From Natural Language to SPARQL : a prototype

Petar Mitankin, Ontotext

First Project Meeting of MOLTO

8 September 2010

The Conversion Problem

Given:

- ontology
- GF grammars

Find:

- an algorithm that converts *grammatically correct phrases* into *ontology constructions*

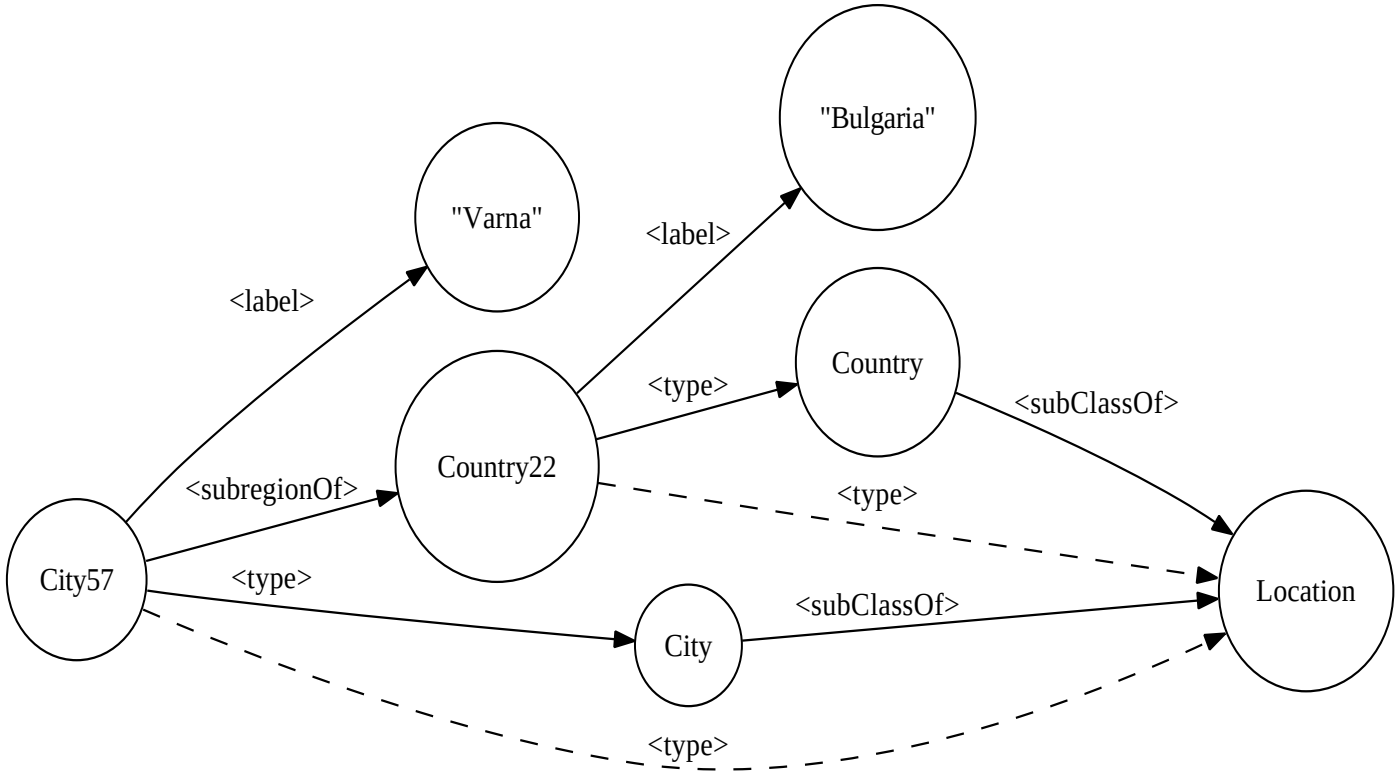
We deal with a concrete instance of the Conversion Problem.

The concrete ontology

- PROTON: classes for named entities and relations between named entities
- dataset: $29,104$ named entities = $6,006$ persons + $8,259$ organizations + $12,219$ locations + $2,620$ job titles

By *ontology* we mean both the scheme that is used to represent the data (PROTON) and the dataset.

The concrete ontology as a directed graph



arcs $\approx 500,000$

arcs + automatically inferred arcs $\approx 1,000,000$

SPARQL

$$\frac{\text{SPARQL}}{\text{ontology}} = \frac{\text{SQL}}{\text{relational database}}$$

```
SELECT DISTINCT ?from ?label ?to WHERE {
   ?from ?label ?to .
}
```

from	label	to
$node'_1$	$label_1$	$node''_1$
$node'_2$	$label_2$	$node''_2$
...		
$node'_N$	$label_N$	$node''_N$

Example: all organizations

```
SELECT DISTINCT ?x WHERE {
   ?x <type> <Organization> .
}
```

x
$node_1$
$node_2$
...
$node_K$

Example: all persons that work as project manager at Ontotext

```
SELECT DISTINCT ?person WHERE {
   ?person <hasPosition> ?jobPos .
   ?jobPos <withinOrganization> ?org .
   ?org <label> "Ontotext".
   ?jobPos <hasTitle> ?jobTit .
   ?jobTit <label> "Project Manager".
}
```

What follows from the SPARQL examples?

$$SPARQL = \frac{\text{SQL}}{\text{relational database}} \cdot \text{ontology}$$

What follows from the SPARQL examples?

SPARQL is nice, but if you want to use it to extract information from our ontology then you have to know PROTON: you have to know very well the graph that we use to represent the data: the names of the nodes, the names of the arcs...

The concrete GF grammars

The Query Grammars:

15 categories: Query, Relation, Kind, Property, Individual, Activity, Name, Loc, Org, Pers, ...
59 functions: ...

The language represented by the Query Grammars:

give me all people
give me all organizations in *L*
give me all persons that work as *JT* at *O*
...

GF is nice: multiple ways to say one and the same thing

64 ways to say
give me all people that work at O:

give me all persons that work at O
give me all people that collaborate in O
give me all persons that collaborate in O
give me the people that work at O
give me the persons that work at O
give me the people that collaborate in O
give me the persons that collaborate in O
give me the names of all people that work at O
give me the names of all persons that work at O
give me the names of all people that collaborate in O
give me the names of all persons that collaborate in O
give me the names of the people that work at O
give me the names of the persons that work at O

GF is very nice: text prediction

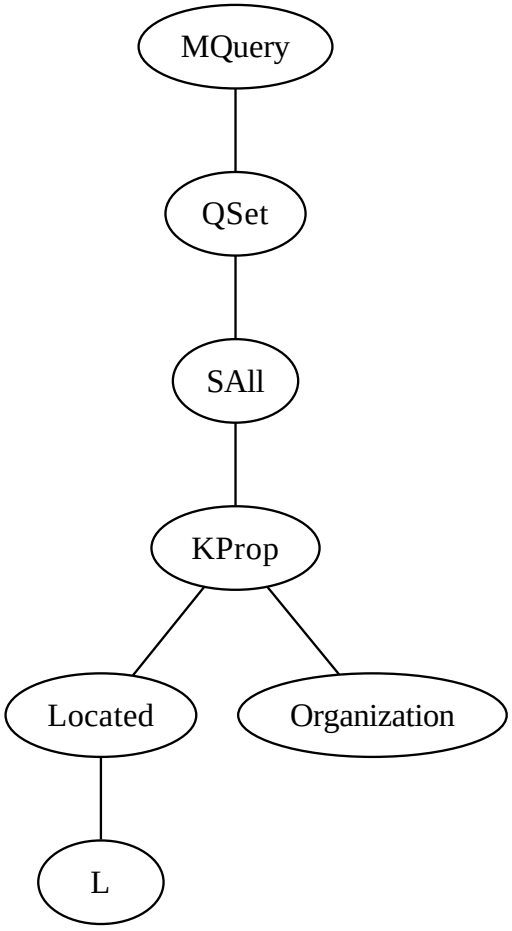
give me

↓

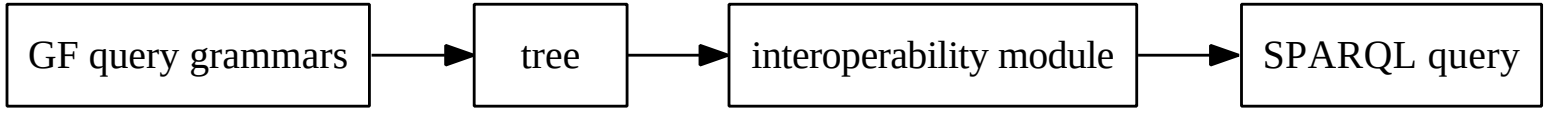
give me a
give me all
give me an
give me information
give me locations
give me names
give me nicknames
give me one
give me organizations
give me other
give me people
...
give me L
give me O
give me P

GF is very very nice: parser

all organizations located in L



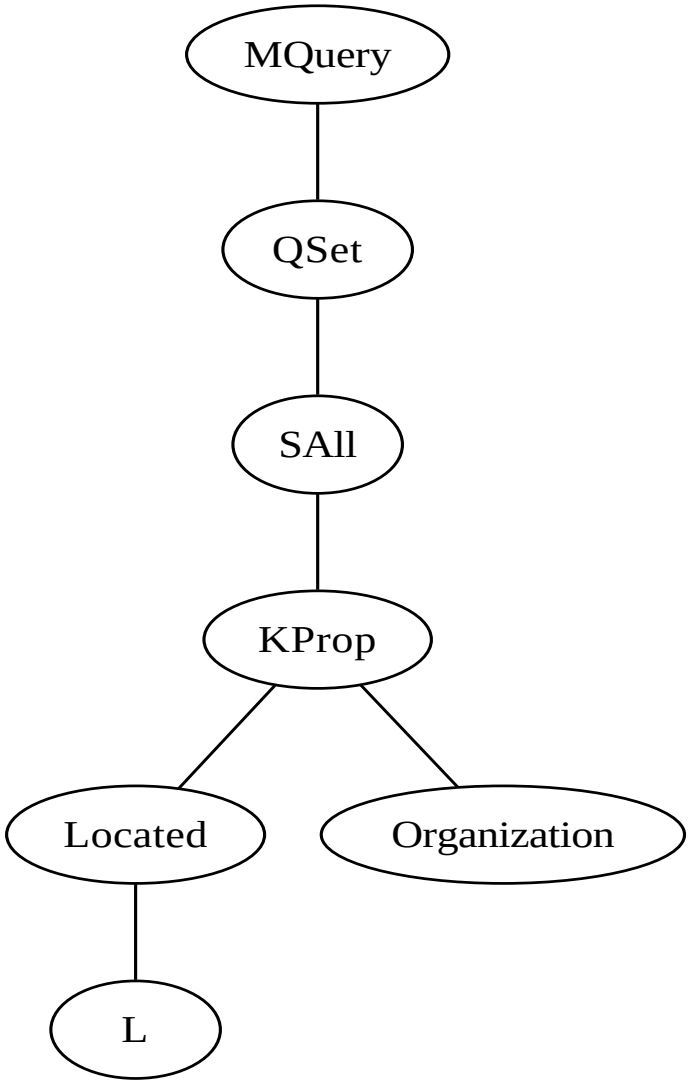
The concrete instance of the Conversion Problem

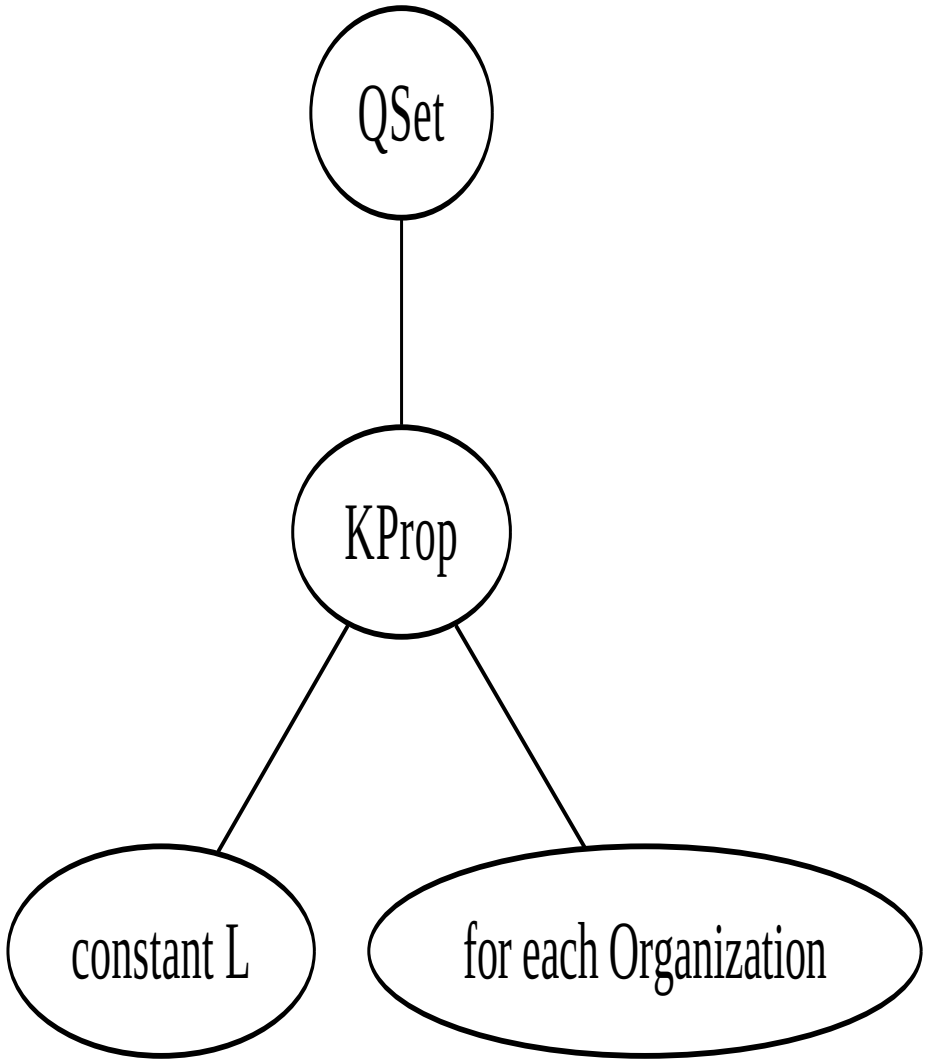


interoperability module = ?

The interoperability module

Step 1: simplify the tree





The interoperability module

Step 2: case study

if the simplified tree ... then

```
SELECT DISTINCT ?organization WHERE {
   ?organization <type> Organization .
   ?organization <locatedIn> ?loc .
   ?loc <label> L .
}
```

else if the simplified tree ... then

SELECT DISTINCT blah blah blah

else if the simplified tree ... then

SELECT DISTINCT blah blah blah

Future work

- generalization of this concrete instance of the Conversion Problem:
  template for interoperability between GF grammars and ontologies
- Enlarge the size of the ontology: FactForge (DBPedia, Freebase, WordNet, ...)
- technical improvements