

# IV-Applications Statistical Models of Knowledge Graphs

Logik für Erklärbare KI: Technische Einführung in das ENEXA Projekt

Maria Schnödt-Fuchs, Alex Goëßmann

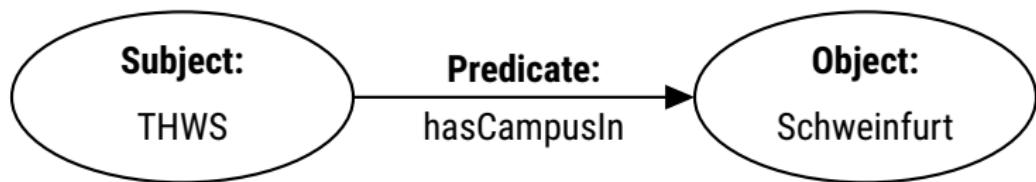
Funded by the  
European Union



15.+16. July, 2024

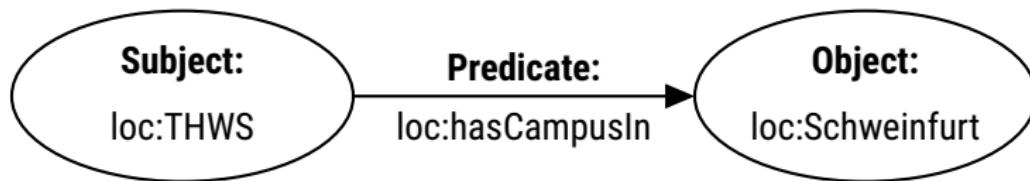
Resource Description Framework (RDF):

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



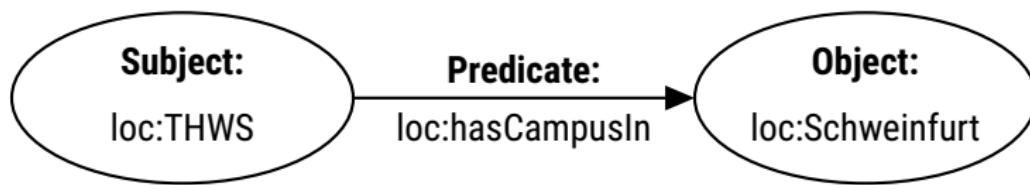
Resource Description Framework (RDF):

- ▶ 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)



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#> .
```

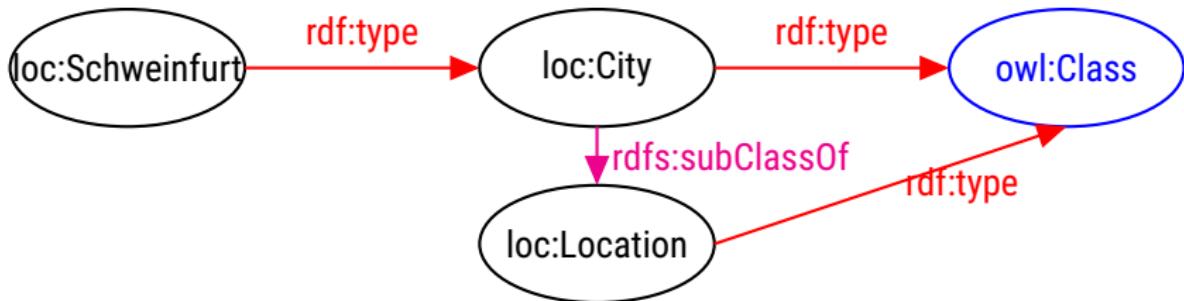
## RDF and RDF Schema (RDFS):

- ▶ **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 .

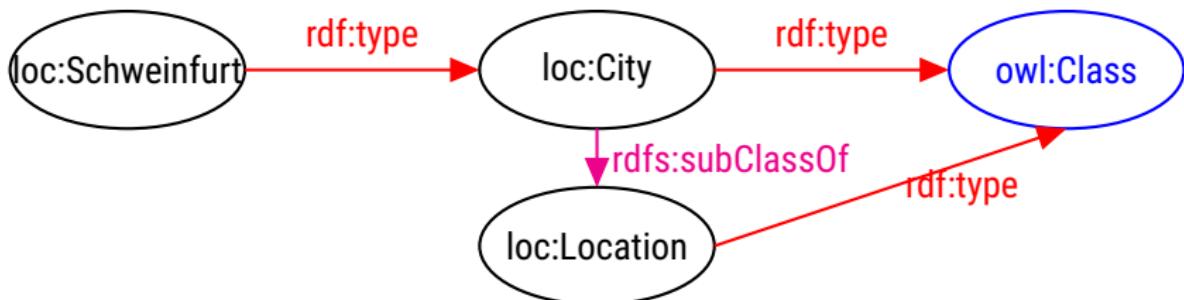
## Web Ontology Language (OWL)

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

# Inference Example: RDFS Subclass Relation



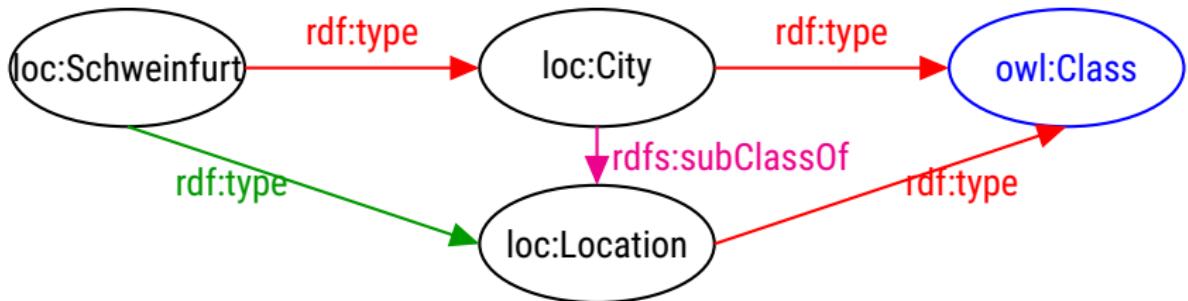
# Inference Example: RDFS Subclass Relation



We apply the built-in rule of RDFS:

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

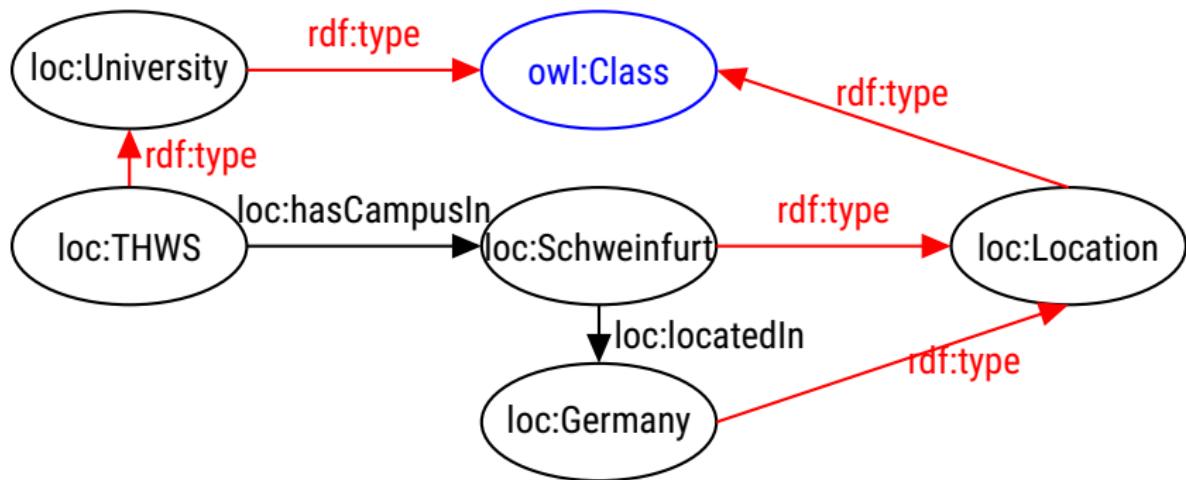
# Inference Example: RDFS Subclass Relation



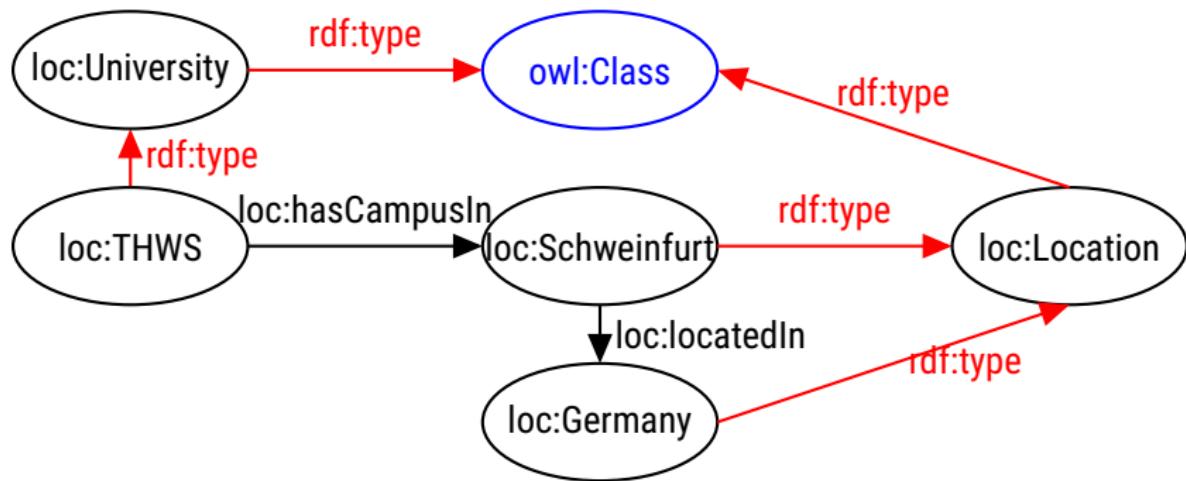
We apply the built-in rule of RDFS:

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

# Inference Example: OWL Role Inclusion Axiom



# Inference Example: OWL Role Inclusion Axiom



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

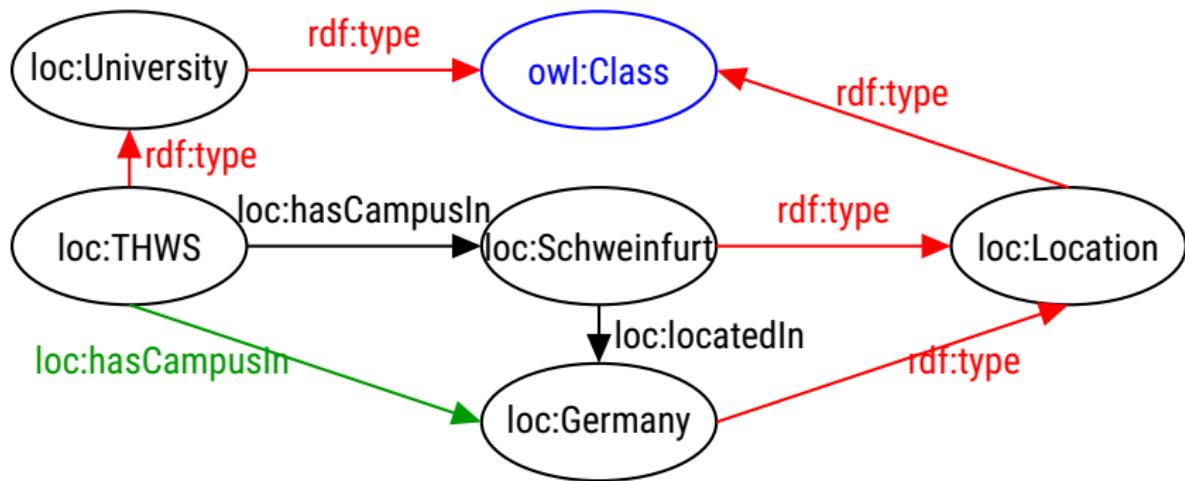
$\forall x, y, z : (\text{hasCampusIn}(x, y) \cap \text{locatedIn}(y, z)) \rightarrow \text{hasCampusIn}(x, z)$

expressed in owl as:

loc:hasCampusIn owl:propertyChainAxiom

(loc:hasCampusIn loc:locatedIn)

# 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)

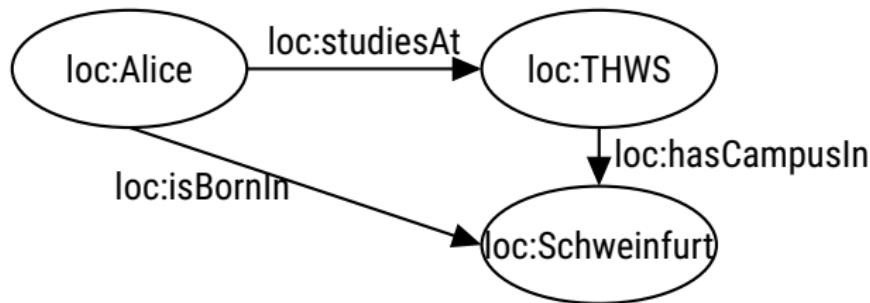
- ▶ Link to Protege Project (can be downloaded in turtle format):  
<https://webprotege.stanford.edu/>
- ▶ Link to WebOWL (need to upload ontology in turtle format):  
<https://service.tib.eu/webowl/>
- ▶ Link to Colab Notebook: [drive.google.com/](https://drive.google.com/)

# Limitation of OWL Expressivity

**Example:** Estimate whether a student lives at her parents

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

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

The formula

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

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 .
}
```

Towards expressing SPARQL Queries in Tensor Networks

## Definition (Grounding Tensor)

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

$$f|_W : \bigtimes_{l \in [n]} A \rightarrow [2]$$

defined as

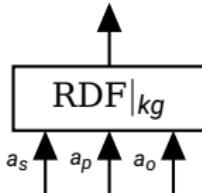
$$f|_W (a_0, \dots, a_{n-1}) = \begin{cases} 1 & \text{if } f(a_0, \dots, a_{n-1}) = 1 \text{ given the world } W \\ 0 & \text{else} \end{cases} .$$

Knowledge Graphs  $kg$  are worlds  $W$ , which are fully specified by the RDF formula. Having a set of variables  $A$  we represent a Knowledge Graph  $kg$  by the tensor

$$\text{RDF}|_{kg} : A \times A \times A \rightarrow [2]$$

where

$$\text{RDF}|_{kg}(a_s, a_p, a_o) = \begin{cases} 1 & \text{if triple } \langle a_s, a_p, a_o \rangle \text{ is in Knowledge Graph } kg \\ 0 & \text{else} \end{cases}$$

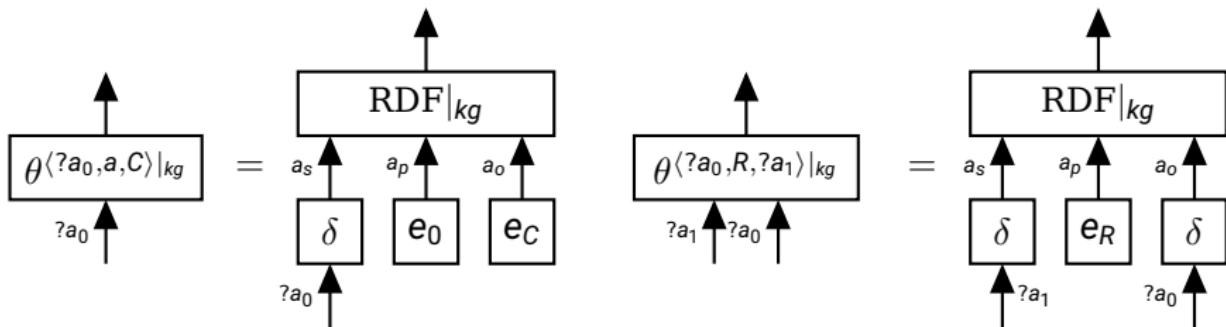


# Basic Graph Patterns

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:  $\langle ?a_0, a, C \rangle$
- ▶ Binary triple pattern with two variables, representing a formula with two projection variables. For example:  $\langle ?a_0, R, ?a_1 \rangle$

Their grounding tensors are slices of the RDF



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

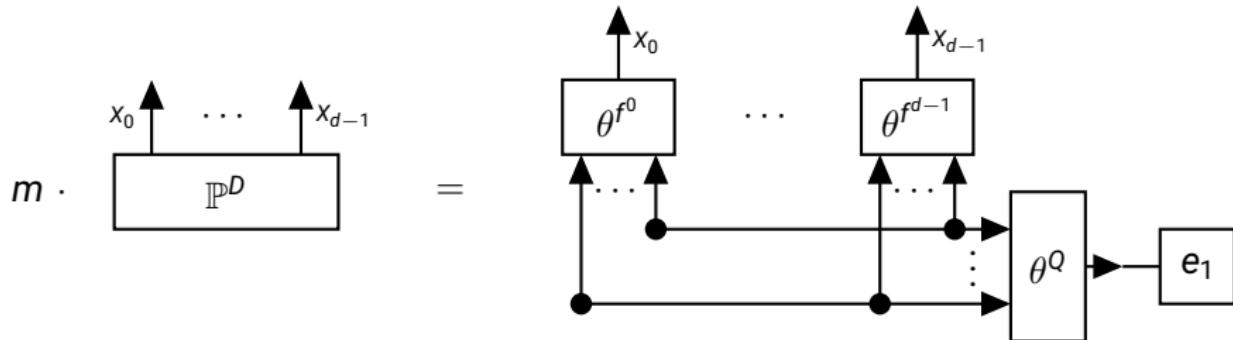
### Connectivity to Samples: Towards building statistical models

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

The extraction is specified by

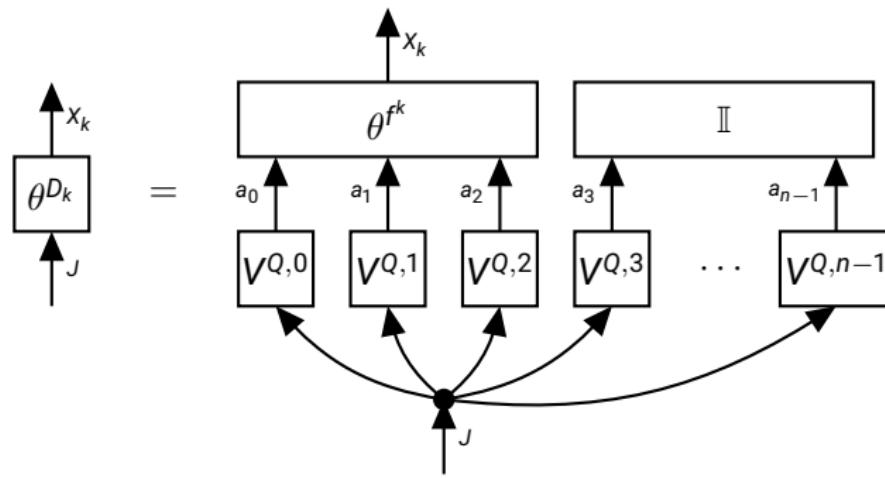
- ▶ **Extraction query**  $Q$  specifying the conditions on a pair of individuals to represent a sample
- ▶ **Atom queries**  $f^k$  extracting the satisfaction of the atom for each pair of individuals

By contraction we get an empirical distribution by



# Representation in Data Cores

Having a CP Decomposition of  $Q|_{kg}$  we build datacores by



and have a representation

$$m \cdot \mathcal{C} \left( \{\mathbb{P}^D\}, \{X_0, \dots, X_{d-1}\} \right) = \mathcal{C} \left( \{\theta^{D_0}, \dots, \theta^{D_{d-1}}\}, \{X_0, \dots, X_{d-1}\} \right).$$

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

## 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