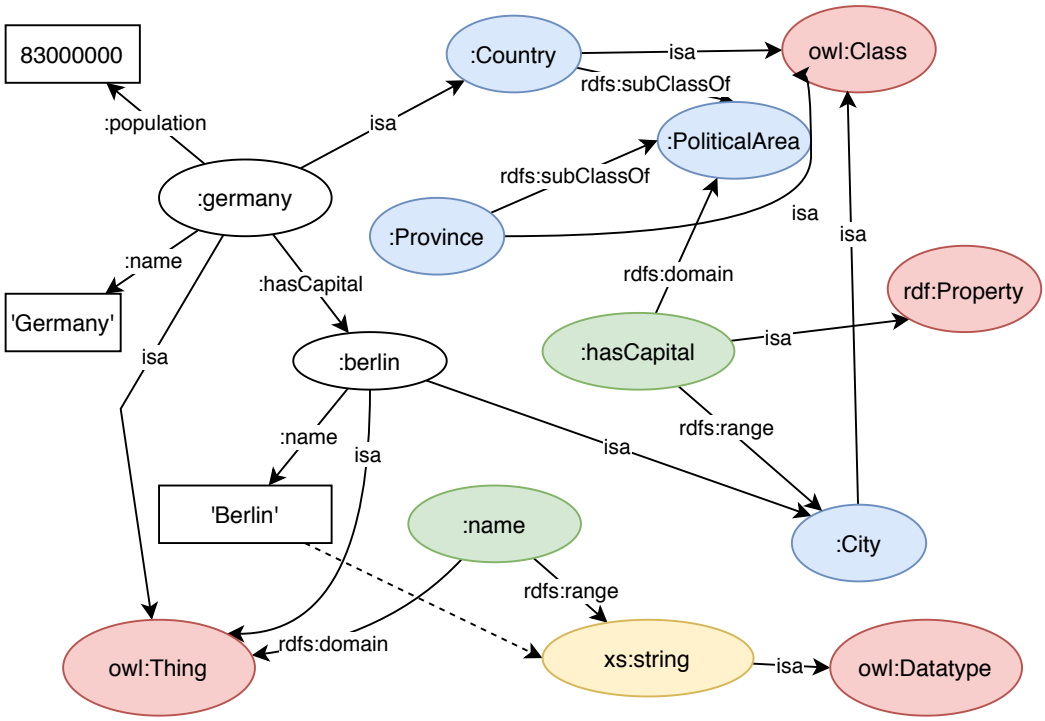

83000000
:population
:germany
isa
:Country
isa
owl:Class
rdfs:subClassOf
:PoliticalArea
rdfs:subClassOf
:Province
isa
isa
:name
'Germany'
:hasCapital
rdfs:domain
rdf:Property
isa
:berlin
:hasCapital
isa
rdfs:range
:name
isa
isa
'Berlin'
:name
:City
rdfs:range
owl:Thing
rdfs:domain
xs:string
isa
owl:Datatype

Yesterday: "Symbolic Reasoning"

Background: Philosophical logics, mathematical logics, model theory aspects: human reasoning *about* properties of the logic).

Each logic, and thus also First-Order Logic provides a framework that can be used for symbolic reasoning:

FOL Formulas are strings, FOL reasoning are algorithms that work on their parse trees.

=> symbolic reasoning: all about Syntax, not Semantics

Formulas are evaluated wrt. first-order-logic structures/interpretations

Syntax: the symbols used for writing formulas:

* logical symbols: $\wedge$, $\exists$, ...
* variables: x, y, ...
* depending on the application: predicate symbols and function symbols, "signature" Σ

for mondial: Σ = {Country, City, name, hasCapital, ...}

FOL Structure: $\mathcal{S} = (I, \mathcal{D})$

$\mathcal{D}$ is the domain ... the things in the real world.

I maps the symbols from Σ to the domain ...

Example: our "real-world-application" contains a (green) frog, and strings and numbers:

$\mathcal{D}$ = { 🐸 } $\cup$ Strings $\cup$ Numbers ...

Signature to talk about the frog and its properties: (1-ary and 2-ary predicates and constant symbols)

$\Sigma = \{$ Frog/1, Green/1, name/2, bob/c0 $\}$

Interpret the symbols in OUR structure/model $\mathcal{S}$ (=current situation):

I (bob) = (🐸) (an element from $\mathcal{D}$)

I (name) = { (🐸 ,"Bob"), ... } (a set of 2-tuples over $\mathcal{D}$)

I (Frog) = {(🐸), ... } (a set of 1-tuples over $\mathcal{D}$)

Knowledge base $\mathcal{K}$: all frogs are green.

$\forall x$: Frog(x) $\rightarrow$ Green(x)

Our $\mathcal{S}$ must be a model of $\mathcal{K}$:

Tableau calculus: what can we derive?

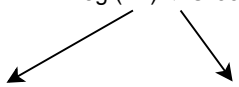
$\forall x$: Frog(x) $\rightarrow$ Green(x)

Frog(bob)

Frog(X1) $\rightarrow$ Green(X1) (introduce a tableau variable X1)

$\neg$ Frog (X1) $\vee$ Green(X1) equivalent

open two branches



$\neg$Frog(X1) Green(X1)

☐ X1 $\leftarrow$ bob ➤ Green(bob)

=> conclusion by reasoning: bob must be green in our $\mathcal{S}$

=> I(Green) $\supseteq$ I(bob)

I(Green) $\supseteq$ { (🐸)}

I practically is a database, containing unary and binary tables:

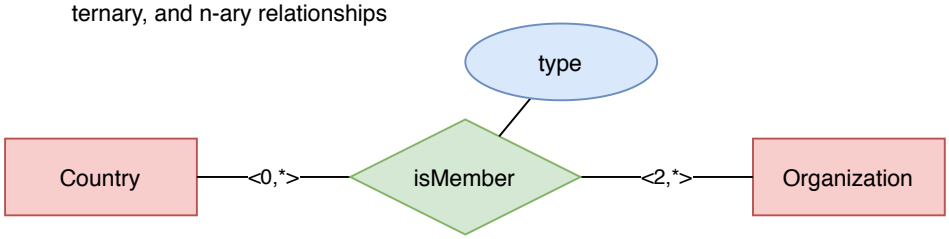
(note: DB is only on the syntax level, so bob <-> 🐸)

name	
bob	"Bob"
:	:

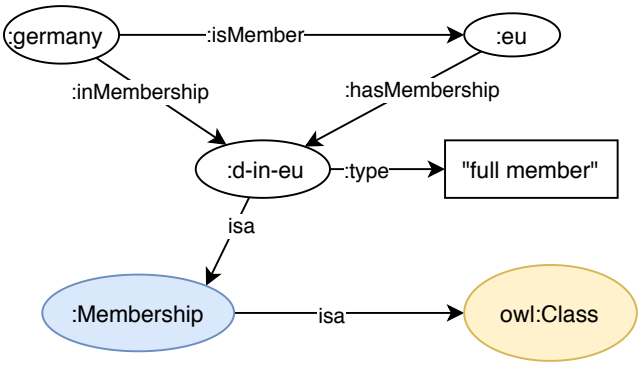
Frog
bob

(the constant bob/c0 is like an object identifier)

RDF (and also UML class diagrams): only binary relationships
ER: relationships may have attributes;
ternary, and n-ary relationships



RDF: Reification of the "Membership" (relationship) instance



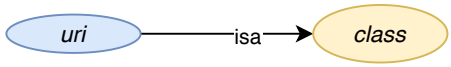
Note: the same holds when modeling relationships with attributes in UML class diagrams. UML provides "association classes" for such cases.

Reification has also to be used for ternary (or n-ary in general) relationships in UML, and in RDF. True ternary relationships are rare. Those who did the BSc here may remember the first exercise sheet of the "Introduction to Databases" lecture.
The reference solutions for it (in German) can be found at <https://www.dbis.informatik.uni-goettingen.de/Teaching/DB-WS1920/DBBlatt1ML.pdf>.

Set-oriented semantics

Class:

- graphically: set of its instances
- relational database:
  (unary) tuples in a unary table
- FOL: interpretation of a unary predicate,
  e.g. $\mathcal{I}$(country) = { (*germany*), (*france*), ... }
  unary tuples(!), each containing an element of the domain
- RDF graph: all the (oval) nodes having an



edge to that class

Relationship

- graphically: set of arrows between things (or things and literals)
- relational database: binary tuples in a table
- FOL: interpretation of a binary predicate
- e.g. $\mathcal{I}$(capital) = { (*germany*, *berlin*), (*france*, *paris*), ...}
  $\mathcal{I}$(area) = {(*germany*, 356000), (*france*, 500000), ... }
  binary tuples each containing a pair of elements from the domain
- RDF graph: set of all tuples connected by an arrow labeled with the property (name)