A pattern-based ontology matching approach for detecting complex correspondences

Dominique Ritze[1], Christian Meilicke[1], Ondřej Šváb-Zamazal[2], and Heiner Stuckenschmidt[1]

[1]University of Mannheim,
dritze@mail.uni-mannheim.de, {christian, heiner}@informatik.uni-mannheim.de
[2]University of Economics, Prague, ondrej.zamazal@vse.cz

Abstract. State of the art ontology matching techniques are limited to detect simple correspondences between atomic concepts and properties. Nevertheless, for many concepts and properties atomic counterparts will not exist, while it is possible to construct equivalent complex concept and property descriptions. We define a correspondence where at least one of the linked entities is non-atomic as complex correspondence. Further, we introduce several patterns describing complex correspondences. In particular, we focus on methods for automatically detecting complex correspondences. These methods are based on a combination of basic matching techniques. We conduct experiments with different datasets and discuss the results.

1 Introduction

Ontology matching is referred to as a means for resolving the problem of semantic heterogeneity [3]. This problem is caused by the possibility to describe the same domain by the use of ontologies that differ to a large degree. Ontology engineers might, for example, chose different vocabularies to describe the same entities. There might also be ontologies where some parts are modeled in a fine grained way, while in other ontologies there are only shallow concept hierarchies in the relevant branches. These kinds of heterogeneities can be resolved by state of the art ontology matching systems, which might e.g. detect that *hasAuthor* and *writtenBy* are equivalent properties and only different vocabulary is used. Moreover a matching system might identify, that *Author* is more general as both concepts *FirstAuthor* and *CoAuthor*.

However, ontological heterogeneities are not restricted to these kind of problems: different modeling styles might require more than equivalence or subsumption correspondences between atomic concepts and properties.[1] Semantic relations between complex descriptions become necessary. This is illustrated by the following example: While in one ontology we have an atomic concept *AcceptedPaper*, in another ontology we have the general concept *Paper* and the boolean property *accepted*. An *AcceptedPaper* in the first ontology corresponds in the second ontology to a *Paper* that has been *accepted*. Such a correspondence, where at least one of the linked entities is a complex concept

[1] Atomic concepts/properties are sometimes also referred to as named concepts/properties resp. concept/property names.

or property description, is referred to as complex correspondence in the following. As main contribution of this paper we suggest an automated pattern based approach to detect certain types of complex correspondences and study its performance by applying it on different datasets. Even though different researchers were concerned with similar topics (see [11]), to our knowledge none of the resulting works was concerned with automated detection in an experimental setting. Exceptions can be found in the machine learning community (see Section 2).

We first discuss related work centered around the notion of a complex correspondence in Section 2. We then present four patterns of complex correspondences in Section 3. In Section 4 we suggest the algorithms we designed to detect occurrences of these patterns. Each of these algorithms is described as a conjunction of conditions, which are easy to check by basic matching techniques. In Section 5 we apply the algorithms on two datasets from the OAEI and show that the proposed techniques can be used to detect a significant amount of complex correspondences. We end with a conclusion in Section 6.

2 Related Work

Complex matching is a well known topic in database schema matching. In [1] the authors describe complex matches as matching corresponding attributes on which some operation was applied, e.g. a name is equivalent with concatenation of a first-name and a last-name. There are several systems dealing with this kind of database schema matching. On the other hand complex matching is relatively new in the ontology matching field. Most of the state of the art matchers just find (simple) correspondences between two atomic terms. However, pragmatic concerns call for complex matching. We also experienced this during discussions at the OM-2008. It turns out that simple correspondences are too limited to capture all meaningful relations between concepts and properties of two related ontologies. This is an important aspect with respect to application scenarios making use of alignments e.g. instance migration scenarios. There are three diverse aspects of complex correspondences: designing (defining), finding and representing them.

In [8] complex correspondences are mainly considered from design and representation aspects. Complex correspondences are captured as correspondence patterns. They are solutions for recurring mismatches being raised during aligning two ontologies. These patterns are now being included within *Ontology Design Patterns* (ODP)[2]. This work considers complex matching as task that had to be conducted by a human user, which might e.g. be a domain expert. Experts can take advantage of diverse templates for capturing complex and correct matching. However, this collection of patterns can also be exploited by some automated matching approach, as suggested and shown in this paper.

In [11] authors tried to find complex correspondences using pattern-based detection of different semantic structures in ontologies. The most refined pattern is concerned

[2] In the taxonomy of patterns at the ODP portal (`http://ontologydesignpatterns.org/wiki/OPTypes`) category AlignmentODP corresponds best with the patterns in this paper, while category CorrespondeceODP is a more general category.

with 'N-ary' relation detection. After detecting an instance of the pattern (using query language and some string-based heuristics) additional conditions (mainly string-based comparisons) over related entities wrt. matching are checked. While there are some experiments with pattern detection in one ontology, experiments with matching tasks are missing.

Furthermore, in [12] the authors consider an approach for pattern-based ontology transformation useful for diverse purposes. One particular use case is ontology matching where this method enables finding further originally missed correspondences. Ontologies are transformed according to transformation patterns and then any matcher can be applied. Authors hypothesize that matchers can work with some structures better than with others. This approach uses *Expressive alignment language*[3] based on [2] which extends the original INRIA alignment format. This language enables to express complex structures on each side of an alignment (set operators, restriction for entities and relations). Furthermore it is possible to use variables and transformation functions for transforming attribute values. "Basically, complex correspondences are employed indirectly in the ontology matching process at a pre-processing step where ontology patterns are detected and transformed [13]." Unlike, in this paper complex correspondences are detected directly taking advantage of information from not only two ontologies being aligned but also from a reference alignment composed of simple correspondences.

Regarding ontology matching, there are a few matchers trying to find complex correspondences based on machine learning approaches (see [9] for a general description). A concrete matching system is presented in [6]. These approaches take correspondences with more than two atomic terms into account, but require the ontologies to include matchable instances. However, ontologies often contain disjoint sets of instances, such that for each instance of one ontology there exists no counterpart in the other ontology and vice versa. The approach proposed in this paper does not require the existence of matchable instances at all.

3 Complex Correspondence Patterns

In the following we propose four patterns for complex correspondences that, due to a preparatory study, we expect to occur frequently within ontology matching problems. We first report about our preparatory study, followed by a detailed presentation of each pattern. Each pattern is also explained by an example depicted in Figure 1. Without explicitly mentioning it, we will refer to Figure 1 throughout this section. Further we use $\mathcal{O}_1$ and $\mathcal{O}_2$ to refer to two aligned ontologies, and we use prefix notation $i\#C$ to refer to an entity C from ontology $\mathcal{O}_i$.

First of all we had to collect different types of complex correspondences. We considered the examples found in [9] and also profited from the discussion of the consensus track at OM 2008, which highlighted the need for complex correspondences.[4] After we had a few ideas, we started observing two sets of ontologies manually to detect concrete examples for complex correspondences. The specific ontologies which we examined are the SIGKDD, CMT, EKAW, IASTED, and CONFOF ontologies of the conference dataset

[3] `http://alignapi.gforge.inria.fr/language.html`
[4] `http://nb.vse.cz/~svabo/oaei2008/cbw08.pdf`

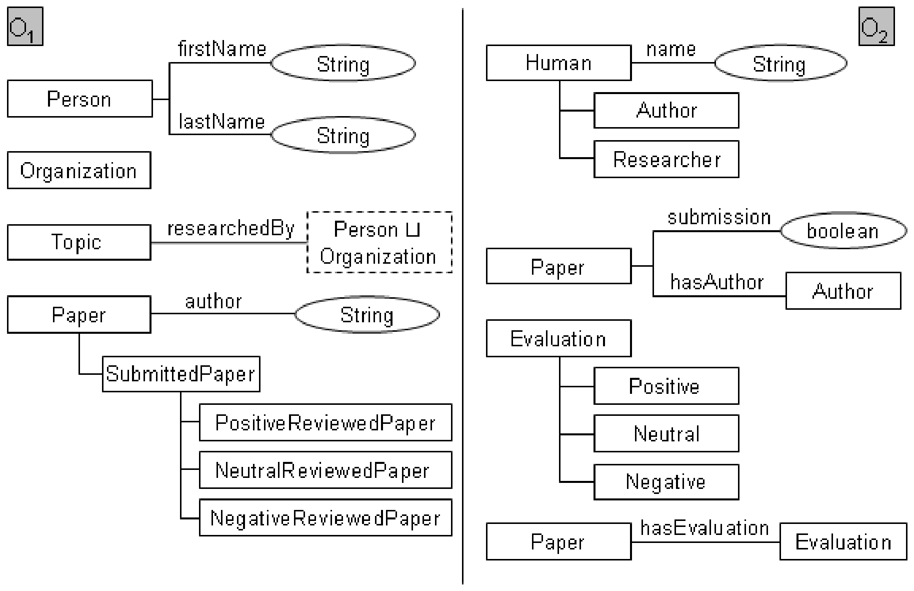

Fig. 1. Two example ontologies to explain the complex patterns

and ontologies 101, 301, 302, 303, and 304 of the benchmark track. The first dataset describes the domain of conferences. This seems to be suitable [10] because most persons dealing with ontologies are academics and know this topic already. Therefore it is easier to understand complex interdependencies in this domain instead compared to an unfamiliar domain like e.g. medical domains. The OAEI Benchmark ontologies attend the domain bibliography which is also well-known by academics. Another reason for choosing these ontologies are the existing and freely available reference alignments. For the conference dataset an alignment is available for every pair of two ontologies. Only for each combination with ontology 101 an alignment is available for the benchmark ontologies, resulting in four matching tasks. In Section 4 we will explain in how far and for which purpose a reference alignment, which consists of simple correspondences, is required.

The first three patterns are very similar, nevertheless, it will turn out that different algorithms are required to detect concrete complex correspondences. In accordance with [8] we will refer to them as *Class by Attribute Type pattern*, *Class by Inverse Attribute Type pattern*, and *Class by Attribute Value pattern*. In the following we give a formal description as well as an example for each pattern.

Class by Attribute Type pattern *(CAT)* This pattern occurs very often when we have disjoint sibling concept. In such a situation the same pattern can be used to define each of the sibling concepts.

Formal Pattern: $1\#A \equiv \exists 2\#R.2\#B$
Example: $1\#PositiveReviewedPaper \equiv \exists 2\#hasEvaluation.2\#Positive$

With respect to the ontologies depicted in Figure 1 we can construct correspondences of this type for the concepts Positive-, Neutral-, and NegativeReviewedPaper.

Class by Inverse Attribute Type pattern *(CAT^{-1})* The following pattern requires to make use of the inverse $2\#R^{-1}$ of property $2\#R$, since we want to define $1\#A$ as subconcept of $2\#R$'s range.

Formal Pattern: $1\#A \equiv 2\#B \sqcap \exists 2\#R^{-1}.\top$
Example: $2\#Researcher \equiv 1\#Person \sqcap \exists 1\#researchedBy^{-1}.\top$

Given an ontology which contains a property and its inverse property as named entities, it is possible to describe the same correspondences as *Class by Attribute Type pattern* and as *Class by Inverse Attribute Type pattern*. Nevertheless, an inverse property might often not be defined as atomic entity in the ontology or might be named in a way which makes a correct matching harder.

Class by Attribute Value pattern *(CAV)* While in the *Class by Attribute Type pattern* membership to a concept was a necessary condition, we now make use of nominals defined by concrete data values.

Formal Pattern: $1\#A \equiv \exists 2\#R.\{\ldots\}$ (where $\{\ldots\}$ is a set of concrete data values)
Example: $1\#submittedPaper \equiv \exists 2\#submission.\{true\}$

Another typical example is the distinction between *LateRegisteredParticipant* and *EarlyRegisteredParticipant*. In particular, the boolean variant of the pattern occurs to distinguish between complementary subclasses. However, in general there might be more than two relevant values. The following correspondence is a more complex example: $1\#StudentPassedExam \equiv \exists 2\#hasExamScore.\{A, B, C, D\}$.

Property Chain pattern *(PC)* [5] In the following we assume that in $\mathcal{O}_1$ property $1\#author$ relates a paper to the name of its author, while in $\mathcal{O}_2$ $2\#author$ relates a paper to its author and the datatype property $2\#name$ relates a person to its name. Under these circumstances a chain of properties in $\mathcal{O}_2$ is equivalent to an atomic property in $\mathcal{O}_1$.

Formal Pattern: $1\#R \equiv 2\#P \circ 2\#Q$
Example: $1\#author \equiv 2\#hasAuthor \circ 2\#name$

Conventional matching systems focus only on correspondences between atomic entities. Therefore, a matcher might detect a similarity between $1\#R$ and $2\#P$ and one between $1\#R$ and $2\#Q$, but will finally decide to output the one with higher similarity. This observation already indicates that state of the art matching techniques can

[5] Correspondence patterns library [8] explicitly contains *(CAT)* and *(CAV)*, other two patterns *(PC)* and *(CAT^{-1})* are not explicitly presented there.

be exploited to generate complex correspondences. In particular, we will argue in the next section, that it is possible to detect complex correspondences by combining simple techniques in an intelligent way.[6]

4 Algorithms

The techniques we are using for detecting complex correspondences are based on combinations of both linguistic and structural methods. In the following we shortly list and describe these approaches. The structural techniques require the existence of a reference alignment $\mathcal{R}$ that consists of simple equivalence correspondences between atomic concepts. In particular, it would also be possible to use a matcher generated (and partially incorrect) alignment, but in our first experiments we wanted to avoid any additional source of error.

Structural Criteria To decide whether two or more entities are related via complex correspondences, information about their position in the ontology hierarchy is required. Therefore, we have to check whether two concepts are in a subclass resp. superclass relation, or are even equivalent concepts. It might also be important to know if two concepts are non overlapping, disjoint concepts. Properties are connected to the concepts hierarchy via domain and range restrictions, which are thus also important context information. All of these notions are clearly defined within a single ontology, however, we extend these notions to a pair of aligned ontologies. $\mathit{1\#C}$ is also referred to as a subconcept of $\mathit{2\#D}$ if there exists a correspondence $\mathit{1\#C'} = \mathit{2\#D'} \in \mathcal{R}$ such that $\mathcal{O}_1 \models \mathit{1\#C} \subseteq \mathit{1\#C'}$ and $\mathcal{O}_2 \models \mathit{2\#D'} \subseteq \mathit{2\#D}$.

Syntactical Criteria The most efficient methods used in ontology matching are based on string comparisons e.g. comparing concept id (the fragment of the concepts URI) resp. label to compute a similarity between ontological elements. We also make use of this basic method by computing a similarity measure between normalized strings based on the Levenshtein measure [4]. For the sake of simplicity we refer to the maximum value obtained from id and label comparison as label similarity in the following. For some operations we need to determine the head noun of a given compound concept/property label. Thus, we can e.g. detect that `Reviewer` is the head noun of `ExternalReviewer`. Sometimes we are simply interested in the first part of a label, sometimes in the head noun and sometimes in the remaining parts.

Data type Compatibility Two data types are compatible if one data type can be translated into the other and vice versa. This becomes relevant whenever datatype properties are involved. We determined compatibility in a wide sense. E.g. data type `String` is compatible to every other data type while `Date` is not compatible to `Boolean`.

[6] Even experts tend to avoid the introduction of complex correspondences. The property chain $\mathit{1\#R} \equiv \mathit{2\#P} \circ \mathit{2\#Q}$, for example, is sometimes reflected by one (two) correspondence(s) $\mathit{1\#R} \equiv \mathit{2\#P}$ or (and) $\mathit{1\#R} \equiv \mathit{2\#Q}$. See for example the reference alignment for OAEI benchmark test case 301 where $\mathit{101\#date} \equiv \mathit{301\#hasYear}$ and $\mathit{101\#year} \equiv \mathit{301\#hasYear}$ which should be replaced by $\mathit{101\#date} \circ \mathit{101\#year} \equiv \mathit{301\#hasYear}$.

A more detailed description can be found in [7]. Overall we emphasize that our methodology does not exceed basic functionalities which we normally would expect to be part of any state of the art matching system.

Class by Attribute Type pattern A correspondence $\mathit{1}\#A \equiv \exists \mathit{2}\#R.\mathit{2}\#B$ of the *CAT* type is generated by our algorithm, if all following conditions hold.

1. The string that results from removing the head noun from the label of $\mathit{1}\#A$ is similar to the label of $\mathit{2}\#B$.
2. There exists a class $\mathit{2}\#C$ that is a superclass of $\mathit{2}\#B$, range of $\mathit{2}\#R$ and has also a label similar to $\mathit{2}\#R$.
3. The domain of $\mathit{2}\#R$ is a superclass of $\mathit{1}\#A$ due to $\mathcal{R}$.

Notice that these conditions are a complete description of our approach for detecting the *CAT* pattern. The following example will clarify why such a straightforward approach works.

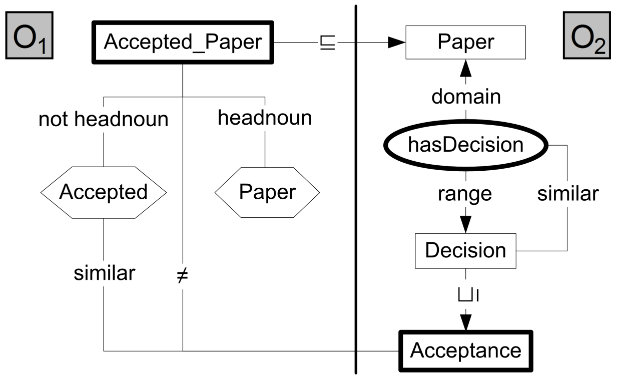

Fig. 2. Conditions relevant for detecting *CAT* correspondence $\mathit{1}\#Accepted\_Paper \equiv \exists \mathit{2}\#hasDecision.\mathit{2}\#Acceptance$.

With respect to the ontologies depicted in Figure 2 our approach will detect that $\mathit{1}\#Accepted\_Paper \equiv \exists \mathit{2}\#hasDecision.\mathit{2}\#Acceptance$. The label of *Accepted_Paper* can be split up into prefix `Accepted` and head noun `Paper`. On the one hand the string `Accepted` is similar to `Acceptance`, but on the other hand *Accepted_Paper* = *Acceptance* is not contained in $\mathcal{R}$. Object property *hasDecision* accomplishes all conditions required by our algorithm: *Acceptance* has a superclass *Decision* which is the range of *hasDecision* and the labels `Decision` and `hasDecision` are similar. Moreover the domain of *hasDecision* is a superclass of *Accepted_Paper* due $\mathcal{R}$, which contains correspondence $\mathit{1}\#Paper = \mathit{2}\#Paper$.

Class by Inverse Attribute Type pattern A correspondence $1\#A \equiv 2\#B \sqcap \exists 2\#R^{-1}.\top$ of the *CAT*$^{-1}$ type is generated if all following conditions hold.

1. The labels of $1\#A$ and $2\#R$ are similar.
2. There exists a concept $2\#B$ which both is a proper subset of the range of $2\#R$
3. and which is, due to the $\mathcal{R}$, a superclass of $1\#A$.

Notice that for the *CAT* pattern we did not demand similarity between $1\#A$ and $2\#R$. This is related to the fact that the label of a property often describes some aspects of its range and not its domain (e.g. *hasAuthor* relates a paper to its author). Thus, the label of a property is relevant for the inverse pattern *CAT*$^{-1}$. The other two conditions are related to structural aspects and filter out candidates that are caused by accidental string similarities.

Class by Attribute Value pattern Although above we described the pattern *CAV* in general, our algorithm will only detect the boolean variant of this pattern. A correspondence $1\#A \equiv \exists 2\#R.\{\mathit{true}\}$ is generated by our algorithm, if all following conditions hold.

1. The range of the datatype property $2\#R$ is `Boolean`.
2. In the following the label of $1\#A$ is split into its head noun $\mathtt{hn}(1\#A)$ and the remaining part of the label $\neg\mathtt{hn}(1\#A)$. Again, $\neg\mathtt{hn}(1\#A)$ is split into a first part $\neg\mathtt{hn}_1(1\#A)$ and a remaining part $\neg\mathtt{hn}_2(1\#A)$.
   (a) $\mathtt{hn}(1\#A)$ is similar to the label of $2\#R$'s domain.
   (b) $\neg\mathtt{hn}(1\#A)$ is similar to the label of $2\#R$.
   (c) $\neg\mathtt{hn}_1(1\#A)$ is similar to the label of $2\#R$.
3. The domain of $2\#R$ is a superclass of $1\#A$ due to $\mathcal{R}$.

Given a non-boolean datatype property range, more sophisticated techniques are required to decide which set of values is adequate for which concept. In our case this distinction is based on condition 2c. If the similarity value does not exceed a certain threshold, we generate $1\#A \equiv \exists 2\#R.\{\mathit{false}\}$ instead of $1\#A \equiv \exists 2\#R.\{\mathit{true}\}$. An example detected in our experimental study is $1\#Early\_Registered\_Participant \equiv \exists 2\#earlyRegistration.\{\mathit{true}\}$ exploiting $1\#Participant \equiv 2\#Participant$ in $\mathcal{R}$.

Property Chain pattern A correspondence $1\#R \equiv 2\#P \circ 2\#Q$ of type *PC* is generated, if all following conditions hold.

1. Due to $\mathcal{R}$, the domain of $1\#R$ is a subclass or superclass of the domain of $2\#P$.
2. The range of $2\#P$ is a subclass or superclass of the domain of $2\#Q$.
3. Datatype properties $1\#R$ and $2\#Q$ have a compatible data range.
4. The labels of $1\#R$ and $2\#P$ are similar.
5. The label of $2\#Q$ is `name` or is contained in the label of $1\#R$ resp. vice versa.

Due to the condition that range of $2\#P$ and domain of $2\#Q$ are in a superclass relation, the successive application of the properties can be ensured. Often $1\#R$ maps a class onto a name, therefore especially properties which are labeled with `name` are potential mapping candidates. An example for this pattern has already been given in the previous section. With respect to Figure 1 we have $1\#R$ = $1\#author$, $2\#P$ = $2\#hasAuthor$, $2\#Q$ = $2\#name$. The property $1\#author$ relates a paper to the name of its author, $2\#hasAuthor$ relates a paper to its author and $2\#name$ an author to its name. Thus, a chain of properties is required to express $1\#author$ in the terminology defined by $\mathcal{O}_2$.

A second set of conditions aims to cover a different naming strategy. The first three conditions are the same as above, but the last ones have to be replaced as follows.

4. The labels of $1\#R$ and $2\#Q$ are similar.
5. The labels of $2\#P$ and its range or the labels of the properties $2\#P$ and $2\#Q$ are similar.

An example, depicted in Figure 4, of a property chain that fulfills these conditions: $1\#hasYear$ = $2\#date \circ 2\#year$ where $2\#date$ is an object property with $2\#Date$ as abstract range.

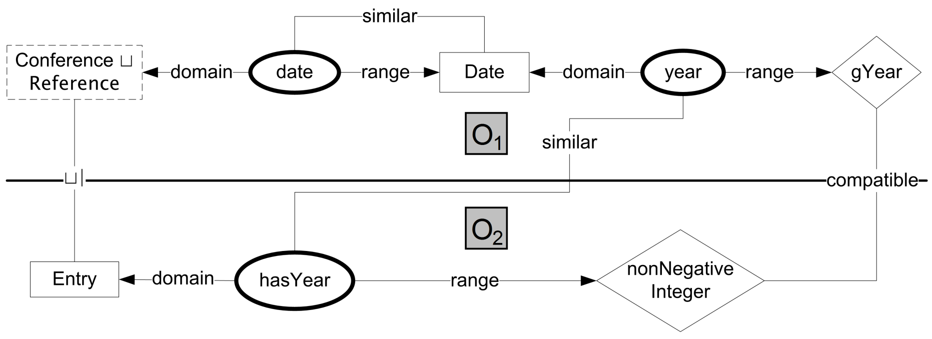

Fig. 3. Conditions relevant for detecting *PC* correspondence $1\#hasYear \equiv 2\#date \circ 2\#year$

For all patterns of the *class by* and *property chain* family we additionally check for each candidate correspondence whether there exists a constituent that already occurs in the reference alignment. In this case we trust the simple correspondence in the reference alignment and do not generate the complex correspondence.

5 Experiments

The algorithms described in the previous section have been implemented in a matching tool available at `http://dominique-ritze.de/complex-mappings/`. We applied our tool on three datasets referred to as CONFERENCE 1, CONFERENCE 2 and BENCHMARK. These datasets have been taken from corresponding tracks of the Ontology Alignment Evaluation Initiative (OAEI). As BENCHMARK we refer to the matching tasks #301 - #304 of the OAEI Benchmark track. We abstained from using the other test cases, because they are generated by systematic variations of the #101 ontology, which

do not exceed a certain degree of structural difference. The CONFERENCE 1 dataset consists of all pairs of ontologies for which a reference alignment is available. Additionally, we used the reference alignment between concepts created for the experiments conducted in [5] to extend our datasets. This dataset is referred to as CONFERENCE 2 and has not been regarded while looking for complex correspondences.
Notice that all conditions in our algorithms express hard boolean constraints. The only exception is the threshold that determines whether two strings are similar. Therefore, we conducted our experiments with different thresholds from 0.6 to 0.9.

	Correct Correspondences (true positives)												Incorrect Correspondences (false positives)											
Type	*CAT* & *CAT*$^{-1}$				PC				$\sum$				*CAT* & *CAT*$^{-1}$				PC				$\sum$			
Threshold	0.6	0.7	0.8	0.9	0.6	0.7	0.8	0.9	0.6	0.7	0.8	0.9	0.6	0.7	0.8	0.9	0.6	0.7	0.8	0.9	0.6	0.7	0.8	0.9
CONFERENCE 1	7	5	5	0	1	1	1	1	8	6	6	1	16	8	6	2	5	3	2	1	21	11	8	3
CONFERENCE 2	3	3	2	0	0	0	0	0	3	3	2	0	8	6	5	0	14	11	11	7	22	17	16	7
BENCHMARK	0	0	0	0	17	17	17	17	17	17	17	17	0	0	0	0	2	2	1	0	2	2	1	0
$\sum$	**10**	**8**	**7**	**0**	**18**	**18**	**18**	**18**	**28**	**26**	**25**	**18**	**24**	**14**	**11**	**2**	**21**	**16**	**14**	**8**	**45**	**30**	**25**	**10**

Table 1. Results with four different thresholds

Table 1 gives an overview on the results of our experiments. We carefully analyzed all generated correspondences and divided them in correct (true positives) and incorrect ones (false positives). One might first notice that we did not include a column for the *CAV* pattern. Unfortunately, only two correct and one incorrect correspondence of this type have been detected in the CONFERENCE 1 dataset. Remember that we only focused on boolean datatype properties. A more general strategy might result in higher recall. Nevertheless, to our knowledge all correspondences of the boolean *CAV* have been detected and even with low thresholds only one incorrect correspondence accrued.

Obviously there is a clear distinction between different datasets. While our matching system detected correct complex correspondences of *class by* types in the CONFERENCE datasets, none have been detected in the BENCHMARK dataset. Nearly the same holds vice versa. This is based on the fact that the ontologies of the BENCHMARK dataset are dedicated to the very narrow domain of bibliography and do not strongly vary with respect to their concept hierarchy, while differences can be found with regard to the use of properties. The CONFERENCE ontologies on the other hand have very different conceptual hierarchies.

Correspondences of the pattern *CAT* and *CAT*$^{-1}$ can be found in both CONFERENCE 1 & 2 datasets. As expected we find the typical relation between precision and recall on the one hand and the chosen threshold on the other hand: low thresholds cause low precision of approx 30% and allow to detect a relatively high number of correct correspondences. A nearly balanced ratio between true and false positives is reached with a threshold of 0.8.

For the *PC* pattern a threshold of 0.6 results in 18 correct and 21 incorrect correspondences. Surprisingly, the number of correct correspondences does not decrease with increasing threshold, although the number of incorrect correspondences decreases significantly. This is based on the fact that the relevant entities occurring in the *PC* pattern are very often not only similar but identical after normalization (e.g. concept *Date* and property *date*). This observation indicates that there is still room for improvement by choosing different thresholds for different patterns.

Another surprising result is the high number of false property chains in the CONFERENCE 1 and in particular in the CONFERENCE 2 dataset compared to the BENCHMARK dataset. Due to the existence of a reference alignment with high coverage of properties for the BENCHMARK dataset many incorrect property chains have not been generated. Their constituents already occurred in simple correspondence of the reference alignment. The same does not hold for the CONFERENCE datasets. There are many properties that have no counterpart in one of the other ontologies.

Our experimental study points to the problem of evaluating the quality of a complex alignment. Due to the fact that complex correspondences are missing in the reference alignments, our results cannot be compared against a gold standard, resulting in missing recall values. Even though it might be possible to construct a complete reference alignment for a finite number of patterns, it will be extremely laborious to construct a complete reference alignment, which contains all non-trivial complex correspondences. Nevertheless, a comparison against the size of the simple reference alignments might deliver some useful insights. The number of property correspondences in the union of all BENCHMARK reference alignments is 139 (only 63 concept correspondences), while we could find 17 additional property chains with our approach. For the CONFERENCE datasets we counted 275 concept correspondences (only the CONFERENCE 1 dataset comprised additionally 12 property correspondences). Here we detected 12 complex correspondences of different class by types. These results indicate that the proposed complex ontology matching strategy increased recall by approx. 4% with respect to concept correspondences and by approx. 10% with repect to property correspondences.

Interpreting these results, we have to keep in mind that the generation of complex correspondences is much harder compared to the generation of simple correspondences. While a balanced rate of correct and incorrect correspondences will not be acceptable for simple matching tasks, a similar result is positive with respect to the complex matching task which we tackle with our approach.

6 Conclusion

We proposed a pattern based approach to detect different types of complex correspondences. Our approach does not rely on machine learning techniques, which require the availability of instance correspondences. On the contrary, it is based on state of the art matching techniques and additionally exploits an input alignment which consists of simple correspondences. In an experimental study we have shown that our approach, which is simply based on checking conditions specific to a particular pattern, is sufficient to detect a significant amount of complex correspondences, while the number of false positives is relatively low, if considering that complex correspondences are quite hard to detect.

Although first results are promising, we know that the task of verifying the correctness of complex correspondences requires human interaction. A pattern based approach, as proposed in this paper, will in most cases fail to generate highly precise alignments. This is based on the fact that the generation of complex correspondences is significantly harder compared to the task of generating simple correspondences. Suppose, given concept *AcceptedPaper* of $\mathcal{O}_1$, a user is searching in $\mathcal{O}_2$ for an equivalent concept. First of all, there are as much simple hypotheses available as there are atomic concepts in $\mathcal{O}_2$. The situation changes dramatically when there exists no atomic counterpart and a complex correspondence is required. The search space explodes and it becomes impossible for a human expert to evaluate each possible combination. We know that the proposed patterns covers only a small part of an infinite search space. Nevertheless, this small part might still be large enough to find a significant fraction of those correspondences that will not be detected at all without a supporting system.

Acknowledgment The work has been partially supported by the German Science Foundation (DFG) under contract STU 266/3-1 and STU 266/5-1 and by the IGA VSE grant no. 20/08 ”Evaluation and matching ontologies via patterns”.

References

1. A. Doan and A. Y. Halevy. Semantic-integration research in the database community. *AI Magazine*, pages 83–94, 2005.
2. J. Euzenat, F. Scharffe, and A. Zimmermann. Expressive alignment language and implementation. deliverable 2.2.10, Knowledge web, 2007.
3. J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, 2007.
4. V. I. Levenshtein. Binary codes capable of correcting deletions and insertions and reversals. *Doklady Akademii Nauk SSSR*, pages 845–848, 1965. In Russian. English Translation in Soviet Physics Doklady, 10(8) p. 707710, 1966.
5. C. Meilicke, A. Tamilin, and H. Stuckenschmidt. Repairing Ontology Mappings. In *Proceedings of the 22nd Conference on Artificial Intelligence*, Vancouver, Canada, 2007.
6. H. Qin, D. Dou, and P. LePendu. Discovering Executable Semantic Mappings Between Ontologies. *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, pages 832–849, 2007.
7. D. Ritze. Generating Complex Ontology Alignments, University Mannheim (Bachelor thesis), 2009.
8. F. Scharffe. *Correspondence Patterns Representation*. PhD thesis, University of Innsbruck, 2009.
9. H. Stuckenschmidt, L. Predoiu, and C. Meilicke. Learning Complex Ontology Alignments A Challenge for ILP Research. In *Proceedings of the 18th International Conference on Inductive Logic Programming*, 2008.
10. O. Šváb, V. Svátek, P. Berka, D. Rak, and P. Tomášek. OntoFarm: Towards an Experimental Collection of Parallel Ontologies. In *Poster Proceedings of the International Semantic Web Conference*, 2005.
11. O. Šváb-Zamazal and V. Svátek. Towards Ontology Matching via Pattern-Based Detection of Semantic Structures in OWL Ontologies. In *Proceedings of the Znalosti Czecho-Slovak Knowledge Technology conference*, 2009.
12. O. Šváb-Zamazal, V. Svátek, J. David, and F. Scharffe. Towards Metamorphic Semantic Models. In *Poster session at European Semantic Web Conference*, 2009.
13. O. Šváb-Zamazal, V. Svátek, and F. Scharffe. Pattern-based Ontology Transformation Service. In *Proceedings of the 1st International Conference on Knowledge Engineering and Ontology Development*, 2009.