Semantisches Wissensmanagement im Unternehmen: Konzepte, Technologien, Anwendungen

Prof. Dr. Stefan Linus Zander

Kapitel 4: Fortschrittliche Modellierungskonzepte in Semantic MediaWiki

Inhalte

1. Motivation
2. Einführendes Beispiel
3. Ein erster Lösungsansatz
4. Darstellung komplexer Sachverhalte mittels Subobjects
5. Abfrage von Subobjects
6. Tipps

Leitfragen

1. Wie lassen sich komplexe Inhalte in Wissensgraphen abbilden ?
2. Wie lassen sich n-äre Beziehungen in Semantic MediaWiki umsetzen ?

Einführendes Beispiel (1/2)

Nicht alle Aspekte eines Gegenstandsbereichs lassen sich durch Property-Value-Paare abbilden.

Beispiel

> *"Meryl Streep played Margaret Thatcher in The Iron Lady"*

Properties `played role` und `played in` mit folgenden Statements:

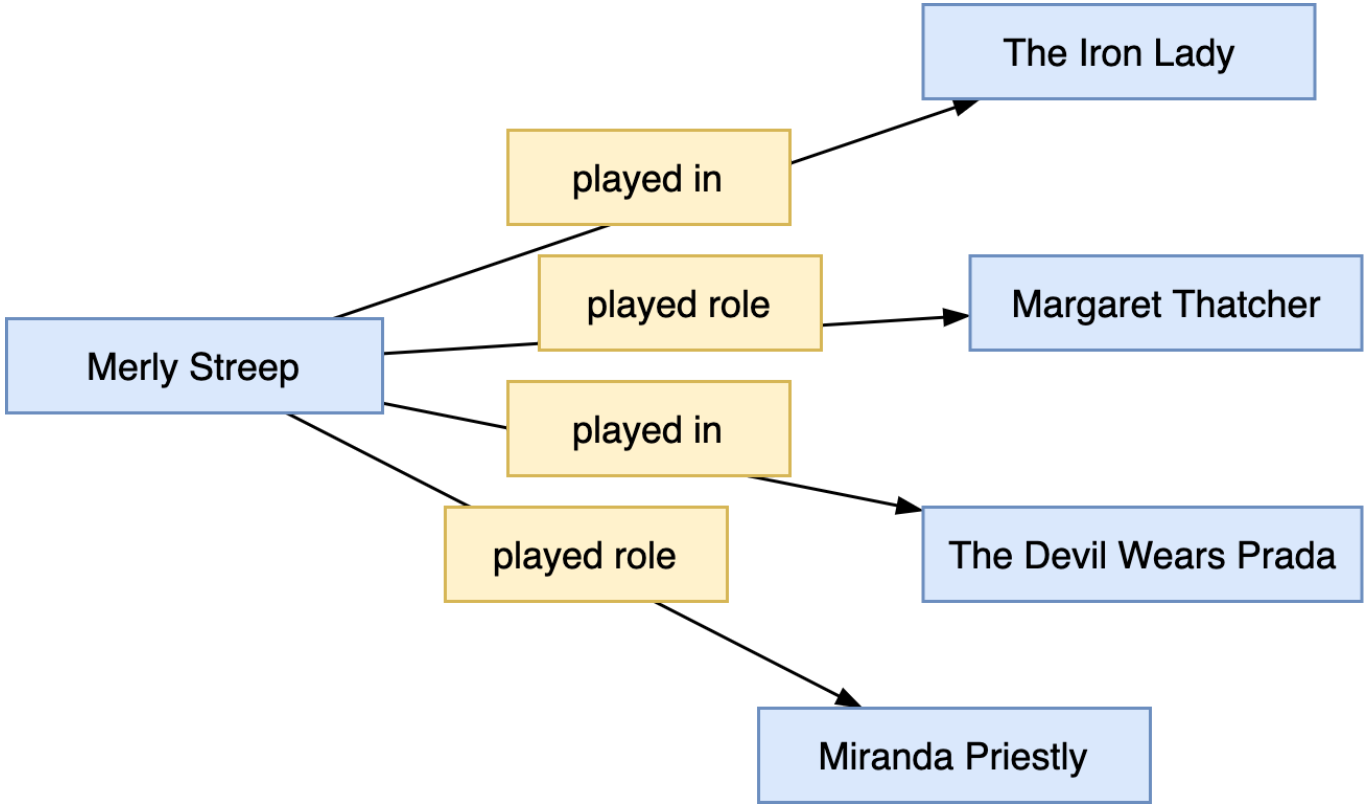
- Meryl Streep played in The Iron Lady
- Meryl Streep played role Margaret Thatcher

Allerdings existieren noch weitere Rollenbesetzungen von Meryl Streep:

- Meryl Streep played in The Devil Wears Prada
- Meryl Streep played role Miranda Priestly

Einführendes Beispiel (2/2)

Daraus resultiert der folgende konzeptuelle Graph:



Frage: Welche Rolle spielte nun Merly Streep im Film The Iron Lady ?

Ein erster Lösungsansatz

1. Erstellung einer neuen Seite mit dem Namen `Streep's role in The Iron Lady` mit folgenden Eigenschaften:

```
[[role::Margaret Thatcher]]
[[film::The Iron Lady]]
```

2. Hinzunahme der Annotation auf der Meryl Streep Seite:

```
[[played::Streep's role in The Iron Lady]]
```

Aufgabe:

1. Stellen Sie obigen Modellierungsansatz in einem konzeptuellen Graphen dar
2. Beurteilen Sie obigen Modellierungsansatz im Hinblick auf Flexibilität, Wartbarkeit, Datensparsamkeit. Beachten Sie insbesondere, dass Meryl Streep in Ihrer beruflichen Laufbahn in mehr als 89 Filmen (Quelle: IMDB.com) mitgewirkt hat.

Ein Lösungsansatz mit Subobjects

Wie Sie bereits aus Datenbanken wissen handelt es sich bei dem einführenden Beispiel modellierungstechnisch um **n-äre Beziehungen**.

Diese lassen sich in Semantic MediaWiki mittels Subobjects abbilden.

Beispiel:

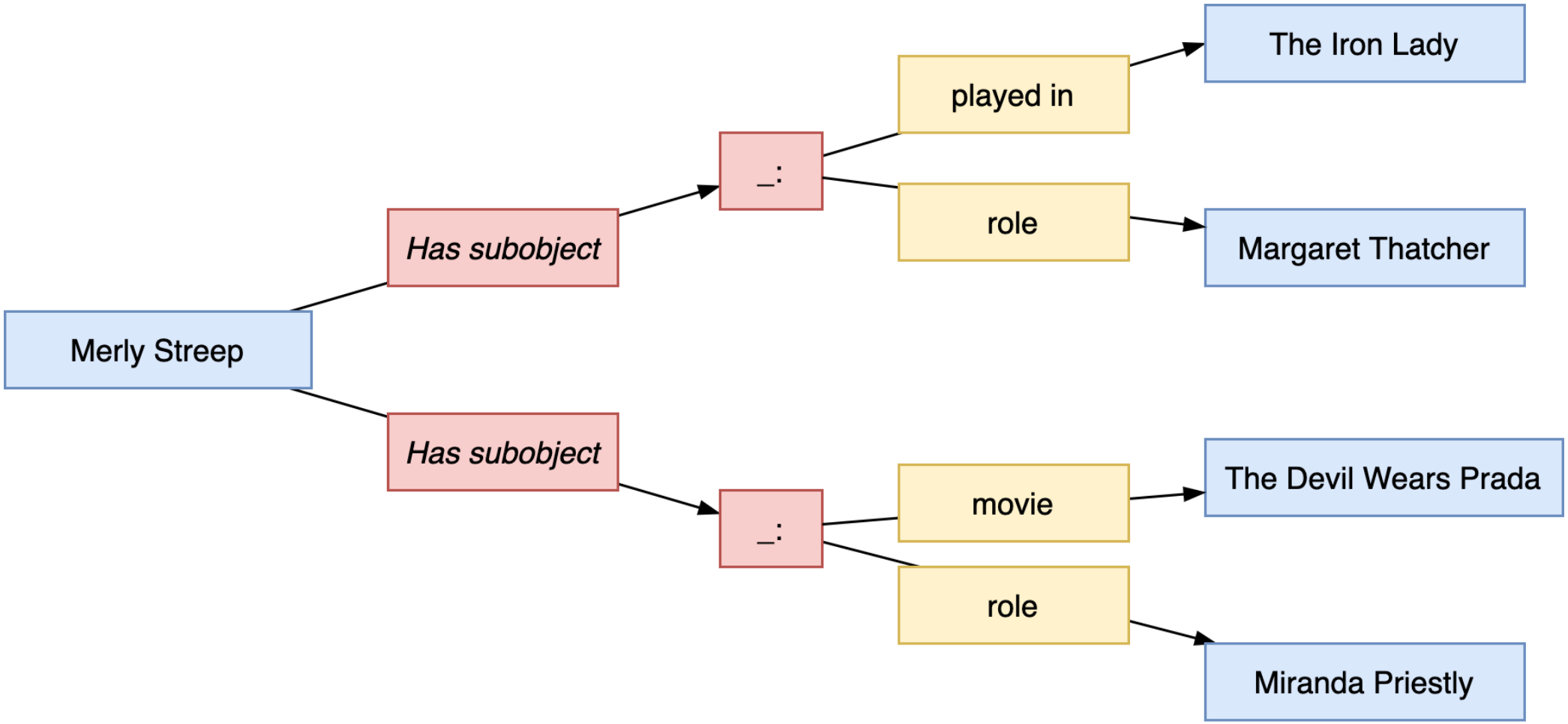
Erstellung eines neuen Subobjects auf der Meryl Streep Seite mit den properties `movie` und `role`

```
{{#subobject:The Iron Lady
 |role=Margaret Thatcher        <!-- refers to role -->
 |movie=The Iron Lady           <!-- refers to played_in -->
}}
```

Aufgaben:

1. Stellen Sie den neuen Modellierungsansatz in einem konzeptuellen Graphen dar
2. Beurteilen Sie den gerade entworfenen Modellierungsansatz erneut nach Flexibilität, Wartbarkeit, Datensparsamkeit.

Konzeptionelle Darstellung



Tipp

Um die **Abfragekomplexität** zu verringern kann es gelegentlich sinnvoll sein, innerhalb eines Subobjects Informationen zu hinterlegen, auf welcher Seite es eingebettet ist (in RDF: zu welchem Subject es gehört). Dies kann mit einem zusätzlichen frei zu definierenden Property realisiert werden, dessen Wert dem Namen der Seite entspricht, auf der das Subobject eingebettet ist.

Beispiel

Stating that the subobject `The Iron Lady` was embedded on the `Meryl Streep` page

```
{{#subobject:The Iron Lady
 ...
 |played by=Meryl Streep  <!-- a more general property: 'refers_to_page' -->
}}
```

Hinweis:
Falls das Subobject mittels einem Template transkludiert wird, so kann für die Angabe des Seitennamens das Magic Word `{{PAGENAME}}` verwendet werden.

Explizit benannte Subobjects können von überall aus in einem Semantic MediaWiki adressiert und in Annotationen genutzt werden:

```
[[played::Meryl Streep#The Iron Lady]]
```

Bei **anonymen Subobjects** ist dies nicht möglich, da ein interner Bezeichner als Identifier vergeben wird, der im Sourcecode der wiki Seite nicht zu sehen ist.

Beispiel: Speicherung des Seitennamens im Subobject

```
<noinclude>
This is the "Project member with role" subobject template.
It should be called in the following format:
<pre>
    {{Project member with role
     |project= #filled with {{PAGENAME}}
     |role=
     |start_date=
     |end_date=
     |member=
    }}
</pre>
Edit the page to see the template text.
</noinclude><includeonly>
{{#subobject:
 |Refers to project={{PAGENAME}} <!-- refers to name of parent page -->
 |Has role={{{role|}}}
 |Has start date={{{start_date|}}}
 |Has end date={{{end_date|}}}
 |Has member={{{member|}}}
 |type=project_membership
}}
</includeonly>
```

Multiple Values for the same Subobject Property

a) Add Multiple Lines

```
{{#subobject:mysubobject
 |Has property 1=value1
 |Has property 1=value2
 |Has property 3=value1
 ...
}}
```

TODO: Check if it works

b) Separate Values using a Pipe

```
{{#subobject:mysubobject
 |Has property=value1|value2
 ...
}}
```

- Separate values using `|` as delimiter

The possibility of using pipes `|` for setting multiple values was deprecated starting with Semantic MediaWiki 3.0.0 and will be removed in a later version. It is strongly recommended to migrate to using the `|+sep` parameter.

c) Separate Values using Named Separator

```
{{#subobject:mysubobject
 |Has property 1=Value 1;Value 2;Value 3|+sep=;
 |Has property 2=12+22+3+4+5+6+7+8+9+10|+sep=+
 |Has property 3=123,1234,12345,|+sep=,
 |Has property 4=One,or,two,more,values|+sep
 ...
}}
```

- Semantic MediaWiki 1.9.0 introduces `|+sep=...` to identify the separator
- Provides greates flexibility and can be combined with PageForms
- Preferred method in most cases

Alternatively, the `{{#arraymap:...` parser function can be used to separate multiple values for one subobject property.

Source: https://www.semantic-mediawiki.org/wiki/Help:Adding_subobjects#Specifying_multiple_values_for_the_same_property

Subobjects: Querying using #ask

Using `#subobject` does not print out anything on the screen. To show the subobject data directly on the page where they are defined use an ask query `{{#ask: [[-Has subobject::{{FULLPAGENAME}}]] }}` with inverse property and add it after the definition of the subobjects.

Example

1. Define two subobjects with identifiers "first" and "second":

```
{{#subobject:first
|property1=value1
|property2=value2
}}

{{#subobject:second
|property1=value3
|property2=value4
}}
```

2. Use the `#ask`-query to print out the subobjects' data:

```
{{#ask:
 [[-Has subobject::{{FULLPAGENAME}}]]
 |?property1
 |?property2
}}
```

Please note that the `Has subobject` property need to be specified as inverse property (note the '`-`' in front of the property) in order to make the query work.

Source: https://www.semantic-mediawiki.org/wiki/Help:Subobjects_and_queries

Searching for Pages with certain Subobject Properties

In some business cases, it is necessary to search for pages with certain subobject properties and display their properties.
The solution is to use subqueries to query for certain subobject properties (e.g. a type property) – so called subproperties.

Example

```
{{#ask:
 [[Has subobject::<q>[[YourSubobjectProperty::Foo]]</q>]]
 |?YourParentProperty1
 |?YourParentProperty2
}}
```

- The query `[[Has subobject::]]` (without the "`-`") queries for the (parent) pages that have certain **subproperties**.
- With the subquery, you can then select certain subobject properties.
- As the query asks for parent pages, you can select properties of the parent pages as printouts.

Source: https://www.semantic-mediawiki.org/wiki/Help:Subobjects_and_queries

Query for Subobject Data on specific Parent Pages

Motivation

Display the items (represented as subobjects) of all book orders issued in 2020.

Assumption

Items are represented as subobjects embedded in order pages.

Solution

1. Draw a data graph of the involved entities
2. Formulate the query conditions for the parent pages
3. Insert the parent page query conditions into a subquery
4. The subquery becomes the value of a query condition using the inverse `Has subobject` property

Example

Build the subquery (i.e. the query conditions for parent pages)

```
[[Category:Buchbestellung]]
[[Bestelldatum::>1.1.2020]] [[Bestelldatum::<31.12.2020]]
```

Build the full query and embed the subquery

```
{{#ask:
 [[-Has subobject::<q>[[Category:Buchbestellung]]
 [[Bestelldatum::>1.1.2020]] [[Bestelldatum::<31.12.2020]]</q>]]
 |?Verfasser
 |?Buchtitel
 ...
}}
```

Subobjects: Show Properties of the Subobject's Parent Page

In some cases it is necessary to query for properties that are defined on the subobject's parent page.

a) Using Property Chains

Example

```
{{#ask:
 [[-Has subobject::{{FULLPAGENAME}}]]
 |?-has subobject.YourParentProperty1
 |?-has subobject.YourParentProperty2
 |?YourSubobjectProperty1
 |?YourSubobjectProperty2
}}
```

- Chain members must be of type page
- Last member can be of any type
- Chaining depth is not limited by default

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

b) Using Templates

1. Query for a property of a subobject.
2. Use `format=template`.
3. Add another query in this template where you ask for `[[Has subobject::{{{1}}}]]`. Has subobject returns the parent page of a subobject. The subobject is queried in the first query and passed on to the template as `{{{1}}}`.

Example

```
{{#ask:
 [[YourSubobjectProperty::Foo]]
 |?YourSubobjectProperty1
 |?YourSubobjectProperty2
 |format=template
 |template=YourTemplate
}}
```

In `Template:YourTemplate` add the following:

```
{{#ask:
 [[Has subobject::{{{1}}}]]
 |?YourParentProperty1
 |?YourParentProperty2
}}
```

Combining Parent Page and Subobject Data in Queries

This can be achieved with subqueries and inverse property chaining in the properties' selection part of inline queries.

Example

```
{{#ask:
 [[-Has subobject::<q>[[Category:Buchbestellung]]
 [[Bestelldatum::>1.1.2020]] [[Bestelldatum::<31.12.2020]]</q>]]
 |?-Has subobject.Besteller         <!-- selecting parent page property via inverse property chaining -->
 |?-Has subobject.Kostenstelle      <!-- selecting parent page property via inverse property chaining -->
 |?Verfasser
 |?Buchtitel=Titel
 |?Erscheinungsjahr=Jahr |+align=center
 |mainlabel=-
 |format=broadtable
 |class=smwtable-clean sortable
 |headers=plain
}}
```