OWL Matching Patterns Backed by Naming and Ontology Patterns

Ondřej Šváb-Zamazal and Vojtěch Svátek

Department of Information and Knowledge Engineering, University of Economics,
W. Churchill Sq.4, 130 67 Prague 3, Czech Republic
{ondrej.zamazal, svatek}@vse.cz

Abstract. The paper introduces a novel concept of ontology matching patterns firmly based on the OWL language. Such a matching pattern consists of source and target ontology patterns, and involves naming patterns for detecting the ontology patterns as well as the matching pattern as such. We provide definitions of the matching pattern's constituents along with illustrative examples. Furthermore, we discuss differences between two kinds of matching patterns. Finally, we demonstrate the applicability of these patterns within ontology alignment evaluation and ontology transformation.

Keywords: Ontological Engineering, Ontology Matching, Ontology Design Patterns

1 Introduction

Nowadays *ontologies*[1] are an omnipresent means of knowledge representation, interlinking atomic entities as well as complex entity descriptions. Being by definition 'formal specifications of shared conceptualizations' [1], ontologies should enable clear communication between machines and/or people. However, there usually exists more than one ontology even within a single community, which makes mutual interoperability complicated. Discrepancies between ontologies are called *ontology mismatches. Ontology matching* [2] aims at facilitating interoperability between different ontologies related to the same domain of interest. The output of ontology matching is a set of correspondences called *alignment.* A *correspondence* is a relation (mostly logical equivalence) between two entities, each belonging to different ontology. We can distinguish between a simple and a complex correspondence. A *simple correspondence* aligns two atomic entities, while a *complex correspondence* deals with at least one complex entity description.

Shortly after the semantic web emerged, the idea of using design patterns as 'solutions to recurring design problems' was applied to ontologies [3, 13]. Currently the portal *OntologyDesignPatterns.org*, built as main result of the most

[1] Particularly, we have in mind OWL ontologies, see http://www.w3.org/TR/owl2-primer/.

prominent ontology design pattern initiative, collects ontology design patterns dealing with e.g. reasoning features of ontologies, recurrent content-based design patterns, or naming conventions for ontologies. One of the most recent categories of ontology patterns, called *Alignment Design Patterns* [8], aims to capture recurrent relations between entities across two ontologies.

In this paper we present the novel notion of matching pattern firmly based on the OWL language, which contains (structural) ontology patterns and (ontology) naming patterns. Although in our previous works we occasionally (often implicitly) referred to such patterns, this is the first attempt to systematically explain this notion and present a synoptic view of its applications.

Generally, a matching pattern[2] is a pattern dealing with (at least) two ontologies. These patterns could include *OWL entities*, *OWL expressions*, *OWL axioms*, and correspondences between entities and/or expressions across ontologies, see Section 3.

In principle, the users of ontologies or software tools can benefit from matching patterns for:

– Designing (with different level of automation and different final applications) of *alignments*.
– Model *transformation*. This is the topic of Section 4.2 describing exploitation of matching patterns within transformation patterns. Transformation could be useful for e.g.
  • better alignment [12],
  • easier ontology import [15],
  • better tractability by a reasoner.
– Evaluation of matchers [11], [16], see Section 4.1.

In the next section we provide an overview of the state of the art. Afterwards, Section 3 presents matching patterns from the PatOMat framework where definitions are given along with illustrative examples. This section is concluded with discussion about differences wrt. Scharffe's alignment design patterns. In Sections 4.1 and 4.2 there are presented two applications of matching patterns. The paper is wrapped up with Conclusions and Future Work in Section 5.

2 State of the Art

Alignment design patterns and its library have been introduced by F. Scharffe [9, 8]. As written on the portal,[3] 'alignment patterns are template representing frequent types of alignments occurring when aligning ontologies'. The patterns are divided into five general categories: *attribute correspondence*[4], *class correspondence*, *relation correspondence*, *individual correspondence*, and *composite patterns*.

[2] We originally used a synonymous term 'mapping patterns' which we introduced in [16].
[3] http://ontologydesignpatterns.org/wiki/Category:AlignmentOP
[4] Scharffe's notions of attribute and relation roughly correspond to that of OWL property.

Let us look at one example from [9] where in the "Wine Ontology"[5] there is a class *BordeauxWine* representing wine made in the region around Bordeaux. In the "Ontologie du Vin"[6] there is no direct conceptualization of this wine, however, these wines are expressed as instances of *Vin* class with an attribute *terroir* indicating the wine provenance. In the pattern library there is *Class by Attribute Value* pattern such that a correspondence links one class (BordeauxWine) to another class (Vin) the scope of which is narrowed by the value of an attribute (terroir) of a certain class (Location), see Figure 1.

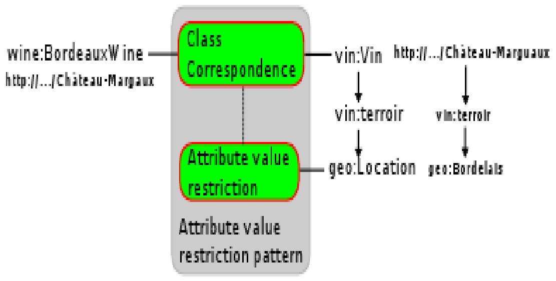
wine:BordeauxW ine
http://.../Château-Margaux
Class Correspondence
vin:Vin
http://.../Château-Marguaux
vin:terroir
vin:terroir
Attribute value restriction
geo:Location
geo:Bordelais
Attribute value restriction pattern

Fig. 1. Class by Attribute Type

The instantiation of this pattern can be used in mediation tasks such as data migration, query rewriting and ontology merging.

The example above actually represents a *complex correspondence*. In [6] the authors automatically detected several complex correspondences by specifying detection algorithms for each of four alignment patterns. This approach has originaly been based on availability of a reference alignment, which was one of the important information on the input for algorithms. However, this has been overcome in the new version of the approach [7]. Furthermore, there has been introduced a tool that can detect such complex correspondences based on declaratively expressed matching conditions using an XML-based specification. Newly, the detection is based on linguistic analysis of concept or property labels, which increased precision.

A somewhat different kind of 'non-standard' alignment structures deals with the so-called *heterogeneous correspondences* [4]. A heterogeneous correspondence connects a class in one ontology with a property in another ontology, e.g. the class Marriage with the property marriedTo. The underlying formalism allows to propagate subsumption information from the class hierarchy in one ontology to a property hierarchy in another ontology, and vice versa. While such processing is useful at the schema level, it is not directly applicable to the level

[5] `http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine.owl`
[6] `http://www.scharffe.fr/ontologies/OntologyDuVin.wsml`

of instance data: if we wanted to e.g. migrate data on married couples from the knowledge base structured according to the first ontology to the knowledge base structured according to the second ontology, we would also have to consider the properties connecting the 'objectified' class Marriage to the marriage participants, such as 'husbandIn' and wifeIn. Consequently, the situation would fall within the scope of our 'homogeneous' matching patterns, as the atomic property marriedTo would be matched to a complex property expression (though one laying beyond current OWL DL fragment): $marriedTo \equiv (husbandIn \bigcirc wifeIn^{-}) or (wifeIn \bigcirc husbandIn^{-})$.

3 Matching Patterns in the PatOMat Framework

A *matching pattern* includes two ontology patterns (the source one and the target one) and the set of correspondences between entities of the source and entities of the target.

The representation of *ontology patterns* is based on the OWL 2 language, namely, on its, OWL-DL profile. However, while an OWL ontology refers to particular entities, e.g. to class *Person*, in the patterns we generally use *placeholders*. Entities are specified (i.e. placeholders are instantiated) at the time of instantiation of a pattern.

Definition 1 (Ontology Pattern). Ontology pattern *is a triple* ⟨`E`, `Ax`, `NDP`*⟩, *such that* `E` *is a non-empty set of* entity declarations*,* `Ax` *a (possibly empty) set of* axioms*,*[7] *and* `NDP`* *a (possibly empty) set of alternative* naming detection patterns*.*

Entity declarations[8] concern classes, properties and individuals (all at the level of placeholders). Properties can be object, data or annotation ones. Annotation properties enable to capture information about parts of ontology pattern that are not part of the logical meaning of the ontology. *Axioms* are facts about entities included in the transformation; we assume them to be OWL 2, OWL-DL profile axioms in Manchester syntax. Annotations cannot be referred to in an alignment part; however, they can be used in the matching process as such. Finally, the NDPs capture the naming aspect of the ontology pattern as needed for its detection, see Section 3.1.

Example 1 In Figure 2 there is an example of source OP where a complex class description is defined using a value restriction requiring the value 'Accept' for the property 'hasStatus'. This complex class description is an unnamed entity that is subclass of the named class 'Paper'.

Definition 2 (Pattern Alignment). *Let* `OP1` *and* `OP2` *be ontology patterns. A* pattern alignment *from* `OP1` *(called source pattern) to* `OP2` *(called target pattern) is a tuple* $PA = $ ⟨`AL`, `SL`*⟩*, in which* `AL` *is a non-empty set of* correspondences*, and* `NL`* *is a (possibly empty) set of naming links.*

[7] To say, other than entity declarations.

[8] Corresponding to axioms with *rdf:type* property.

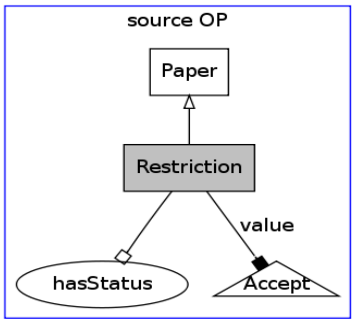

Fig. 2. Example of source Ontology Pattern

A correspondence is a triple $\langle E_1, E_2, R \rangle$ where E_1 is an entity placeholder from $OP1$, E_2 is an entity placeholder from $OP2$ and R is a relation from OWL 2. Entity placeholders E_1 and E_2 are either atomic entities or complex entity descriptions using e.g. atomic negation, conjunction, disjunction, value restriction, cardinality restriction etc. as it is possible to do in OWL-DL language.

Naming links capture the way to detect entities being involved in an alignment. They refer to constituents of NDPs from ontology patterns, see Section 3.1.

Definition 3 (Matching Pattern). *A* matching Pattern *`MP` is a triple* $\langle$*`OP1`, `PA`, `OP2`*$\rangle$ *such that `OP1`, `OP2` are ontology patterns and `PA` is a pattern alignment from `OP1` to `OP2`.*

Example 2 For example, there is a matching pattern between complex class description (Figure 2) and named class ‘AcceptedPaper’:

– $OP1$: E={Class: ?A. ObjectProperty: ?p. Individual: ?a},
  Ax={(?p value ?a) SubClassOf: ?A},
– $OP2$: E={Class: ?B. Literal: ?X. Literal: ?Y},
  Ax ={?B annotation:discr_property ?X. ?B annotation:value ?Y},
– AP : LI={?A EquivalentTo: (?p value ?a).}.

The target OP contains two annotation properties that capture information that is not directly present in the ontology pattern, i.e. the discriminating property and its value. There could be for instance the following assignments of entity placeholders: $OP1$:$?A = Paper$, $?p = hasStatus$, $?a = Accept$, $OP2$:$?B = AcceptedPaper$, $?X = hasStatus$, $?Y = Accept$, see Figure 3.

Note that the ontology patterns part of matching pattern is restrained to OWL DL, while the pattern alignment part also allows constructs from the OWL 2 language as such.

3.1 Naming Patterns within Matching Patterns

The attention paid to naming patterns follows from the finding that untrivial and useful regularities can be observed in ontology entity naming [14]. Oper-

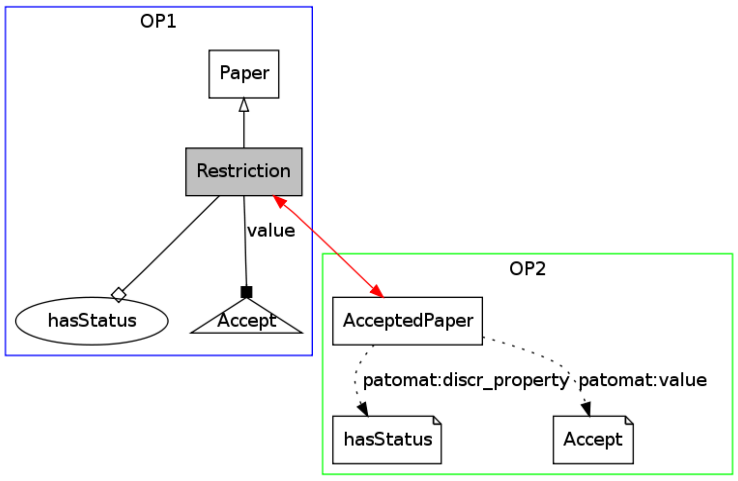

Fig. 3. Example of Matching Pattern

ational naming patterns consist of naming operations, which can be divided into *passive* ones, applied for checking purpose, and *active* ones, for naming a new entity.[9] In this paragraph we will introduce passive operations, NDPs and naming links. In [12] there are presented active operations and naming transformation patterns. While both passive and active operations can be plugged into naming transformation patterns, only passive operations can be used in naming detection patterns.

We start with the definition of naming detection pattern:

Definition 4. *A* naming detection pattern *is a set of passive naming operations,* `NDP` $= \{no_1, no_2, \ldots, no_n\}$. *All* no_i *have as operands entity placeholders from the ontology pattern to which* `NDP` *belongs, constants, or another passive naming operation.*

Example 3 As an example of NDP with two operations we can take the following:

$$\{\text{comparison(?B, head\_term(?p)), exists(verb\_form(?C))}\}$$

For instance, if ?B is 'Decision', ?p is 'hasDecision' (with 'Decision' as head term) and ?C is 'Acceptance' (with 'accept' as verb form) then the pattern succeeds.

Naming patterns can be generally defined on any lexical aspect of an ontology: URI of entities, its fragment, labels, comments etc. By default, we consider naming patterns applied over fragments of URIs.

Currently, the list of passive naming operations considered (and implemented) in our framework is as follows:

– delimiter_xxx(?A): checks whether entity name uses given delimiter, e.g. underscore, camel-case or hyphen where xxx stands for the delimiter,

[9] A passive naming operation often has its active variant.

- verb_form(?A): checks whether noun $?A$ has a correspoding verb form. Currently, it is implemented using WordNet[10] and its semantic information "derivationally related forms", where we check whether particular related form is a verb.[11] It stops when it finds the first verb form.
- head_noun(?A): returns a term on which all other tokens, from the whole phrase, are dependent. Similarly, there is a head_term(?p) for properties,
- complement_head_noun(?A): returns a complement of the head_noun to the union of all tokens appearing in an entity name,
- passive_verb(?A): returns a passive voice of verb from entity name of ?A,
- hyponym_of(?A): returns a hyponym of entity name of ?A,
- get_verb_form(?A): returns a verb form for given noun $?A$, see operation verb_form(?A).

Finally, we will define the notion of naming link:

Definition 5. *A* naming link *is a quadruple* $\langle\ X, Y, R, t\ \rangle$ *where* X *and* Y *are passive naming operations, entity placeholders or constants from the* $OP1$ *and* $OP2$*, respectively.* R *is a string-based measure with its respective threshold* t*.*

Example 4 As an example of a naming link with an equality measure we can show the following:

$$\{\langle\ ?A,\ head\_noun(?B),\ equal,\ 1.0\ \rangle\}$$

For instance, if ?A is 'Author' and ?B is 'PosterAuthor' (with 'Author' as head noun) then the equality measure succeeds.

Example 5 We provide one more example of matching pattern. The source and target ontology patterns are depicted at Figure 4, and their representation is as it follows:

- $OP1$: E={Class: ?A, ?B, ?C. ObjectProperty: ?p},
  Ax={?C SubClassOf: ?B. (?p some ?C). ?p domain: ?A. ?p range: ?B},
  NDP={head_term(?p)=?B},
- $OP2$: E={Class: ?D},
  NDP={verb_form(complement_head_noun(?D))},
- AP : LI={(?p some ?C) EquivalentTo: ?D},
  NL={?D=hyponym_of(?A), get_verb_form(?C)=complement_head_noun(?D)}.

This matching pattern is inspired by the so-called CAT pattern from [7]. Due to the naming aspect this pattern can be succesfully detected.

Discussion To conclude this section, we will discuss differences between matching patterns and alignment design patterns [8]. Matching patterns are based on:

[10] `http://wordnet.princeton.edu/`

[11] Using Stanford Log-linear Part-Of-Speech Tagger, available at: `http://nlp.stanford.edu/software/tagger.shtml`

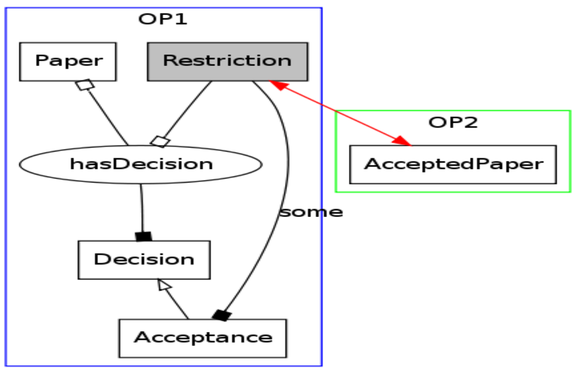

Fig. 4. Example of complex matching pattern

- *OWL language* as for describing parts of ontologies being aligned as well as for expressing alignment, i.e. semantic relations between them. Using the OWL language gives us ready-to-use direct semantics. In comparison with alignment design patterns, which come with its own language, using one language for expressing both, entities and correspondences, leads to a mixture of alignment and definitions. On the other hand we have specifically divided the parts for entities and part dealing with alignment. Another possible disadvantage of using the OWL language could be limited means of representation regarding data-level operations on individuals such as *transforming data values*. This could be additionally solved by introducing specific functions within dedicated namespace.
- *ontology patterns* with entity placeholders, which are constituents of matching pattern. This gives us a possibility to include any ontology pattern within matching pattern, in contrast to alignment design patterns where the parts on each side are not considered stand-alone.
- *naming patterns*: The naming aspect, besides structural aspect, is considered as important information for detection of ontology patterns as such or a matching pattern as a whole. Although we have done several experiments with detection of (logical) ontology patterns using the naming aspect in the past [10], there is still ample space for future work.

4 Application of Matching Patterns

This section presents two applications which are more-or-less based on matching patterns: ontology alignment evaluation and ontology transformation.

4.1 Matching Patterns Employed within Ontology Alignment Evaluation

Ontology alignment evaluation is an important activity of assessing the fitness of different methods and tools with respect to different domains and settings.

Since 2004 there is Ontology Alignment Evaluation Initiative (OAEI) which is annual international initiative for comparing the state of the art ontology matching methods and systems. Traditionally, evaluation is based on computing precision and recall (inspired by Information Retrieval) either based on reference alignment or statistical approximations using manual labeling. There are usually simple correspondences taken into account. However, we might even be interested in more complex alignment structures (patterns), which could reveal interesting details about the relationship of the two ontologies, e.g. the situation when an entity from one ontology can potentially be aligned on both a parent and a child from the other ontology.

Since 2007 we provide matching pattern based data mining evaluation method within one of six OAEI tracks, conference track [16], [11]. This method is based on large-scale mining over the alignment results with meta-data. The input to the mining process can be not only the name of the matching system, name and nature of the ontologies aligned, the type of correspondence (such as equivalence/subsumption) and the subjective posterior evaluation, but also the information whether the given correspondence is part (and what part) of a certain matching pattern. This enables us to discover hypotheses discovered via data mining (over ontology alignment data including information about matching patterns). Particularly, we provide frequent associations, which can become useful feedback to the development and tuning of matching systems, complementary to the feedback provided by Information Retrieval measures with respect to reference alignment.

Example 6 For instance we will show one matching pattern indicating incorrect correspondences due to the inconsistency. Matching pattern 'disjoint siblings', see Figure 5, contains of simultaneous correspondences between class A and two sibling classes C and D where A is from one ontology and C and D are from another ontology. Furthermore, two sibling classes C and D are disjoint.

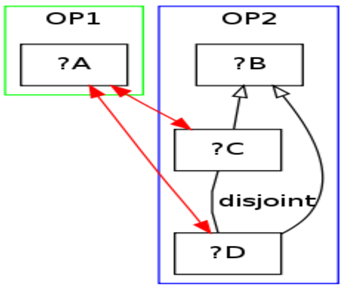

Fig. 5. Example of Matching Pattern employed within evaluation

In evaluation within OAEI 2007, this matching pattern was detected 112 times by system SEMA. Subsequently, during data mining corresponding hypothesis was found which could be rephrased as: *Correspondences that are pro-*

duced by system SEMA are by 1343% more often part of MP9 than correspondences produced by all systems (on average). In two consecutive OAEIs, 2008 and 2009, another system, the DSSim, found this pattern 124 and 295 times respectively. These systems could be improved by avoiding this pattern instances from their alignment results.

4.2 Matching Patterns Employed within Ontology Transformation

Matching patterns can be considered as a transition between two different styles which capture the same conceptualization. These differences could be an obstacle to using existing ontologies in more advanced semantic web scenarios, in particular:

– Two ontologies using different styles are difficult to *match* [12] or to *import* [15] to one another. Few matching systems support complex matching structures that bridge such heterogeneity, never mind considering schema merging and/or data migration.
– Opting for a style when designing an ontology may have dramatic impact on the usability and performance of *reasoners*, as some features cause performance problems for certain reasoners (for a specific reasoner, this has been investigated e.g. in [5]).

In order to enable this transition we have designed (within PatOMat project) and implemented ontology transformation framework [12] using *transformation patterns.* Transformation patterns can be based on matching patterns considering its equivalence correspondences. But there are important differences in other aspects. Regarding the purpose, while matching patterns are meant to represent recurring alignment structures at the ontological level, transformation patterns express how one structure can be transformed to another, conceptually similar structure.

Furthermore, analogously to a correspondences within an alignment pattern there is a pattern transformation part in a transformation pattern. In this part there are transformation links between entities. These links can be defined between homogeneous entities (equivalence correspondences), heterogeneous entities (*eqHet*)[12] and between real and annotation literals (*eqAnn*). The last two are not present in matching patterns.

Regarding transformation as such, transformation operations are defined over atomic entities: i.e. renaming, adding and removing; and over axioms: i.e. adding and removing. Complex descriptions are also considered within a transformation pattern; however, in comparison with alignment patterns they are only meaningful as part of some axiom. It does not make sense to add/remove an unnamed entity (e.g. a restriction class) unless it is involved in some axiom. This means that in the case of matching we consider as matchable components atomic entities and/or (even unnamed) complex entity descriptions. On the other hand, in

[12] Between different kinds of entities.

the case of transformation we only consider atomic entities and axioms as wholes as transformable components.

Example 7 For instance we will show one transformation pattern based on Example 2 where is one equivalence transformation link (but now between two atomic entities) and two transformation links eqAnn between real concepts and their annotation counterparts, see Figure 6. Due to annotations, it should be possible to make reverse transformation.

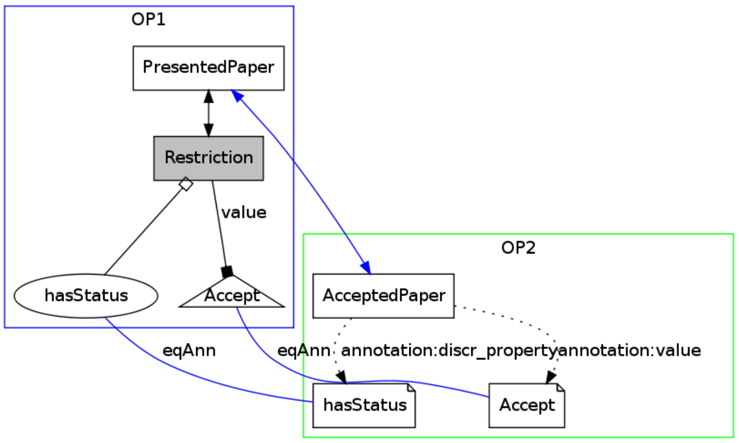

Fig. 6. Example of Transformation Pattern based on Matching Pattern

5 Conclusions and Future Work

This paper deals with matching patterns as an intersection of two hot topics of recent year in semantic web: Ontology Matching and Ontology Design Patterns. There are already Alignment Design Patterns within OntologyDesignPortal. In this paper we present matching patterns from PatOMat framework which are, newly, based on OWL language and ontology patterns. Matching pattern also includes naming aspect as a naming pattern. Matching patterns have potentially various applications. We present two applications: one from ontology alignment evaluation and one from ontology transformation.

We plan to set up experiment dealing with detection based on naming patterns for ontology patterns and matching patterns. Regarding transformation application, in the near future, we are about to consider, so far neglected, data migration aspect of transformation pattern. It means we have to figure out data migration scenario for each transformation pattern in connection with corresponding matching pattern on the one side and on the other side harmonize transformation pattern shape described in [12] accordingly.

This work has been partially supported by the CSF grant P202/10/1825 (PatOMat project).

References

1. W. Borst. *Construction of Engineering Ontologies.* PhD thesis, University of Tweenty, Enschede, The Netherlands, 1997.
2. J. Euzenat and P. Shvaiko. *Ontology matching.* Springer-Verlag, Heidelberg (DE), 2007.
3. A. Gangemi. Ontology design patterns for semantic web content. In *International Semantic Web Conference (ISWC2005)*, 2005.
4. C. Ghidini and L. Serafini. Reconciling concepts and relations in heterogeneous ontologies. In *Proceedings of the 3rd European Semantic Web Conference (ESWC-2006)*, pages 50–64, 2006.
5. H. Lin and E. Sirin. Pellint - a performance lint tool for pellet. In *Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions (OWLED-2008)*, 2008.
6. D. Ritze, C. Meilicke, O. Šváb-Zamazal, and H. Stuckenschmidt. A pattern-based ontology matching approach for detecting complex correspondences. In *Ontology Matching workshop (OM-2009)*, 2009.
7. D. Ritze, J. Völker, C. Meilicke, and O. Šváb-Zamazal. Linguistic analysis for complex ontology matching. In *Ontology Matching workshop (OM-2010).* Accepted.
8. F. Scharffe. *Correspondence Patterns Representation.* PhD thesis, University of Innsbruck, 2009.
9. F. Scharffe and D. Fensel. Correspondence patterns for ontology mediation. In *Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW2008)*, Acitrezza, Italy, September 2008. Springer.
10. O. Šváb-Zamazal, F. Scharffe, and V. Svátek. Preliminary results of logical ontology pattern detection using SPARQL and lexical heuristics. In *Proceedings of the Workshop on Ontology Patterns (WOP-2009)*, 2009.
11. O. Šváb-Zamazal and V. Svátek. Empirical knowledge discovery over ontology matching results. In *Proceedings of the 1st workshop on Inductive Reasoning and Machine Learning on the Semantic Web (IRMLes-2009).*
12. O. Šváb-Zamazal, V. Svátek, and L. Iannone. Pattern-based ontology transformation service exploiting OPPL and OWL-API. In *Knowledge Engineering and Knowledge Management by the Masses. EKAW-2010.*, 2010. Accepted.
13. V. Svátek. Design patterns for semantic web ontologies: Motivation and discussion. In *Conference on Business Information Systems (BIS-04)*, 2004.
14. V. Svátek, O. Šváb-Zamazal, and V. Presutti. Ontology naming pattern sauce for (human and computer) gourmets. In *Workshop on Ontology Patterns (WOP-2009)*, CEUR. CEUR, 2009.
15. V. Svátek, O. Šváb-Zamazal, and M. Vacura. Adapting ontologies to content patterns using transformation patterns. In *Workshop on Ontology Patterns (WOP-2010)*, CEUR. CEUR, 2010. Accepted.
16. O. Šváb, V. Svátek, and H. Stuckenschmidt. A study in empirical and 'casuistic' analysis of ontology mapping results. In *Proceedings of the 4th European conference on The Semantic Web (ESWC-07)*, pages 655–669, Berlin, Heidelberg, 2007. Springer-Verlag.