">Jen</rdfs:label>
  <ex1:allergy rdf:resource="http://example1.org/#Citrus" />
  <ex1:location>
   <rdf:Description>
    <ex1:lat rdf:datatype="&xsd;decimal">53.3</ex1:lat>
    <ex1:long rdf:datatype="&xsd;decimal">-9.0</ex1:long>
   </rdf:Description>
  </ex1:location>
 </ex1:Person>
</rdf:RDF>
```

RDFa

- Embed RDF into HTML
- Not so intuitive

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>Recipe for Coffee Parfait</title>
  <base href="http://example.org/" />
 </head>
 <body vocab="http://example.org/#" lang="en"
       prefix="rdfs: http://www.w3.org/2000/01/rdf-schema#">
  <div typeof="Recipe" resource="#CoffeeParfait">
   <h1 property="rdfs:label">Coffee Parfait</h1>
   <p>Time: <span property="minutes" datatype="xsd:integer" content="25">25 mins</span></p>
   <h2>Ingredients:</h2>
   <ul rel="ingredient">
    <li about="#EggYolk" property="rdfs:label">Egg Yolk</li>
    <li about="#Sugar" property="rdfs:label">Sugar</li>
    <li about="#Cream" property="rdfs:label">Cream</li>
    <li about="#Coffee" property="rdfs:label">Coffee</li>
   </ul>
  </div>
 </body>
</html>
```

JSON-LD

- Embed RDF into JSON
- Not completely aligned with RDF

```
{
 "@context": {
  "xsd": "http://www.w3.org/2001/XMLSchema#",
  "@base": "http://example.com/",
  "@vocab": "http://example.com/#",
  "label": "http://www.w3.org/2000/01/rdf-schema#label",
  "minutes": {
   "@id": "minutes",
   "@type": "xsd:integer"
  },
  "@language": "en"
 },
 "@id": "#CoffeeParfait",
 "@type": "Recipe",
 "label": "Coffee Parfait",
 "minutes": "25",
 "ingredient": [
  { "@id": "#EggYolk", "label": "Egg Yolk"},
  { "@id": "#Sugar", "label": "Sugar"},
  { "@id": "#Cream", "label": "Cream"},
  { "@id": "#Coffee", "label": "Coffee"}
 ]
}
```

Turtle

- Readable format

ex1:Jen	rdf:type	ex1:Person
ex1:Jen	rdf:type	ex1:Female
ex1:Jen	rdfs:label	"Jen"@en
ex1:Jen	ex1:allergy	ex1:Citrus
ex1:Jen	ex1:location	_:loc
_:loc	ex1:lat	"53.3"^^xsd:decimal
_:loc	ex1:long	-9.0^^xsd:decimal

```
@base <http://ex1.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ex1: <http://ex1.org/#> .
<#Jen> a <http://ex1.org/#Person> , ex1:Female ;
    rdfs:label "Jen"@en ; <#allergy> <#Citrus> ;
    ex1:location [ ex1:lat 53.3 ; ex1:long -9.0 ] .
```

Relative URIs

Prefixes

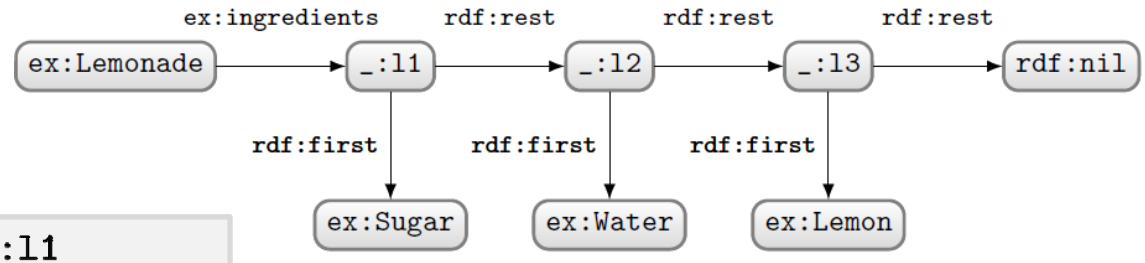
Repeat S (‘;’) SP (‘,’)

rdf:type

Datatype shortcuts

Blank node shortcuts

Turtle: Collections Shortcut



ex:Lemonade	ex:ingredients	_:l1
_:l1	rdf:first	ex:Sugar
_:l1	rdf:rest	_:l2
_:l2	rdf:first	ex:Water
_:l2	rdf:rest	_:l3
_:l3	rdf:first	ex:Lemon
_:l3	rdf:rest	rdf:nil

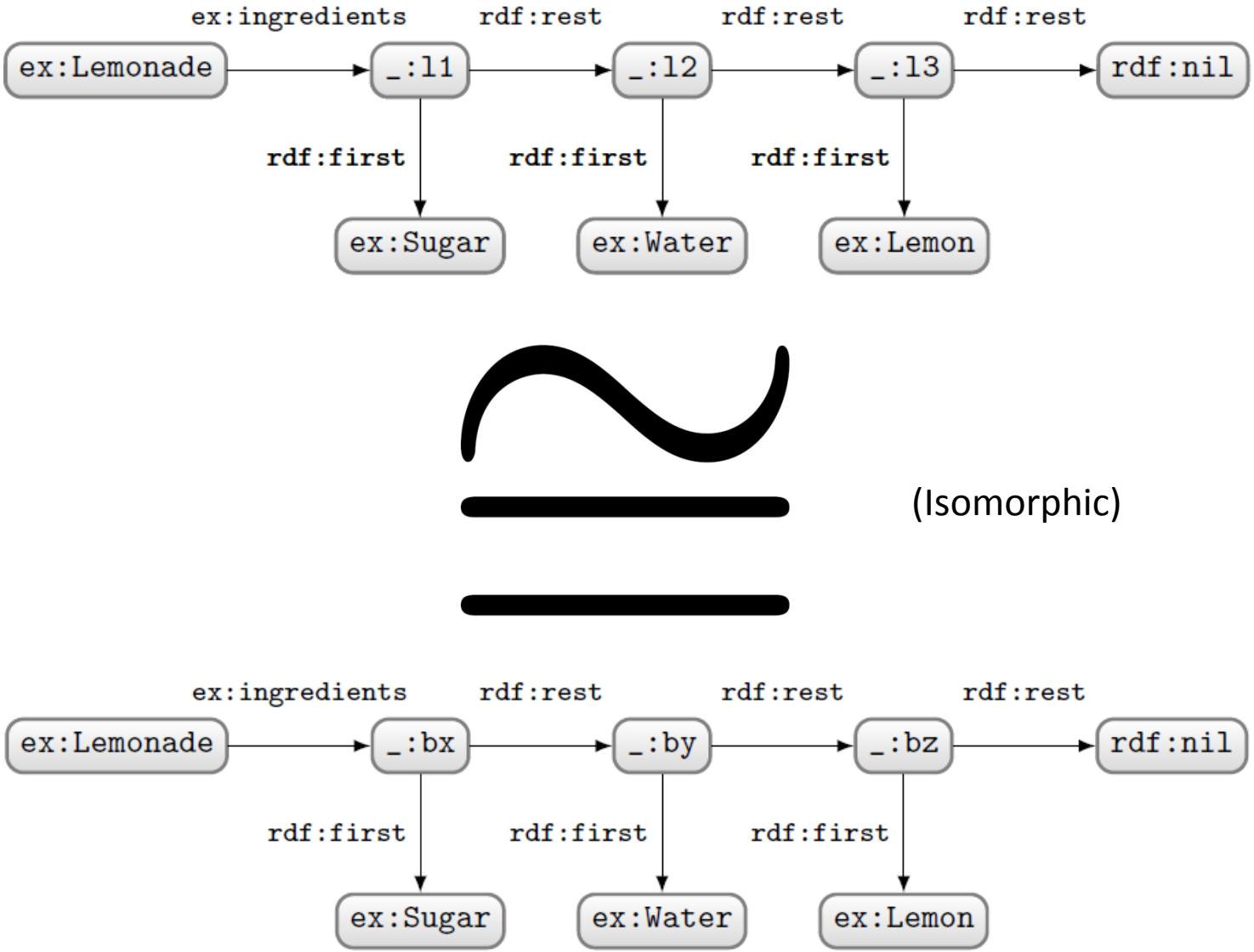
Only possible with blank nodes!

```
@base <http://example.org/#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
  <Lemonade> <ingredients> [
   rdf:first <Sugar> ; rdf:rest [
     rdf:first <Water> ; rdf:rest [
       rdf:first <Lemon> ; rdf:rest rdf:nil
     ]
   ] .
```

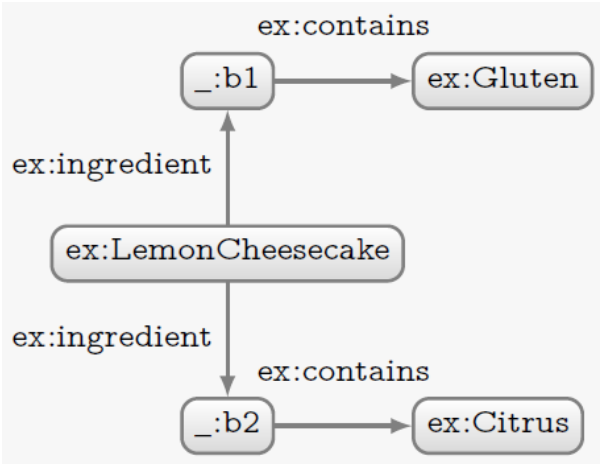
```
@base <http://example.org/#> .
  <Lemonade> <ingredients> ( <Sugar> <Water> <Lemon> ) .
```

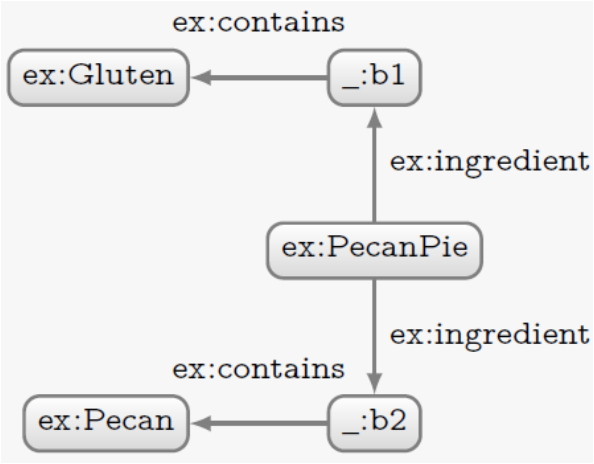
BLANK NODES ADD COMPLEXITY

Blank nodes names aren't important …

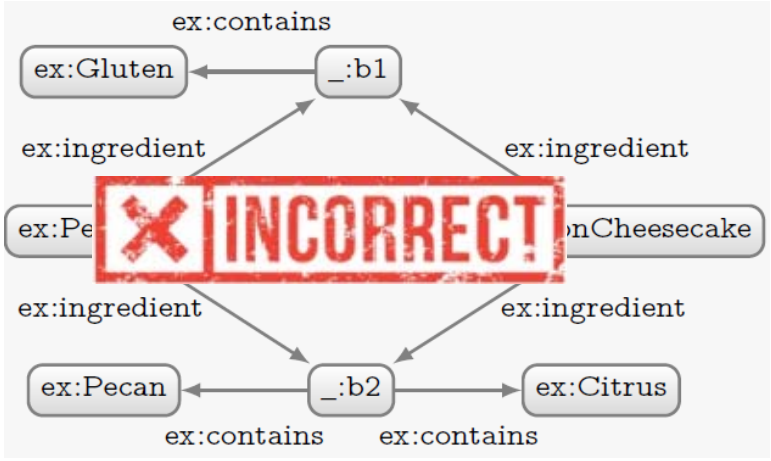


Blank nodes are local identifiers

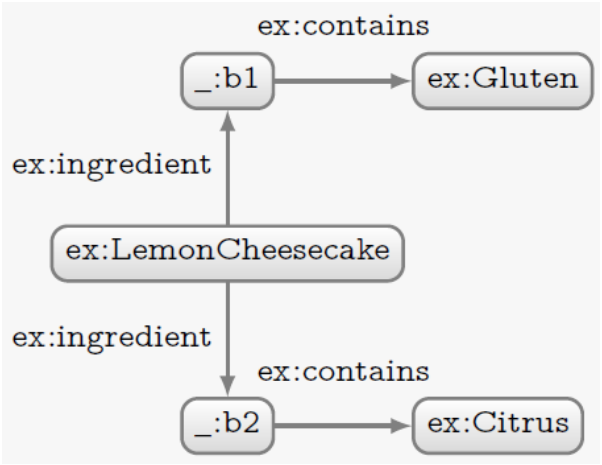


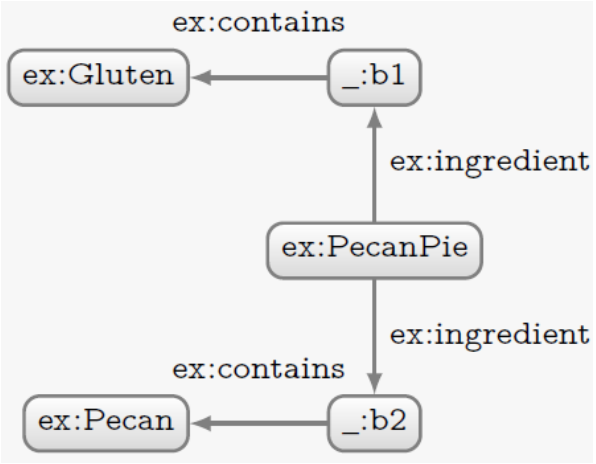


How should we combine these two RDF graphs?

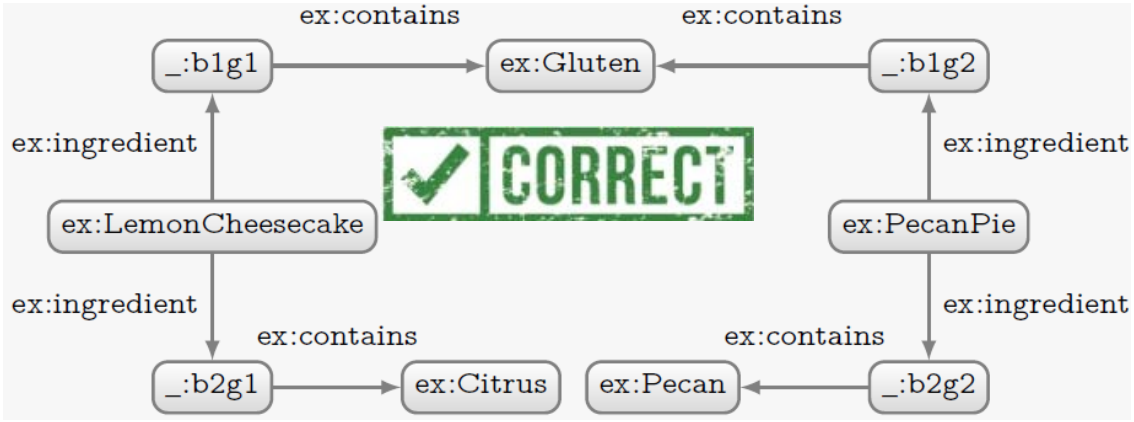


Need to perform an RDF merge

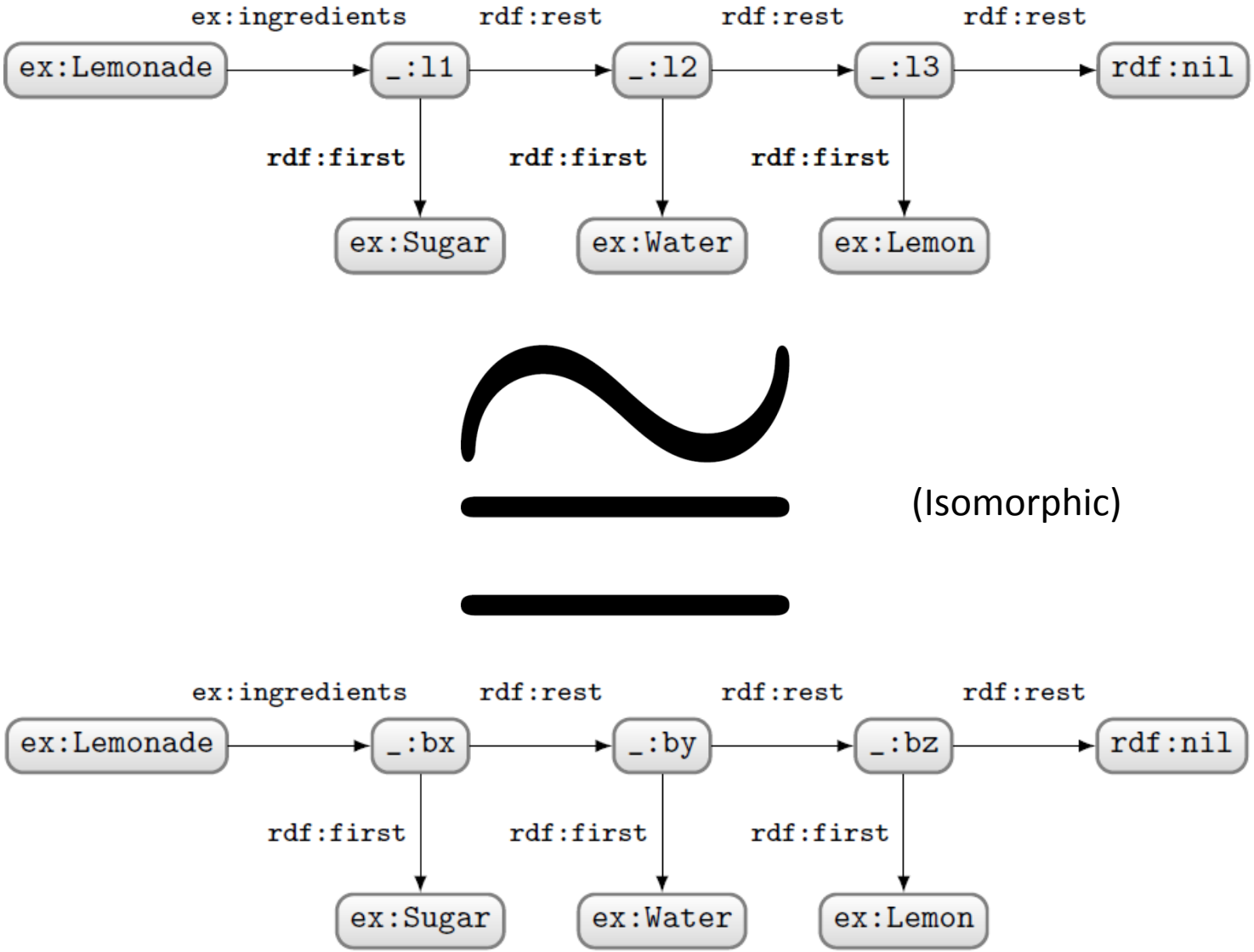




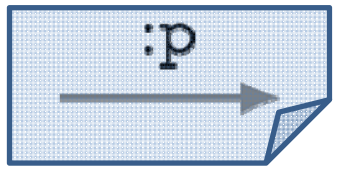
How should we combine these two RDF graphs?

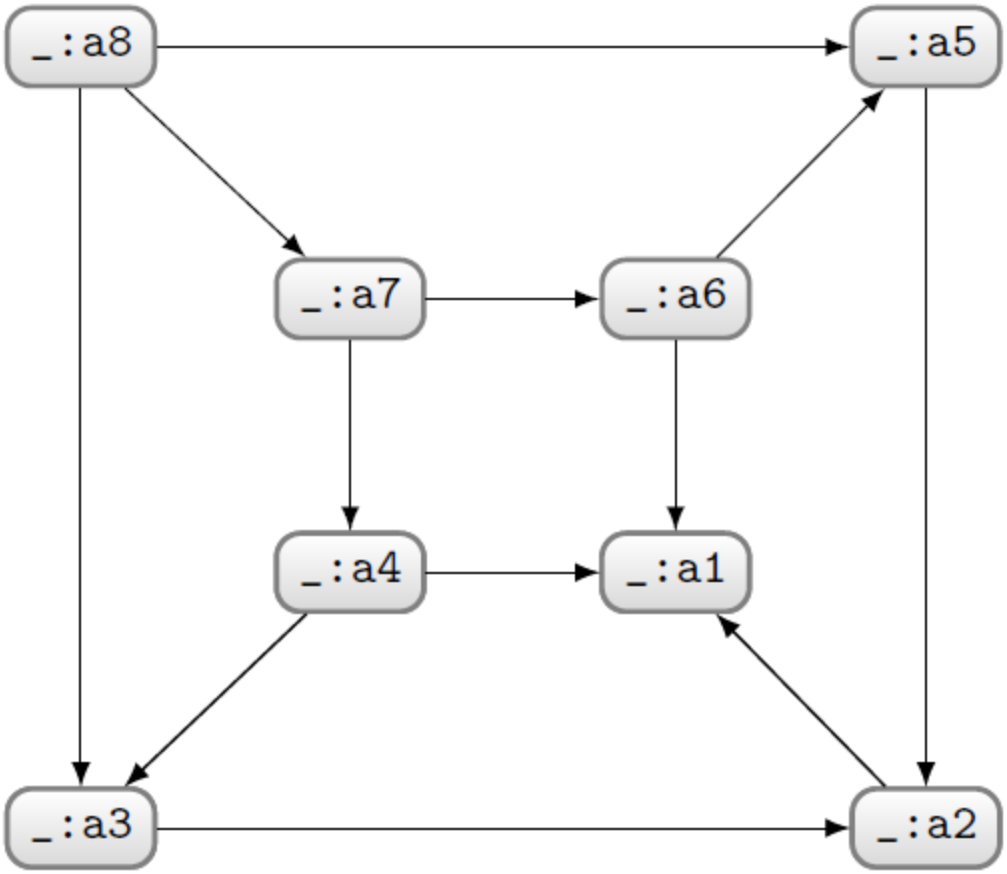


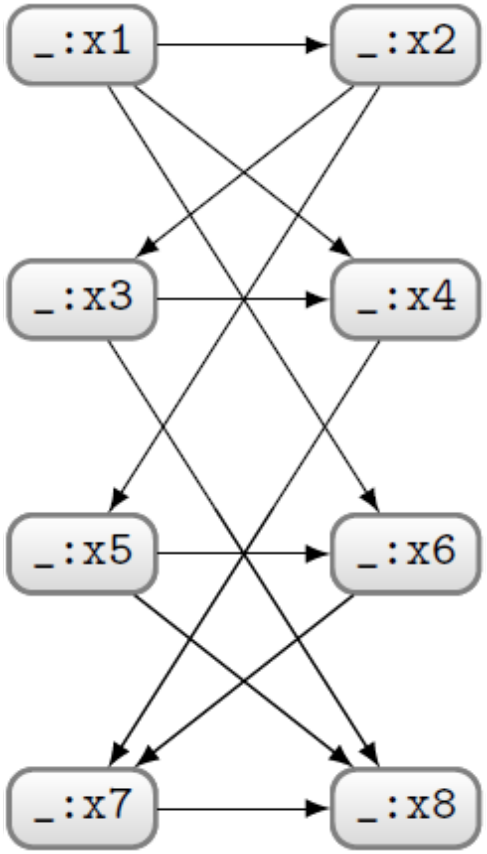
Are two RDF graphs the “same”?



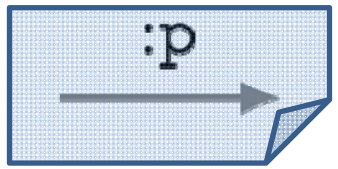
Are two RDF graphs the “same”?

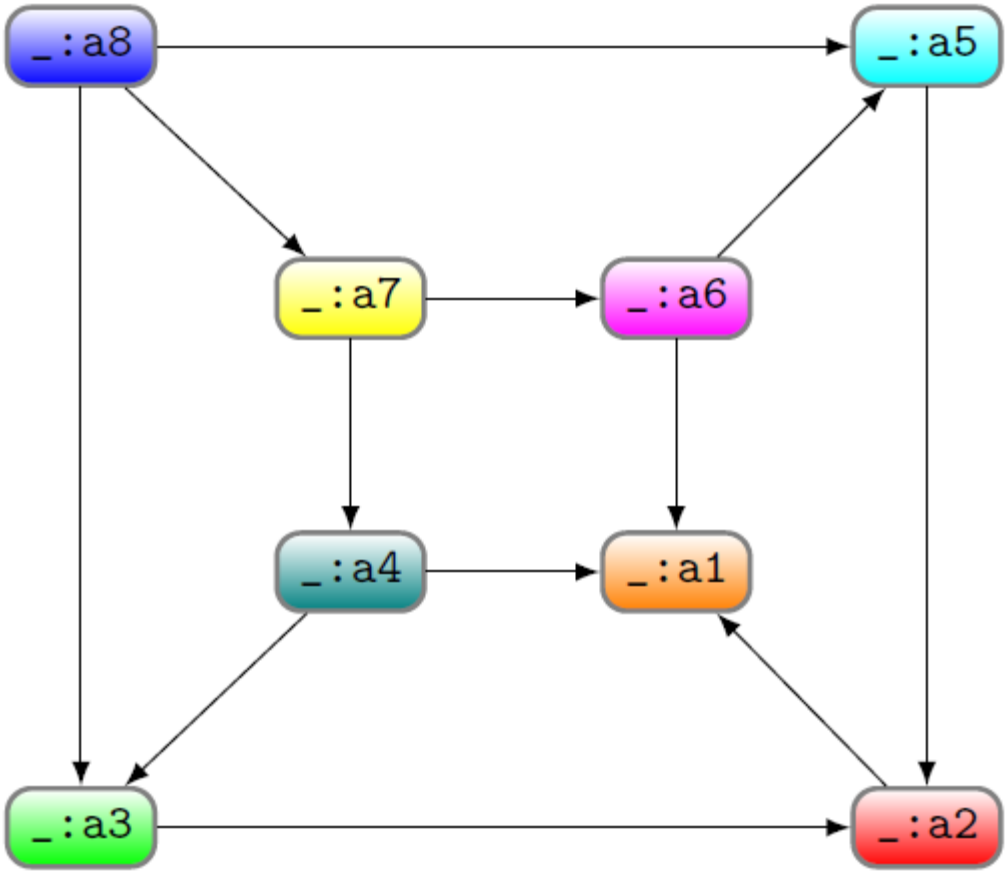


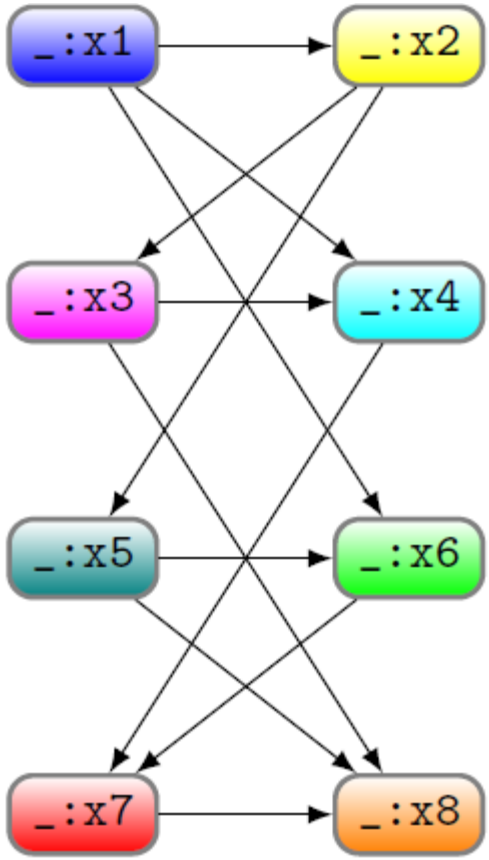




Are two RDF graphs the “same”?







GI-COMPLETE

RECAP

RDF: Resource Description Framework

DATA:

`http://ex.org/Ireland`

Ireland

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

`http://ex.org/Dublin`

Dublin

(Dublin,population,1000000)

RDF is based on triples:

(Ireland,capital,Dublin)

(subject,predicate,object)

RDF = Resource Description Framework

- Structure data on the Web!

- RDF based on triples:
  - subject, predicate, object
  - A set of triples is called an RDF graph

- Three types of RDF terms:
  - IRIs (any position)
  - Literals (object only; can have datatype or language)
  - Blank nodes (subject or object)

RDF = Resource Description Framework

- Modelling in RDF:
  - Describing resources
  - Classes and properties form core of model
  - Try to break up higher-arity relations
  - Collections: standard way to model order/lists

- Syntaxes:
  - N-Triples: simple, line-delimited format
  - RDF/XML: legacy format, horrible
  - RDFa: embed RDF into HTML pages
  - JSON-LD: embed RDF into JSON
  - Turtle: designed to be human friendly

RDF = Resource Description Framework

- Two operations on RDF graphs:
  - Merging: keep blank nodes in source graphs apart
  - Are they the “same” modulo blank node labels: isomorphism check!

Questions?