Reengineering and Refactoring

Eva Blomqvist
eva.blomqvist@liu.se
Department of Computer and Information Science (IDA)
Linköpings universitet
Sweden

Slides partly by Aldo Gangemi, STLab, ISTC-CNR, Italy
Outline

- Introduction and motivation
- From any data source to the Semantic Web, e.g. DB etc.
- Semion method and tool
- Writing Semion rules
Why do I need transformations?

- **A scenario**
  - My system fetches knowledge from different sources in LOD
  - Each of these sources uses its own ontology/vocabulary
- **Another scenario**
  - I have legacy data in a DB or in a custom XML format
  - This data should be integrated with RDF data

How to arrive at a homogeneous representation of knowledge expressed with heterogeneous schemas/vocabularies?
Motivations

the Web of Data is fed by “triplifiers”, tools able to transform content to Linked Data

lack of good practices for knowledge representation and organization

triplifiers implement various methods typically based on bulk recipes which allow for no or limited customization of the process

the transformation relies on predetermined implicit assumptions on the domain semantics of the non-RDF data source
An Example

I want to aggregate the two graphs

dbpedia: Bob_Marley

foaf:name

"Bob Marley"

"Marley, Bob"

skos:prefLabel

"Marley, Bob"

skos:Concept

nyt: 65169961111056171853

dbpedia: Person

rdf:type

"Marley, Bob"

skos:prefLabel

"Bob Marley"

I want to aggregate the two graphs

rdf:type

"Bob Marley"

skos:prefLabel

"Bob Marley"

I want to aggregate the two graphs

rdf:type

"Bob Marley"

skos:prefLabel

"Bob Marley"

I want to aggregate the two graphs
A DB stores data and answers queries

- Aldo is 48
- Aldo works in Rome
- Aldo does research on Semantic Web

- STLab was founded in 2008
- STLab is in Italy
- STLab does research on semantic technologies

<table>
<thead>
<tr>
<th>nome</th>
<th>data di nascita</th>
<th>luogo di lavoro</th>
<th>temi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aldo</td>
<td>08-16-962</td>
<td>Roma</td>
<td>Semantic Web</td>
</tr>
<tr>
<td>Alfio</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>laboratori</th>
<th>data_di_fondazione</th>
<th>sede</th>
<th>temi</th>
</tr>
</thead>
<tbody>
<tr>
<td>STLab</td>
<td>2008</td>
<td>Italia</td>
<td>Tecnologie Semantiche</td>
</tr>
<tr>
<td>LOA</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Complex queries?

- *Who is interested in “Semantic Web” and is working in the same country as STLab is located?*

```
workplace | same concept? | seat  
Rome     |               | Italy 

<table>
<thead>
<tr>
<th>nome</th>
<th>data di nascita</th>
<th>luogo_di_lavoro</th>
<th>temi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aldo</td>
<td>08-16-962</td>
<td>Roma</td>
<td>Semantic Web</td>
</tr>
<tr>
<td>Alfio</td>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>laboratori</th>
<th>data_di_fondazione</th>
<th>sede</th>
<th>temi</th>
</tr>
</thead>
<tbody>
<tr>
<td>STLab</td>
<td>2008</td>
<td>Italia</td>
<td>Tecnologie Semantiche</td>
</tr>
<tr>
<td>LOA</td>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>
```

No answer
Is mapping enough?

<table>
<thead>
<tr>
<th>workplace</th>
<th>mapped ...</th>
<th>seat</th>
<th>same country?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rome</td>
<td></td>
<td>Italy</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>nome</th>
<th>data di nascita</th>
<th>luogo_di_lavoro</th>
<th>temi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aldo</td>
<td>08-16-96</td>
<td>Roma</td>
<td>Semantic Web</td>
</tr>
<tr>
<td>Alfio</td>
<td>....</td>
<td>....</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>laboratori</th>
<th>data_di_fondazione</th>
<th>sede</th>
<th>temi</th>
</tr>
</thead>
<tbody>
<tr>
<td>STLab</td>
<td>2008</td>
<td>Italia</td>
<td>Tecnologie Semantiche</td>
</tr>
<tr>
<td>LOA</td>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>

Persons

Labs
Publishing DB data as RDF on the Web

```
<table>
<thead>
<tr>
<th>nome</th>
<th>data di nascita</th>
<th>luogo di lavoro</th>
<th>temi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aldo</td>
<td>08-16-96</td>
<td>Roma</td>
<td>Semantic Web</td>
</tr>
<tr>
<td>Alfio</td>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>laboratori</th>
<th>data_di_fondazione</th>
<th>sede</th>
<th>temi</th>
</tr>
</thead>
<tbody>
<tr>
<td>STLab</td>
<td>2008</td>
<td>Italia</td>
<td>Tecnologie Semantiche</td>
</tr>
<tr>
<td>LOA</td>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>
```
Data linking and querying on the web of data

- Who is interested in “Semantic Web” and is working in the same country as STLab is located?
How to answer really tough queries?

 › Who is interested in “Semantic Web” and is working in a country where the unemployment rate is lower than 5%?
Knowledge transformation issues

- Syntactic interoperability bottleneck (platform + data model)
  - e.g. rdb, eav, xml, text, prolog, N3
  - e.g. kos rdb with adjacency list, path enumeration

- Semantic interoperability bottleneck (logical + conceptual level)
  - e.g. rdb with: lexical, statistical, formal data
  - e.g. two different databases on the same topic

- Social, pragmatic interoperability bottleneck (privacy, sustainability, policy)
  - e.g. different requirements, organizational contexts, etc.
Dealing with web semantics: current state

- Much enthusiasm, a lot of nice, different ideas
- Much confusion and mutual misunderstanding between “scruffies” and “neats”
  - Pushing formal semantics beyond its limits (e.g. the “owl:sameAs” dispute)
  - Doing ad-hoc apps
  - Mixing up strings, classes, terms, concepts, topics, tags, etc.
  - Trivializing transformation from social to formal semantics (e.g. when translating a syntactic frame directly to an OWL construct)
Some techniques for semantic data reuse

- Virtual linked data
  - Automatic RDB schema conversion to RDFS
  - RDB data browsing and on-demand automatic conversion to RDF
  - Sample tools: Sparql endpoint+D2R
  - Dataset example: IMDB
  - +Time to usage –Flexibility

- Ontology-based access with ad-hoc queries
  - (DL-Lite) ontology to be designed separately
  - Ad-hoc SQL query on RDB, “embedded” in class spec
  - On demand ontology-based navigation
  - Sample tools: Mastro+Quonto
  - +Complexity –Flexibility –Time to usage

- Physical linked data with custom ontologies
  - Custom conversion of RDB/XML to one or more OWL ontologies
  - Custom conversion of data to RDF-OWL datasets that can be published and queried
  - Sample tools: Sparql endpoint+Semion
  - Sample datasets: DBpedia, data.cnr.it
  - +Flexibility ±Time to usage ±Complexity

- Key aspects
  - Mapping specification
  - Consumable RDF data semantics
Transformation patterns

Types of transformation patterns

1. Direct structural morphism

<table>
<thead>
<tr>
<th>Broader</th>
<th>Narrow concept</th>
<th>Broad concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>:Broader rdf:type dbs:Table</td>
<td>:Narrow_concept rdf:type dbs:Column</td>
<td>:France rdf:type dbs:Datum</td>
</tr>
<tr>
<td>:Broader</td>
<td>Paris</td>
<td>France</td>
</tr>
</tbody>
</table>

2. Semantic interpretation

<table>
<thead>
<tr>
<th>Broader</th>
<th>Narrow concept</th>
<th>Broad concept</th>
</tr>
</thead>
</table>
Transformation patterns (cont.)

3. Re-interpretation, e.g. through alignment patterns
   - :Narrow_concept ----> skos:Concept
     - also as mediated semantic interpretation
     - also as revised semantic interpretation

4. Production: new entities, vocabulary/string manipulation
   - :Narrow_concept ----> :Concept
   - “%p” ----> :hyponym
   - “plant flora plant_life” ----> :Plant, :Flora, :PlantLife
Semion - Method and Tool
A common recipe

- each table is a rdfs:Class
- each table record is an owl:Individual
- each table column is a rdf:Property
Example

Class: Person

DatatypeProperty: firstName

DatatypeProperty: lastName

Individual: Person1
Type: Person
Facts: firstName "Aldo"
lastName "Gangemi"

Individual: Person2
Type: Person
Facts: firstName "Valentina"
lastName "Presutti"

...
Implications

- Limited customization of the transformation process
- Difficulty in adopting good practices of knowledge reengineering and ontology design
- Limited exploitation of OWL expressivity for describing the domain
The Semion Approach

original source

schema/data

Mediator
ontology

ontology from schema

individuals from data

syntactic reengineering

refactoring

A-box refactoring

T-box refactoring

ontology from data
The Linguistic Meta-Model (LMM)

LMM allows a semiotic cognitive representation of knowledge based on the so-called semiotic triangle.

LMM plays the role of a mediator ontology.

A mediator is an ontology that describes the entities used for expressing the informal semantics that we use for organizing our knowledge.

Most knowledge representation schemata can be aligned to the semiotic triangle.
just extract RDF triples!

Reengineering process

Data source
- Define/acquire data source
- Define/acquire mapping description

Data source + MM + mapping
- Apply reengineering

RDF dataset

Refactoring process

RDF dataset
- Define alignments to LMM
- Perform alignments to LMM

LMM aligned dataset
- Define alignments to a formal semantics MM
- Perform alignments to the formal semantics MM

LMM aligned dataset
- Define alignments to a logic MM
- Perform alignments to the logic MM

express the domain semantics
The Semion Reengineer

- It does not add any semantics, but just the RDF format
- Semion needs the meta-model of the structure of the source (and some code)
- Currently supports RDBMS and XML
- Supported sources can be extended by providing new reengineering services as an OSGi bundle (not available yet)
Basic idea
A meta-model for RDBs
Example of transformation of a DB

Class: Table

ObjectProperty: hasRecord
  Domain: Table
  Range: Record
  inverseOf: isRecordOf

ObjectProperty: isRecordOf
  Domain: Table
  Range: Record
  inverseOf: hasRecord

Individual: Person
  Type: Table
  Facts: hasRecord AldoGangemi

Individual: AldoGangemi
  Types: Record
  Facts:
  hasDatum AldoGangemiFirstName
  hasDatum AldoGangemiLastName
  isRecordOf Person

<table>
<thead>
<tr>
<th>Person</th>
<th>FirstName</th>
<th>LastName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aldo</td>
<td>Gangemi</td>
<td></td>
</tr>
<tr>
<td>Valentina</td>
<td>Presutti</td>
<td></td>
</tr>
<tr>
<td>Eva</td>
<td>Blomqvist</td>
<td></td>
</tr>
</tbody>
</table>
Reengineering Rules

- **Table** -> individual of **dbs:Table**
- **Column** -> individual of **dbs:Column**
- **Record** -> individual of **dbs:Record**
- **Field** -> individual of **dbs:Datum**
Primary Keys

- primary keys are used for URI generation

<table>
<thead>
<tr>
<th>Person</th>
<th>FirstName</th>
<th>LastName</th>
<th>id (PK)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Aldo</td>
<td>Gangemi</td>
<td>stlab.istc-cnr1</td>
</tr>
</tbody>
</table>

Individual: Person_stlab.istc-cnr1
Type: Record
**Foreign Keys**

- Foreign keys identifies relations between tables and are mapped to relations between individuals

<table>
<thead>
<tr>
<th>Person</th>
<th>FirstName</th>
<th>LastName</th>
<th>id (PK)</th>
<th>affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Aldo</td>
<td>Gangemi</td>
<td>stlab.istc-cnr1</td>
<td>istc-cnr</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Institute</th>
<th>id (PK)</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>istc-cnr</td>
<td>Istituto di Scienze e Tecnologie della Cognizione – CNR</td>
</tr>
</tbody>
</table>

Individual: Person_stlab.istc-cnr1  
Type: Record  
Facts: hasDatum Person_stlab.istc-cnr1_affiliation

Individual: Person_stlab.istc-cnr1_affiliation  
Type: Datum  
Facts: hasContent Institute_istc-cnr

Individual: Institute_istc-cnr  
Type: Record  
Facts: hasDatum Institute_istc-cnr_name
...and for XML
The Semion Refactorer

- Allows to align a data set expressed with a specific vocabulary/ontology to another vocabulary/ontology
- Is expressed as a set of rules
- Rules are expressed in a human readable syntax called SemionRule Syntax and can be transformed into
  - SWRL rules for reasoning
  - SPARQL CONSTRUCT for pure refactoring
- Rules realize recipes that can be saved (*refactoring patterns*)
Basic idea
The Reengineering View
The Refactoring View
SemionRule Syntax

dbs = <http://ontologydesignpatterns.org/ont/iks/dbs_l1.owl#> .
owl = <http://www.w3.org/2002/07/owl#> .

myRule[
  is(dbs:Table, ?x) . has(dbs:hasColumn, ?x, ?y)
  ->
  is(owl:Class, ?x)
]

as a SPARQL CONSTRUCT

PREFIX dbs: <http://ontologydesignpatterns.org/ont/iks/dbs_l1.owl#> .
PREFIX owl: <http://www.w3.org/2002/07/owl#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

CONSTRUCT{ ?x rdf:type owl:Class }
WHERE{
  ?x rdf:type dbs:Table .
  ?x dbs:hasColumn ?y
}
...and as a SWRL rule

```
<swrl:Variable rdf:ID="x"/>
<swrl:Variable rdf:ID="y"/>
<swrl:Imp>
  <swrl:body rdf:parseType="Collection">
    <swrl:ClassAtom>
      <swrl:classPredicate rdf:resource="&dbs;Table"/>
      <swrl:argument1 rdf:resource="#x"/>
    </swrl:ClassAtom>
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&dbs;hasColumn"/>
      <swrl:argument1 rdf:resource="#x"/>
      <swrl:argument2 rdf:resource="#y"/>
    </swrl:IndividualPropertyAtom>
  </swrl:body>
  <swrl:head rdf:parseType="Collection">
    <swrl:ClassAtom>
      <swrl:classPredicate rdf:resource="&owl;Class"/>
      <swrl:argument1 rdf:resource="#x"/>
    </swrl:ClassAtom>
  </swrl:head>
</swrl:Imp>
```
Stanbol Rule Syntax

In Stanbol a rule is defined as

\[ \text{ruleName}[\text{body} \rightarrow \text{head}] \]

where:

- The ruleName identifies the rule
- The body is a set of atoms that must be satisfied when evaluating the rule
- The head or consequent is a set of atoms that must be true if the condition is evaluated to be true
- Both body and head consist of a list of conjunctive atoms
  - \( \text{body} = \text{atom}_1 . \text{atom}_2 . \ldots . \text{atom}_N \)
  - \( \text{head} = \text{atom}_1 . \text{atom}_2 . \ldots . \text{atom}_M \)
- The conjunction \( \land \) in Stanbol Rules is expressed with the symbol “. ”
Rule Atoms

- An atom is the smallest unit of a rule
  - e.g.: in predicate calculus
    \[ Person(x) \Rightarrow hasFather(x, y) \]
    Person(\(*\)) and hasFather(\(*, *, \)) are two atoms

- In Stanbol basic atoms are
  - Class assertion atom – is(foaf:Person, ?x)
  - Individual assertion atom – has(foaf:knows, ?x, ?y)
  - Data value assertion atom – values(foaf:firstName, ?x, ?fname)
  - ...

- There are also comparison atoms, string and integer manipulation atoms

- The atoms may contain
  - constants: they consist of URIs or Literals (values)
    - e.g. http://dbpedia.org/resource/Bob_Marley is a constant, but “Bob Marley”^^xsd:string is a constant too
  - variables: any identifier preceded by ?
    - e.g. ?x is a variable, but also ?y is a variable
Sample rule

Considering Stanbol Rules, the FOL formula

\[ \text{hasFather}(x,y) \land \text{hasBrother}(y,z) \Rightarrow \text{hasUncle}(x,z) \]

becomes

\[
\text{myRule[ has(\text{<http://myont.org/hasFather>}, ?x, ?y) . has(\text{<http://myont.org/hasBrother>}, ?y, ?z) -> has(\text{<http://myont.org/hasUncle>}, ?x, ?z) ]}
\]
Namespace Prefixes

- URIs are useful, but sometime too long for humans
- We can use namespace prefixes instead of full URIs in rule atoms
- e.g:
  
  myont = <http://myont.org/>
  
  myRule[ has(myont:hasFather, ?x, ?y) .
          has(myont:hasBrother, ?y, ?z) ->
          has(myont:hasUncle, ?x, ?z) ]
Define a refactoring recipe

we want to use the FOAF vocabulary instead of SKOS

- *skos:* Concept
- *rdf:type*
- *skos:prefLabel*
- *nyt:* 65169961111056171853
- *skos:prefLabel*
Define a refactoring recipe

\[
\text{skos} = \langle \text{http://www.w3.org/2004/02/skos/core}\rangle. \\
\text{foaf} = \langle \text{http://xmlns.com/foaf/0.1}\rangle. \\
\]

\[
\text{conceptToPerson}[ \text{is(skos:Concept, } ?x) \rightarrow \text{is(foaf:Person, } ?x) ]. \\
\text{labelRule}[ \text{values(skos:prefLabel, } ?x, ?y) \rightarrow \text{values(foaf:name, } ?x, ?y) ]
\]
Exercise

- Download Semion and launch it as follows:
  - (Mac) java -jar -Xmx512m -XstartOnFirstThread /LocalPathname/it.cnr.istc.semion.tool-0.6-SNAPSHOT.one-jar.jar
  - (Win) java -jar -Xmx512m \LocalPathname \it.cnr.istc.semion.tool-0.6-SNAPSHOT.one-jar.jar

- Connect to the indicated database and perform reengineering first, and alignment (refactoring) second