

Semantisches Wissensmanagement im Unternehmen: Konzepte, Technologien, Anwendungen

Prof. Dr. Stefan Linus Zander

Kapitel 3.3: Semantische Suche

Didaktischer Aufbau der Einheit

Dieses Kapitel ist **zweigeteilt**:

- **Teil 1** behandelt **Syntax** und **syntaktische Beschreibung** von Abfragen
- **Teil 2** beschäftigt sich mit dem **strukturellen Aufbau** von Abfragen aus konzeptueller Sicht

Warum ist eine derartige Betrachtungsweise sinnvoll ?

Vorbemerkungen

SMW verfügt über eine **eigene Anfragesprache**

- Anlehnung an [Wiki-Syntax](#)
- interne Umsetzung der Anfragen in Abfragen für den internen Speicher (z.B. SQL bei Verwendung von MySQL als internen Speicher)

Formale Semantik der Anfrage

- durch Abbildung in **OWL-DL-Klassenkonstrukte** gegeben
~> d.h. eine Anfrage ermittelt Instanzen einer entsprechenden OWL-Klasse
- Nutzung der Anfragesprache auf Spezialseite oder in **inline queries**

Einbindung in Wiki-Seiten

- Anzeige der [Abfrageergebnisse](#) auf der Wiki-Seite stets aktuell

Polynomielle Komplexität der Anfragebeantwortung

- Anfragesprache unterstützt daher (wie auch OWL DL) keine [benannten Variablen](#)
(Beispiel: Personen, die in Stadt_x geboren wurden und in Stadt_x gestorben sind) (mindestens NP-hart)

Part 1: Syntax and Semantics

The #ask Query Language (AQL)

Semantic MediaWiki includes an easy-to-use query language called **AQL – #ask Query Language**, which enables users to access the wiki's knowledge. The **syntax** is similar to the syntax of **annotations**. AQL can be used on the **special page** `Special:Ask`, in **concepts**, and in **inline queries**.

Semantic queries specify two things:

1. Which **pages** or **subobjects** to select
2. What **information to display** about those pages

All queries must state some **conditions** that describe what is asked for.

Pages can be selected by

- **name**,
- **namespace**,
- **category**, and most importantly by
- **property values**.

Example

```
[[Located in::Germany]]
```

selects all pages with property `Located in` and value `Germany`.

Anatomy of AQL

Semantic MediaWiki defines its own **query language** called **AQL – #ask Query Language**. It allows to **retrieve** pages (or subobjects) based on the information they contain.

AQL allows to **query** for

- **pages** (e.g. the wiki pages of all running projects)
- **property values** (e.g. the full names of all employees of an organisation)
- **subobjects** (subgraphs embedded in pages ~> will be discussed later in this course)

Main Idea

To ask for pages with some specified annotations in order to retrieve additional information from those pages. Those annotations in question are used as query conditions.

Structure

#ask Queries consist of **four parts**

1. The **#ask** **parser function**
2. **Query conditions**, i.e., what pages to select
3. **Printout statements**, i.e., data to be displayed
4. **Display options**, i.e., how data are displayed

Example

```

{{#ask:
  [[Category:City]]
  [[Located in::Germany]]
  |?Population
  |?Area#km² = Size in km²
  |format=ul
}}
  
```

<!-- Parser Function -->
 <!-- Query Conditions -->
 <!-- Printout Statements -->
 <!-- Display Options -->

A Word about the Condition Syntax...

The **markup text** for formulating **query conditions** is exactly similar to the annotations embedded in wiki pages.

The syntax for asking for pages that satisfy some condition is exactly the syntax for explicitly asserting that this condition holds.

The following queries show what this means:

- `[[Category:Actor]]` gives all pages directly or indirectly (through a sub-, subsub-, etc. category) in the category.
- `[[Born in::Boston]]` gives all pages annotated as being about someone born in Boston.
- `[[Height::180cm]]` gives all pages annotated as being about someone having a height of 180cm.

Query Algebra – Part 1: Conjunctions

Conditions can be **combined**.

```
[[Category:Actor]] [[Born in::Boston]] [[Height::180cm]]
```

Similar, more readable notation

```
[[Category:Actor]]  
[[Born in::Boston]]  
[[Height::180cm]]
```

When using **many conditions** in **one query**, the result is **narrowed down** to those pages that meet **all the requirements**.
Thus we have a **logical AND**.

Note that queries only return the articles that are **positively known to satisfy the required properties**
⇒ If there is no property for the height of some actor, that actor page will not be selected.

Lexical Peculiarities

SMW will ignore some characters such as trailing spaces or comma in numbers depending on the datatype used. SMW also treat synonymous page names as **identical resources** → "Semantic wiki", "Semantic_wiki" and "semantic wiki" all refer to the same page.

Query Algebra – Part 2: Disjunctions

Disjunctions are **OR-conditions** that allow several **alternative conditions** on query results.

A **disjunction** requires that at least one of the possible alternatives is satisfied (ie. **logical OR**).

Semantic MediaWiki has two ways of writing **disjunctions** in queries:

- The operator **OR** is used for taking the **union of two queries**
- The operator **||** is used for disjunctions in **property values, page names, and category names**

Examples

```
[[Born in::Boston]] OR [[Born in::New York]]
```

Describes people who were born in Boston **OR** New York

```
[[Born in::Boston|New York]]
```

The same query written in a more **concise form**

Note that **||** does not always offer an alternative to **OR**.

For example, `[[Born in::Boston]] OR [[Category:Actor]]` cannot be expressed with **||**.

OR operates on the query, not on a single element of the query. In the following query, the **category name** needs to be repeated:

```
[[Category:Actor]] [[Born in::Boston]] OR [[Category:Actor]] [[Born in::New York]]
```

Wildcards and Search Operators

Wildcards are written as `+` and allow **any value** for a given condition¹.

For example, `[[Born in::+]]` returns all pages that have any value for the property `Born in`.

Comparators are **special symbols** like `<` or `>`². They are placed after `::` in **property conditions**.

- `>>` and `<<`: "greater than" and "less than"
- `>` and `<`: "greater than or equal" and "less than or equal" by default
- `≥` and `≤`: "greater than or equal" and "less than or equal"
- `!`: "not" ("unequal")
- `~`: «like» comparison for texts and pages
- `!~`: «not like» comparison for texts and pages

When applying **comparators** to pages, then the **title of the page** (without namespace prefix) is used.

Comparators work only for **property values** and not for conditions on categories.

¹ Please note that `+` can only be used by itself¹.

² See https://www.semantic-mediawiki.org/wiki/Help:Search_operators

Search Features

SMW supports a number of additional search features, which are not discussed in detail here¹:

- [Search operators](#) shows how to refine search conditions and criteria using operators such as comparators or wildcards.
- [Unions \(OR\)](#) of results describes how [disjunctions](#) (OR-conditions) can be used to combine query results on alternative conditions.
- [Single page restriction](#) section describes how to directly select some pages, or pages from a given namespace.
- [Specify range of pages](#)
- [Use namespace restrictions](#)
- [Subqueries and property chains](#)
- [Work with value substitutions](#) describes how templates and variables can be used in a query to substitute value components
- [Distance queries](#)

See https://www.semantic-mediawiki.org/wiki/Help:Selecting_pages

AQL Inverse Properties

Sometimes, it is necessary to **invert the direction of properties** in queries, in particular when asking for pages that contain a **subobject**.

Definition **Inverse properties** do not ask for pages that contain a matching annotation but for the **object value** of the annotation on pages where the property is used.

Quelle: eigene Definition angelehnt an https://www.semantic-mediawiki.org/wiki/Help:Inverse_properties

Example

```
{{#ask: [[has capital-::Germany]] }}
```

- In this example, we do not ask for the page that contains a property with the given value but rather for the **value of the property** **has capital** entered to the page **Germany**.
- In consequence **has capital-** has the meaning **is capital of**.

⇒ Inverse properties can be used in all SMW interfaces that take properties, but not when adding data to a page

Part 2: Formulating Query Conditions

How to Formulate Query Conditions

Example

The following domain knowledge is modelled in a Semantic MediaWiki

*"A research group has a number of employees being members of it.
These employees work in different projects, where each project has different topics it is concerned with."*

Question:

How can we satisfy the information need of displaying all the research topics a research group's members are associated with through their project work on the research group's wiki page?

...or in other words

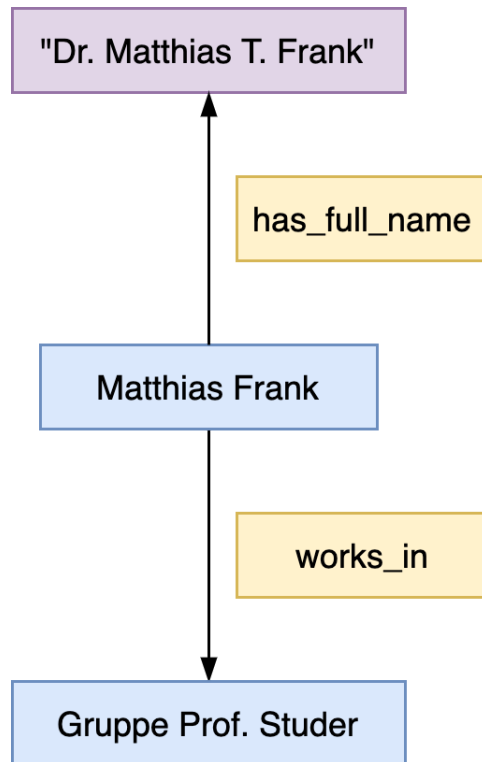
Which topics are a research group working on?¹

How can we model and satisfy this information need in Semantic MediaWiki ?

¹ Assuming that topic information is encoded on the project pages using, e.g., `has_topic::Smart_Data` statements.

Formulating Queries I

First consider the **structural composition** of the data model used to represent the universe of discourse (or an excerpt of it).



This **graphical representation** serves as basis for the formulation of **query conditions**.

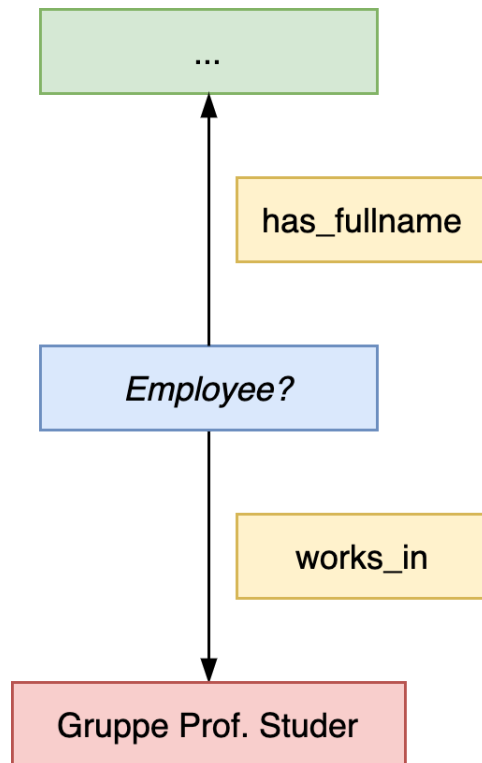
- Therefore, think of query conditions as
~> **graphs with conditional node values**.
- Also consider the **structural semantics** of involved elements.

Structural Semantics of the knowledge graph

- `Matthias Frank` is a wiki page in the `main` namespace
- The page holds two assertions
 - a `has_full_name`-property the value of which is a Literal (datatype: `Text`)
 - a `works_in`-property the value of which is the wiki page representing Prof. Studer's research group

Formulating Queries I (Part 2)

The structural composition of the knowledge graph can then be transformed in a **query graph**



The **query graph** then helps in formulating

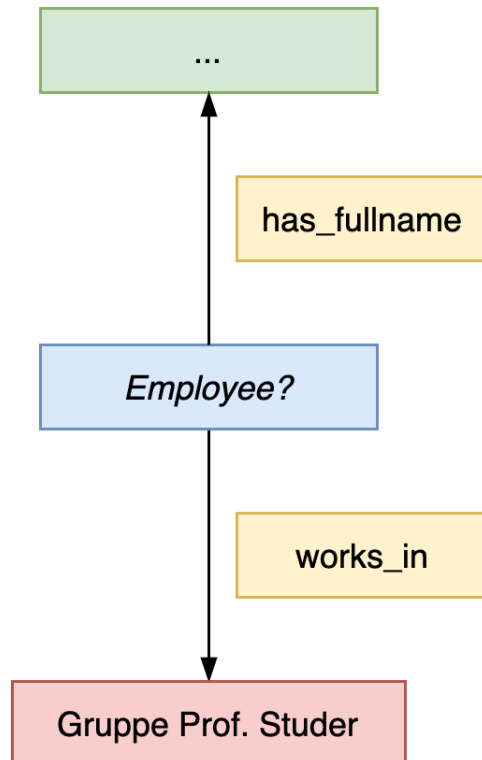
- query conditions and
- the structural components of the universe of discourse

Query Conditions and structural components

- Retrieve all pages that are...
 - ...of type `Employee` and
 - ...that participate in a `works_in`-relationship to the `Gruppe Prof. Studer` wiki page.
- From all matching pages...
 - ...retrieve the value (object) of the `has_full_name`-property

Formulating Queries I (Part 3)

Once the structural components of the query graph are clear, the query can be formulated using **AQL**



Example

```

{{#ask
  [[Category:Employee]]
  [[works_in::Gruppe Prof. Studer]]
  |?has_full_name = Fullname
  |format=ul
}}
  
```

Description

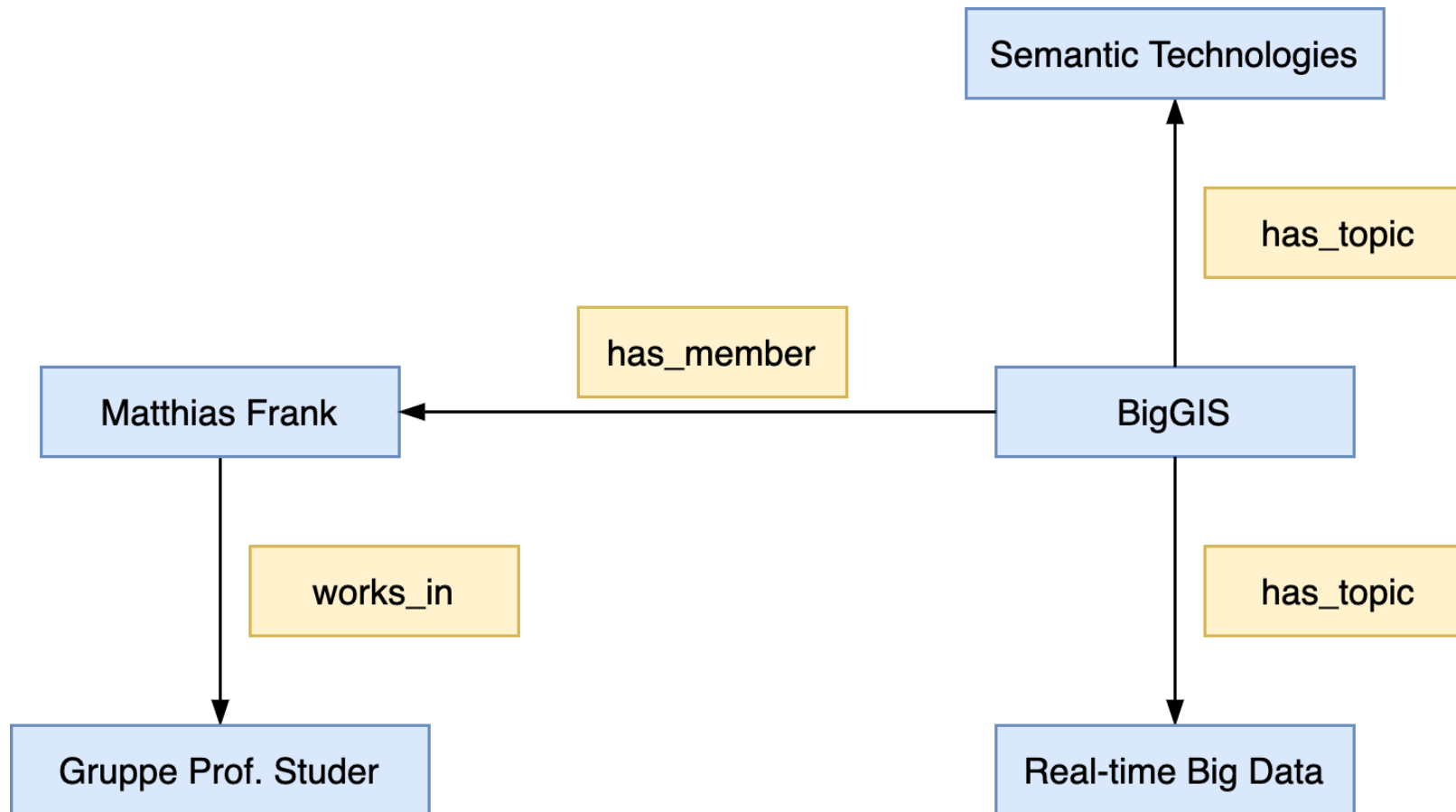
- Retrieve all pages that are...
 - ...of type `Employee` and
 - ...that participate in a `works_in`-relationship to the `Gruppe Prof. Studer` wiki page.
- From all matching pages...
 - ...retrieve the value (object) of the `has_full_name`-property

Explanatory Information

Consider the following conceptual **query graph semantics** of the previous example:

- yellow boxes represent **properties**;
 - properties need to be pre-determined in a query, i.e., you can, e.g., not ask which properties exist between two pages in the main namespace.
- the red box represents a **specific wiki page** in the main namespace
- the blue boxes represent **query variables** that are to be filled with the values in the course of **evaluating** the **query conditions**
- the green boxes are the **results** returned by processing the query and that are being displayed.

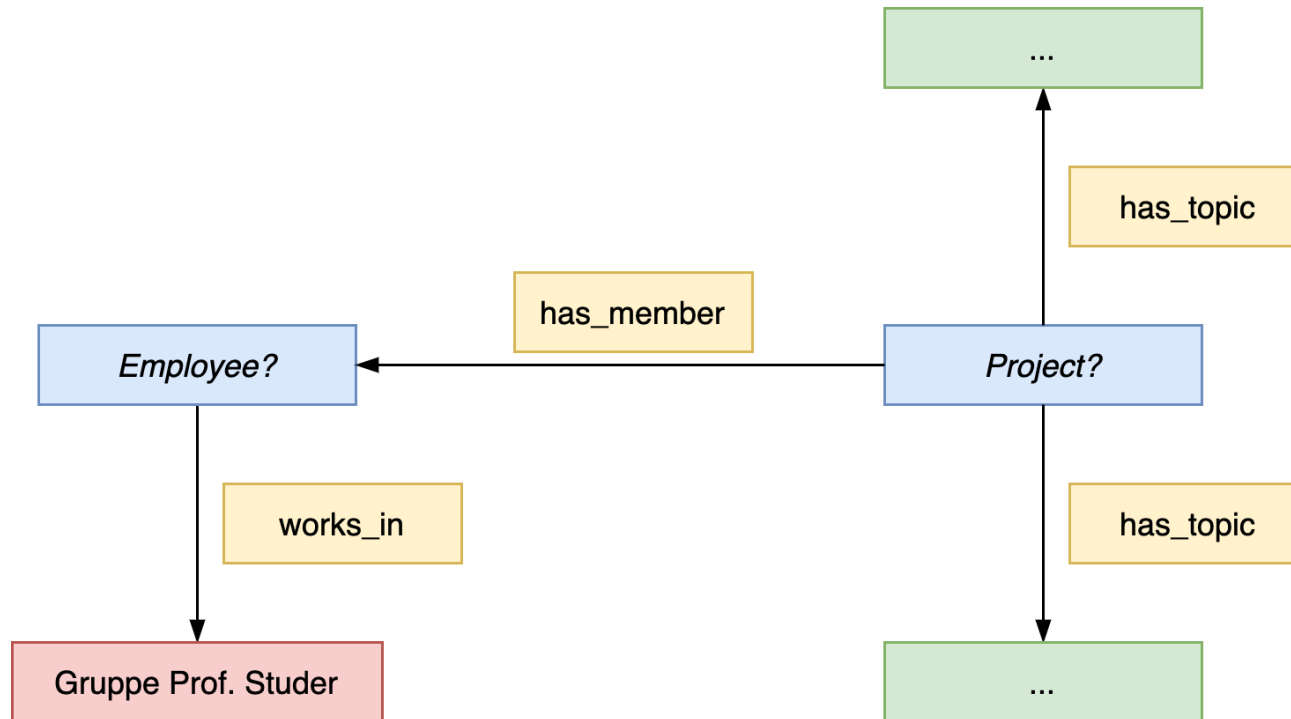
Formulating Complex Queries



- Consider the following excerpt of a **knowledge graph** representing the **group** an **employee** works in and a **project** she is member of together with the **topics** the project is concerned with.
- Be aware of the **structural semantics** (ie., how pages are linked together) that exists between employees, projects, and topics.

Formulating Complex Queries: The Query Graph

If we want to retrieve **all topics**, members of Prof. Studer's group are working on, the **query graph** looks as follows:



The **query graph** consists of two variables and thus two subgraphs

- the **project** subgraph
- the **employee** subgraph

It needs to be transformed into an **inner query** and an **outer query**¹

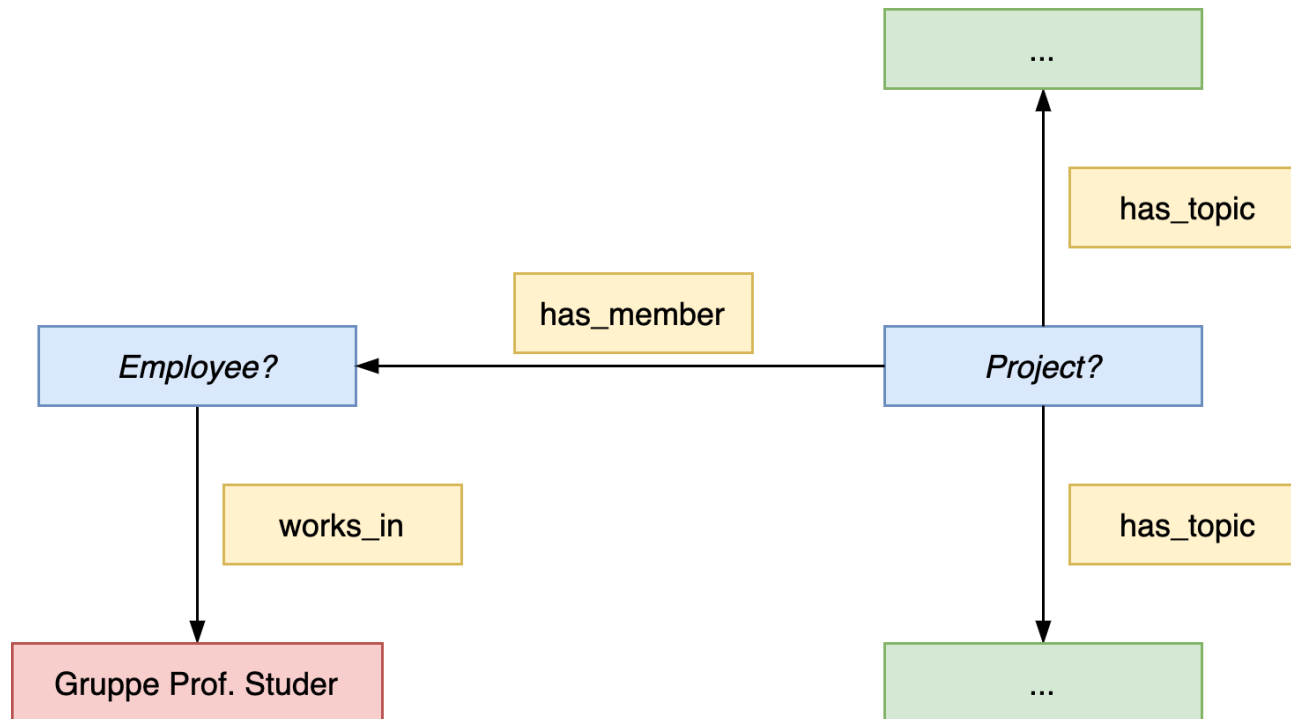
- the **inner query** represents specific **employees**²
- the **outer query** represents **projects** they are members of

¹ In order to determine the inner and outer query, interpret the query graph as projects with specific employees as members.

² The values which participate as conditions in the outer query

Formulating Complex Queries: The Query

If we want to retrieve **all topics**, members of Prof. Studer's group are working on, the **query graph** looks as follows:



The #ask Query¹

```

{{#ask: [[has_member::<q>[[works_in::{{PAGENAME}}}]]</q>]]
|?has_topic=
|mainlabel=-
|format=valuerank
}}

```

- The **inner query** asks for employees working in the group (cf. `works_in`)
- The **outer query** retrieves projects in which these employees participate (cf. `has_member`)

When formulating queries in Semantic MediaWiki, always consider the structural semantics of pages (ie., how pages are linked together via properties).

¹ Assuming, that the query is placed on the group's page.