Complex Correspondences for Query Patterns Rewriting

Pascal Gillet, Cassia Trojahn, Ollivier Haemmerlé, and Camille Pradel

IRIT & Université de Toulouse 2, Toulouse, France
pascalgillet@ymail.com,{cassia.trojahn,ollivier.haemmerle,camille.pradel}@irit.fr

Abstract. This paper discusses the use of complex alignments in the task of automatic query patterns rewriting. We apply this approach in SWIP, a system that allows for querying RDF data from natural language-based queries, hiding the complexity of SPARQL. SWIP is based on the use of query patterns that characterise families of queries and that are instantiated with respect to the initial user query expressed in natural language. However, these patterns are specific to the vocabulary used to describe the data source to be queried. For rewriting query patterns, we experiment ontology matching approaches in order to find complex correspondences between two ontologies describing data sources. From the alignments and initial query patterns, we rewrite these patterns in order to be able to query the data described using the target ontology. These experiments have been carried out on an ontology on the music domain and DBpedia ontology.

1 Introduction

Despite the fact that SPARQL is the standard *de facto* language for querying RDF data, its complexity may restrict its use at a large scale, specially for non-expert RDF users. Translating natural language queries into SPARQL ones is the object of researches in both Natural Language Processing and Semantic Web fields. Within the SWIP system [11], users express queries in natural language sentences and pre-written query patterns are instantiated with respect to a syntactic analysis of the initial query. Several possible interpretations of the queries are shown to the user which selects the query he/she is interested in. Unlike other proposals, such as the one in [5, 4], where user queries are limited to keywords, or in [16], where users can express their queries using a visual query language, the originality of SWIP is related to the use of query patterns.

The main principle behind query patterns states that, in real applications, the submitted queries are variations of few typical query families (i.e., the family of queries asking for the *actors playing in movies* or the queries asking for the *members of a musical band*, or the *albums of a musical artist*). On the one hand, the use of patterns avoids exploring the whole ontology to link the semantic entities identified from the keywords since the potential relations are already expressed in the patterns. The process thus benefits from the pre-established families of frequently expressed queries for which real information needs exist.

A query pattern can also be seen as the projection of a subgraph of the underlying knowledge base, used as a mediator for the translation between the query expressed in natural language and the corresponding SPARQL query. On the other hand, one of the main limitations of the query pattern-based approach is that reusing patterns across different data sources can be done in a very limited extent. For each data source to be queried, the corresponding query patterns have to be (manually) built. Rewriting query patterns based on a vocabulary to query patterns based on another vocabulary is a task that can be carried out with the help of ontology alignments.

This paper discusses the use of complex correspondences for automatic query patterns rewriting. While the usefulness of simple correspondences has long been recognised, query rewriting requires more expressive links between ontology entities expressing the true relationships between them. At a lower level of abstraction, ontology alignments have been used to support the task of SPARQL query rewriting [2, 9, 8]. However, query patterns and SPARQL refer to different levels of expressivity in their representations, where query patterns are articulated as a set of subpatterns and rely solely on the $\mathcal{T}$Box of ontologies. Hence, we experiment ontology matching approaches in order to find complex correspondences between two ontologies describing two different data collections. From the complex alignments and source query patterns, we rewrite these patterns in order to be able to query the data collection described using the target ontology. Experiments have been carried out on an ontology on the music domain and DBpedia ontology. As a main outcoming, we have a set of manually validated complex correspondences, from which query patterns can be reused across the data sets described using those ontologies.

The rest of the paper is organised as follows. First, we introduce complex correspondences and query patterns (§2). Then, we present the approach for query pattern rewriting that is based on the use of complex correspondences (§3). Next, the experiments are discussed (§4). Finally, we discuss related work (§5) and conclude the paper (§6).

2 Foundations

2.1 Complex correspondences

Matching two ontologies is the process of generating an alignment between them [6]. An alignment A is directional and refers to a source ontology O and a target ontology O', denoted $A_{O \to O'}$:

Definition 1 (Alignment). *An alignment $A_{O \to O'}$ between two ontologies O and O' is a set of correspondences $A_{O \to O'} = \{c_1, c_2, ..., c_n\}$, where each c_i is a triple $\langle e_i, e'_i, r \rangle$, where:*

- *whether the correspondence is **simple**, then it relates one and only one entity (i.e., a class or a property) e_i of O to one and only one entity e'_i of O' (1:1);*

– *or the correspondence is* ***complex****, and it involves one or more entities in a logical formulation (1:n,m:1,m:n), where e_i refers to a subset of elements $\in O$, and e'_i refers to a subset of elements $\in O'$, and these elements are related using the constructors of a formal language (First-Order Logic or Description Logics);*
– *r is a relation, e.g., equivalence ($\equiv$), more general($\sqsupseteq$), more specific ($\sqsubseteq$), holding between e_i and e'_i;*
– *additionally, a value n (typically in [0,1]) is assigned to c_i indicating the degree of confidence that the relation r holds between the e and e'.*

The correspondence $\langle e_i, e'_i, r\rangle$ is unique in $A_{O\to O'}$. On the other hand, e_i or e'_i may be present in more than one correspondence c_i. The alignment $A_{O\to O'}$ is said *complex* if it contains at least one complex correspondence. In the rest of this paper, a correspondence c_i, which is a triple $\langle e_i, e'_i, r\rangle$ (suffix notation), will be noted $e_i\ r\ e'_i$. For example, $Film \sqsubseteq Work$ is a simple correspondence and asserts that $Film$ in O is more specific than $Work$ in O'. On the other hand, we can have the following two complex correspondences :

$$\forall x, Short\_Film(x) \equiv Film(x) \wedge duration(x, y) \wedge y \leq 59 \quad (1)$$
$$\forall x, Biopic(x) \equiv Film(x) \wedge Celebrity(y) \wedge topic(x, y) \quad (2)$$

(1) asserts that a $Short\_Film$ in O is equivalent to a $Film$ in O' whose duration is less than 59 minutes, and (2) asserts that a *Biopic* (or biographical film) in O is equivalent to a $Film$ in O' whose the topic is about a famous person. We can either use First-Order Logic (FOL) or the constructors of Description Logics (DL) for expressing the complex correspondences. In FOL, a *class* corresponds to a unary predicate with one variable, a *property* to a binary predicate with two variables, and an *instance* to a constant. In DL, a class corresponds to a (atomic) concept, a property to a role, and an instance to an individual. We can equally transpose logical statements from DL to FOL, and conversely, as long as the DL fragment is always respected. In particular, we take advantage of the expressivity allowed by $\mathcal{SHOIN}$, describing the following DL operators: $\neg C$ (negation of concepts), $C \sqcap C$ (intersection or conjunction of concepts), $C \sqcup C$ (union or disjunction of concepts), $\exists R.C$ (existential restriction), $\forall R.C$ (universal restriction), $\leq nR$ (at most restriction), $\geq nR$ (at least restriction). For instance, the formula (1) becomes (3) and the formula (2) becomes (4) :

$$Short\_Film \equiv Film \sqcap \exists duration. \leq 59 \quad (3)$$
$$Biopic \equiv Film \sqcap \exists topic.Celebrity \quad (4)$$

In a (complex) correspondence formula expressed in FOL, a variable may occur in several entities in left and right operands. Intuitively, two classes referring to the same single variable are bound and are first connected by a simple correspondence (with a subsumption relation, otherwise there is no need for an additional complex correspondence to characterise the entities involved). For instance, the formulas (1) and (2) are *consistent* only if $Court\_metrage \sqsubseteq Film$ and $Biopic \sqsubseteq Film$, respectively. The use of DL in (4) requires to explicitly assert the simple correspondence $Biopic \sqsubseteq Film$ first, as we do not know if $Biopic$ is a specialisation or rather a generalisation of $Film$ or $Celebrity$ otherwise.

2.2 Query patterns

A pattern p^O targetting an ontology O is composed of an RDF graph which is the prototype of a relevant family of queries. A pattern can be composed of several subpatterns sp_i, such as $p^O = \{sp_1, sp_2, ..., sp_n\}$. Subpatterns are assigned minimal and maximal cardinalities, making these subgraphs optional or repeatable when generating the final SPARQL query. Formally, a query pattern can be defined as follows [12]:

Definition 2 (Query pattern). *Let G be a graph and v a vertex of the graph, we denote by $G\backslash v$ the graph deprived of the vertex v and all the arcs incident to the vertex. A query pattern p is a triple (G, Q, SP) such as :*

- *G is a RDF connected graph that describes the general structure of the pattern and represents a family of requests;*
- *Q is a subset of elements in G, called qualifier elements; these are typical of the pattern and will be taken into account during the association of the user request to the pattern in question. A qualifier element is either a vertex (class or data type), or an arc (object or data property) in G;*
- *SP is the set of subpatterns sp in p such that $\forall sp = (SG, v, card_{min}, card_{max}) \in SP$, we have:*
  - *SG is a subgraph of G and v is a vertex of SG (and thus of G), such as $G\backslash v$ is not connected (v is a joint vertex in G, also called junction vertex) and admits $SG\backslash v$ as a connected component (i.e. all the vertices in this connected component belong to the subpattern subpattern's graph);*
  - *At least one vertex or an arc of SG is a qualifier element;*
  - *$card_{min}, card_{max} \in \mathbb{N}$ et $0 \leq card_{min} \leq card_{max}$; are respectively the minimum and maximum cardinality of sp that define the optional and repeatable characteristics of sp.*

Figure 2.2 shows an example of a query pattern which deals with the events, and the performers involved, where musical works have been performed (or conversely) with the corresponding artist(s) and release date. The pattern is composed of three subpatterns named *live*, *artist* and *date*. All of them are optional, and only the subpatterns *live* and *artist* are repeatable: it is considered that a musical work can have only one release date, but a musical work may be the work of several artists and can be performed many times.

3 Patterns rewriting approach

The rewriting approach takes as input a complex alignment $A_{O \to O'}$ and a set $P=\{p_1^O,...,p_n^O\}$ of query patterns p_i^O, and outputs a set $P'=\{p_1^{O'},...,p_n^{O'}\}$ of query patterns $p_j^{O'}$. The intuition is that every subgraph from the input patterns has potentially a (complex) correspondence associating its entities to entities in the target ontology. We consider that the subpattern is the relevant unit of semantic information constituting the patterns. Each subpattern is ideally replaced with

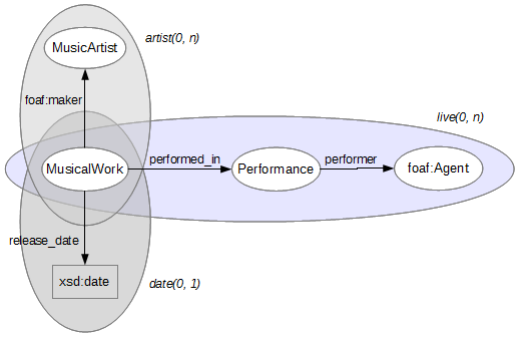

Fig. 1. Query pattern asking for events and their performers, where musical works have been performed (or conversely), with the corresponding artist(s) and release date.

an *equivalent* subgraph corresponding to a logical statement relating concepts and properties of the target ontology. This statement is the target part of the correspondence, if any in the alignment, in which the source part *matches* the initial subpattern. But the subpatterns and the correspondences in the alignment may not have the same granularity (correspondences can be either simple or can relate smaller subgraphs). Thus, we define an algorithm that is similar to a Depth-First Search algorithm (DFS) for traversing and searching graph data structures in the input query patterns. It starts at the largest subgraph, i.e. the subpattern, and recursively explores its subgraphs (i.e. subpattern > RDF triples > classes and properties), until a correspondence is found for the considered subgraph (in which case, the target graph is written to the subpattern being outputted) or a class or property is reached. If at the end of this process, there are entities that have not been translated, the whole subpattern will be discarded[1]. The operation is repeated for each subpattern in the input patterns.

The approach is inherently limited by the use of ontology alignment, which is itself an incomplete process. The subpattern is the indivisible expression of a need for information: it can be rewritten by chunks but if it is not fully rewritten at the end of the process, it is discarded. Thus, the conservation of the semantics of original patterns directly depends on the completeness of the input alignment (coverage of the source ontology, quality of correspondences, etc.). We consider that some loss of (semantic) information is acceptable, and that it can be filled with other techniques (for instance, by interacting with the user). However, it is out of the scope of this paper. Figure 3 illustrates the rewriting of the pattern depicted in Figure 2.2 (with the input alignment given later in §4.3, Table 1). The subpattern named *live* is rewritten to *live'*, following the complex correspondence #6 in the alignment. The subpattern named *artist* is rewritten to *artist'*, following the complex correspondence #2 in the alignment (in this case, only the first term of the disjunction $artist \sqcup author \sqcup creator \sqcup musicComposer$ appears in the resulting subpattern, for the sake of simplicity and readability).

[1] In this case, the pattern is still connected, i.e. there is a chain connecting each pair of vertices.

Finally, the subpattern named *date* could not be directly rewritten since there is no correspondence for this subgraph. Instead, the property *release_date* is rewritten to *releaseDate*, following a simple correspondence, and the data type *xsd:date* remains *xsd:date*.

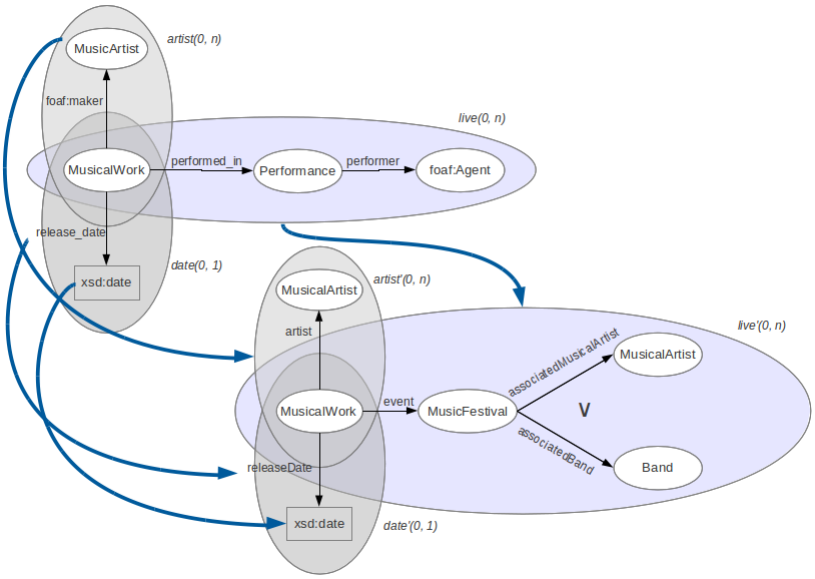

Fig. 2. Example of query pattern rewriting.

4 Experiments and discussion

4.1 Data sets

SWIP provides two sets of query patterns, one for the MusicBrainz collection described in terms of the Music ontology[2] (containing 249 $\mathcal{T}$Box entities), and another for querying the $\mathcal{A}$Box of the Cinema IRIT[3] ontology (containing 300 $\mathcal{T}$Box entities). We have carried out our experiments using the Music ontology and DBpedia 3.8[4] ontology (containing 2213 entities), in order to rewrite query patterns targeting MusicBrainz collection into patterns targeting DBpedia. The music set of patterns is composed of 5 query patterns and 19 subpatterns.

4.2 Preliminary experiments

In a first series of experiments, we used a set of simple correspondences for rewriting patterns. These correspondences come from a merge of alignments generated

[2] `http://musicontology.com/`
[3] `http://ontologies.alwaysdata.net/cinema`
[4] `http://wiki.dbpedia.org/Ontology?v=181z`

by OAEI 2012 matching systems (the reader can refer to [7] for details). Overall, 67% of the entities in the Music ontology were covered in the alignment. 25 out of 60 entities in the query patterns for this ontology could be replaced by a target entity (coverage of 41%). In terms of subpatterns, only 2 out of the 19 subpatterns could be fully rewritten using the alignment. Although we have found a handmade (reference) alignment between Music and DBpedia ontologies[5], we could not use it because it is mostly composed by correspondences linking classes only, using subsumption relations, and few equivalences could be inferred from them. Despite the fact that the quality of the alignment from the matchers was not measured, these first experiments highlighted the limitations in replacing individually the entities in the patterns. Nevertheless, we needed to accurately assess this deficiency and to know to what extent we could rewrite query patterns. It turned out that simple correspondences are not sufficient to capture all the meaningful relations between entities of two related ontologies.

4.3 Complex correspondences

Very few systems are able to find complex correspondences. First, we tried the tool described in [14], which finds complex correspondences using a set of predefined patterns. Besides the two ontologies to align, this tool takes an alignment as input. We used the tool on the pair Music-DBpedia, with (i) the handmade alignment between Music and DBpedia ontologies, and (ii) the merged alignment from the matchers used in the preliminary experiments (§4.2). In both cases, few correct correspondences could be identified. We tried then the successor of this tool described in [15], which benefits of natural language processing techniques instead of requiring an input alignment, and we obtained similar results.

Hence, we manually created a set of 28 complex correspondences (along 11 simple ones) for the pair Music-DBpedia, guided by the query patterns for Music[6]. A subset of them is presented in Table 1. The idea behind using complex correspondences is that every subgraph pattern has potentially a (complex) correspondence in the target ontology. Given that the subpattern is the relevant unit of semantic information constituting the patterns, we isolated them, and for each of them, we tried to find an equivalent logical statement relating concepts and properties of the target ontology. For constructing the complex correspondence set, we used as basis a set of simple correspondences (the left operand refers to an entity of the Music ontology, and the right operand refers to an entity of the DBpedia ontology): MusicalManifestation $\sqsubseteq$ MusicalWork, MusicalWork $\equiv$ MusicalWork*, MusicArtist $\equiv$ MusicalArtist*, Performance $\sqsubseteq$ Event, Performer $\equiv$ MusicalArtist, foaf:Group $\sqsupset$ Band, MusicGroup $\equiv$ Band*, SoloMusicArtist $\sqsubseteq$ MusicalArtist, Track $\sqsubseteq$ MusicalWork, Track $\equiv$ Song, and Record $\sqsubseteq$ MusicalWork[7].

For each correspondence, we identified the complex correspondence patterns that characterise it, from the patterns proposed in the literature: *Class*

[5] `http://knoesis.org/projects/BLOOMS/#Resources_for_Download`

[6] The resulting alignment do not cover all possible correspondences between Music-DBpedia, but a subset of them where entities appear in the query patterns.

[7] Correspondences with an asterisk were discovered using OAEI matchers.

#1	**CAV** MusicalManifestation ⊓ ∃release_type.album ≡ Album
#2	**CAT ≡ OR** MusicManifestation ⊓ ∃foaf:maker.MusicArtist ≡ MusicalWork ⊓ (∃artist.owl:Thing ⊔∃author.Person ⊔ ∃creator.Person ⊔ ∃musicComposer.MusicalArtist)
#3	**CAV ⊑ CAT** MusicalManifestation ⊓ ∃release_type.live ⊑ MusicalWork ⊓ ∃recordedIn.PopulatedPlace
#4	**CAV + CAT ⊐ CAT** MusicalManifestation ⊓ ∃release_type.soundtrack ⊓ ∃composer.foaf:Agent ⊐ Film ⊓ ∃musicComposer.MusicalArtist
#5	**PC ≡ OR** MusicGroup ⊓ ∃bio:event(bio:Birth ⊓ ∃bio:date.xsd:dateTime) ≡ Band ⊓ (∃formationDate.xsd:date ⊔ ∃formationYear.xsd:gYear ⊔ ∃activeYearsStartYear.xsd:gYear)
#6	**PC ≡ CAT(OR)** MusicalWork ⊓ ∃performed_in(Performance ⊓ ∃performer.foaf:Agent) ≡ MusicalWork ⊓ ∃event(Event ⊓ (∃associatedMusicalArtist.MusicalArtist ⊔ ∃associatedBand.Band))
#7	**CAT ≡ CAT** Track ⊓ ∃duration.xsd:decimal ≡ MusicalWork ⊓ ∃runtime.Time
#8	**CAT ≡ OR + CAT-1** foaf:Agent ⊓ ∃member_of.foaf:Group ≡ Person ⊓ (∃bandMember.Band ⊔ ∃formerBandMember.Band)
#9	**AND + PC ≡ AND(CAT-1)** Membership ⊓ (∃event:agent.foaf:Agent ⊓ ∃group.foaf:Group ⊓ ∃event:time.(event: TemporalEntity ⊓ (∃tl:start.xsd:date ⊓ ∀ tl:end.¬xsd:date)) ≡ Person ⊓ (∃bandMember.Band ⊓ ∀ formerBandMember.¬Band)
#10	**AND(PC) ≡ AND** Membership ⊓ ∃event:agent.foaf:Agent ⊓ ∃event:time.(event:TemporalEntity ⊓ ∃tl:start.xsd:date) ≡ Event ⊓ (∃pastMember.Person ⊓ ∃startDate.xsd:date)
#11	**PC ≡ CAT** SoloMusicArtist ⊓ ∃bio:event(bio:Birth ⊓ ∃bio:date.xsd:dateTime) ≡ MusicalArtist ⊓ ∃birthDate.xsd:date
#12	**CAT ≡ OR(CAT)** foaf:Agent ⊓ ∃collaborated_with.foaf:Agent ≡ (Artist ⊓ ∃associatedAct.Artist) ⊔ (Person ⊓ ∃partner.Person)

Table 1. 12 out of 28 handmade complex correspondences between Music and DBpedia ontologies.

by Attribute Type [14] (CAT), *Class by Inverse Attribute Type* [14] (CAT-1), *Class by Attribute Value* [14] (CAV), *Attribute Value Restriction* [18] (AVR), equivalent to CAV, *Property Chain* [14] (PC), *Aggregation* [18] (AGR), equivalent to PC, *Inverse Property* [15] (IP), *Union* [18] (OR), and *Intersection* [18] (AND). Although no new pattern was discovered, stating that, for our case, the patterns proposed in the literature cover all types of generated correspondences, several of our correspondences are in fact compositions of them (Table 1). For instance, the left operand in the correspondence #4 is an assembly of the patterns CAV and CAT. In the scope of this paper, however, we do not define any algebra which would describe how patterns can be composed or associated to represent the structure of complex correspondences (definition of basic properties and laws such as associativity, commutativity and distributivity). We have also manually generated 52 multilingual complex correspondences (along 13 simple correspondences) for the Cinema and DBpedia ontologies. For instance, the correspondence expressing the relation between the artists that are awarded the Cesar Award in Cinema (source ontology) and DBpedia (target ontology) : $Artiste \sqcap \exists estRecompenseA.CesarDuCinema \equiv Artist \sqcap \exists cesarAward(Award \sqcap \exists event.FilmFestival)$.

4.4 Rewriting SPARQL queries and query patterns

From the set of complex correspondences for the pair Music-DBpedia and the query patterns for Music, we applied our approach (§3) for rewriting Music patterns in terms of the DBpedia vocabulary. Before rewriting patterns, we evaluated the use of these correspondences for rewriting SPARQL queries. For doing so, we defined a set of rules for translating a complex correspondence pattern into RDF graph patterns (Table 2). These rules are intended to be used for guiding the process of SPARQL rewriting. Following these rules, we managed to rewrite the 25 first SPARQL queries from the benchmark training data in QALD 2013[8]. The training data include 100 natural language questions for MusicBrainz with the corresponding SPARQL queries, as well as the answers these queries retrieve. The queries have been rewritten in order to interrogate DBpedia.

ID	**Formal pattern**	**SPARQL rewriting rule**
CAT	A ≡ ∃R.B	?x a A → { ?x R B }
CAT-1	A ≡ B ⊓ ∃R-.T	?x a A → { ?x a B . ?y R ?x }
CAV	A ≡ ∃R.{...}	?x a A → { ?x R "..."^^*ex:dataType* }
AVR	A ≡ B ⊓ ∃R.{...}	?x a A → { ?x a B . ?x R *SomeValue* . }
PC	R ≡ P.(A ⊓ ∃Q)	?x R ?y → { ?x P A . A Q ?y }
IP	R- ⊑ P	?x R ?y → ?y P ?x
OR	A ≡ B ⊓ (∃R.T ⊔ ∃Q.T)	?x a A → {?x a B . {B R ?y} UNION {B Q ?z}}
AND	A ≡ B ⊓ (∃R.T ⊓ ∃Q.T)	?x a A → {?x a B . ?x R ?y ; Q ?z .}

Table 2. Complex correspondence patterns and SPARQL rewriting rules.

As an example, consider the query in Table 3 asking if there are *members of the Ramones who are not named Ramone* (question #25) over MusicBrainz, and the same query rewritten for DBpedia. The MusicBrainz result answers *false*, while the DBpedia result asserts the opposite. The DBpedia request is not less correct: if we return the actual query solutions (SELECT) instead of testing whether or not the query pattern has a solution (ASK), we find that Clem Burke in DBpedia was a member of The Ramones under the name "Elvis Ramone", while the MusicBrainz data set directly refers to him with this alias. In fact, both sets of instances do not fully intersect and they are not necessarily/correctly interlinked[9]. From this point of view, 18 of the 25 rewritten queries are correct and *consistent* with the queries for MusicBrainz: they do not necessarily give the same results, but they do answer the same question. 3 of these 18 results give the same number of solutions with exactly the same literals. 5 out of the 7 remaining results give no solution at all (no instance). And finally, the 2 last results are not fully correct since the complex correspondences ahead are not correct themselves. For instance, $MusicalManifestation \sqcap \exists release\_type.live \sqsubseteq MusicalWork \sqcap \exists recordedIn.PopulatedPlace$ turns out to be erroneous since the albums which have been recorded in a recording studio are equally selectable

[8]Open challenge on Multilingual Question Answering over Linked Data: `http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=task1&q=3`

[9]`http://wiki.dbpedia.org/Interlinking?v=vn`

within the property *recordedIn*: we wrongly thought it was reserved for live performances only. Thus, these results allowed us to validate our complex correspondences and remove those that prove to be incorrect.

ASK	ASK
WHERE {	WHERE {
?band foaf:name 'Ramones' .	?band foaf:name 'Ramones'**@en** .
?artist foaf:name ?artistname .	?artist foaf:name ?artistname .
?artist mo:member_of ?band .	**{?band dbo:bandMember ?artist}**
	UNION
	{?band dbo:formerBandMember ?artist} .
FILTER (NOT regex(?artistname,"Ramone"))	FILTER (NOT regex(?artistname,"Ramone"))
}	}

Table 3. DBpedia rewritten query (right) from MusicBrainz query (left). Namespace prefix bindings were ommitted (*dbo* refers to dbpedia and *mo* to music).

Next, we rewrote the Music query patterns in terms of the DBpedia vocabulary. Using the 11 simple correspondences and the 28 complex correspondences, we achieved a rewriting percentage of 90% of the Music patterns: on the 19 subgraphs (subpatterns) identified in the patterns for Music, we were able to transform 17 of them. For the Cinema patterns, we were able to rewrite 45 out of 51 subpatterns from the Cinema IRIT query patterns. Then, the patterns rewritten from Music were injected in the SWIP system along the DBpedia data set, in order to demonstrate the relevance of rewriting query patterns in the whole process. We managed to run five queries from QALD and originally intended to MusicBrainz. The generated SPARQL queries are (semantically) correct as long as (i) the correspondences involved do not apply any disjunction of terms, which is not currently supported in SWIP (in this case, only the most likely term is kept), and (ii) the source and target in the correspondences involved have the same information level (basically, equivalence).

5 Related work

Rewriting query patterns using complex correspondences is the novel aspect of this paper. With respect to complex correspondence generation, different approaches have emerged in the literature in the last years. A common approach is based on complex correspondence patterns [18, 17, 14, 15] (§4.3). Walshe [19] proposes refining elementary correspondences by identifying which correspondence pattern best represents a given correspondence. Following a different strategy, Qin *et al.* [13] propose an iterative process that combines terminological, structural, and instance-based matching approaches for mining *frequent queries*, from which complex matching rules (represented in FOL) are generated. Nunes *et al.* [10] present a two-phase instance-based technique for complex datatype property matching, where the first phase identifies simple property matches and the second one uses a genetic programming approach to detect complex matches. Recently, Arnold [1] uses state-of-the-art matchers for generating initial correspondences

that are further (semi-automatically) enriched by using linguistic, structural and background knowledge-based strategies. Although different strategies have been proposed, very few matchers for generating complex correspondences are available or use EDOAL, an expressive alignment language [3], for representing them. With respect to query patterns rewriting, the problem can be seen, at a lower level of abstraction, as a problem of SPARQL rewriting. Correndo *et al.* [2] propose a set of SPARQL rewriting rules exploiting both (complex) ontology alignments and entity co-reference resolution. Zheng *et al.* [20] propose to rewrite SPARQL queries from different contexts, where context mappings provide the articulation of the data semantics for the sources and receivers. Makris *et al.* [9, 8] present the SPARQL-RW rewriting framework that applies a set of predefined (complex) correspondences. They define a set of correspondence types that are used as basis for the rewriting process (i.e., *Class Expression*, *Object Property Expression*, *Datatype Property*, and *Individual*). However, the way the set of complex correspondences is established is not described. Our naive approach for rewriting query patterns is close to these SPARQL rewriting proposals in the sense of using complex correspondences. One of the difficulties is the lack of established ways for automatically identifying them. Although m:n complex correspondences are proposed at conceptual level, few concrete examples are available in the literature. Most of our correspondences are n:m. As most proposals, we start from a set of (automatically) discovered simple correspondences. Finally, our method is applied in an applicative context of rewriting patterns for a question answering system over RDF data.

6 Conclusions and future work

This paper has discussed the use of complex correspondences for rewriting query patterns, aiming at reusing query families across data sets that overlap. Although we could not fully evaluate the rewriting process mainly due to the fact that SWIP does not treat pattern disjunctions, we were able to validate our approach on a subset of manually validated complex correspondences. This opens several opportunities for future work. First, the structure of query patterns in SWIP could evolve so that they match the structure of the complex correspondences we have established. In particular, SWIP would benefit from the disjunction of subpatterns or specification of instances in patterns. Second, we plan to represent our complex correspondences using EDOAL. Third, we plan to formalise the composition of complex correspondence patterns, which are thereby the building blocks to obtain richer correspondences, following a grammar defining a set of rules for rewriting logical statements in FOL or DL. The grammar must define the properties of pattern precedence, transitivity, associativity, commutativity, and distributivity. Finally, we plan to propose an approach for (multilingual) complex correspondence generation, exploiting specially the $\mathcal{A}$Box of ontologies.

References

1. P. Arnold. Semantic enrichment of ontology mappings: Detecting relation types and complex correspondences. In *25th GI-Workshop on Foundations of Databases*, 2013.
2. G. Correndo, M. Salvadores, I. Millard, H. Glaser, and N. Shadbolt. SPARQL Query Rewriting for Implementing Data Integration over Linked Data. In *1st International Workshop on Data Semantics (DataSem 2010)*, March 2010.
3. J. David, J. Euzenat, F. Scharffe, and C. Trojahn. The Alignment API 4.0. *Semantic Web*, 2(1):3–10, 2011.
4. S. Elbassuoni and R. Blanco. Keyword search over rdf graphs. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 237–242. ACM, 2011.
5. S. Elbassuoni, M. Ramanath, R. Schenkel, and G. Weikum. Searching RDF Graphs with SPARQL and Keywords. *IEEE Data Eng. Bull.*, 33(1):16–24, 2010.
6. J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag, Berlin, Heidelberg, 2007.
7. P. Gillet, C. Trojahn, and O. Haemmerlé. Réécriture de patrons de requêtes à l'aide d'alignements d'ontologies. In *Atelier Qualité et Robustesse dans le Web de Données, IC*, 2013.
8. K. Makris, N. Bikakis, N. Gioldasis, and S. Christodoulakis. SPARQL-RW: transparent query access over mapped RDF data sources. In *15th International Conference on Extending Database Technology*, pages 610–613. ACM, 2012.
9. K. Makris, N. Gioldasis, N. Bikakis, and S. Christodoulakis. Ontology mapping and SPARQL rewriting for querying federated RDF data sources. In *2010 Conference on On the Move to Meaningful Internet Systems*, pages 1108–1117, 2010.
10. B. Nunes, A. Mera, M. Casanova, K. Breitman, and L. A. Leme. Complex Matching of RDF Datatype Properties. In *6th Workshop on Ontology Matching*, 2011.
11. C. Pradel, O. Haemmerlé, and N. Hernandez. A Semantic Web Interface Using Patterns: The SWIP System. In *Graph Structures for Knowledge Representation and Reasoning*, LNCS, pages 172–187. Springer Berlin Heidelberg, 2012.
12. C. Pradel, O. Haemmerlé, N. Hernandez, et al. Des patrons modulaires de requêtes sparql dans le système swip. *23es Journées d'Ingénierie des Connaissances*, 2012.
13. H. Qin, D. Dou, and P. LePendu. Discovering executable semantic mappings between ontologies. In *OTM International Conference*, pages 832–849, 2007.
14. D. Ritze, C. Meilicke, O. Sváb-Zamazal, and H. Stuckenschmidt. A pattern-based ontology matching approach for detecting complex correspondences. In *4th Workshop on Ontology Matching*, 2009.
15. D. Ritze, J. Völker, C. Meilicke, and O. Sváb-Zamazal. Linguistic analysis for complex ontology matching. In *5th Workshop on Ontology Matching*, 2010.
16. A. Russell and P. R. Smart. Nitelight: A graphical editor for sparql queries. In *Poster and Demo Session at the 7th International Semantic Web Conference*, 2008.
17. F. Scharffe. *Correspondence Patterns Representation*. PhD thesis, University of Innsbruck, Innsbruck, 2009.
18. F. Scharffe and D. Fensel. Correspondence patterns for ontology alignment. In *Knowledge Engineering: Practice and Patterns*, pages 83–92. Springer, 2008.
19. B. Walshe. Identifying complex semantic matches. In *9th International Conference on The Semantic Web: Research and Applications*, pages 849–853, 2012.
20. X. Zheng, S. E. Madnick, and X. Li. SPARQL Query Mediation over RDF Data Sources with Disparate Contexts. In *WWW Workshop on Linked Data on the Web*, 2012.