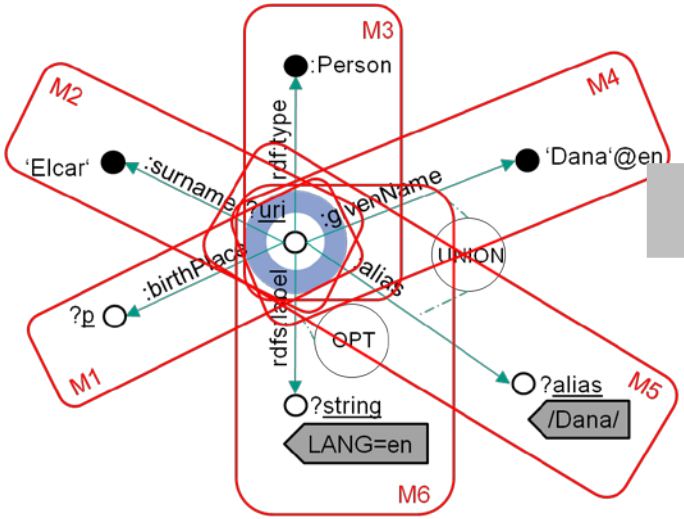
SPARQL Query Verbalization for Explaining Semantic Search Engine Queries

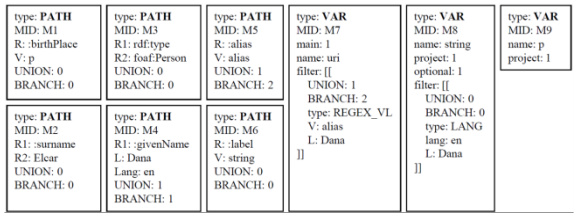
Basil Ell,[1] **Andreas Harth**,[1] **Elena Simperl**[2]

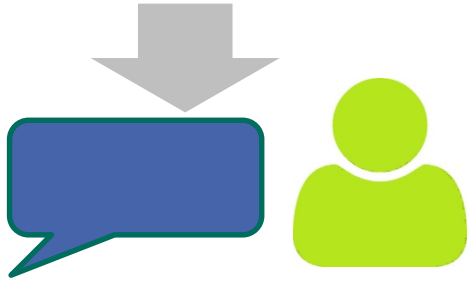
11th Extended Semantic Web Conference 2014
28 May 2014, Anissaras/Heronissou, Crete, Greece

[1] Institute of Applied Informatics and Formal Description Metthods (AIFB), Karlsruhe, Germany
[2] University of Southampton, United Kingdom

```
SELECT ?uri ?string ?p WHERE {
  ?uri rdf:type :Person .
  ?uri :birthPlace ?p .
  ?uri :surname 'Elcar' .
  { ?uri :givenName 'Dana'@en . } UNION {
    ?uri :alias ?alias .
    FILTER(regex(?alias, 'Dana')) .
  }
  OPTIONAL {
    ?uri rdfs:label ?string .
    FILTER(lang(?string)='en')
  }
}
```







Overview

- Motivation
- Anatomy of a query verbalization
- Main ideas
- Example
- Evaluation
- Conclusions

Motivation (1/2)

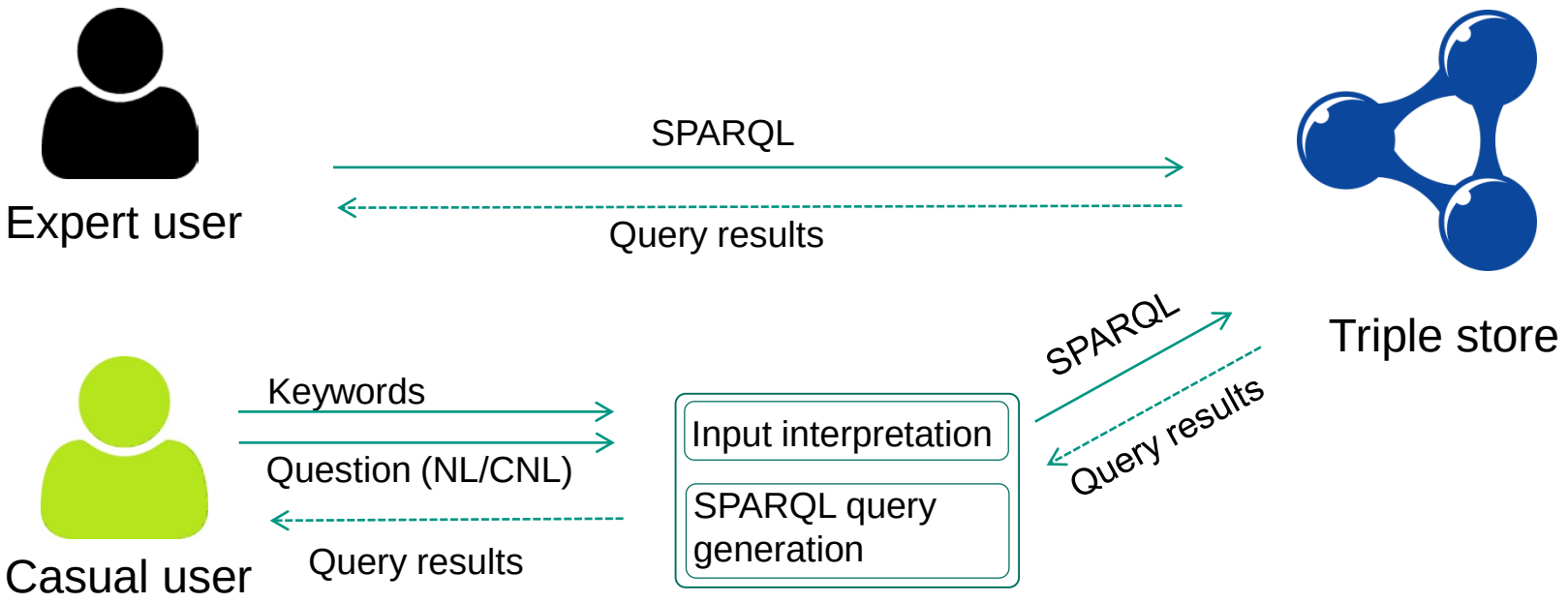
Expert user

SPARQL

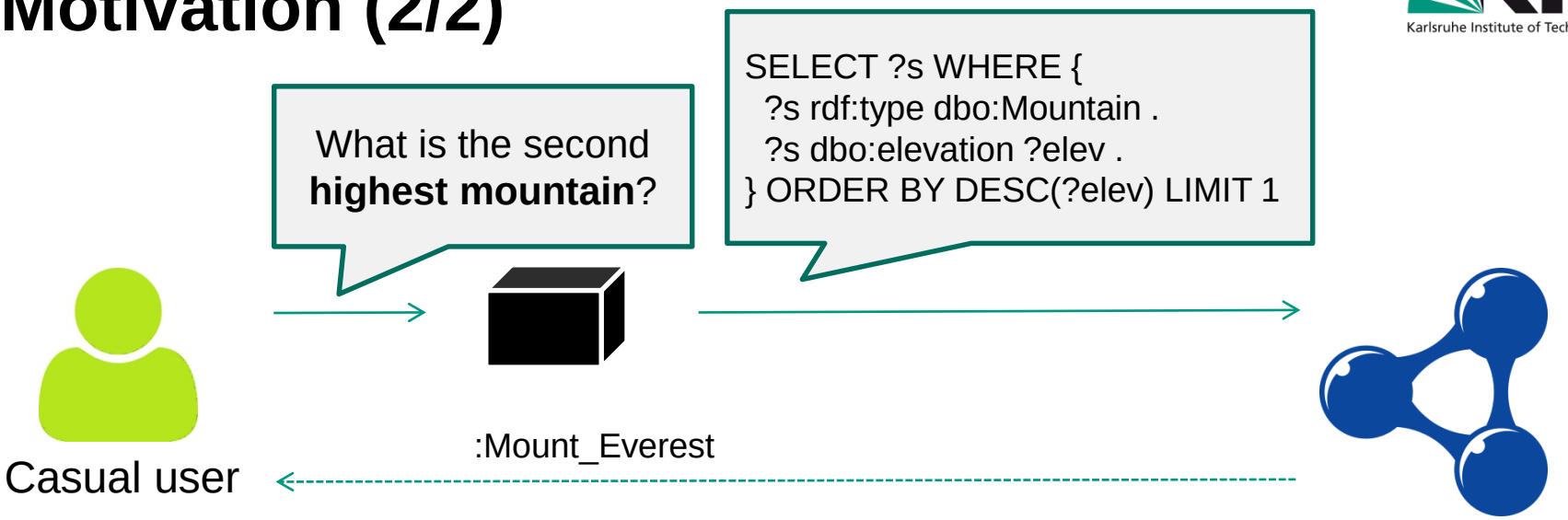
Query results

Triple store

Motivation (1/2)

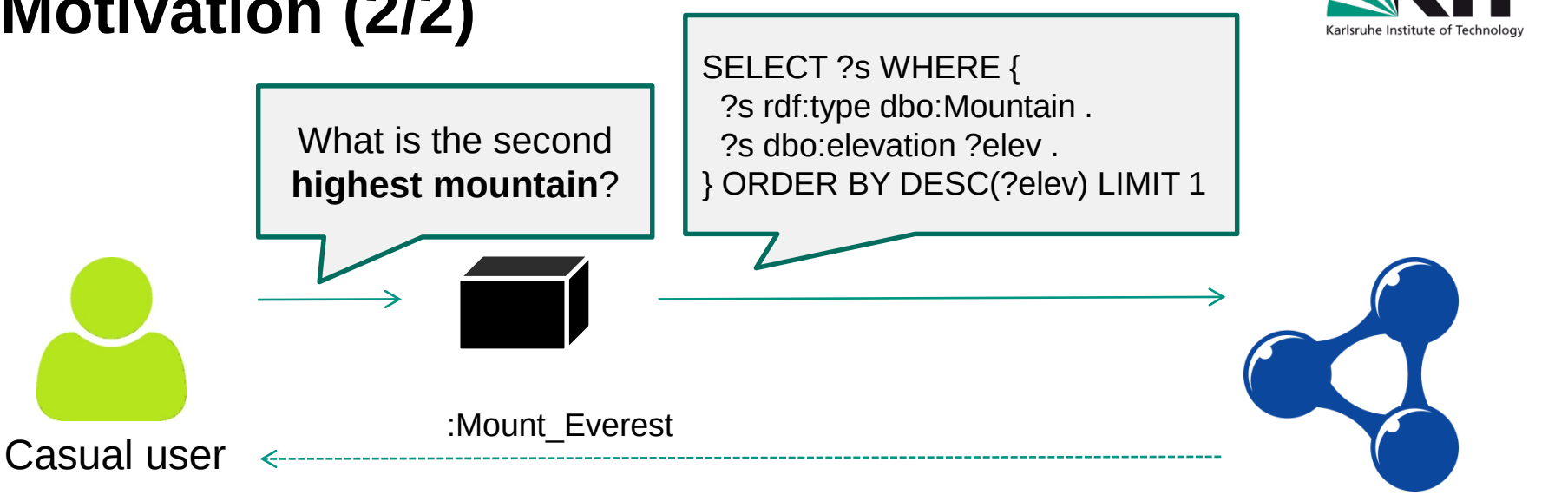


Motivation (2/2)

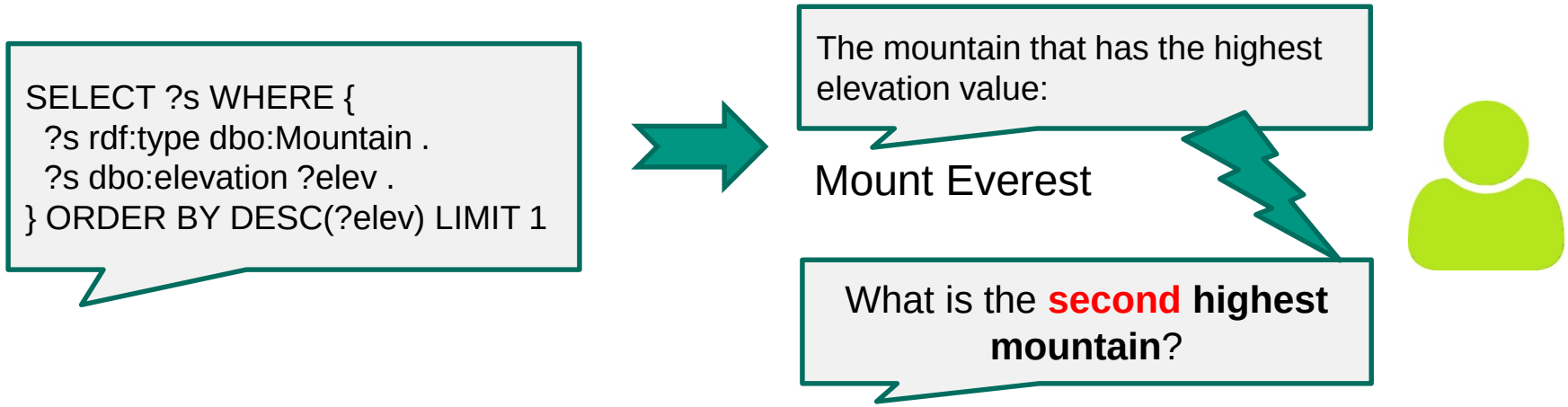


Unless the user does not understand the generated query, the user cannot trust the results.

Motivation (2/2)



Unless the user does not understand the generated query, the user cannot trust the results.



Anatomy of a query verbalization (1/4)

subject	constraints	requests
The country	that has the highest number of languages	Show also, if available, its English labels.

```
SELECT ?uri ?string WHERE {
   ?uri rdf:type onto:Country .
   ?uri dbo:language ?language .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER(lang(?string) = 'en')
   }
} ORDER BY DESC (COUNT(?language)) LIMIT 1
```

Anatomy of a query verbalization (1/4)

subject	constraints	requests
The country	that has the highest number of languages	Show also, if available, its English labels.

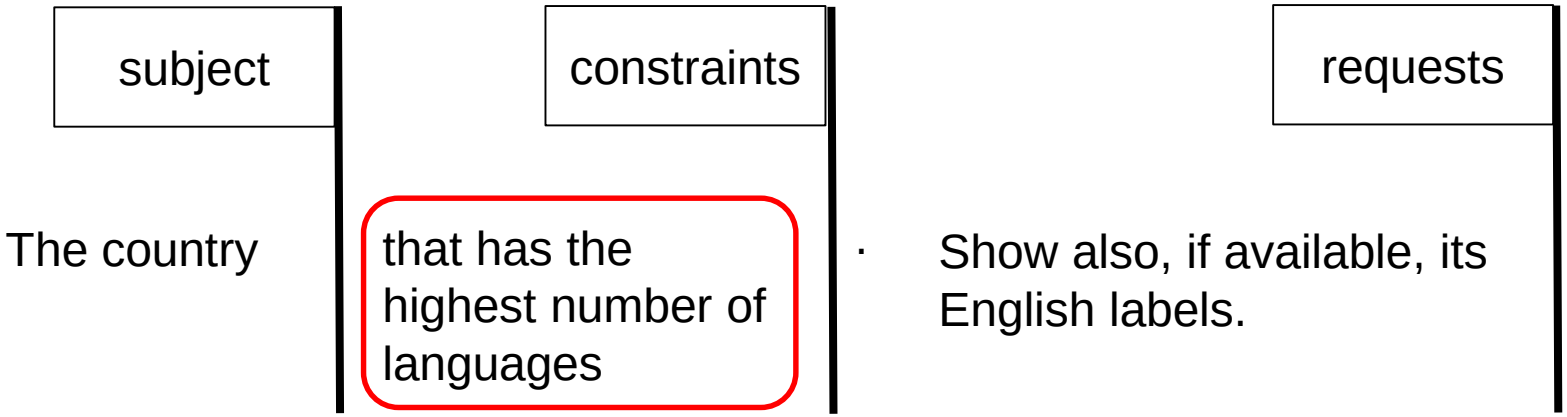
```
SELECT ?uri ?string WHERE {
   ?uri rdf:type onto:Country .
   ?uri dbo:language ?language .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER(lang(?string) = 'en')
   }
} ORDER BY DESC (COUNT(?language)) LIMIT 1
```

Anatomy of a query verbalization (1/4)

subject	constraints		requests
The country	that has the highest number of languages	·	Show also, if available, its English labels.

```
SELECT ?uri ?string WHERE {
   ?uri rdf:type onto:Country .
   ?uri dbo:language ?language .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER(lang(?string) = 'en')
   }
} ORDER BY DESC (COUNT(?language)) LIMIT 1
```

Anatomy of a query verbalization (1/4)




```
SELECT ?uri ?string WHERE {
   ?uri rdf:type onto:Country .
   ?uri dbo:language ?language .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER(lang(?string) = 'en')
   }
} ORDER BY DESC (COUNT(?language)) LIMIT 1
```

Anatomy of a query verbalization (1/4)

subject	constraints	requests
The country	that has the highest number of languages	Show also, if available, its English labels.

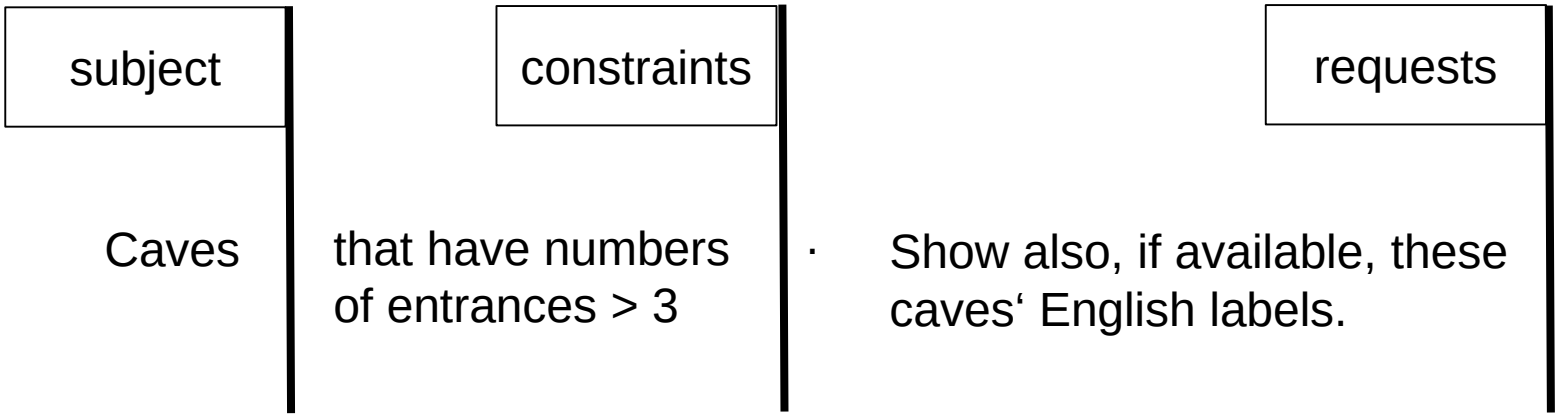
```
SELECT ?uri ?string WHERE {
   ?uri rdf:type onto:Country .
   ?uri dbo:language ?language .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER(lang(?string) = 'en')
   }
} ORDER BY DESC (COUNT(?language)) LIMIT 1
```

Anatomy of a query verbalization (1/4)

subject	constraints		requests
The country	that has the highest number of languages	.	Show also, if available, its English labels.

```
SELECT ?uri ?string WHERE {
   ?uri rdf:type onto:Country .
   ?uri dbo:language ?language .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER(lang(?string) = 'en')
   }
} ORDER BY DESC (COUNT(?language)) LIMIT 1
```

Anatomy of a query verbalization (2/4)



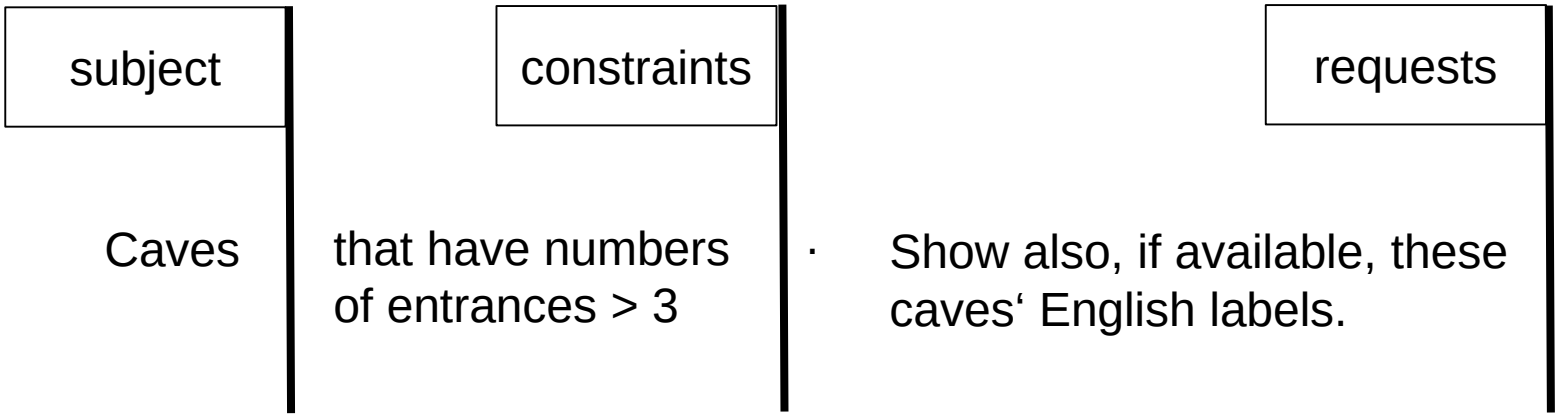

```
SELECT ?uri ?string WHERE {
   ?uri rdf:type dbo:Cave .
   ?uri dbo:numberOfEntrances ?entrance .
   FILTER (?entrance > 3) .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER (lang(?string) = 'en')
   }
}
```

Anatomy of a query verbalization (2/4)

subject	constraints	requests
Caves	that have numbers of entrances > 3	Show also, if available, these caves‘ English labels.

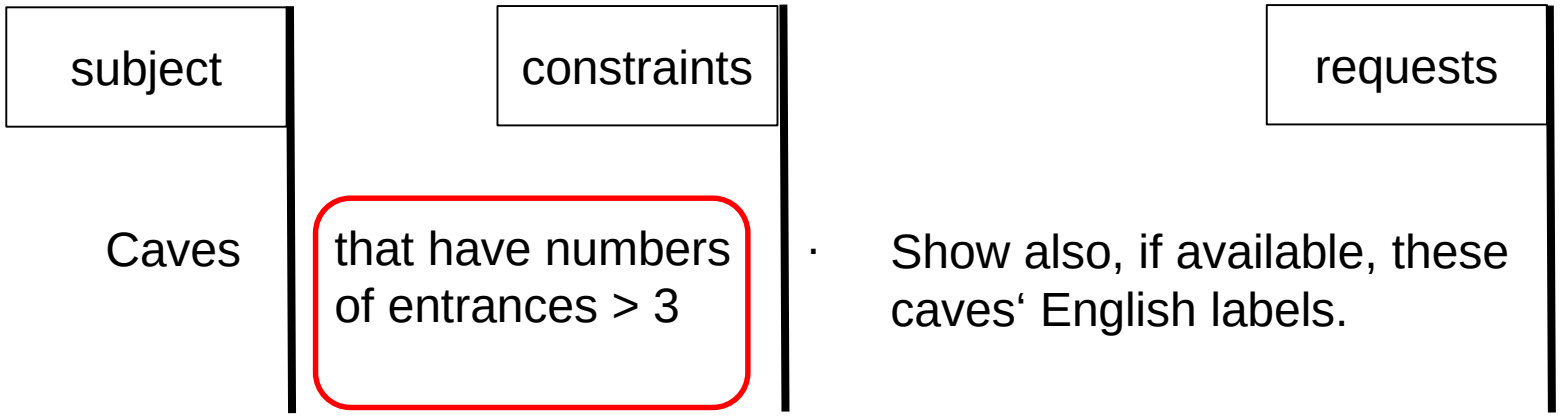
```
SELECT ?uri ?string WHERE {
   ?uri rdf:type dbo:Cave .
   ?uri dbo:numberOfEntrances ?entrance .
   FILTER (?entrance > 3) .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER (lang(?string) = 'en')
   }
}
```

Anatomy of a query verbalization (2/4)



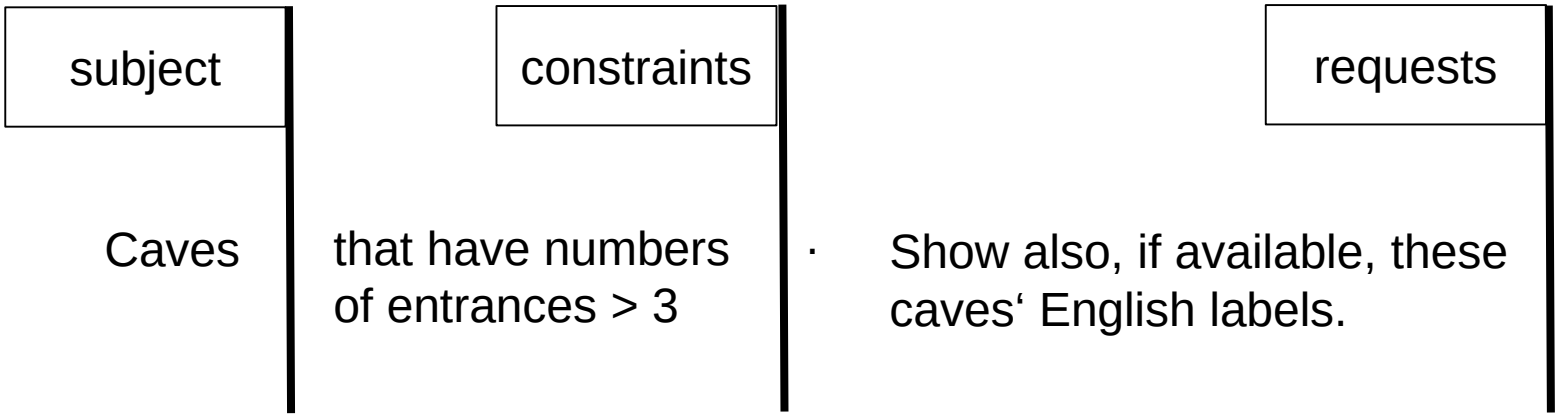

```
SELECT ?uri ?string WHERE {
   ?uri rdf:type dbo:Cave .
   ?uri dbo:numberOfEntrances ?entrance .
   FILTER (?entrance > 3) .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER (lang(?string) = 'en')
   }
}
```

Anatomy of a query verbalization (2/4)




```
SELECT ?uri ?string WHERE {
   ?uri rdf:type dbo:Cave .
   ?uri dbo:numberOfEntrances ?entrance .
   FILTER (?entrance > 3) .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER (lang(?string) = 'en')
   }
}
```

Anatomy of a query verbalization (2/4)




```
SELECT ?uri ?string WHERE {
   ?uri rdf:type dbo:Cave .
   ?uri dbo:numberOfEntrances ?entrance .
   FILTER (?entrance > 3) .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER (lang(?string) = 'en')
   }
}
```

Anatomy of a query verbalization (2/4)

subject	constraints	requests
Caves	that have numbers of entrances > 3	Show also, if available, these caves‘ English labels.

```
SELECT ?uri ?string WHERE {
   ?uri rdf:type dbo:Cave .
   ?uri dbo:numberOfEntrances ?entrance .
   FILTER (?entrance > 3) .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER (lang(?string) = 'en')
   }
}
```

Anatomy of a query verbalization (3/4)

subject	constraints	requests
Distinct cities	that have population urbans > 2000000 or that have a population > 2000000	Show also, if available, these cities‘ English labels.

```
SELECT DISTINCT ?uri ?string WHERE {
   ?uri rdf:type onto:City.
   { ?uri prop:population ?population. }
     UNION
   { ?uri prop:populationUrban ?population. }
   FILTER (xsd:integer(?population) > 2000000) .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER(lang(?string)='en')
   }
}
```

Anatomy of a query verbalization (3/4)

subject	constraints	requests
Distinct cities	that have population urbans > 2000000 or that have a population > 2000000	· Show also, if available, these cities‘ English labels.

```
SELECT DISTINCT ?uri ?string WHERE {
   ?uri rdf:type onto:City.
   { ?uri prop:population ?population. }
     UNION
   { ?uri prop:populationUrban ?population. }
   FILTER (xsd:integer(?population) > 2000000) .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER(lang(?string)='en')
   }
}
```

Anatomy of a query verbalization (3/4)

subject	constraints	requests
Distinct cities	that have population urbans > 2000000 or that have a population > 2000000	· Show also, if available, these cities‘ English labels.

```
SELECT DISTINCT ?uri ?string WHERE {
   ?uri rdf:type onto:City.
   { ?uri prop:population ?population. }
     UNION
   { ?uri prop:populationUrban ?population. }
   FILTER (xsd:integer(?population) > 2000000) .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER(lang(?string)='en')
   }
}
```

Anatomy of a query verbalization (3/4)

subject	constraints	requests
Distinct cities	that have population urbans > 2000000 or that have a population > 2000000	Show also, if available, these cities‘ English labels.

```
SELECT DISTINCT ?uri ?string WHERE {
   ?uri rdf:type onto:City.
   { ?uri prop:population ?population. }
     UNION
   { ?uri prop:populationUrban ?population. }
   FILTER (xsd:integer(?population) > 2000000) .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER(lang(?string)='en')
   }
}
```

Anatomy of a query verbalization (3/4)

subject	constraints	requests
Distinct cities	that have population urbans > 2000000 or that have a population > 2000000	Show also, if available, these cities‘ English labels.

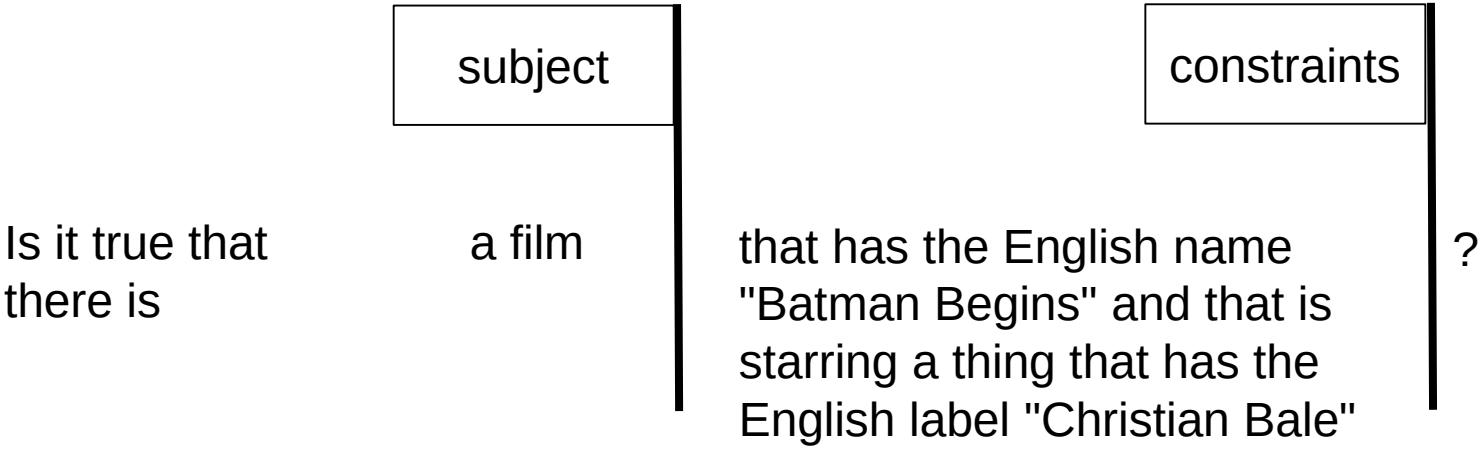
```
SELECT DISTINCT ?uri ?string WHERE {
   ?uri rdf:type onto:City.
   { ?uri prop:population ?population. }
     UNION
   { ?uri prop:populationUrban ?population. }
   FILTER (xsd:integer(?population) > 2000000) .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER(lang(?string)='en')
   }
}
```

Anatomy of a query verbalization (3/4)

subject	constraints	requests
Distinct cities	that have population urbans > 2000000 or that have a population > 2000000	Show also, if available, these cities‘ English labels.

```
SELECT DISTINCT ?uri ?string WHERE {
   ?uri rdf:type onto:City.
   { ?uri prop:population ?population. }
     UNION
   { ?uri prop:populationUrban ?population. }
   FILTER (xsd:integer(?population) > 2000000) .
   OPTIONAL {
     ?uri rdfs:label ?string .
     FILTER(lang(?string)='en')
   }
}
```

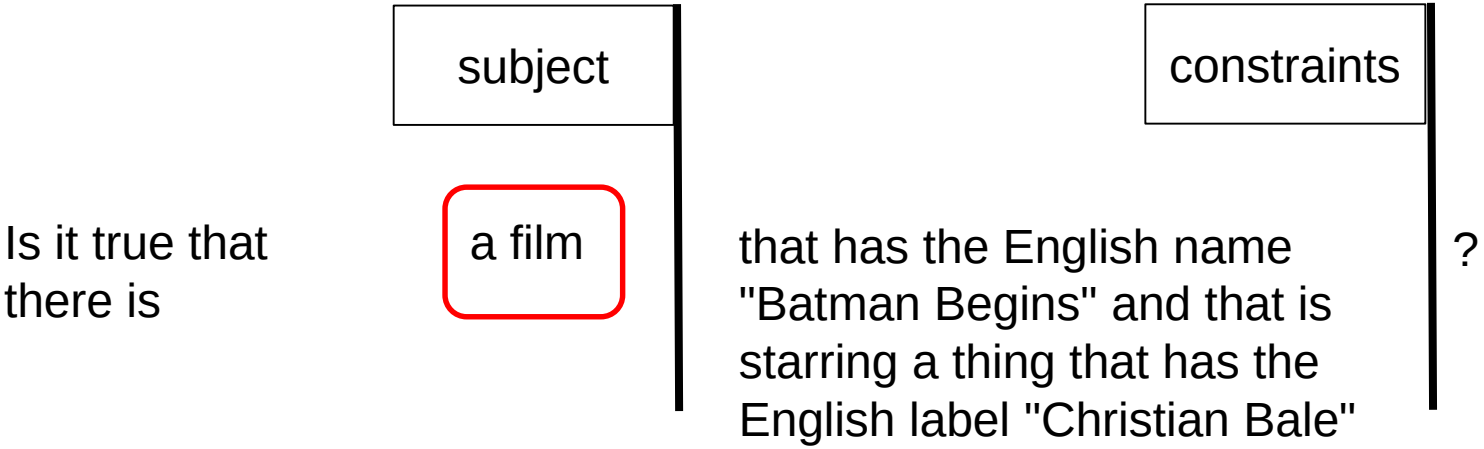
Anatomy of a query verbalization (4/4)




```
ASK WHERE {
   ?film rdf:type onto:Film .
   ?film onto:starring ?actors .
   ?actors rdfs:label 'Christian Bale'@en .
   ?film foaf:name 'Batman Begins'@en
}
```

> Main Idea

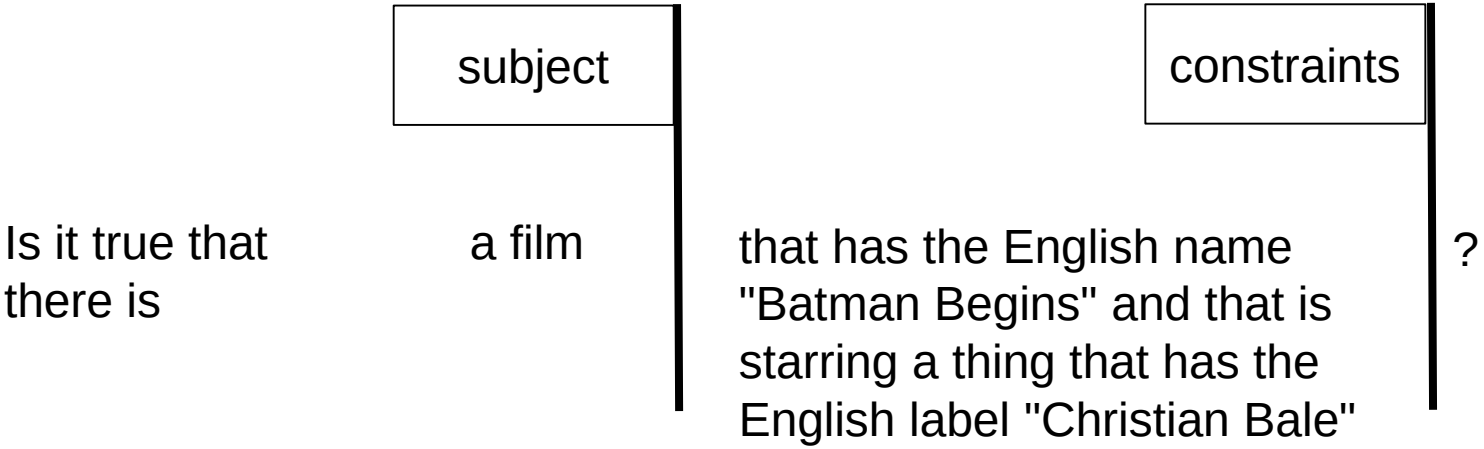
Anatomy of a query verbalization (4/4)




```
ASK WHERE {
   ?film rdf:type onto:Film .
   ?film onto:starring ?actors .
   ?actors rdfs:label 'Christian Bale'@en .
   ?film foaf:name 'Batman Begins'@en
}
```

> Main Idea

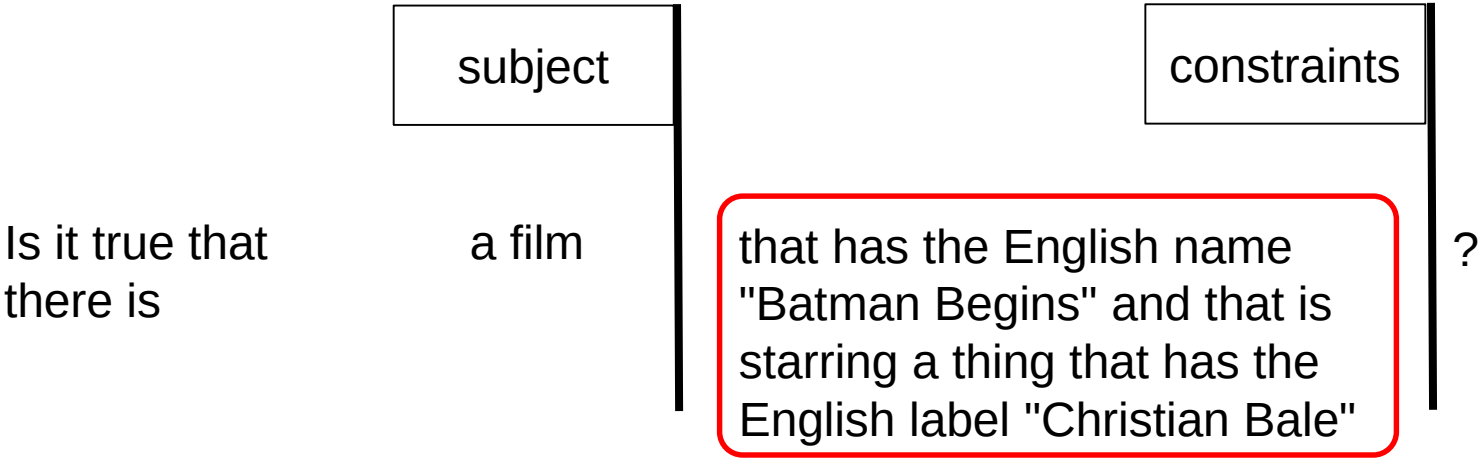
Anatomy of a query verbalization (4/4)




```
ASK WHERE {
   ?film rdf:type onto:Film .
   ?film onto:starring ?actors .
   ?actors rdfs:label 'Christian Bale'@en .
   ?film foaf:name 'Batman Begins'@en
}
```

> Main Idea

Anatomy of a query verbalization (4/4)




```
ASK WHERE {
    ?film rdf:type onto:Film .
    ?film onto:starring ?actors .
    ?actors rdfs:label 'Christian Bale'@en .
    ?film foaf:name 'Batman Begins'@en
}
```

> Main Idea

Main idea

- Decompose a query into independently verbalizable messages (top-down approach)
- Retrieve labels via URI look-up
- Verbalize messages using templates
- Assemble verbalized messages

- Templates are mostly schema-agnostic
  - Templates depend on linguistic features of properties
  - Verbalizer is domain-independent
  - Schema-specific templates can be added
  - Lemon dictionaries could be used

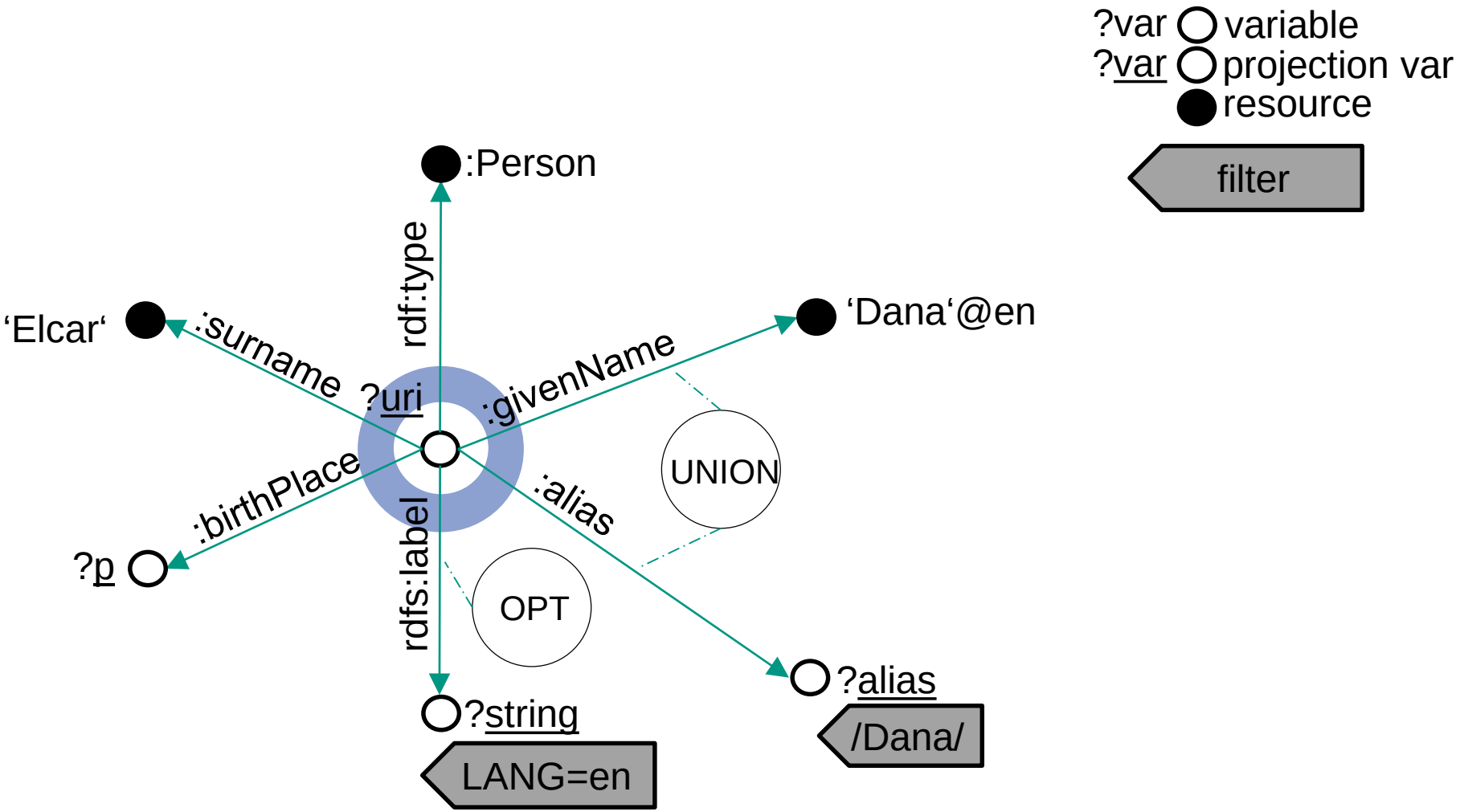
> Example

Example – SPARQL query

```
01 SELECT ?uri ?string ?p WHERE {
02   ?uri rdf:type :Person .
03   ?uri :birthPlace ?p .
04   ?uri :surname 'Elcar' .
05   { ?uri :givenName 'Dana'@en . } UNION {
06     ?uri :alias ?alias .
07     FILTER(regex(?alias, 'Dana')) .
08   }
09   OPTIONAL {
10     ?uri rdfs:label ?string .
11     FILTER(lang(?string)='en')
12   }
13 }
```

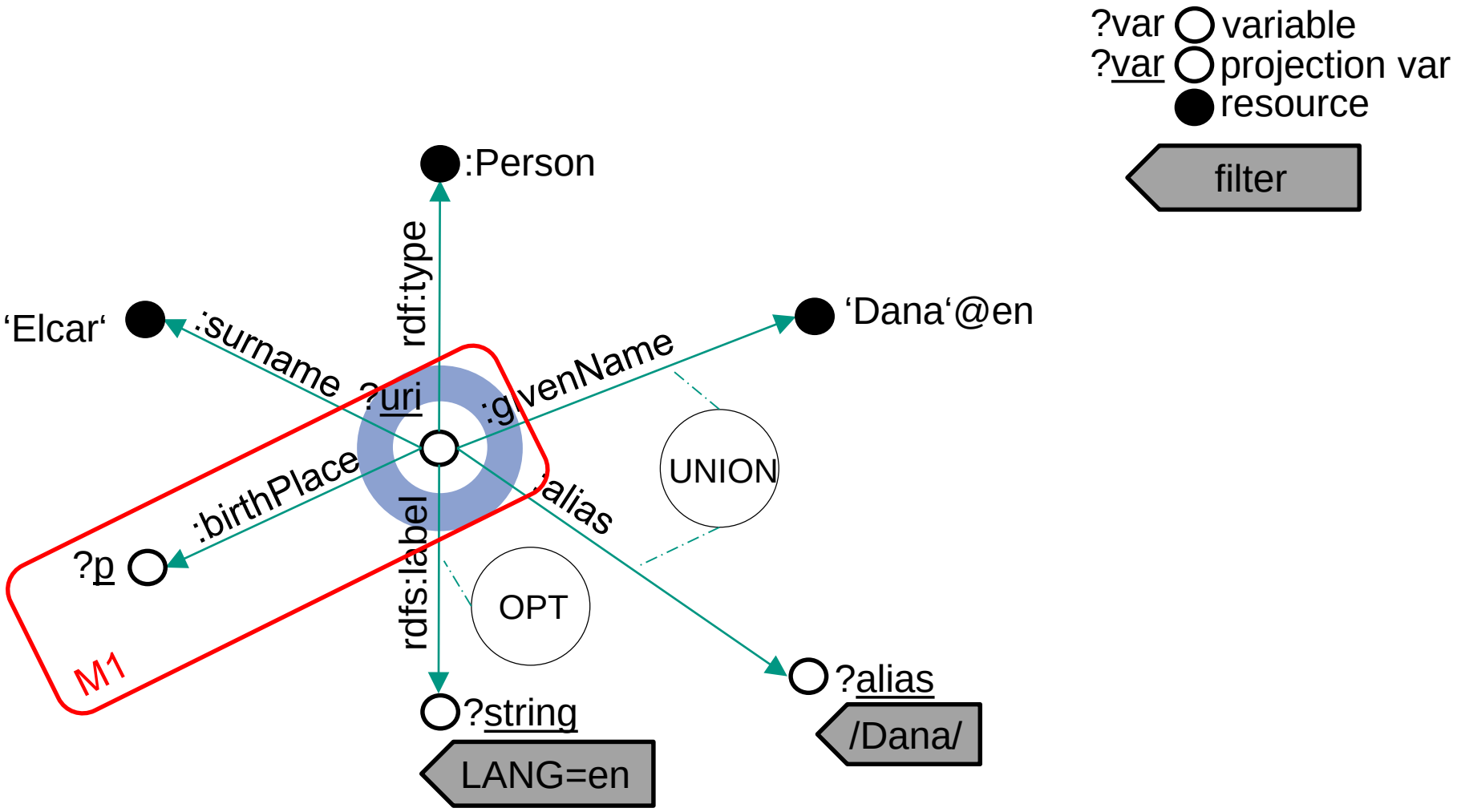
> Graph rep.

Example query – graph representation



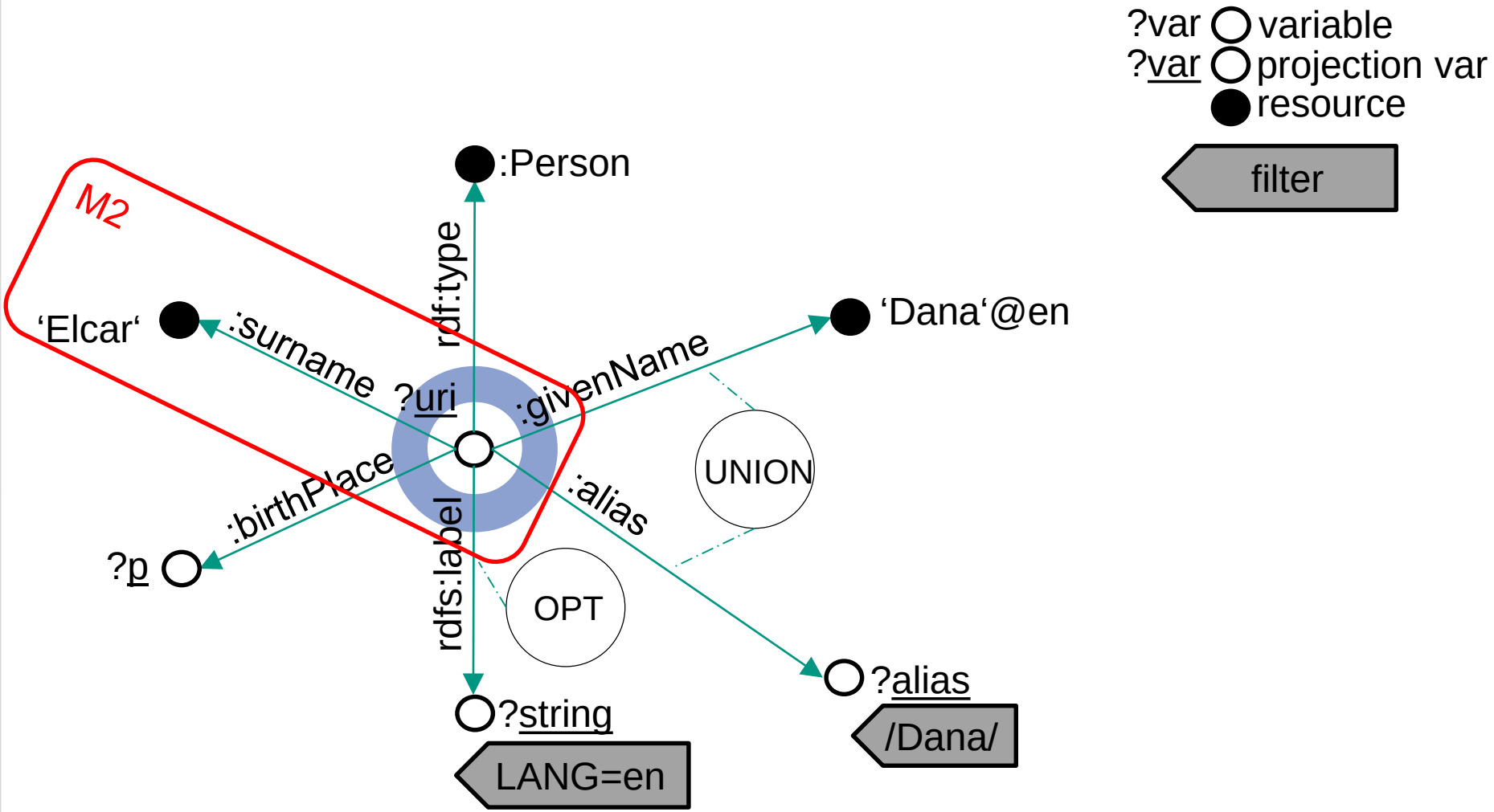
> Message rep.

Example query – graph representation



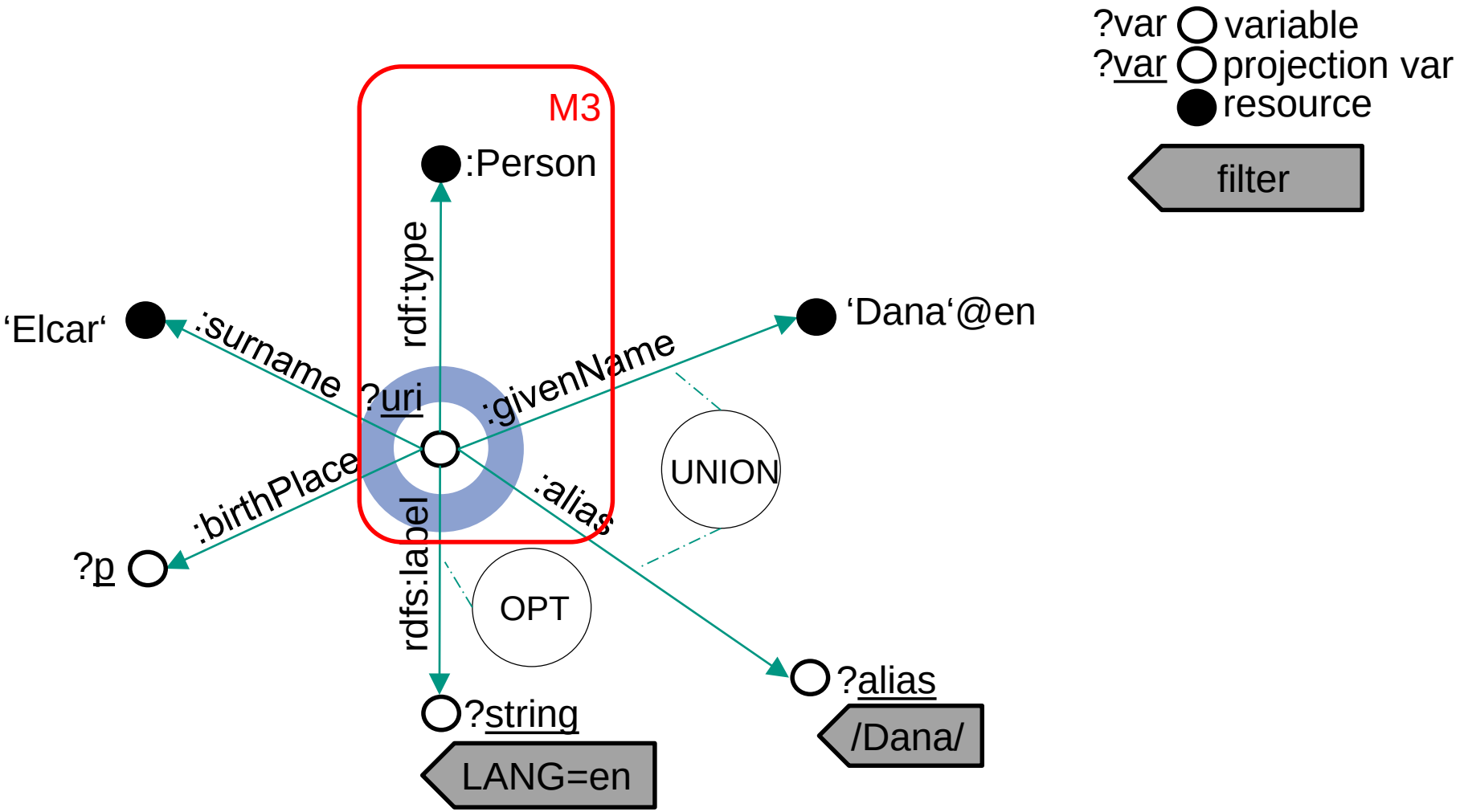
> Message rep.

Example query – graph representation



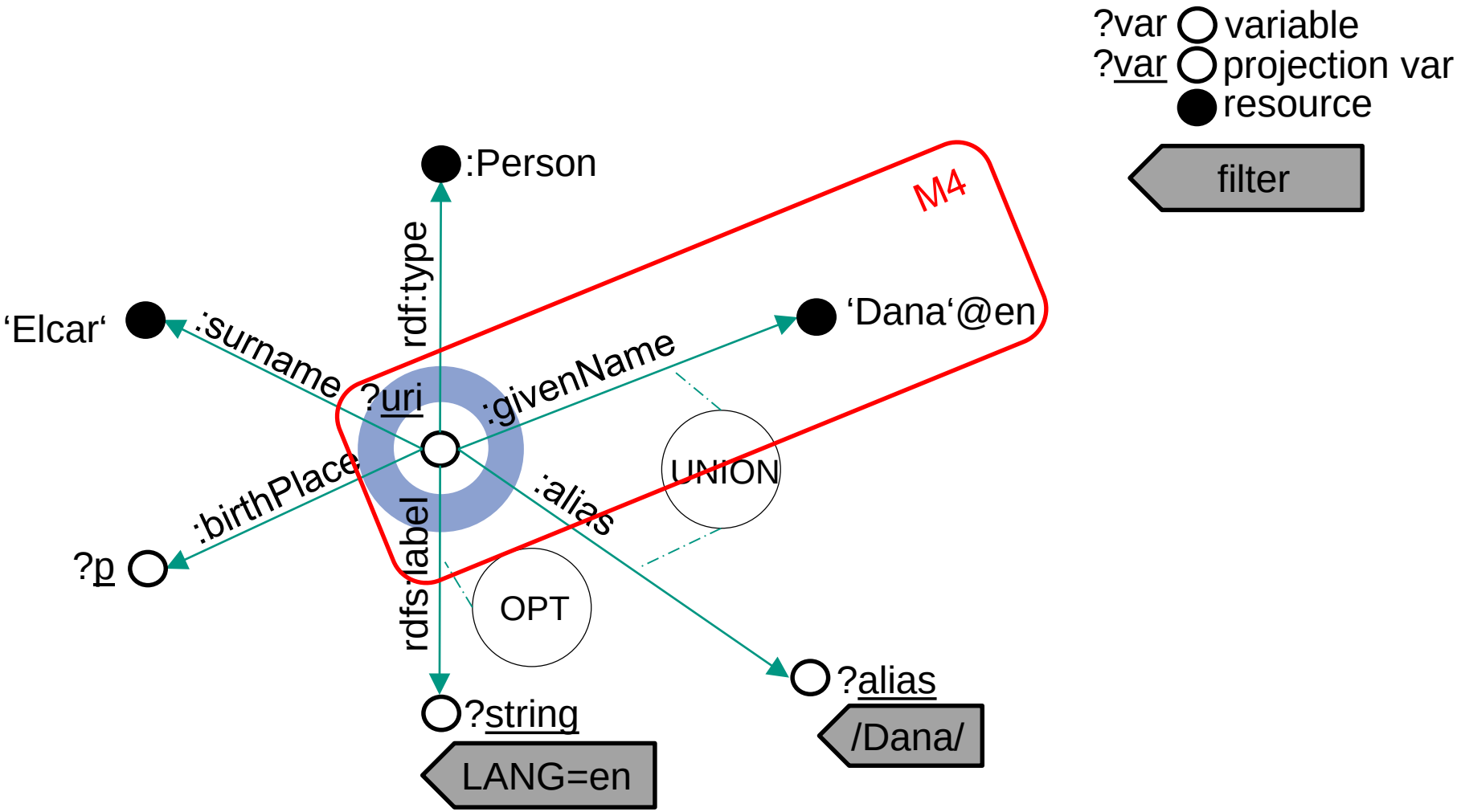
> Message rep.

Example query – graph representation



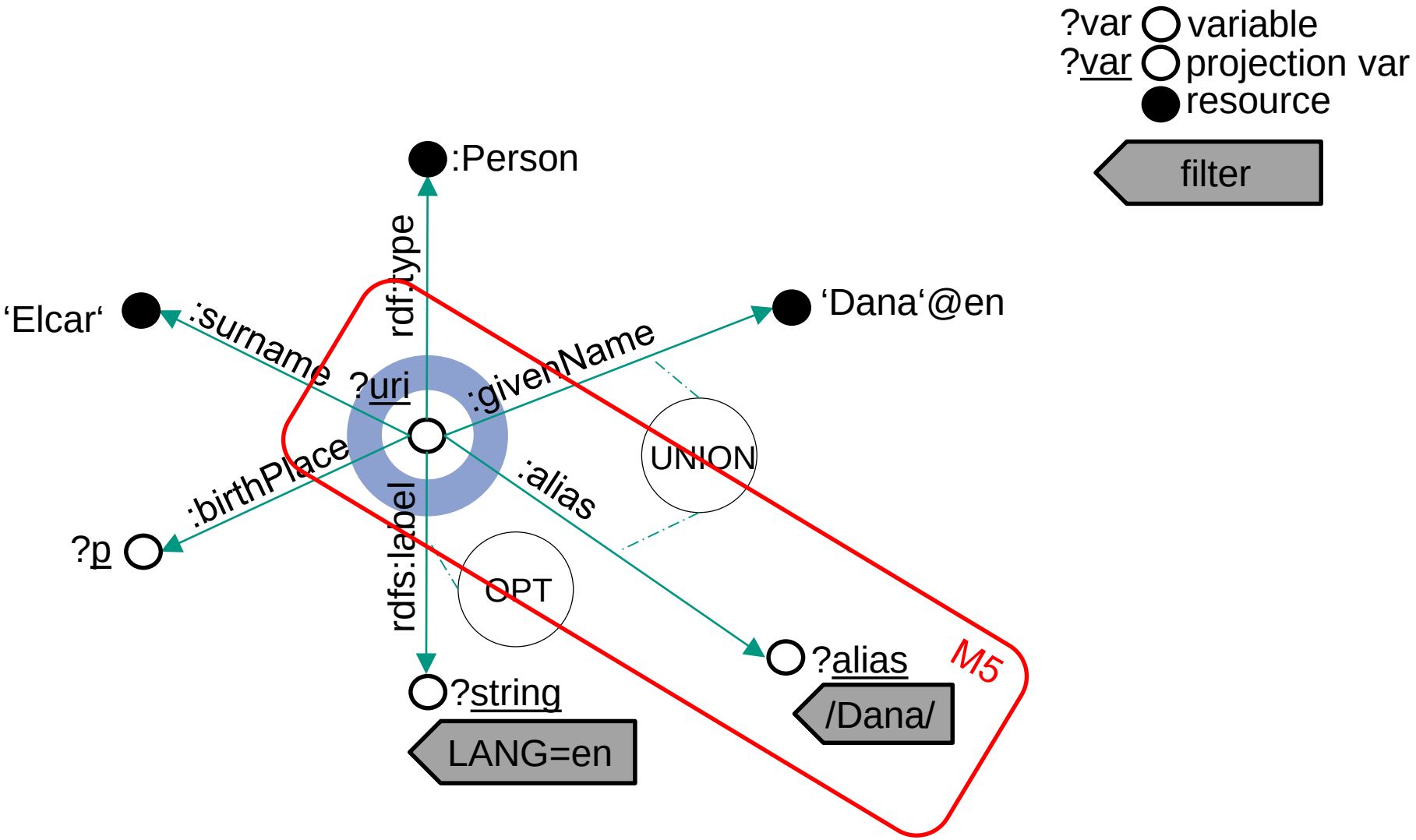
> Message rep.

Example query – graph representation



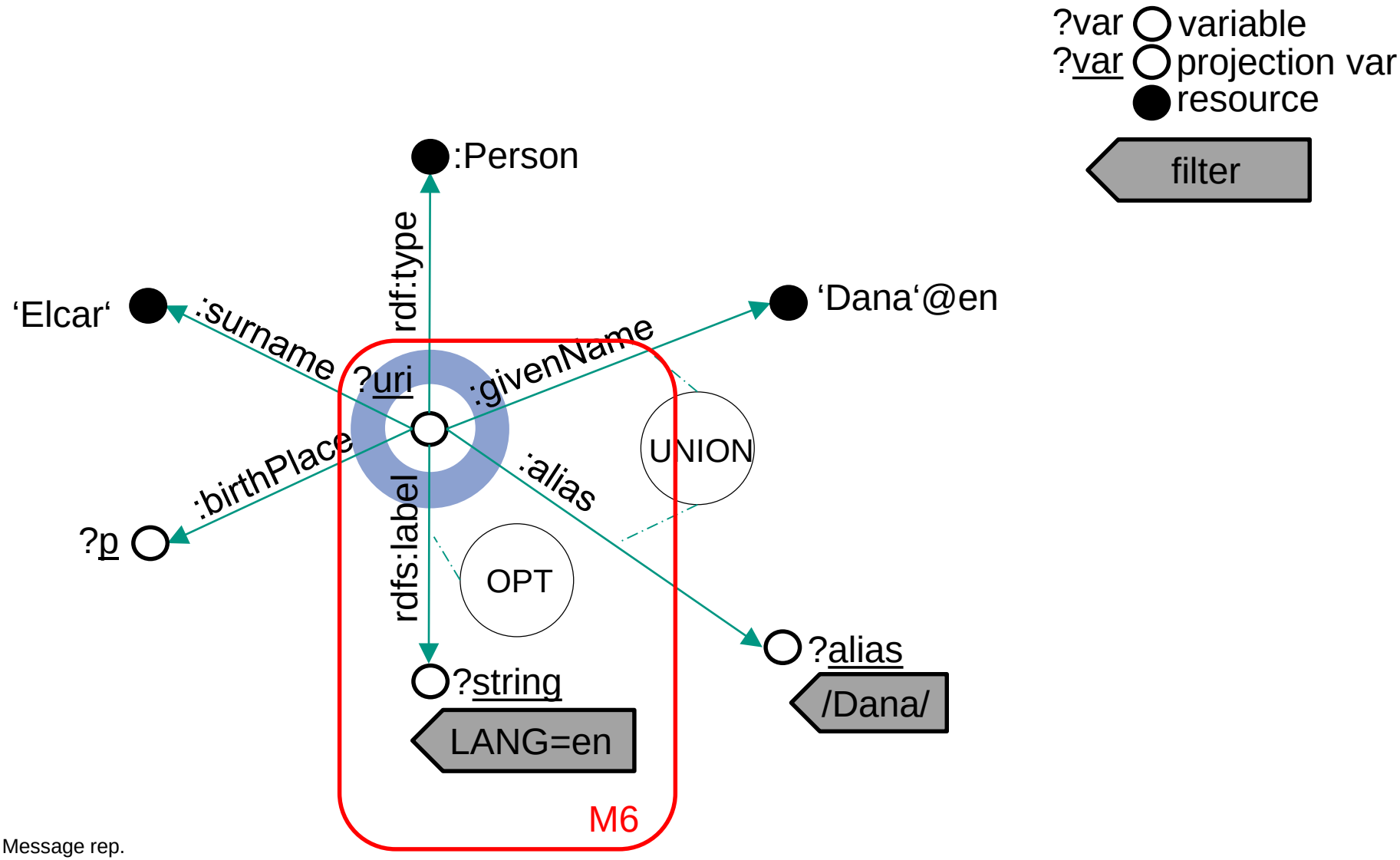
> Message rep.

Example query – graph representation



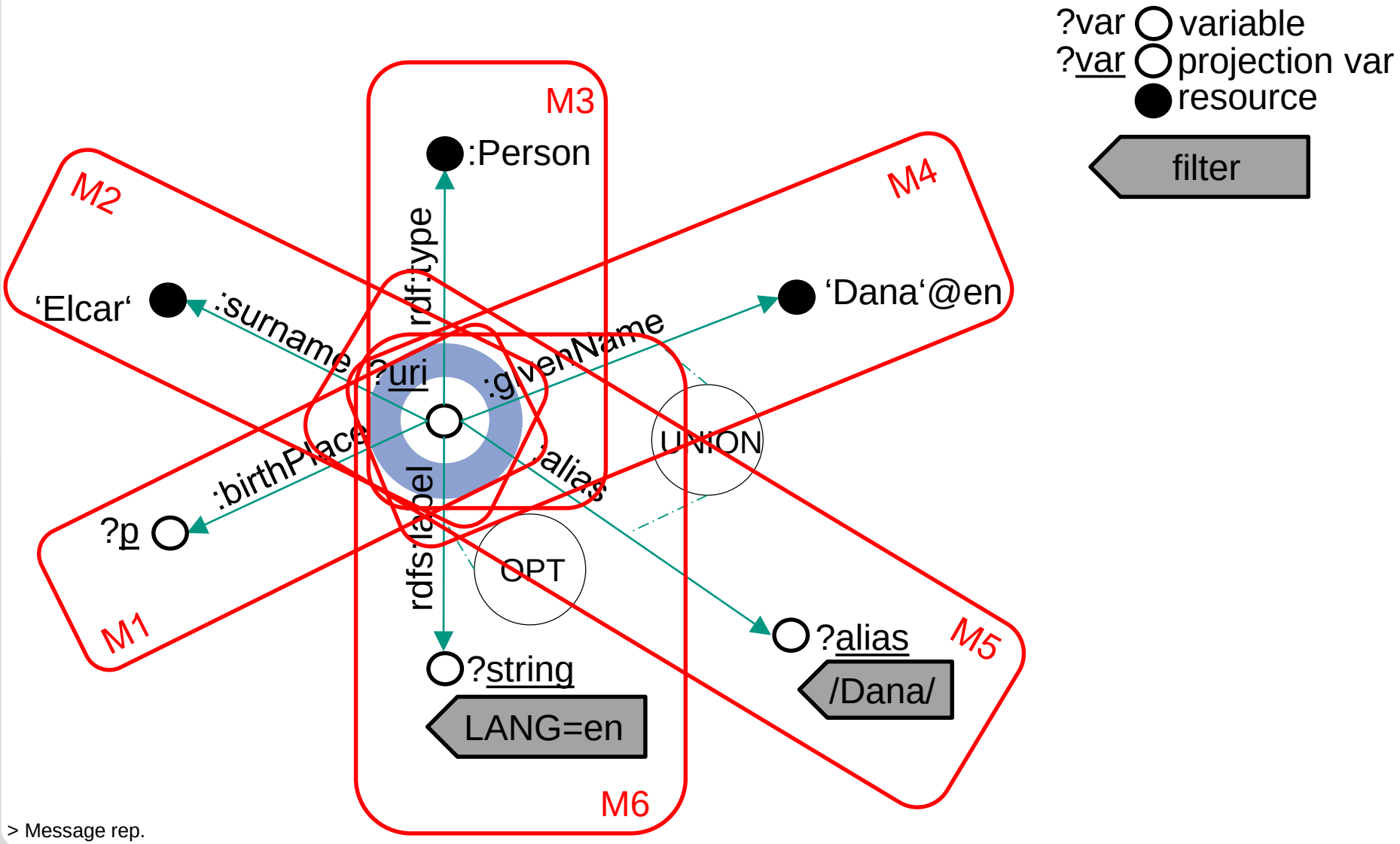
> Message rep.

Example query – graph representation



> Message rep.

Example query – graph representation



> Message rep.

Example query – message rep.

type: **PATH** MID: M1 R: :birthPlace V: p UNION: 0 BRANCH: 0	type: **PATH** MID: M3 R1: rdf:type R2: foaf:Person UNION: 0 BRANCH: 0	type: **PATH** MID: M5 R: :alias V: alias UNION: 1 BRANCH: 2	type: **VAR** MID: M7 main: 1 name: uri filter: [[UNION: 1 BRANCH: 2 type: REGEX_VL V: alias L: Dana]]	type: **VAR** MID: M8 name: string project: 1 optional: 1 filter: [[UNION: 0 BRANCH: 0 type: LANG lang: en L: Dana]]	type: **VAR** MID: M9 name: p project: 1
type: **PATH** MID: M2 R1: :surname R2: Elcar UNION: 0 BRANCH: 0	type: **PATH** MID: M4 R1: :givenName L: Dana Lang: en UNION: 1 BRANCH: 1	type: **PATH** MID: M6 R: :label V: string UNION: 0 BRANCH: 0			

> Example verb.

Example – Subject verbalization

- Control variables

$A \hat{=}$ The DISTINCT modifier is used
$B \hat{=}$ Main entity is counted as in `SELECT(COUNT ?main)`
$C \hat{=}$ Result set is limited with LIMIT as in `LIMIT 10`
$D \hat{=}$ ME needs to be verbalized in singular (LIMIT = 1)
$E \hat{=}$ Main entity is ordered as in `ORDER BY DESC(?main)`
$F \hat{=}$ Main entity has multiple types as in
`?main rdf:type ex:A. ?main rdf:type ex:B.`

- Subject template (excerpt)

`Abcdef` 'Distinct ' . $D{TPL}
→ Distinct scientists
`aBcdef` 'Number of ' . $D{TPL}
→ Number of scientists
`AbcdEf` 'The distinct ' . $D{TPL}
→ The distinct scientists
`abCdef` 'Not more than ' . $D{L}
.' '.$D{TPL}
→ Not more than 10 scientists
`abcDEf` 'The ' . $D{TSI}
→ The scientist

(Due to depedencies: 36 cases instead of 2^6=64)

Example – Subject verbalization

Control variables

$A \mathrel{\hat{=}}$ The DISTINCT modifier is used

$B \mathrel{\hat{=}}$ Main entity is counted as in `SELECT(COUNT ?main)`

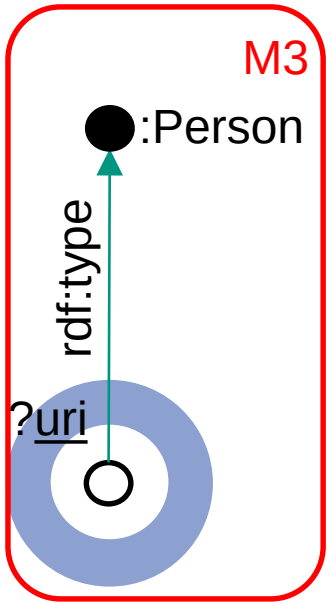
$C \mathrel{\hat{=}}$ Result set is limited with LIMIT as in `LIMIT 10`

$D \mathrel{\hat{=}}$ ME needs to be verbalized in singular (LIMIT = 1)

$E \mathrel{\hat{=}}$ Main entity is ordered as in `ORDER BY DESC(?main)`

$F \mathrel{\hat{=}}$ Main entity has multiple types as in

```
?main rdf:type ex:A. ?main rdf:type ex:B.
```



→ Case: abcdef

→ Result: “People“

Subject template (excerpt)

`Abcdef` 'Distinct ' . $D{TPL}
→ Distinct scientists

`aBcdef` 'Number of ' . $D{TPL}
→ Number of scientists

`AbcdEf` 'The distinct ' . $D{TPL}
→ The distinct scientists

`abCdef` 'Not more than ' . $D{L}
.' '.$D{TPL}
→ Not more than 10 scientists

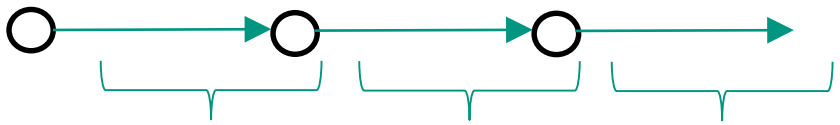
`abcDEf` 'The ' . $D{TSI}
→ The scientist

(Due to depedencies: 36 cases instead of 2^6=64)

Example – Constraint verbalization (1/2)

Control variables

A ≙ the variable is the first optional variable and this variable is not NOTBOUND
B ≙ the variable has exactly one type
C ≙ the variable has more than one type
D ≙ the property is reversed
E ≙ the variable is counted
F ≙ plural form is required
G ≙ ordered by variable
H ≙ descending order
I ≙ the variable is OPTIONAL and NOTBOUND
J ≙ the property is numeric



Classes of properties

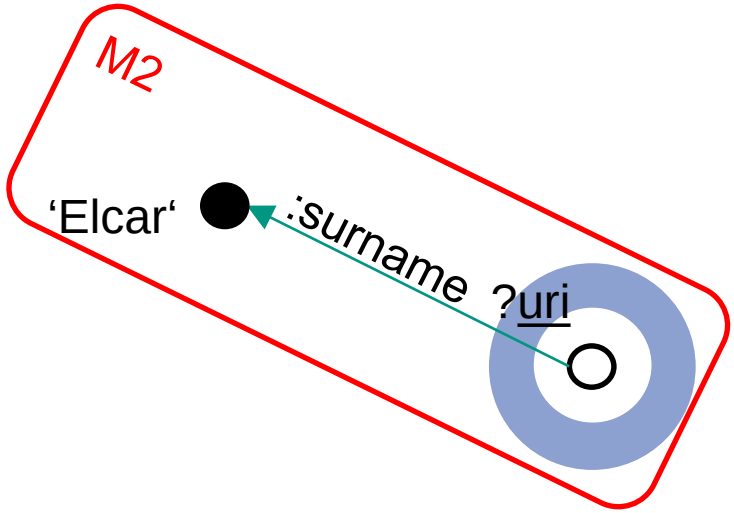
№	Examples	Expansions	№	Examples	Expansions
1	email, hasColor	X has an email Y X has a color Y R: Y is an email of X R: Y is a color of X	5	collaboratesWith	X collaborates with Y R: Y collaborates with X
2	knows	X knows Y R: Y is known by X	6	visiting R: Y is visited by X	X is visiting Y
3	brotherOf, isBrotherOf	X is brother of Y R: Y has a brother X	7	locatedInArea	X is located in area Y in which X is located
4	producedBy, isMadeFrom	X is produced by Y X is made from Y R: Y produces X R: Y is used to make X	8	marriedTo	X is married to Y R: Y is married to X

schema-independence, based on POS

Example – Constraint verbalization (2/2)

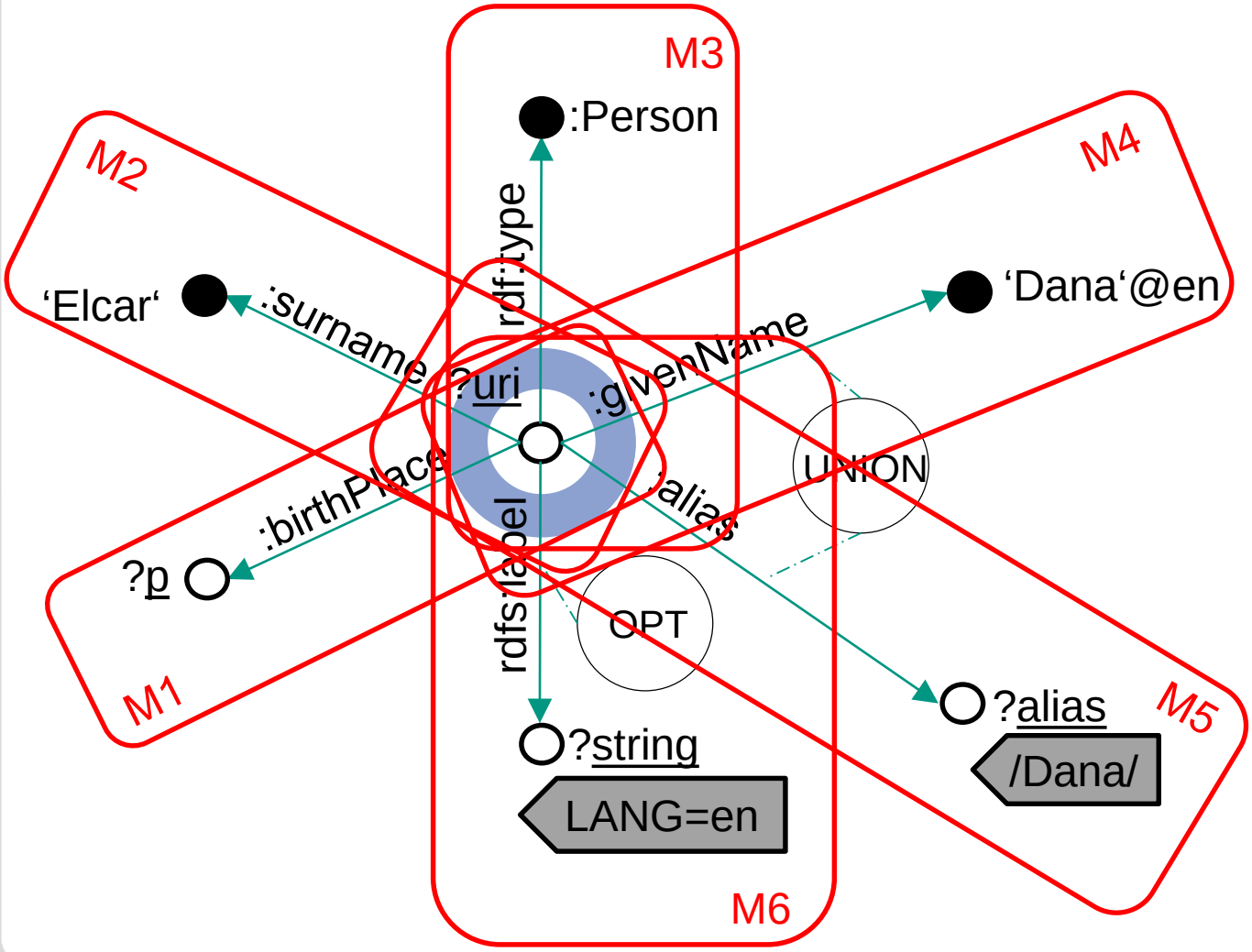
- RV-Constraint template (class 1) (excerpt)

```
abcdefghIJ 'that has no ' . $D{PSI}
           → that has no email
abcdEFGHij 'that have the highest
           number of '.$D{PPL}
           → that have the highest
           number of languages
abcDefghIj 'that is not '.$D{A}
           .' '.$D{PSI}.' of a thing'
           → that is not a holder
           of a thing
```

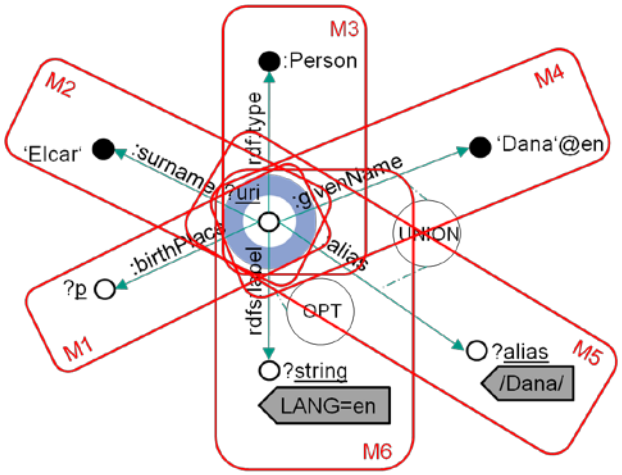


Property class: C1
→ Case: abcdeFghij
→ Result: „ that have a surname “Elcar““

Example – verbalization result



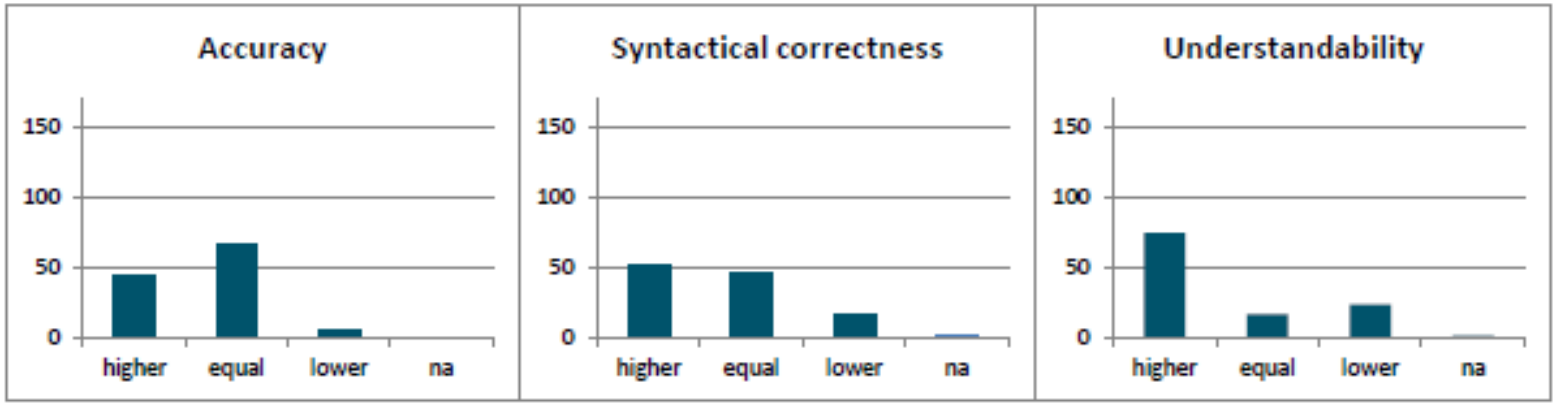
Example – verbalization result



M3 People
M2 that have a surname "Elcar"
and
M1 that have birth places
and
M4 that have the English given name "Dana"
or
M5 that have aliases that match the expression /Dana/
.
M6 Show also, if available,
these people's English labels
and
?p these people's birth places
.

Evaluation (1/2)

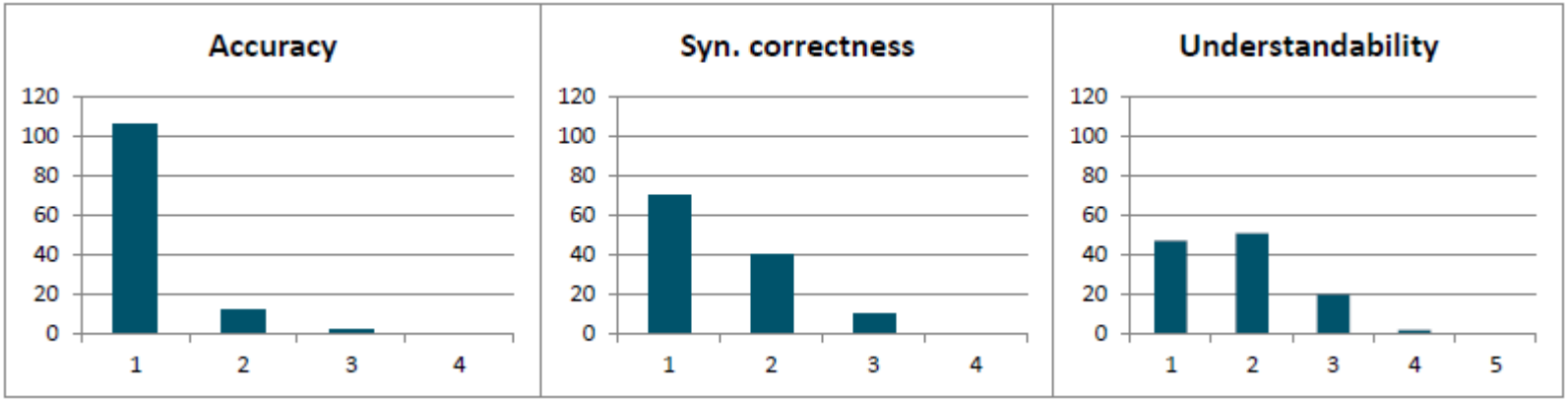
- Comparative evaluation: Spartiqulation vs. SPARQL2NL [Ngonga Ngomo et al., 2013] (bottom-up approach)



6 evaluators, 38 verbalizations

Evaluation (2/2)

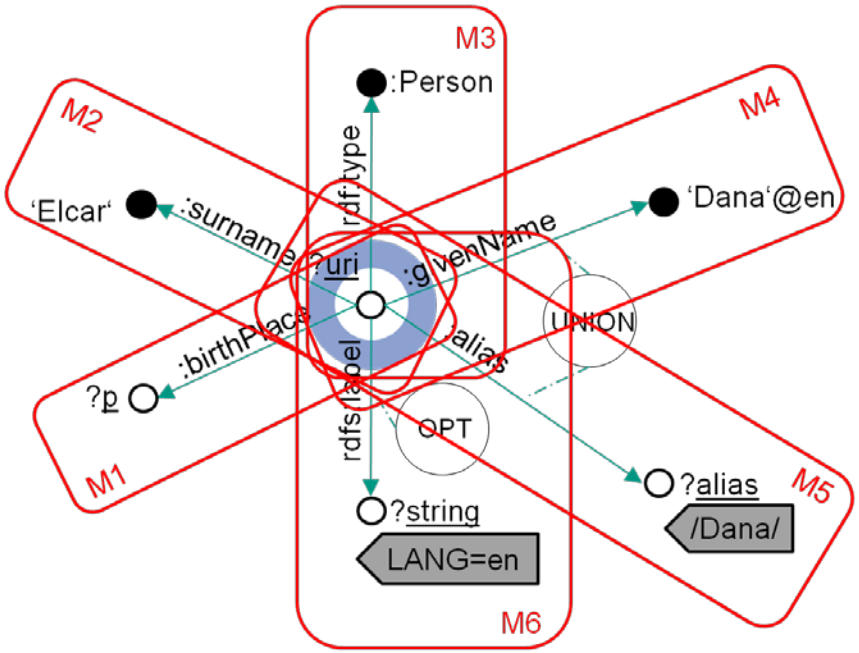
- Non-comparative evaluation



6 evaluators, 40 verbalizations

Conclusions

- Verbalization of SPARQL queries allow users to observe discrepancies between intended questions and generated queries
- Domain-independent approach:
  - Templates are based on linguistic properties of properties
- Evaluation shows
  - high accuracy,
  - acceptable syntactical correctness
  - outperforms SPARQL2NL i.t.o. understandability



?question

#eswc2014Ell

The authors acknowledge the support of the European Commission's Seventh Framework Programme FP7/2007-2013 (PlanetData, Grant 257641) and FP7-ICT-2011-7 (XLike, Grant 288342).