TOWARDS SEMI-AUTOMATIC GENERATION OF R2R MAPPINGS

Valéria Magalhães Pequeno
Vânia M. Ponte Vidal
Tiago da Silva Vinuto
A mapping specifies the equivalence between ontology terms in different ontologies

Important when integrating multiple knowledge bases
  - different datasets use different vocabularies to represent the same concept of the real-world
Introduction – R2R mappings

- Declarative language
- Based on SPARQL
  - Easy to the users understand the generated mapping
- For publish mappings between RDF vocabularies
- Ready for use
  - We can generate RDF triples to the target ontology from R2R mapping
  - The mappings can be published on the web
Introduction – R2R mappings

- Example 1:

  Source Ontology: Dbpedia
  Target Ontology: MyMusic

  mp: CMA1
  a r2r:classMapping;
  r2r:prefixDefinitions "dbo: <http://...> . mo:<http://...>";
  r2r:sourcePattern "?SUBJ a dbo:RecordLabel";
  r2r:targetPattern "?SUBJ a mo:Label".

  ⟨ s rdf:type dbo:RecordLabel ⟩   ⟷   ⟨ s rdf:type mo:Label ⟩
Introduction - Problem

- When more complex mapping is necessary

Source Ontology: MySpace

```
mp: MCA2
  a r2r:ClassMapping;
  r2r:prefixDefinitions "mo:<···>.myspo:<···>";
  r2r:sourcePattern "?SUBJ a myspo:MusicArtist;
                     myspo:recordLabel ?r;
  r2r:targetPattern "?u a mo:Label";
  r2r:transformation "?u= concat(?SUBJ, xpath-encode-for-uri(?r))" .
```

Target Ontology: MyMusic

```
< s myspo:recordLabel "apple" >   < u rdf:type mo:Label >
```

Deep knowledge about the ontologies

Deep knowledge about the R2R language
Our Proposal

- A mapping pattern library that:
  - Describe usual mapping problems between RDF vocabularies
  - Provides solutions that facilitates the generation of mappings
    - For example, semi-automatically generate R2R mappings
  - Provides templates handle with simple and complex mappings
  - Present constraints between the different mappings to guarantee that the whole set of mappings between the source and the target ontologies generate correct instances
Our Proposal

- Definition of a formal language (mapping assertions) that:
  - Declaratively define mappings
  - Highest level of abstraction than R2R mapping
  - Concise
  - Support the most part of the data alignment usually necessary to transform RDF data
Mapping Pattern - Library

Class
- CM1: Semantically equivalent class
- CM2: Semantically non-equivalent class

Object Property
- OM1: Direct Object Property
- OM2: Path Property
- OM3: Embedded Class

Datatype Property
- DM1: 1-1 Direct Mapping
- DM2: 1-1 Direct Mapping with Transformation Function
- DM3: 1-n Direct Mapping
- DM4: 1-1 Datatype Property throw a Path
- DM5: 1-1 Datatype Property throw a Path with Transformation Function
- DM6: 1-N Datatype Property throw a Path
Mapping Pattern - Template

- Name (of the pattern)
- Alias
  - Alternative names or synonyms for the pattern
- Problem
  - Description of the goals of the pattern
- Context
  - The applicability of the pattern
- Force
- Solution
  - Description of the solution using Mapping Assertions, Mapping Rules and R2R Mapping
- Examples of use
Mapping Pattern - Example

- **Name:** Semantically Non-Equivalent Class Mapping
- **Alias:** CM2

- **Problem:** How should we specify the mapping of the instances of a class $C_S$ in $V_S$ into instances of a class $C_T$ in $V_T$?
Mapping Pattern – CM2 (cont.)

- **Context:**
  - $C_T$ and $C_S$ are classes in vocabularies $V_T$ and $V_S$, respectively
  - $A_1, \ldots, A_n$ are datatype properties whose domain is $C_S$
  - $C_T$ and $C_S$ are NOT semantically equivalent, i.e. they do not represent the same object of the real world
  - $f$ is a condition of selection (a predicate) over instances of $C_S$ ($f$ is optional)
  - The terms may have the same name or different names in the different ontologies

- **Example:**

  **Source Ontology: MySpace**
  - `myspo:MusicArtist`
  - `myspo:recordLabel`

  **Target Ontology: MyMusic**
  - `mo:Label`
  - `ex:labelName`

  **Example:**
  - Class `mo:Label` corresponds to class/property combination `myspo:MusicArtist[myspo:recordLabel]`
Mapping Pattern – CM2 (cont.)

- **Force:**
  - The mapping can be complete or partial

- **Solution:**
  - **Mapping Rule:** \( C_T(u) \leftrightarrow C_S(s); \ f(s) \; ; \ \text{hasUri}[A_1,\ldots,A_n](s, u) \)

  If \(<s \text{ rdf:type } C_S>\) such that \(f(s) = \text{true}\) and \(u = \text{hasUri}[A_1,\ldots,A_n](s)\), then \(<u \text{ rdf:type } C_T>\)
Mapping Pattern – CM2 (Example)

- **Force:**
  - The mapping can be complete or partial

- **Solution:**
  - **Mapping Rule:** $C_T(u) \leftarrow C_S(s); \text{ hasUri}[A_1,\ldots,A_n](s, u)$

  If $<s \text{ rdf:type } C_S>$ such that $f(s) = \text{true}$ and $u = \text{hasUri}[A_1,\ldots,A_n](s, s)$, then $<u \text{ rdf:type } C_T>$

- **Example:**

  **Mapping Rule:**
  \[ R_1: \text{mo:Label}(u) \leftarrow \text{myspo:MusicArtist}(s) ; \text{ hasUri}[\text{myspo:recordLabel}](s, u) \]
Mapping Pattern – CM2 (cont.)

- **Force:**
  - The mapping can be complete or partial

- **Solution:**
  - **Mapping Rule:** $C_T(u) \leftarrow C_S(s); f(s); \text{hasUri}[A_1, \ldots, A_n](s, u)$
    
    If $<s \text{ rdf:type } C_S>$ such that $f(s) = \text{true}$ and $u = \text{hasUri}[A_1, \ldots, A_n](s, u)$, then $<u \text{ rdf:type } C_T>$

  - **Mapping Assertion:** $\psi: C_T \equiv C_S[A_1, \ldots, A_n] / f$
Mapping Pattern – CM2 (cont.)

- **Force:**
  - The mapping can be complete or partial

- **Solution:**
  - **Mapping Rule:** \( C_T(u) \leftarrow C_S(s); \ f(s); \ \text{hasUri}[A_1, \ldots, A_n](s, u) \)
    
    If \( \langle s \ \text{rdf:type} \ C_S \rangle \) such that \( f(s) = \text{true} \) and \( u = \text{hasUri}[A_1, \ldots, A_n](s) \), then \( \langle u \ \text{rdf:type} \ C_T \rangle \)

  - **Mapping Assertion:** \( \psi: C_T \equiv C_S[A_1, \ldots, A_n] / f \)

  - **Example:**

  Source Ontology: MySpace
  
  - `myspo:MusicArtist`
  - `myspo:recordLabel`

  Target Ontology: MyMusic
  
  - `mo:Label`
  - `ex:labelName`

  **Mapping Rule:**
  
  \( R_1: \ \text{mo:Label}(u) \leftarrow \text{myspo:MusicArtist}(s) ; \ \text{hasUri}[\text{myspo:recordLabel}](s, u) \)

  **Mapping Assertion:**
  
  \( CMA_2: \ \text{mo:Label} \equiv \text{myspo:MusicArtist}[\text{myspo:recordLabel}] \)
Mapping Pattern – CM2 (cont.)

▪ Force:
  ▪ The mapping can be complete or partial

▪ Solution:
  ▪ Mapping Rule: $C_T(u) \leftarrow C_S(s); f(s) \text{ hasUri}[A_1, \ldots, A_n](s, u)$
  
  If $<s \text{ rdf:type C}_S>$ such that $f(s) = true$ and $u = \text{hasUri}[A_1, \ldots, A_n](s)$, then $<u \text{ rdf:type C}_T>$

  ▪ Mapping Assertion: $\psi: C_T \equiv C_S[A_1, \ldots, A_n] / f$

  ▪ R2R Mapping: template T2

```xml
#Class Mapping
#CMA $\Psi_c: C_T \equiv C_S[A_1, \ldots, A_n] / f$
mp: mca2
  a r2r: ClassMapping;
  r2r:prefixDefinitions “prefixExp”;  
  r2r:sourcePattern “?SUBJ a s:C_S sQuery”;  
  r2r:targetPattern “?s a s:C_T”;  
  r2r:transformation “?s = generateUri(?SUBJ, A_1, \ldots, A_n)”.
```
Mapping Pattern – CM2 (cont.)

- R2R Mapping (Example):
  
  Template T2

  #Class Mapping
  #CMA \( \Psi: C_T \equiv C_S[A_1, \ldots, A_n] / f \)
  mp: mca2
    a r2r: ClassMapping;
    r2r:prefixDefinitions "prefixExp";
    r2r:sourcePattern "?SUBJ a s:C_S sQuery";
    r2r:targetPattern "?s a s:C_T";
    r2r:transformation "?s = generateUri(?SUBJ, A_1, \ldots, A_n)".

  CMA_2: mo:Label \equiv myspo:MusicArtist[myspo:recordLabel]

  Source Ontology: MySpace
  Target Ontology: MyMusic

  myspo:MusicArtist

  mo:Label

  myspo:recordLabel

  ex:labelName

  MySpace

  MyMusic

  r2r:sourcePattern "?SUBJ a myspo:MusicalArtist, myspo:recordLabel ?x";
  r2r:targetPattern "?s a mo:Label";
  r2r:transformation "?s = generateUri(?SUBJ, ?x)".
APPLYING MAPPING PATTERNS TO GENERATE R2R MAPPINGS

- Based on our pattern library, we created algorithms to semi-automatically generate R2R mappings.

- The process to create R2R mappings to transform instances from an ontology into another one consists of two steps:
  1. Define the MAs that formally specify the relationships between the target ontology and the source ontology.
  2. Generate a set of R2R mappings based on the MAs generated in step 1, in order to populate the target ontology with values from the source(s) ontology(ies).

- RBA (R2R by Assertions)
  - A tool for helping the designer in the process of definition of the mappings, which uses the proposed patterns.
Conclusions

- We presented a proposal to semi-automatically generate R2R mappings using mapping patterns.

- The current proposal allows us:
  - To specify mappings between terms of different ontologies in a clear and concise way.
  - Generate mappings that are ready to use in real scenarios.
  - Group the most common mapping problems.

- We had developed a tool, the RBA.

- We intend to carry out a deep study to show how our proposal is generally useful.
Thank you very much for your attention!
The RBA

- The target ontology’s schema
- The source ontology’s schema
- The source ontology
  - The RDF triples of the source ontology
  - The file’s name to keep the RDF triples of the target ontology
RBA – Mapping’s specification
RBA – Mapping’s specification
RBA – Generating R2R Mappings