IV-Applications
Statistical Models of Knowledge Graphs

Foundations of Neuro-Symbolic AI

Alex Goessmann

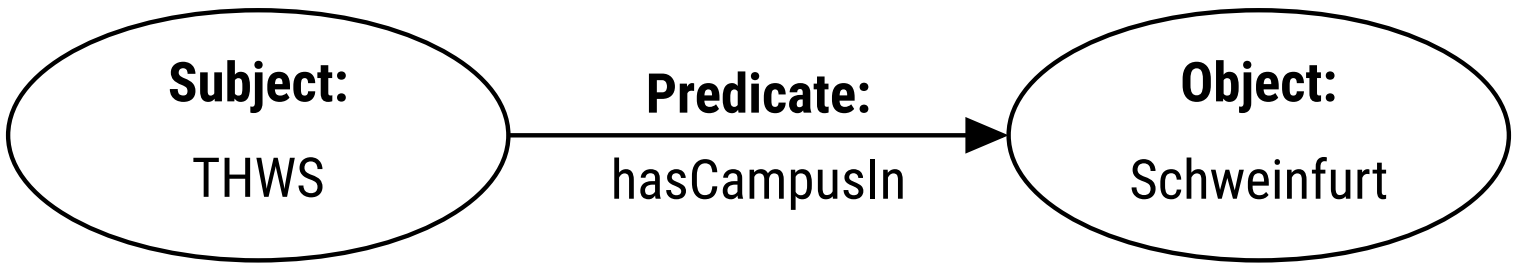
University of Applied Science Würzburg-Schweinfurt

Summer Term 2026

Knowledge Graphs: Data stored in Graphs

Resource Description Framework (RDF):

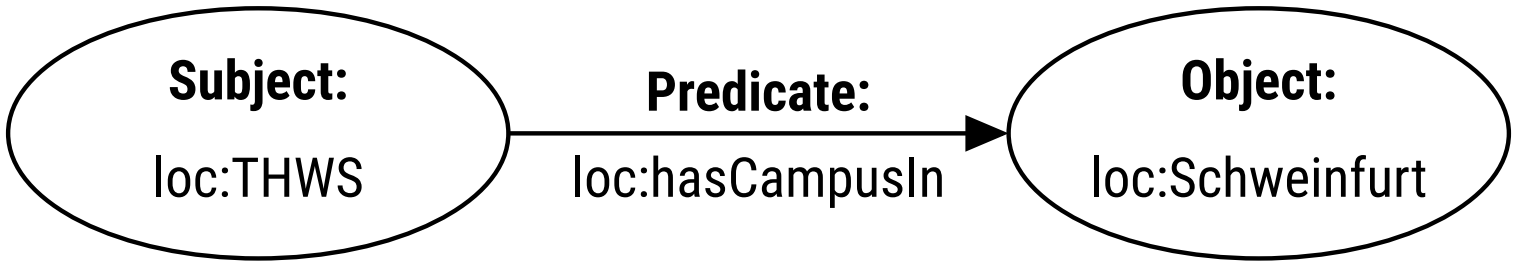
- Data is stored in form of triples (Subject,Predicate,Object)



Knowledge Graphs: Data stored in Graphs

Resource Description Framework (RDF):

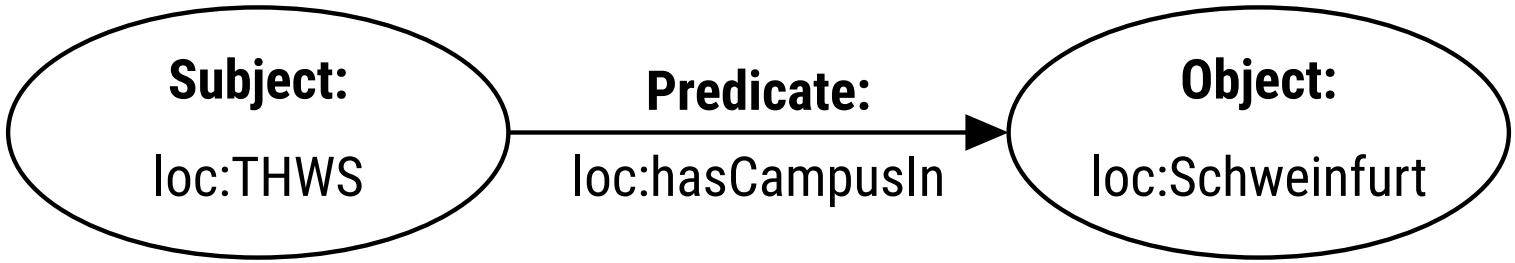
- Data is stored in form of triples (Subject,Predicate,Object)
- All entities are described by Uniform Resource Identifiers (URI), using prefix notation (e.g.: loc)



Knowledge Graphs: Data stored in Graphs

Resource Description Framework (RDF):

- Data is stored in form of triples (Subject,Predicate,Object)
- All entities are described by Uniform Resource Identifiers (URI), using prefix notation (e.g.: loc)



Representation in Turtle syntax:

```
@prefix loc: <www.locationdemo.de/location/ontology#> .

loc:THWS loc:hasCampusIn loc:Schweinfurt .
```

RDF, RDFS and OWL: Relation to Formal Logic

@prefix rdf : < http://www.w3.org/1999/02/22-rdf-syntax-ns# > .
@prefix rdfs : < http://www.w3.org/2000/01/rdf-schema# > .
@prefix owl : < http://www.w3.org/2002/07/owl# > .

RDF and RDF Schema (RDFS):

- **Class memberships:** City(Schweinfurt)

  loc : Schweinfurt rdf : type loc : City .
- **Subclass hierarchies:** $\forall x : \text{City}(x) \rightarrow \text{Location}(x)$

  loc : City rdfs : subClassOf loc : Location .

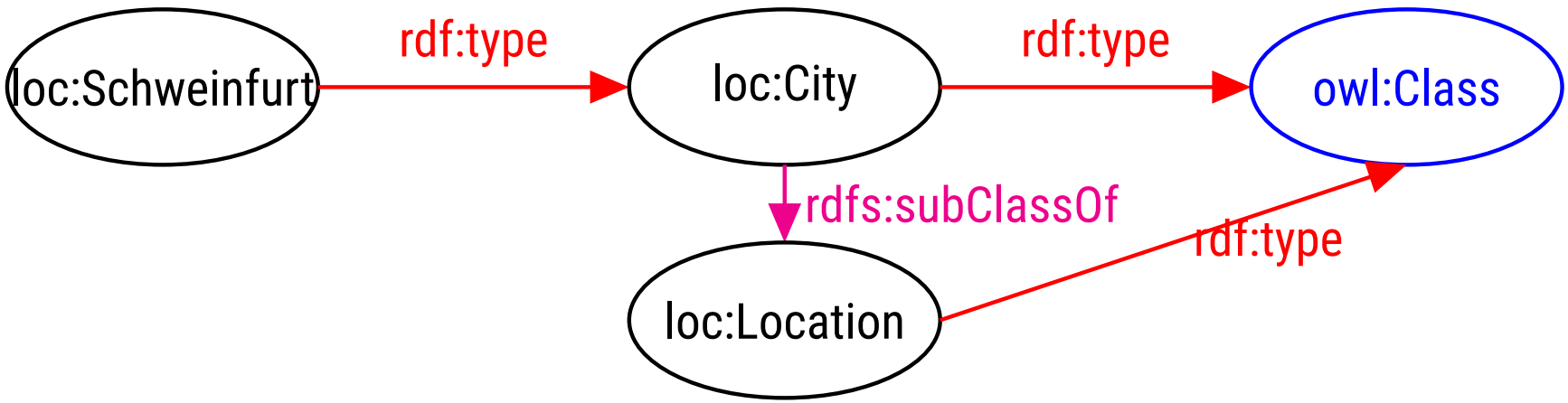
Web Ontology Language (OWL)

- **Classes:** Class(City)

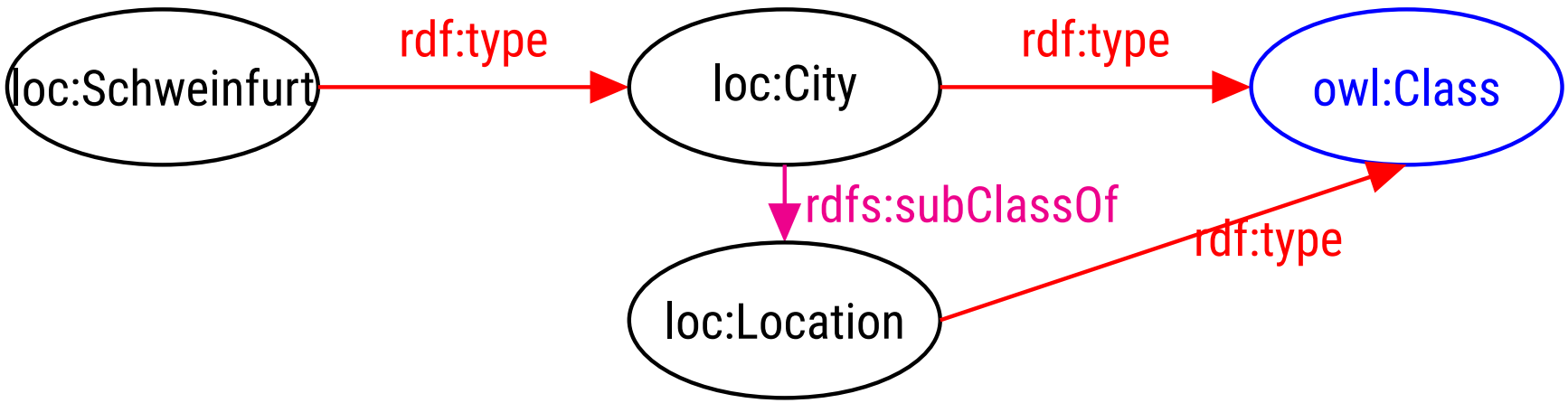
  loc : City rdf : type owl : Class .
- **Object properties:** Property(hasCampusIn)

  loc : hasCampusIn rdf : type owl : ObjectProperty .

Inference Example:
RDFS Subclass Relation



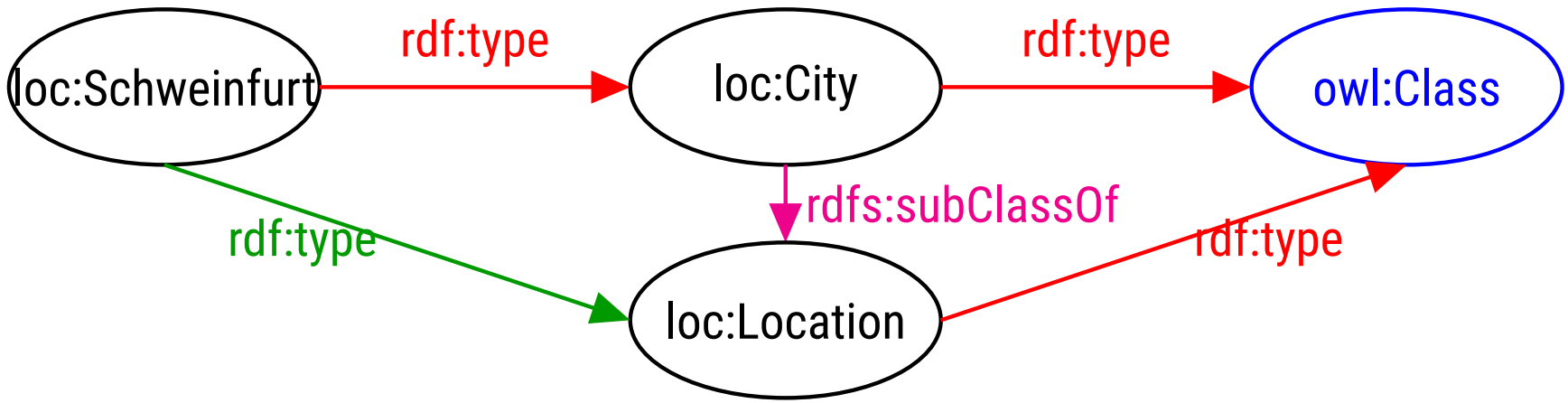
Inference Example: RDFS Subclass Relation



We apply the built-in rule of RDFS:

$$\forall x, y, z: \quad \text{rdf:type}(x, y) \cap \text{rdfs:subClassOf}(y, z) \rightarrow \text{rdf:type}(x, z)$$

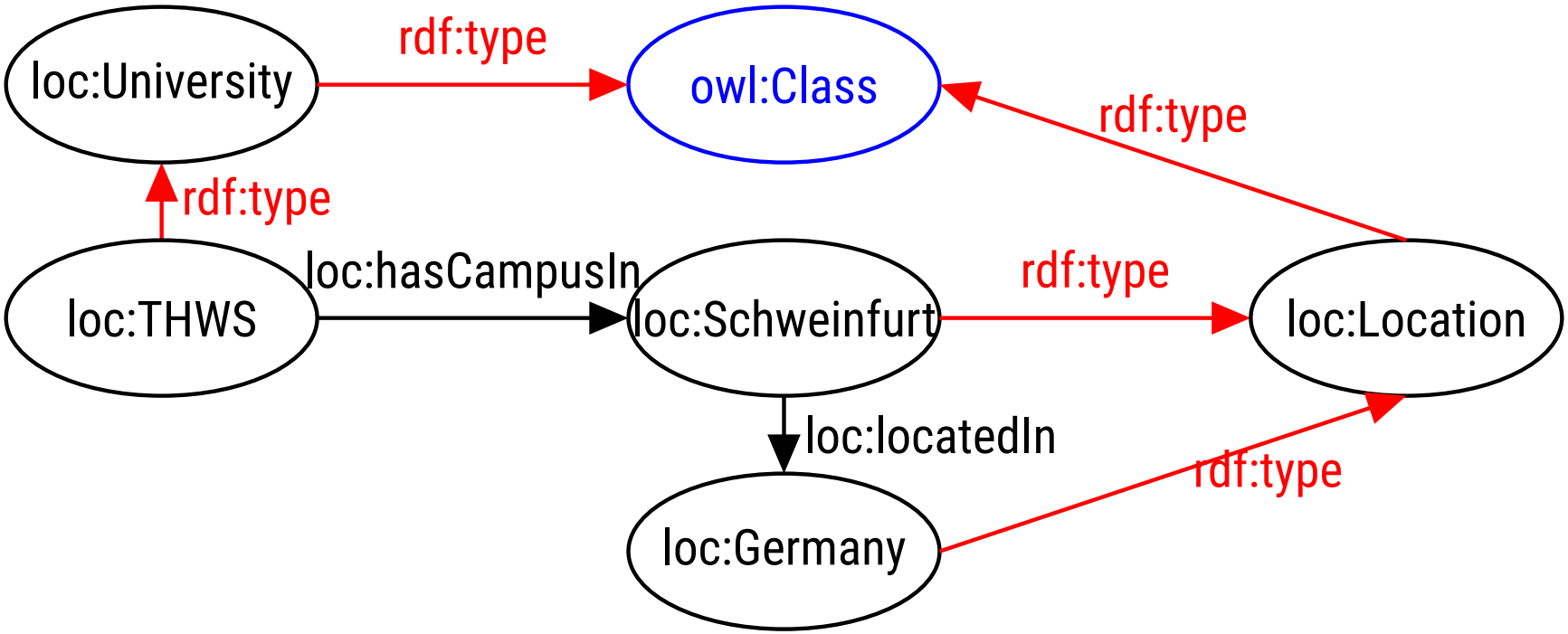
Inference Example: RDFS Subclass Relation



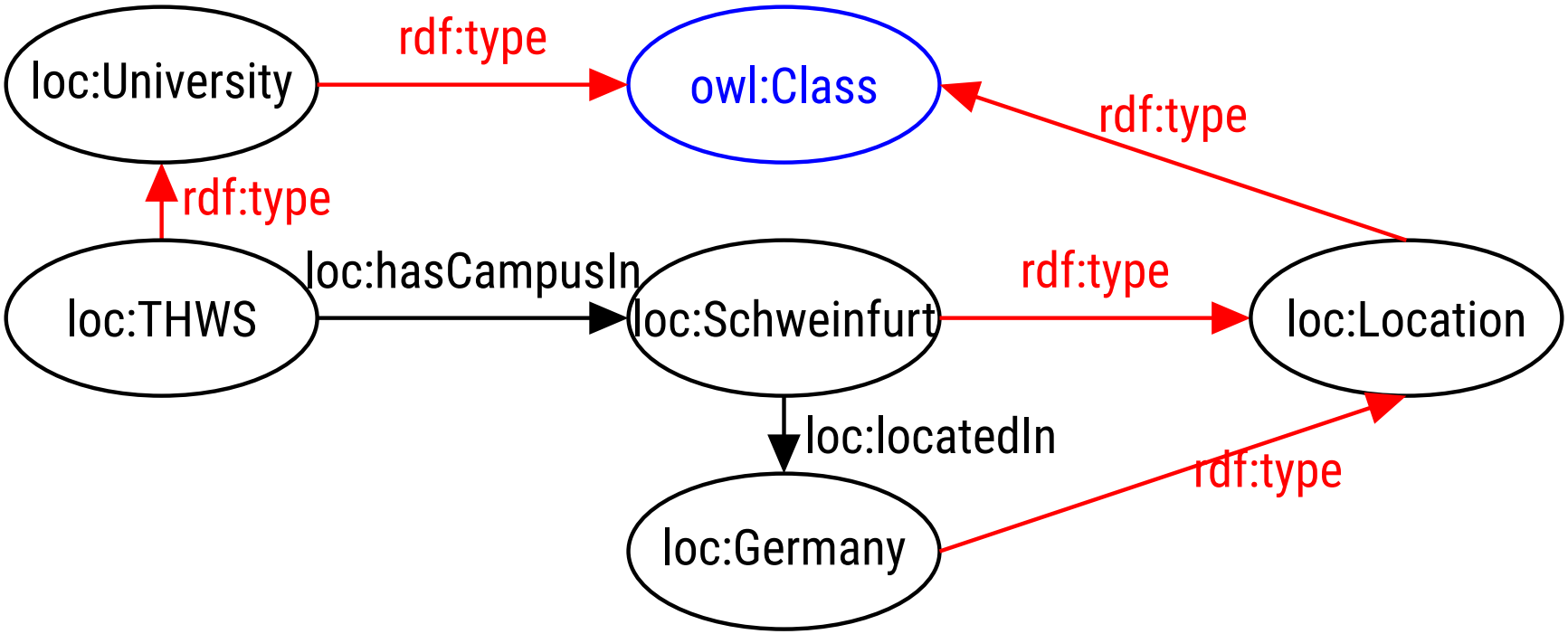
We apply the built-in rule of RDFS:

$$\forall x, y, z: \quad \text{rdf:type}(x, y) \cap \text{rdfs:subClassOf}(y, z) \rightarrow \text{rdf:type}(x, z)$$

Inference Example:
OWL Role Inclusion Axiom



Inference Example: OWL Role Inclusion Axiom



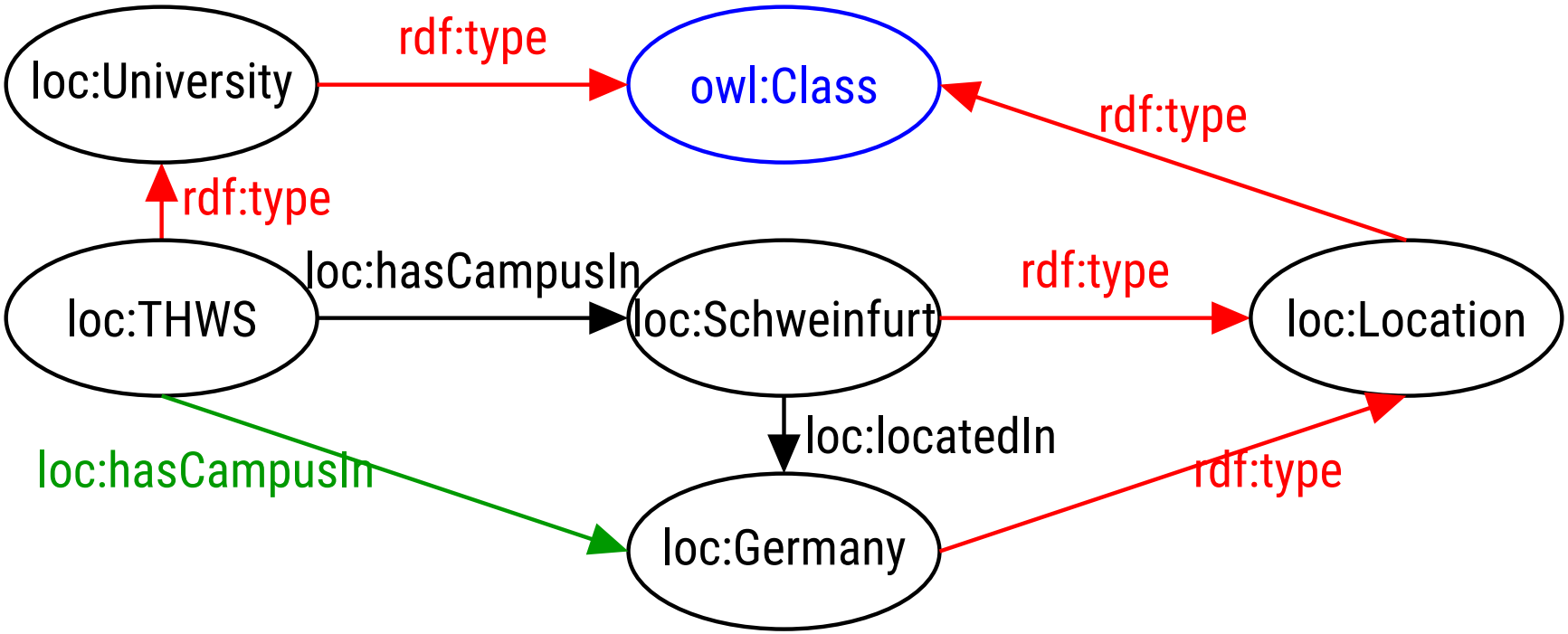
Let us infer using a role inclusion axiom defined by hand:

$$\forall x, y, z : (\text{hasCampusIn}(x, y) \cap \text{locatedIn}(y, z)) \rightarrow \text{hasCampusIn}(x, z)$$

expressed in owl as:

loc:hasCampusIn owl:propertyChainAxiom
(loc:hasCampusIn loc:locatedIn) .

Inference Example: OWL Role Inclusion Axiom



Let us infer using a role inclusion axiom defined by hand:

$$\forall x, y, z : (\text{hasCampusIn}(x, y) \cap \text{locatedIn}(y, z)) \rightarrow \text{hasCampusIn}(x, z)$$

expressed in owl as:

loc:hasCampusIn owl:propertyChainAxiom
(loc:hasCampusIn loc:locatedIn) .

Demonstration: Knowledge Graph for THWS

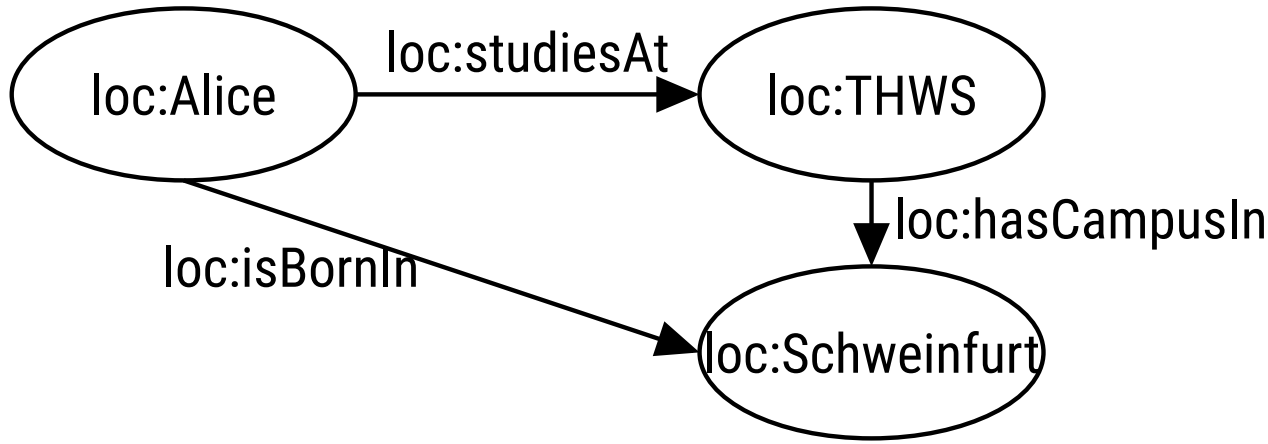
- Link to Protege Project (can be downloaded in turtle format): https://webprotege.stanford.edu/
- Link to WebVOWL (need to upload ontology in turtle format): https://service.tib.eu/webvowl/
- Link to Colab Notebook: drive.google.com/

Limitation of OWL Expressivity

Example: Estimate whether a student lives at her parents

$$\forall x, y, z : \big(\mathrm{studiesAt}(x, y) \cap \mathrm{hasCampusIn}(y, z) \cap \mathrm{isBornIn}(x, z)\big) \rightarrow \mathrm{livesAtParents}(x)$$

This amounts to looking for patterns like this:



Limitation of OWL (and other Description Logics)

- Cannot represent the formula without enlarging the ontology
- Uncertainties are not supported in classical logics

SPARQL **Queries**

The formula

$$\forall x, y, z : \big(\text{studiesAt}(x, y) \cap \text{hasCampusIn}(y, z) \cap \text{isBornIn}(x, z)\big) \rightarrow \text{livesAtParents}(x)$$

can be materialized by the SPARQL Query

```
INSERT{
        ?x rdf:type loc:livesAtParents .
}
WHERE{
        ?x loc:studiesAt ?y .
        ?y loc:hasCampusIn ?z .
        ?x loc:isBornIn ?z .
}
```

Grounding Tensors

Towards expressing SPARQL Queries in Tensor Networks

Definition (Grounding Tensor)

Given a specific world X_W, with a set of objects A, the grounding of a formula f with n arguments is the tensor

$$f|_{x_W} : \bigtimes_{l \in [n]} A \to [2]$$

defined as

$$f|_{x_W}\left(a_{o_0}, \ldots, a_{o_{n-1}}\right) = \begin{cases} 1 & \text{if } f\left(a_{o_0}, \ldots, a_{o_{n-1}}\right) = 1 \text{ given the world } X_W \\ 0 & \text{else} \end{cases}$$

.

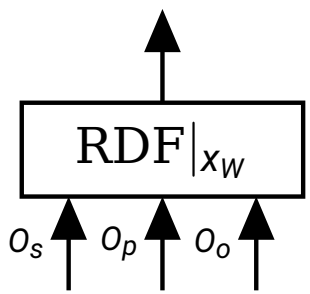
Knowledge Graphs as grounding tensors

Knowledge Graphs $\mathrm{KG}|_{x_W}$ are worlds X_W, which are fully specified by the RDF formula. Having a set of variables A we represent a Knowledge Graph $\mathrm{KG}|_{x_W}$ by the tensor

$$\mathrm{RDF}|_{x_W} : A \times A \times A \to [2]$$

where

$$\mathrm{RDF}|_{x_W}(O_s, O_p, O_o) = \begin{cases} 1 & \text{if triple } \langle O_s, O_p, O_o \rangle \text{ is in Knowledge Graph } \mathrm{KG} \\ 0 & \text{else} \end{cases}$$

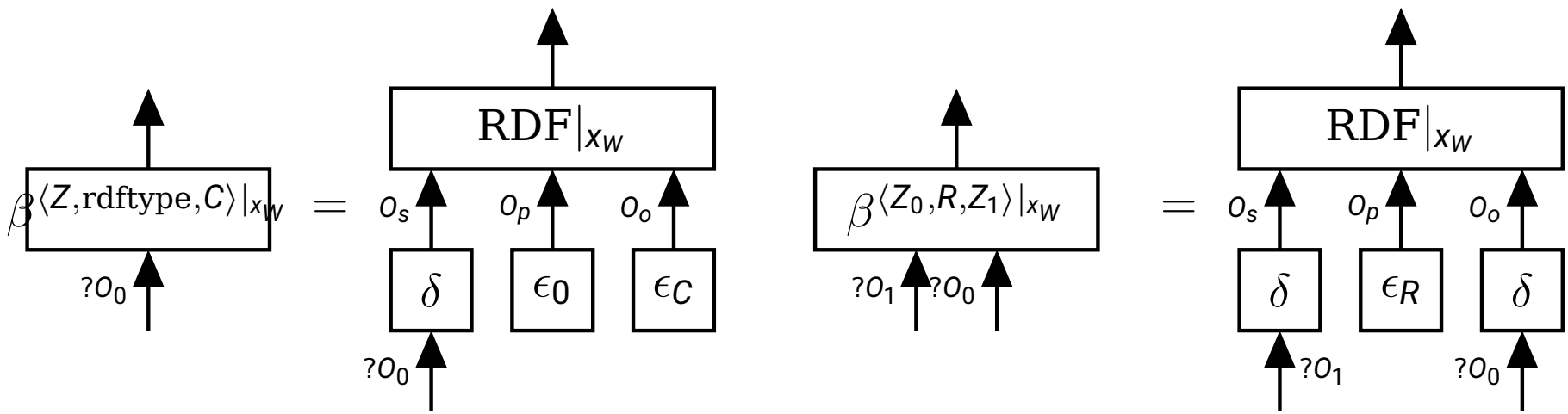


Basic Graph Patterns

Basic Graph Patterns are restrictions of the RDF formula on specific arguments, for example

- Unary triple pattern with one variable, representing a formula with a single projection variable. For example: $\langle Z, \mathrm{rdftype}, C \rangle$
- Binary triple pattern with two variables, representing a formula with two projection variables. For example: $\langle Z_0, R, Z_1 \rangle$

Their grounding tensors are slices of the RDF



Statistical Models of Knowledge Graphs

Limitation of Knowledge Graphs:

- Handling of Uncertainty: How to reason given uncertain knowledge?
- Expressivity of Logics: Which relations can be modelled?

Knowledge Graph
Data described in logics
Logical reasoning about
connectivity

Statistical ("Graphical") Models
Statistical dependencies of links
Probabilistic reasoning about
samples

Connectivity to Samples: Towards building statistical models

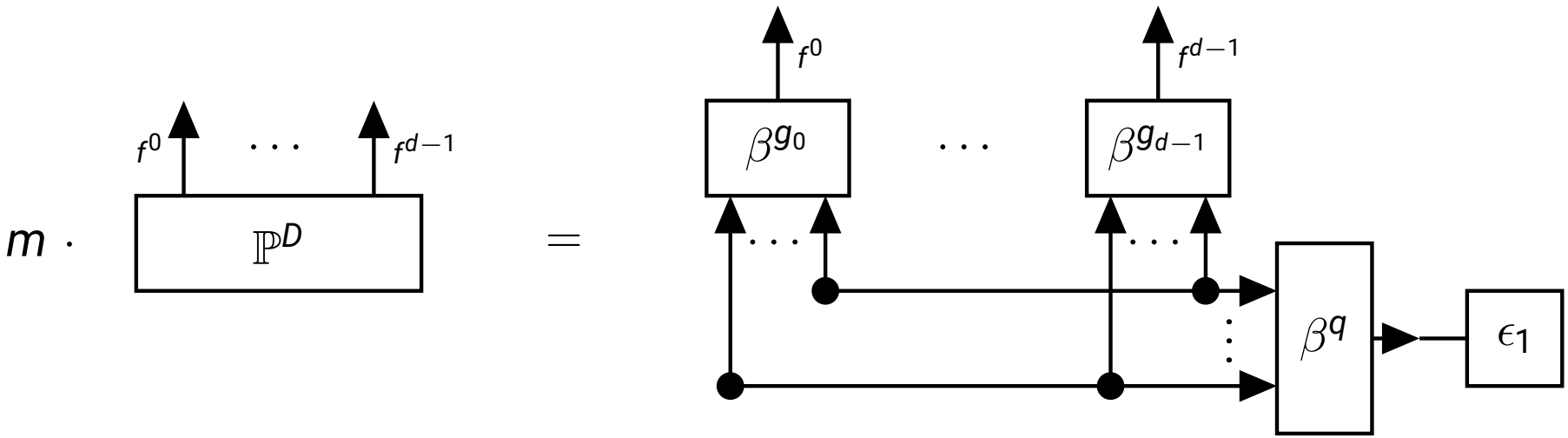
- Identify subgraphs of the Knowledge Graph to be interpreted as independent samples
- Learn and infer graphical models to reason about subgraphs

Extraction of samples for statistical models

The extraction is specified by

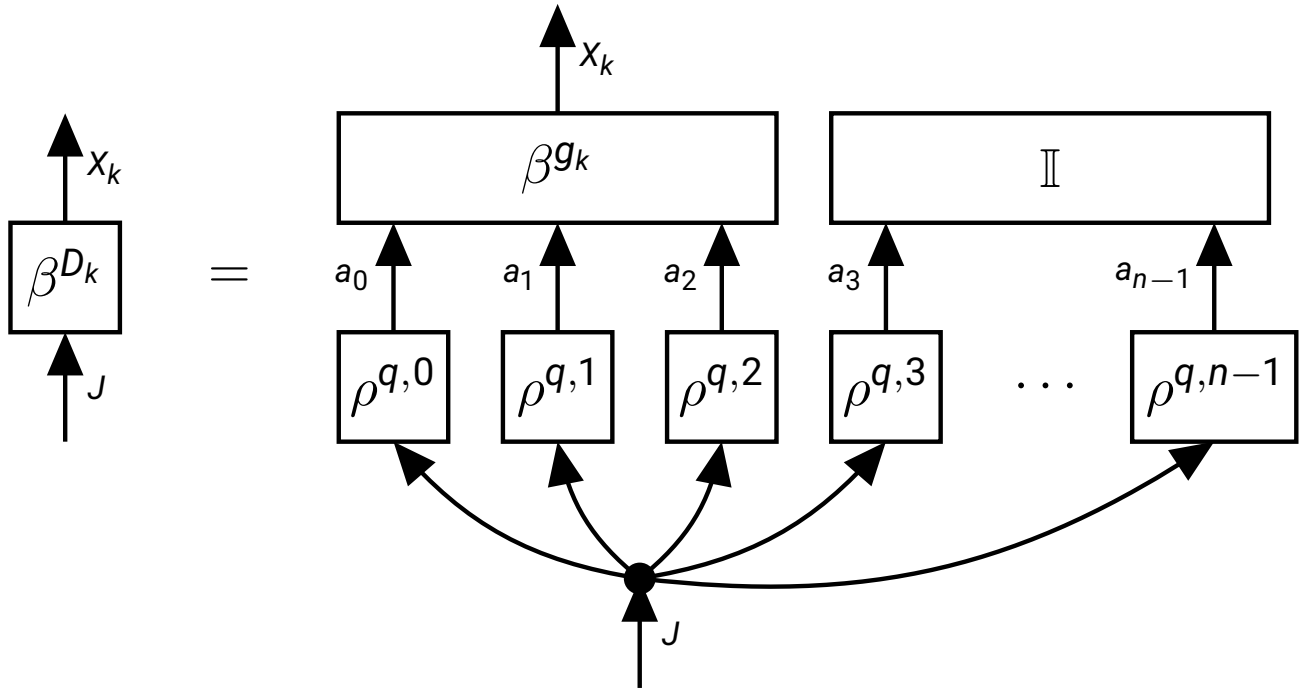
- **Extraction query** q specifying the conditions on a pair of individuals to represent a sample
- **Atom queries** g_k extracting the satisfaction of the atom for each pair of individuals

By contraction we get an empirical distribution by



Representation in Data Cores

Having a CP Decomposition of $q|_{x_W}$ we build datacores by



and have a representation

$$m \cdot \left\langle \mathbb{P}^D \right\rangle_{\left[f^0,\ldots,f^{d-1}\right]} = \left\langle \beta^{D_0}, \ldots, \beta^{D_{d-1}} \right\rangle_{\left[f^0,\ldots,f^{d-1}\right]} .$$

This amounts to a CP Decomposition of $\mathbb{P}^D$, which **can introduce storage overheads**!

Statistical Models of Knowledge Graphs

Learning:

- Extract samples based on the extraction query and the atom queries
- Train a Markov Logic Model based on neuro-symbolic architectures and parameter estimation

Inference:

- Estimate the probability of missing links (by modification of atom truths)
- Generate Knowledge Graphs based on samples of the statistical model