Semantic Web 0 (0) 1
IOS Press


# Survey on complex ontology matching

Élodie Thiéblin *, Ollivier Haemmerlé, Nathalie Hernandez, Cassia Trojahn
*Equipe MELODI, Institut de Recherche en Informatique de Toulouse, Toulouse, France*
*E-mails: elodie.thieblin@irit.fr, ollivier.haemmerle@irit.fr, nathalie.hernandez@irit.fr, cassia.trojahn@irit.fr*

**Abstract.** Simple ontology alignments, largely studied in the literature, link a single entity of a source ontology to a single entity of a target ontology. One of the limitations of these alignments is, however, their lack of expressiveness which can be overcome by complex alignments. While diverse state-of-the-art surveys mainly review the matching approaches in general, to the best of our knowledge, there is no study taking the specificities of the complex matching problem. In this paper, an overview of the different complex matching approaches is provided. This survey proposes a classification of the complex matching approaches based on their specificities (i.e. type of correspondences, guiding structure). The evaluation aspects and the limitations of these approaches are also discussed. Insights for future work in the field are provided.

Keywords: Ontology matching, complex alignment, survey, schema matching

## 1. Introduction

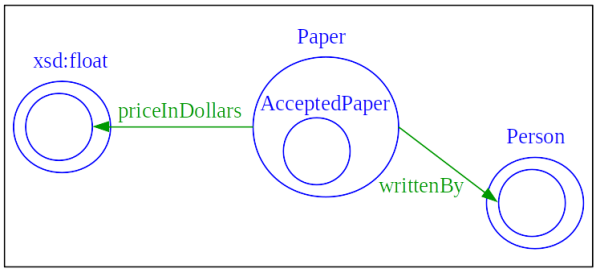
Ontology matching is an essential task for the management of the semantic heterogeneity in open environments. This task is often associated with the schema matching problem [1] as they share the same goal: interoperability. Largely speaking, the matching process aims at generating a set of correspondences (i.e., an alignment) between the entities of different knowledge representation models (e.g., ontologies, schemata, etc.). Two 'paradigms' organise the field. While approaches generating simple correspondences are limited to matching single entities (i.e., linking a single entity from a source ontology to a single entity of a target ontology), complex matching approaches are able to generate correspondences which express the relationships between entities from different ontologies better.

With the increasing amount of knowledge sources made available on the Linked Open Data (LOD) and their variety of modelling choices, the relationships between entities of these sources are required to be more expressive. Simple correspondences (binary links) are not expressive enough to fully overcome ontology conceptual heterogeneity. Earlier works have introduced the need for complex alignments and solutions for it, not only in terms of automatism, but as of representation and management [2]. However, currently, very few complex alignments are available and published on the LOD. One of the reason is the difficulty of producing and evaluate such alignments. This is corroborated in [3], where a survey on ontology matching researchers and future challenges in the field have been discussed. They agree on the fact that there is a need to "automatically discover complex relations, instead of 1:1".

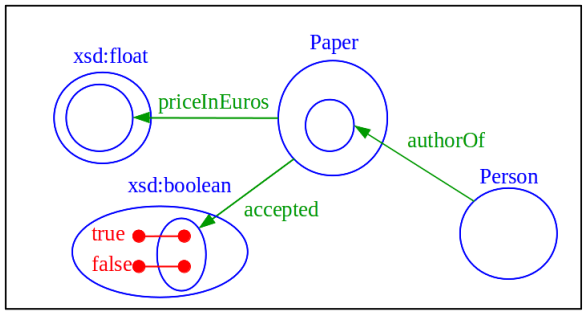
Despite this fact, different complex approaches have emerged in the literature, adopting a diversity of matching strategies and dealing with different knowledge representation models (from database schemata, taxonomies to heavy ontologies). Applications consuming such complex alignments have been proposed for different tasks [4], data translation [5], ontology merging [6], ontology evolution [7].

Diverse surveys in the literature have been focused on the different aspects of the schema and ontology matching [1, 3, 8–13] without paying attention to the specificities of complex matching (i.e., underlying strategy, structure of complex correspondences, etc.). The aim of this survey is to provide an overview of the complex matching approaches dealing with different kinds of knowledge representation models such as ontologies, XML schemata, database schemata, etc. A classification of the approaches based on the specifici-

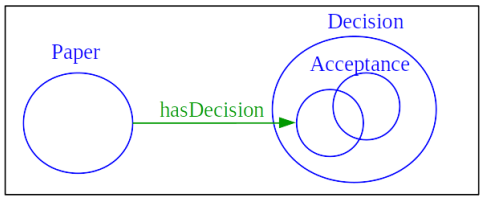
*Corresponding author. E-mail: elodie.thieblin@irit.fr.




(a) $o_1$




(b) $o_2$




(c) $o_3$

Fig. 1. Example ontologies. The format used to represent the ontologies is described in [14]

ties of complex alignments is proposed. The evaluation aspects of these approaches are also overviewed. Limitations of both approaches and evaluations are discussed and insights for future work in the field are provided, in particular to foster the generation of complex alignments on the LOD.

This paper is organised as follows. After background definitions (§2), a classification of complex matching approaches is proposed (§3), followed by a description of state-of-the-art approaches (§4) and their evaluation (§5). The example presented in the next section gives an intuition on how complex alignments can be used, and why they are necessary.

*A motivating example*

Consider three toy ontologies $o_1$, $o_2$ and $o_3$ illustrated in Figure 1, each ontology being used for describing the data of a given conference. Imagine that the three conferences merge into one event and that the three knowledge bases need to be queried by a unique software. To make the knowledge available for this software, different solutions are possible: translate the data of two knowledge bases according to the third ontology, merge the three ontologies and the knowledge bases, rewrite each query asked by the software against a dataset into queries adapted to the two others, etc. Each solution has its pros and cons, but in all cases correspondences between entities of each ontology can help automatising the task.

Links between single entities of the different ontologies can be expressed. The class $o_2$*:Paper* corresponds to the class $o_1$*:Paper* or the class $o_3$*:Paper*. The same kind of simple correspondence can be established between the class $o_2$*:Person* and $o_1$*:Person* (the notion of Person is not considered in $o_3$). However, to express the links between the price of a paper in Dollars and the price of a paper in Euro (1.), accepted papers in each dataset (2.)(3.) and the relations expressed between a paper and its author (4.), *complex correspondences* are needed :

1. $\forall x,y,\ o_1$*:priceInDollars(x,y)* $\equiv$ $o_2$*:priceInEuro(x,conversionFunction(y))*
2. $\forall x,\ o_1$*:AcceptedPaper(x)* $\equiv$ $o_2$*:accepted(x,true)*
3. $\forall x,\ o_3$*:hasDecision(x,y)* $\wedge$ $o_3$*:Acceptance(y)* $\equiv$ $o_1$*:AcceptedPaper(x)*
4. $\forall x,y,\ o_1$*:writtenBy(x,y)* $\equiv$ $o_2$*:authorOf(y,x)*

With the established simple and complex correspondences all the available knowledge in the different bases can be accessed. For example, in the case of the query rewriting solution, considering only simple correspondences such as done in [5] will lead to a loss in information when searching for all the authors of an accepted paper. Approaches such as [4, 15, 16] will retrieve information by processing the complex correspondence (2). Such approaches are able to automatically transform query Q1 written for $o_1$ into query Q2 written for $o_2$.

```
Q1:
SELECT ?author WHERE {
  ?paper rdf:type o1:AcceptedPaper.
  ?paper o1:writtenBy ?author.
}
```

```
Q2:
SELECT ?author WHERE {
  ?paper rdf:type o1:AcceptedPaper.
```

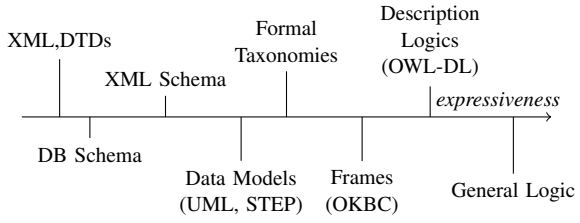



Fig. 2. Various kinds of knowledge representation models sorted by expressiveness (adapted from [17]).

```
  ?paper o1:writtenBy ?author.
}
```

## 2. Background

In this section the scope of this study is specified and the definitions related to alignments are given.

### 2.1. Knowledge representation models

*"An ontology can be viewed as a set of assertions that are meant to model some particular domain. Usually, they define a vocabulary used by a particular application. In various areas of computer science, there are different data and conceptual models that can be thought of as ontologies."* [13]. As quoted, the need for knowledge representation has led to different kinds of knowledge representation models.

Figure 2 presents the different kinds of models sorted by expressiveness and found in the literature dealing with complex matching and which are considered in this survey. The different levels of expressiveness in these knowledge representation models will influence the matching techniques. An approach can match two models of the same kind (e.g., XML schema to XML schema) or of different kinds (e.g., DB schema to DL ontology).

Complex alignments can also occur between representation of other objects such as business processes [18]. As focus of this survey is knowledge representation, therefore, these approaches are out of the scope of this study.

### 2.2. Expressions

Before the definition of alignments, the definition of expressions, which will serve to define the notion of correspondences, is given.

A **simple expression** is composed of a single entity represented by its unique identifier (e.g. an IRI for ontologies). For example, the IRI $o_1$*:Paper* is a simple expression of $o_1$.

A **complex expression** is composed of at least one entity on which a constructor or a transformation function is applied. For example, $\forall x$, $o_2$*:accepted(x,true)* is a complex expression which represents all the papers having the value *true* for the $o_2$*:accepted* property . The constructor used here is a value restriction constructor. A **constructor** is a logic constructor (union, intersection, inverse, transitivity, etc.) or a restriction constructor (cardinality restriction, type restriction, value restriction, etc.).

A **transformation function** is a function that modifies the values of a literal field. It can be an aggregation function (e.g. string concatenation, sum of integers, etc.), a conversion function (e.g. metric conversion, etc.), etc.

In domains such as ontology evolution [7, 19], other constructors and transformation functions exist. However, those are out of the scope of this survey.

### 2.3. Alignment and correspondence

As defined in [13], **ontology matching** is the process of generating an ontology alignment *A* between two ontologies: a source ontology $o_1$ and a target ontology $o_2$[1].

**Definition 1.** *An **ontology alignment** A is directional, denoted* $A_{o_1 \to o_2}$. $A_{o1 \to o2}$ *is a set of correspondences* $A_{o_1 \to o_2} = \{c_1, c_2, ..., c_n\}$.

**Definition 2.** *A **correspondence** $c_i$ is a triple* $\langle e_{o_1}, e_{o_2}, r \rangle$ *which can be written* $e_{o_1}$ *r* $e_{o_2}$. $e_{o_1}$ *and* $e_{o_2}$ *are the **members** of the correspondence: they can be simple or complex expressions with entities from respectively* $o_1$ *and* $o_2$*:*

- *if the correspondence $c_i$ is **simple**, both $e_{o_1}$ and $e_{o_2}$ are simple expressions;*
- *if the correspondence is **complex**, at least one of $e_{o_1}$ or $e_{o_2}$ is a complex expression;*
- *r is a relation, e.g., equivalence ($\equiv$), more general ($\sqsupseteq$), more specific ($\sqsubseteq$), disjointedness ($\perp$), holding between $e_{o_1}$ and $e_{o_2}$;*
- *additionally, a value n (typically in [0,1]) can be assigned to $c_i$ indicating the degree of confidence that the relation r holds between $e_{o_1}$ and $e_{o_2}$.*

[1] The focus here is pairwise ontology matching and compound ontology matching [20] is out of the scope of this paper.

One can indicate if each member of the correspondence is a simple expression, noted *s*, or a complex expression, noted *c*.

A simple correspondence is always *(s:s)* whereas a complex correspondence can be *(s:c)*, *(c:s)* or *(c:c)*. The *(1:1), (1:n), (m:1), (m:n)* notations have been used for the same purpose in the literature [11] (*1* for *s* and *m* or *n* for *c*). However they can be misinterpreted as the alignment arity or multiplicity [21].

In the following, are provided some examples of complex correspondences based on the definitions above and the motivating example ontologies (Figure 1).

1. $c_1$ = $\forall$*x,* $o_1$*:Person(x)* $\equiv$ $o_2$*:Person(x)* is a *(s:s)* simple correspondence.
2. $c_2$ = $\forall$*x,y,* $o_1$*:priceInDollars(x,y)* $\equiv$ $o_2$*:priceInEuro(x,conversionFunction(y))* is a *(s:c)* complex correspondence with a transformation function (here presented in the *conversionFunction(y)* that states that $conversionFunction(y) = changeRate \times y$).
3. $c_3$ = $\forall$*x,* $o_3$*:hasDecision(x,y)* $\wedge$ $o_3$*:Acceptance(y)* $\equiv$ $o_1$*:AcceptedPaper(x)* is a *(c:s)* complex correspondence with constructors. Note that $o_3$*:Paper(x)* is not specified in the left member of the correspondence because $o_3$*:Paper* is the domain of the $o_3$*:hasDecision* property (c.f. figure 1c). Therefore, $o_3$*:Paper(x)* is implied in the left member.
4. $c_4$ = $\forall$*x,y,* $o_1$*:writtenBy(x,y)* $\equiv$ $o_2$*:authorOf(y,x)* is a *(s:c)* complex correspondence with the *inversion* constructor.
5. $c_5$ = $\forall$*x,* $o_2$*:accepted(x,true)* $\equiv$ $\exists$*y,* $o_3$*:hasDecision(x,y)* $\wedge$ $o_3$*:Acceptance(y)* is a *(c:c)* complex correspondence with constructors.

In opposition to a **simple alignment**, a **complex alignment** contains at least one complex correspondence. Complex ontology matching is the process of generating a complex alignment between ontologies. The approaches for generating such kind of alignment are discussed in the next section.

## 3. Classification of complex matchers

Ontology matching approaches have been classified in various surveys [1, 3, 8–13]. These classifications however do not address the specificities of the complex approaches. After giving an overview of the main ontology matching approaches classifications (§3.1), axis for complex matching approaches classification are presented (§3.2).

### *3.1. Classifications of ontology matching approaches*

Euzenat and Shvaiko [1, 13] define three matching dimensions: input, process and output which will be the guiding thread to present the classifications. Most of the classifications so far focus on the input and process dimensions [1, 9, 11–13].

Regarding the input dimension, the *instance vs ontology* classification (called *instance vs schema* in [11]) divides the matchers into those which deal with information from the $\mathcal{TB}ox$ and those which deal with the $\mathcal{AB}ox$. Rahm *et al.* [11] also consider as input the type of auxiliary information used by the approaches (thesaurii, etc.). For the process dimension, Rahm *et al.* [11] propose classification axis such as *element vs structure*, *linguistic vs constraint-based*. All of these classification axis are put together into a taxonomy.

The classification of Rahm *et al.* [11] has been developed and extended by Euzenat and Shvaiko in [1, 13]. For instance, they distinguish whether an input is considered syntactically or semantically by the approach. The two-ways taxonomy ends in basic approach strategies (e.g. string-based, model-based, formal resource-based, etc.).

The classification of schema matching techniques of Doan *et al.* [12] separates *rule-based* techniques from *learning-based* techniques. Considering both input and process dimensions, *rule-based* techniques only exploit schema-level information in specific rules while *learning-based techniques* may exploit data instance information with machine-learning or statistical analysis.

Noy [9] proposes two main categories of ontology matching approaches: the first one includes the matching process is guided by a top-level ontology from which the source and target ontologies derive; the second one includes the heuristic-based and machine-learning techniques.

Regarding the output dimension of the matching approaches, Rahm *et al.* [11] considers the output alignment arity as a characteristic of the approaches which could be integrated into its taxonomy.

In summary, among the ontology matching classifications so far, the one from Euzenat and Shvaiko [13] is the most extensive (all the others can be represented in this classification). However, even if considered, the output dimension of the matching approaches is hardly a basis for classification, whereas it becomes of interest when considering complex correspondences.

More generally, the classifications of ontology matching cited above do not address the specificities of the

complex matching problem. The characteristics of the processes leading to the generation of complex correspondences need to be studied, in particular the kind of structure guiding the discovery of correspondences. The next section presents classification axis for complex ontology matching approaches.

### 3.2. Classification for complex matching approaches

The specificities of the complex matching approaches rely on their output and their process. These are the two axis of the proposed classification. In this section, the different types of output (types of correspondences) and the structures used in the process to guide the correspondence detection are presented (guiding structure).

*Type of correspondence.* The correspondences (output of the matching approaches) are divided into three main categories according to their type: *logical relations*, *transformation functions* and *blocks* (Figure 4). The **logical relations** category stands for correspondences whose complex members are expressed with logical constructors only. In contrast, the **transformation functions** category includes the approaches that generate correspondences with *transformation functions* in its members. The **blocks** correspondences gather entities using a grouping constructor in their members (clusters of entities), not specifying a semantic relation between them. Note that in theory, a correspondence could have members expressed with transformation functions combined with logical constructors but no approach able to generate such kind of correspondences was found. However, some approaches are able to generated both types independent of each other, as depicted in Figure 4.

*Guiding structures.* These categories aim at classifying the (complex) matching approaches based on their process dimension. In particular, it focuses on the structure on which the process generating the correspondences relies:

- **Atomic patterns** The approaches in this category consider the correspondence as an instanciation of an atomic pattern, such as the ones defined by Scharffe [22]. An atomic pattern is a template of a correspondence. A template can represent logical relation or transformation function correspondences. For example, an approach looking for correspondences following this exact pattern: $\forall x,\ o_1{:}A(x) \equiv \exists y\ o_2{:}b(x,y) \wedge o_2{:}C(y)$ falls into this category and in the *logical relation* type of correspondence. An approach searching for $\forall x,y,\ o_1{:}a(x,y) \sqsubseteq \exists\ y_1,y_2 o_2{:}b(x,y_1) \wedge o_2{:}c(x,y_2) \wedge (y = y_1 + y_2)$ falls into this category and in the *transformation function* type of correspondence.
- **Composite patterns** The approaches in this category aim at finding repetitive compositions of an atomic pattern. As for the atomic patterns, the composite patterns can represent both logical relations and transformation functions correspondence patterns. For example, an approach looking for correspondences of the form $\forall x,\ o_1{:}A(x) \equiv o_2{:}B(x) \vee o_2{:}C(x) \vee o_2{:}D(x)...$, where $o_1{:}A$, $o_2{:}B$, $o_2{:}C$, $o_2{:}D$, *etc.* are classes and the number of unions in the right member of correspondences is not *a-priori* defined by the approach, falls into this category. Correspondences representing string concatenation of unlimited number of properties also fall into this category and in the *transformation function* type of correspondence.
- **Path** The approaches in this category detect the correspondences using path-finding algorithms. The resulting correspondence is a property path in $o_1$ put in relation with a path in $o_2$. For example, an approach looking for a path between two pairs of aligned instances described by $o_1$ resp. $o_2$ falls into this category.
- **Tree** The approaches in this category rely on tree structures for correspondence detection. The tree structure can either be a help to the process or the tree structure of a schema. For example, in genetic programming based approaches, a tree structure is used for the construction of the correspondences. In other approaches, a schema is considered as a tree and the approach consists in finding the smallest equivalent tree in an ontology.
- **No structure** As opposite to the other approaches, the approaches of this category do not rely on a structure to guide the correspondence generation. Instead, they discover correspondences more freely.

The process of complex ontology matching also has a dimension of **members expressions pre-definition**. Three categories are proposed regarding if one, both or none members of the correspondences follows an expression template defined by the approach :

- The **fixed to fixed** category includes the matching approaches that always produce correspondences with fixed members expressions. Atomic

patterns-based approaches generate fixed to fixed correspondences as both members' expressions are defined by the pattern.
- The **fixed to unfixed** members expression category covers the matching approaches for which one of the members of the correspondence will always follow the same expression template, while the expression of the other member may vary. For example, an approach aiming at traducing each property of an ontology into a path falls into this category: one of the member will always be one property while the other will be a path of a-priori undefined length.
- The **unfixed to unfixed** members expression category includes the approaches that output correspondences whose members have an undefined expression before-hand. For example, an approach aiming at finding similar paths in two ontologies falls into this category: both members have a-priori undefined length.

A matching approach can exploit many different matching strategy to find complex correspondences. In the following, the matching strategies of the approaches are classified on their guiding structure. Therefore, the same approach can appear in multiple sections.

## 4. Complex alignment approaches

The following sections present the approaches according to our classification. Although these sections are organised according to the guiding structure (Figure 3), a reference to the kind of output and members expression pre-definition is made in the text. The approaches are detailed in paragraphs whose title follows a template : *Name [ref] Type of knowledge representation models, [(s:c), (c:s), (c:c)].*

### 4.1. Atomic patterns

Atomic patterns are used in approaches to detect logical relations as well as transformation functions relations. Table 1 shows the atomic patterns of the correspondences which guide the state-of-the-art approaches of this category.

The atomic pattern based approaches have different strategies for the definition of their patterns. For instance, some rely on the patterns defined by one of the ontologies to align [36], others approaches have their own pattern library [23, 25, 31, 40, 43, 45]. Two main detection techniques appear: structuro-linguistic conditions (called *matching patterns* defined in [55]) [23, 24, 31, 36, 40], and statistical measures [25, 43, 45]. These approaches are detailed in the following.

Table 1
Atomic patterns per approach

| *Work* | *Patterns* |
|---|---|
| Ritze2009 [23] | Class by Attribute Type, Class by Inverse Attribute Type, Class by Attribute Value, Property Chain |
| Ritze2010 [24] | Inverse property, Class by Attribute Type, Class by Inverse Attribute Type, Class by Attribute Value |
| Svab2009 [31] | N-ary relation |
| Rouces2016 [36] | Linguistic patterns of FrameBase |
| Walshe2016 (Bayes-ReCCE) [25] | Class by Attribute Value |
| Jiang2016 (KAOM) [43] | Linear Regression |
| Dhamankar2004 (iMAP) [45] | Conversion functions predefined, basic arithmetic properties |
| Jimenez2015 (BootOX) [40] | RDB schema properties to OWL axioms |

*Ritze et al. [23, 24] Ontology to Ontology, (s:c),* In [23, 24], Ritze *et al.* propose a set of matching conditions to detect correspondence patterns: *Class by Attribute Value*, *Class by Attribute Type*, *Class by Inverse Attribute Type*, *Inverse Properties* and *Property Chain* defined by Scharffe [22]. The conditions are based on the labels of the ontologies entities, the structures of these ontologies and the compatibility of the data-types of data-properties. The matching conditions to detect these patterns are an input to the matching algorithm. The user can add new matching conditions to detect other patterns.

The first approach [23] detects the modifier and head-noun of a label. In the matching conditions, string similarity (Levenshtein distance) is used to detect a potential relation between two entities (e.g. *Acceptance* is similar to *Accepted*). The improved version of the matching conditions [24] refines the syntactic part of the previous work by introducing linguistic analysis such as detection of antonymy, active form, etc. Various linguistic analysis features are studied and incorporated in the matching conditions. In Example 1, the simplified matching conditions to detect inverse property states that if the verb phrase of the label of a property $p_1$ is the active voice of the verb phrase of a la-

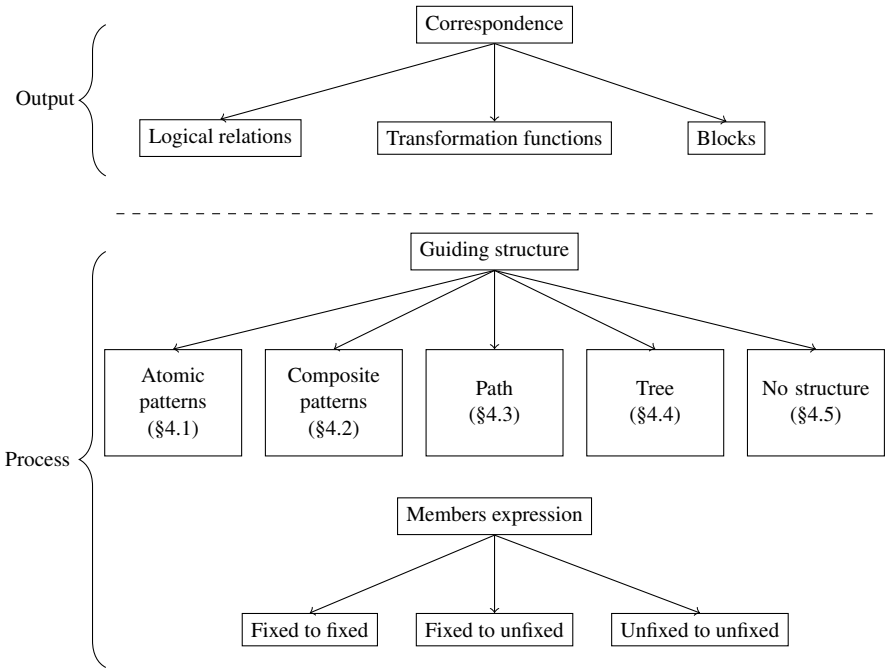



Fig. 3. Two axis to characterise the complex matching approaches: output and process

bel of an other property $p_2$, then $\forall x,y,\ o_1{:}p_1(y,x) \sqsubseteq o_2{:}p_2(x,y)$ is a probable correspondence.

**Example 1.** *Conditions:* $\forall x,y\ o_1{:}p_1(y,x) \sqsubseteq o_2{:}p_2(x,y)$ *iff verb*$(p_1)$ *= active-voice (verb*$(p_2)$*)*
*Correspondence:*
$\forall x,y\ o_1{:}writePaper(y,x) \sqsubseteq o_2{:}writtenBy(x,y)$
*because write = active-form(written)*

The structural matching conditions are the same for both approaches. Example 1 is extended with structural constraints on the range and domain of $p_1$ and $p_2$ : $dom(p_1) \sqsupseteq range(p_2)$ and $range(p_1) \sqsubseteq dom(p_2)$. The subsumption between range and domains of the two properties can be detected by inference on the ontologies structure linked by the simple reference alignment or by an hypernymy relation between the labels.

*Šváb-Zamazal and Svátek [31] Ontology to ontology, (s:c)* This approach is based on structural and naming conditions to detect N-ary relations as defined by the Semantic Web Best Practice (SWBP)[2] in the source ontology. Then other conditions, similar to those of [23] (e.g. similarity between two concept's labels), are used to match the detected N-ary relations to an object property in the target ontology. This approach is similar to Ritze *et al.*'s [23, 24].

[2]https://www.w3.org/TR/swbp-n-aryRelations/

*Rouces et al. [36] Ontology to FrameBase ontology, (s:c) (c:s)* Rouces *et al.* use FrameBase as a mediator ontology for complex alignment discovery. FrameBase is an ontology based on linguistic frames, seen as linguistic patterns in this approach. The approach identifies complex patterns in FrameBase from the linguistic patterns it describes. For each complex pattern identified, a corresponding *candidate property* is created (see Example 2). The names of the properties of the source ontology (the one to be aligned to FrameBase) are pre-processed, for example $o_1$*:birthDate* becomes $o_1$*:hasBirthDate*. The properties of the source ontology are then aligned with simple alignments to the candidate properties created in FrameBase. The similarity of two properties is calculated based on a bag of words cosine from the tokenised property names. Once a source ontology property has been aligned to a created property of FrameBase, it is aligned to its corresponding pattern. The originality of this approach, is that the correspondence patterns on which it relies are encoded in one of the aligned ontologies (FrameBase).

**Example 2.** *Created property: frame:hasBirthDate(s,o)*
*Pattern: frame:BirthEvent(e)* $\wedge$ *frame:hasSubject(e,s)* $\wedge$ *frame:hasDate(e,o)*
*Source property:* $o_1$*:birthDate* $\rightarrow$ $o_1$*:hasBirthDate*
*Simple correspondence:* $\forall x,y,\ o_1{:}hasBirthDate(x,y) \equiv$ *frame:hasBirthDate(x,y)*

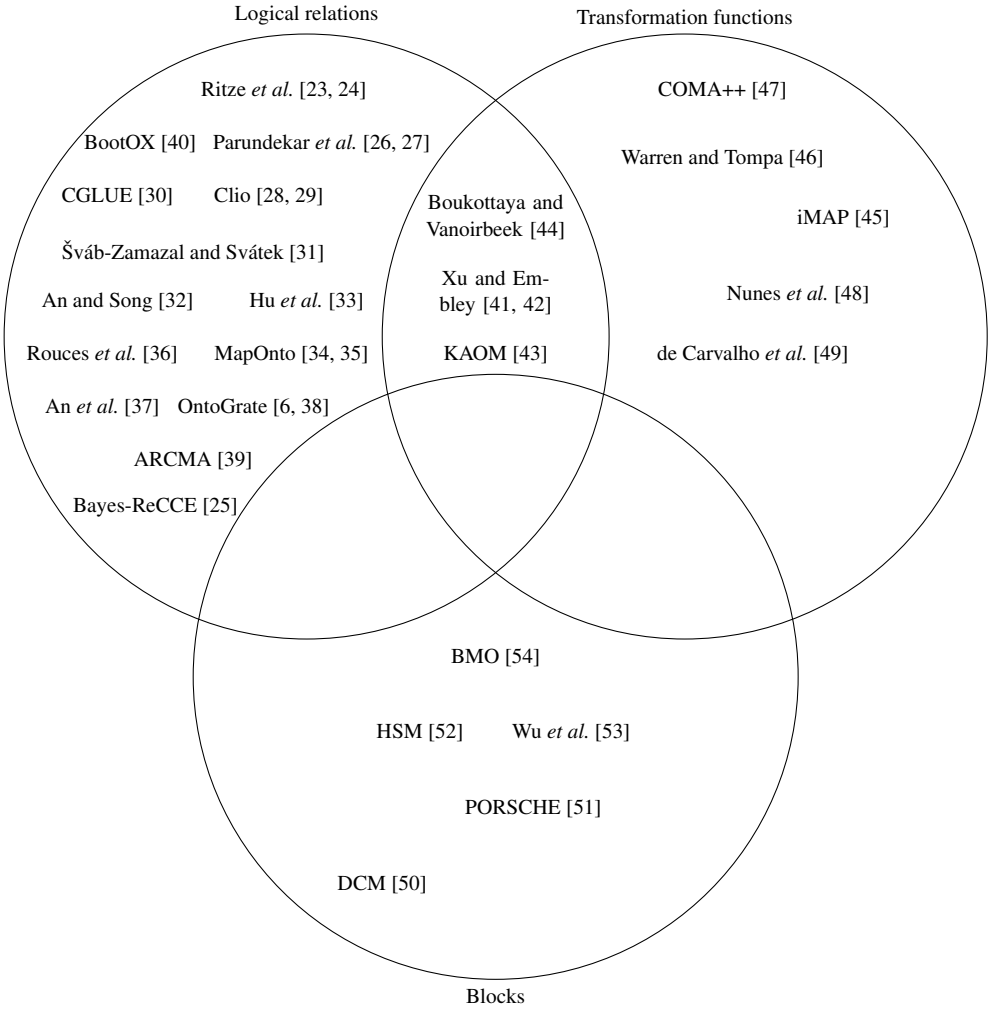



Fig. 4. Classification of the complex matching techniques according to the type of correspondence they output

*Correspondence:*
$\forall x,y,\ o_1$*:birthDate(x,y)* $\equiv \exists z,$ *frame:BirthEvent(z)* $\wedge$ *frame:hasSubject(z,x)* $\wedge$ *frame:hasDate(z,y)*

*Bayes-ReCCE [25] Ontology to ontology, (s:c)* This approach detects *Class Attribute Value Restrictions*, *Class Attribute Type* (and by extension *Class Attribute Existence*) correspondences. Bayes-ReCCE uses the properties of matched instances of two classes $o_1$*:A* and $o_2$*:B*, with $\forall x,\ o_1$*:A(x)* $\sqsupseteq$ $o_2$*:B(x)* in a reference alignment. The matching problem is transformed into the feature-selection problem. The common instances are represented as binary vectors, each feature of the vector represents the presence of an *attribute-value* pair for a given instance. Feature-selection is the process of reducing the search space of features (here *attribute-value* pairs) to keep only relevant features for a model (here a classification). A score is given to each feature. Two metrics are used in the scoring process: information gain (with a closed-world assumption) and beta-binomial class prediction metric based on Bayesian probabilities (compliant with the open-world assumption). For each class, the top-k best features are returned to the user to choose from.

*iMAP, Dhamankar et al. [45] Relational database schema to relational database schema, (c:s)* The iMAP system [45] uses a combination of *searchers* to discover simple and complex correspondences between database schemata. The validity of each correspondence is then checked by a similarity estimator based on the attributes' name similarity and a Naive-Bayes classifier trained on the target data. The correspondences are finally presented to a user who validates or

invalidates them. Each searcher implements a specific strategy. Some of the searchers use atomic patterns for correspondence detection. For instance, the numeric, category and schema mismatch searchers look for correspondences fitting given atomic patterns. These searchers base their confidence in a correspondence on the data value distribution using the Kullback-Leibler divergence measure. The unit conversion searcher is based on string recognition rules in the attributes' names and data (such as "$", "hour", "kg", etc.). The searcher finds the best match function from a predefined set of conversion functions.

*KAOM, Jiang et al. [43] Ontology to ontology, (s:c) (c:s) (c:c)* KAOM generates transformation function correspondences and logical relation correspondences. As the iMap's system [45], KAOM implements different matching strategies: one for detecting transformation function correspondences, the other for logical relation correspondences. Here is presented its transformation function correspondence detection approach, as it uses an atomic pattern. The logical relation correspondence approach is presented in §4.5. The atomic pattern used is a positive linear transformation function between numerical data properties $p_1$ and $p_2$ of respectively $o_1$ and $o_2$. As for some of iMAP's searchers, a Kullback-Leibler divergence measure on the data values is used to define the coefficient $a$ of the linear transformation: $p_1 = a \times p_2$.

*BootOX, Jimenez-Ruiz et al. [40] Database schema to ontology, (c:s)* The BootOX approach [40] produces correspondences between a relational database schema and a target ontology via the creation of a "bootstrapped" ontology. There are two phases to the approach. In the first phase, an ontology is bootstrapped from a relational database schema based on a set of patterns. For example, a non-binary relation table in the source schema produces a class in the bootstrapped ontology. The patterns used in this approach lead to the creation of axioms involving class restrictions in the bootstrapped ontology. R2RML correspondences between the relational database and its bootstrapped ontology are the result of this phase. This bootstrapped ontology is then aligned with the LogMap [56] matcher to the target ontology. LogMap relies on linguistic and structural information to perform the matching. The juxtaposition of the two alignments produce a complex alignment between the source database schema and the target ontology.

Other systems can bootstrap ontologies from relational database schemata [57, 58] but their aim is not to align the schema to an ontology. Therefore, they are out of the scope of this study. In this survey, BootOX is considered in its LogMap extension form.

### 4.2. Composite patterns

Composite pattern-based approaches often focus on one or two types of patterns. Table 2 presents the different composite pattern types detected by the approaches.

Some approaches iteratively construct the member(s) of the correspondence [27, 30, 39, 45, 46] (text searcher of iMap). Others first discover atomic pattern correspondences and merge them in a final (non-iterative) step [26, 47]. Approaches use graph-pattern matching either as detection conditions [44, 51, 53] or over the properties of a mediating ontology [41, 42, 45] (iMap's date searcher). Finally, [50, 52] start by grouping schema attributes before matching the groups. Even though the holistic approaches [50, 52] produce block correspondences (of properties only), it has been decided that these two approaches are composite pattern driven as the *grouping* phase follows a repetitive pattern. Some approaches search for composite patterns inside a tree structure [41, 42, 44, 51]. These approaches could also be classified into tree-guiding structure. However, as their matching process relies on the identification of a composite pattern in those trees, they were classified in this category.

*CGLUE, Doan et al. [30] Taxonomy to taxonomy, (s:c)* The GLUE system [30] is specialised in detecting (s:s) correspondences between taxonomies using machine learning techniques such as joint probability distribution. CGLUE, also presented in [30] is an extension of the GLUE system. It can detect (s:c) class unions in taxonomies such as $\forall x,\ o_1{:}A(x) \equiv o_2{:}B(x) \vee o_2{:}C(x) \vee \ldots$. To detect these unions, the authors make a few assumptions such as: the children of any taxonomic node are mutually exclusive and exhaustive. To find a match to a class $o_1{:}A$, each class-union of $o_2$ is considered a potential candidate. The first candidates are the set of single classes of $o_2$. An adapted beam search finds the $k$ best candidates according to a similarity score given by the GLUE system. The $k$ best candidates are then expanded as unions with the classes of $o_2$ until no improvement is done on the similarity score.

*Parundekar et al. [27] Ontology to ontology, (s:c) (c:s)* In this approach proposed by Parundekar *et al.* [27], the type of correspondences sought is an

Table 2
Composite patterns per approach

| *Work* | *Composite pattern* | *Pattern form* |
|---|---|---|
| Doan2003 (CGLUE) [30] | Class unions | $\forall x, A(x) \equiv B(x) \vee C(x) \ldots$ |
| Parundekar2012 [27] | Disjunction of attribute-value pairs | $\forall x, p(x,v) \equiv p_2(x,v_2) \vee p_2(x,v_3) \vee \ldots$ |
| Kaabi2012 (ARCMA) [39] | Class intersection | $\forall x, A(x) \equiv B(x) \wedge C(x) \ldots$ |
| Parundekar2010 [26] | Conjunction of attribute-value pairs | $\forall$x, $p$(x,$v$) $\wedge$ $p_1$(x,$v_1$) $\wedge \ldots \equiv$ $p_2$(x,$v_2$) $\wedge$ $p_3$(x,$v_3$) $\wedge$ ... |
| Boukottaya2005 [44] | string concatenation, subset merging | $p_1 = concat(p_2, p_3, \ldots)$ ; $p_1 = union(p_2, p_3, \ldots)$ |
| Xu2003 [41] | string concatenation, subset merging | $p_1 = concat(p_2, p_3, \ldots)$; $p_1 = union(p_2, p_3, \ldots)$ |
| Xu2006 [42] | string concatenation, subset merging | $p_1 = concat(p_2, p_3, \ldots)$; $p_1 = union(p_2, p_3, \ldots)$ |
| Dhamankar2004 (iMAP) [45] | string concatenation | $p_1 = concat(p_2, p_3, \ldots)$ |
| Warren2006 [46] | string concatenation of attribute substrings | $p_1 = concat(substr(p_2), substr(p_3), \ldots)$ |
| Arnold2013 (COMA++) [47] | string concatenation | $p_1 = concat(p_2, p_3, \ldots)$ |
| Wu2004 [53] | aggregate or is-a | $\{p_1\} = is-a\{p_2, p_3, \ldots\}$ ; $\{p_1\} = aggregate\{p_2, p_3, \ldots\}$ |
| Saleem2008 (PORSCHE) [51] | Bag of properties/blocks | $\{p_1\} = \{p_2, p_3, \ldots\}$ |
| He2004 (DCM) [50] | Bag of properties/blocks | $\{p_1, \ldots\} = \{p_2, p_3, \ldots\}$ |
| Su2006 (HSM) [52] | Bag of properties/blocks | $\{p_1, \ldots\} = \{p_2, p_3, \ldots\}$ |

*attribute-value* pair matched with an attribute and a disjunction of its acceptable values. In a first step, the approach finds correspondences between *attribute-value* pairs from the linked instances of the two ontologies (instances linked with *owl:sameAs* predicate). The number of instances sharing both attribute-value pairs defines if the correspondence has a subsumption or equivalence relation. A resulting correspondence is for instance: $\forall x, o_1{:}p_1(x,v_1) \sqsupseteq o_2{:}p_2(x,v_2)$. The second step of the approach is, for each subsumption correspondence of the previous step, to merge in a union all the attribute-pairs with a common attribute. The relation of the new correspondence is then re-evaluated according to the number of instances for each member. The final correspondence has the form $\forall x, p(x,v) \equiv p_2(x,v_2) \vee p_2(x,v_3) \vee \ldots$.

**Example 3.** *First step output:*

- $\forall x$, *rdf:type(x,*$o_1$*:AcceptedPaper)* $\sqsupseteq o_2$*:hasStatus(x, "accepted")*
- $\forall x$, *rdf:type(x,*$o_1$*:AcceptedPaper)* $\sqsupseteq o_2$*:hasStatus(x, "camera-ready")*

*Second step output:* $\forall x$, *rdf:type(x,*$o_1$*:AcceptedPaper)* $\sqsupseteq o_2$*:hasStatus(x, "accepted")* $\vee o_2$*:hasStatus(x, "camera-ready")*

*ARCMA, Kaabi et Gargouri [39] Ontology to ontology, (s:c)* Kaabi *et* Gargouri [39] propose ARCMA (Association Rules Complex Matching Approach) to find correspondences of the form $\forall x, o_1{:}A(x) \sqsubseteq o_2{:}B(x) \wedge o_2{:}C(x) \wedge \ldots$. A set of terms is associated with each class: the terms are extracted from the annotations, labels, instance values, instance labels of this class and its subclasses. The detection of the correspondences rely on existing simple correspondences: each class of the right member ($o_2$:B(x), $o_2$:C(x), ...) must be equivalent to a parent of $o_1{:}A$. The correspondences are then filtered based on a value measuring how the sets of terms of each member overlap.

*Parundekar et al. [26] Ontology to ontology, (s:c) (c:s) (c:c)* Parundekar *et al.* [26] look for conjunctions of *attribute-value* pairs, for instance correspondences of the form $\forall x, o_1{:}p_1(x,v_1) \wedge o_1{:}p_2(x,v_2) \wedge \ldots \equiv o_2{:}p_3(x,v_3) \wedge o_2{:}p_4(x,v_4) \wedge \ldots$ The approach starts with pre-processing the two knowledge-bases described by $o_1$ and $o_2$. Only the common instances are kept. Properties that cannot contribute to the alignment are removed. A set of first correspondences (the seed hypotheses) are created between *attribute-value* pairs. An example of a seed hypothesis is $\forall x, o_1{:}p_1(x,v_1) \equiv o_2{:}p_2(x,v_2)$. Starting from these seed hypotheses, the approach implements a heuristic in depth-first exploration of the search space (all the attribute-value pairs conjunctions). The search space is considered as a tree, the root being a seed hypothesis. Each node is an extended version of its parent: an attribute-value pair is added to one member of the parent. The search-tree is pruned following rules based on the variation of instances described by each member. For example if the attribute-value added in a node

is too restrictive or if the support of the ancestor node is the same as the current node, the children of the current node are not explored. The final set of correspondences is filtered to avoid redundancy. The number of instances of each member will determine the correspondence's relation.

*Boukottaya and Vanoirbeek [44] XML schema to XML schema, (s:c) (c:s) (c:c)* Boukottaya *et* Vanoirbeek [44] propose an XML schema matching approach based on the schema tree and linguistic layer of the schema. This approach finds simple correspondences as well as complex ones. The complex ones follow a few patterns such as merge/split, union/selection and join. The first step calculates a similarity between nodes of the source and target schemata. A linguistic similarity is calculated. A datatype similarity is then computed for the linguistically similar nodes. The union/selection and merge/split correspondences are detected based on graph-mapping. If a node *s:address* of the source schema with children leaf nodes (*s:street*, *s:city*) matches a leaf node *t:address* of the target schema, then a concatenation of the children nodes can be matched to the target node. The correspondences are filtered based on their *structural context*: ancestors and children nodes. The access path of each node is written in the final correspondences.

*iMAP, Dhamankar et al. [45] Relational database schema to relational database schema, (c:s)* As seen in the previous section, the iMAP system [45] uses a combination of *searchers* to discover simple and complex correspondences between database schemata. Some of the searchers, use composite patterns for correspondence detection. For instance, the text searcher looks for correspondences between an attribute from the target schema and concatenation of string attributes from the source schema. This searcher starts from ranking all possible simple correspondences between attributes. For that, a Naive-Bayes classifier is trained on the target data values to classify whether a given value can be from the target attribute, the average score given by this classifier to a correspondence is used for the ranking. Once the $k$ best simple correspondences are picked, the process is reiterated but with concatenations of each picked source attribute and each other source attribute as base correspondences. These new correspondences are scored, picked, and so on.

Another searcher implements a composite pattern search: the date searcher. It uses a date ontology as mediating schema containing date concepts (e.g. date, month, year, etc.) and the relations between them (e.g. concatenation, subset, etc.). The attributes of each schema are matched to the date ontologies' entities and the relations between them are reported as transformation functions in the resulting correspondence. The date ontology contains the composite patterns which are discovered by simple graph matching.

*COMA++, Arnold [47] Ontology to ontology, (s:c)* As an improvement of the COMA system [59], Arnold [47] discusses a solution based on a lexical strategy on the ontologies' labels: *n (s:s)* data-property correspondences with the same entity as target (or source), could be merged into a complex one. The initial approach generates simple correspondences with expressive relations such as meronymy *part-of* or holonymy *has-a* besides usual relations ($\sqsupseteq$, $\sqsubseteq$, $\equiv$). The extension for transforming the simple correspondences into a complex one can take into account the type of data-property (e.g. concatenation for string properties or sum for numeric properties). The following example shows a complex correspondence inferred from simple correspondences.

**Example 4.** *Correspondences with same member:*

- $o_1$*:firstName part-of* $o_2$*:fullName*
- $o_1$*:lastName part-of* $o_2$*:fullName*

*Aggregation in a new correspondence:*
*concat(*$o_1$*:firstName,* $o_1$*:lastName) =* $o_2$*:fullName*

*Xu and Embley [41, 42] Schema to schema, (s:c) (c:s)* Xu and Embley [41] propose a similar approach to iMap's date matcher. It uses a user-specified domain ontology as mediator between the two schemata to align. This ontology contains relations between concepts such as composition, subsumption, etc. It is populated thanks to regular expressions applied on source and target data. Simple correspondences (equivalence or subsumption) are first detected using *recognition of expected values* techniques between the source schema (resp. target) attributes and the ontology's concepts. These simple correspondences are kept for the next phase if the number of common values between the schema attribute and the ontology concept are above a threshold.

The relation between the ontology concepts in simple schema-ontology correspondences will become the transformation functions between the attributes they are linked to. For example, *s:street s:city* are two attributes from the source schema and *t:address* is an attribute from the target, schema. In the first matching phase, simple correspondences are drawn with concepts from the mediating ontology $o$:

- *o:Address = t:address*
- *o:Street = s:street*
- *o:City = s:city*

In *o*, the concept *o:Address* has a composition relation with the concepts *o:Street*, *o:City*. Therefore, the output complex correspondence will state that *t:address* is a string concatenation of *s:street* and *s:city*.

The later version of Xu and Embley's approach [42] completes this work with two new confidence calculations for simple attribute matching. The two new calculations do not consider a mediating ontology.

*Warren and Tompa [46] Database schema to database schema, (c:s)* Warren and Tompa [46] focus on finding correspondences between string columns of databases. They deal with correspondences that translate a concatenation of column sub-strings. The approach starts by ranking the columns according to the *q-grams* (sequence of *q* characters) of its values found in target column. Then it looks for matched instances (rows) according to a *tf-idf* formula on co-occurring q-grams. The column that has the smallest editing distance from the target column is put in an initial translation rule. This translation rule is then iteratively refined with addition of sub-strings from other columns.

*Wu et al. [53] Database schema to database schema, (s:c) (c:s)* Wu *et al.* [53] propose a clustering approach to find synonym alignments between database schemata based on web query interfaces. It considers the hierarchical structure of an HTML form. It also considers the values taken in the database rows as the *domain* of an attribute.

The first step consists in finding complex correspondences of the form *(s:c)* or *(c:s)* in which the attribute in the simple member is called the *singleton attribute* and the attributes in the complex member, the *grouped attributes*. Two types of correspondences are sought: *aggregate* and *is-a*. An *aggregate* correspondence shows a value concatenation: *{date}=aggregate{day,month,year}*. A *is-a* correspondence shows a union, sum, etc. of these values: *{passengers}={adults,children,seniors}*. The detection conditions of these correspondences are based on the taxonomy: the label of the parent node of the grouped attributes must be similar to the one of the singleton attribute. For *is-a*, the grouped attributes' domains must be similar to the singleton's one, whereas for *aggregate*, the domain of each grouped attribute must be similar to a subset of the singleton attribute's domain.

Then a clustering technique computes simple correspondences in a holistic manner between the interfaces. Simple correspondences and preliminary complex correspondences are merged. Other complex correspondences may be inferred from this merging phase. Even if the simple matching process is holistic, the detection of the complex correspondences is made one interface to one interface. Thus, the output correspondences are one schema to one schema.

The final step of the approach is user refinement. The system asks the user questions to refine the alignment and tune the parameters of the clustering algorithm and similarity calculation.

*PORSCHE, Saleem et al. [51] XML schema to XML schema, (s:c) (c:s)* PORSCHE (Performance ORiented SCHEma Matching) [51] matches a set of *schema trees* (schemata with a single root) at once. It is a holistic approach. This approach outputs a mediating schema (all the schema merged) as well as correspondences from each source schema to the mediating schema. An initial mediating schema is chosen among the source schema trees. It is then extended by the approach. For each node of each schema, the approach tries to find a corresponding node in the mediating schema. The tokenised labels of the nodes are compared with help of an abbreviation table. The context of a node is also taken into account for the merging, where the ancestors of the nodes must match. The pattern used for the detection of the complex correspondences is: if a non-leaf node $v_n$ is similar to a leaf node $v_l$, a (c:s) correspondence is created between $v_l$ and the leaf nodes descending from $v_n$. The produced correspondences are coherent (leaves with leaves) but approximate. Indeed, the context of a node is not checked in the case of a (s:c) leaf-non-leaf correspondence. No transformation function is specified in the correspondence. They come as bags of properties.

The two following approaches are also holistic: they match many schemata at once. They rely on web query interfaces for their matching.

*DCM, He et al. [50] Database schema to database schema, (s:c) (c:s) (c:c)* DCM (Dual Correlation Mining) [50] is a holistic schema matching system. It alignes database schemata attributes through the web query interfaces of these databases. It uses data-mining techniques (positive and negative correlation mining) on a corpus of web query interfaces to discover complex correspondences. The approach uses attribute co-occurrence frequency as a feature for the correlation algorithm. The first step of the algorithm is to mine

frequently co-occurring attributes from the web query interfaces. These attributes are put together as *groups* (e.g. *{firstName, lastName}*). In the second step, each set of co-occurring attributes (e.g. *{firstName, lastName}*) is put in correspondence with sets of attributes which do not often co-occur with them (e.g. *{author}*). The correspondences are then filtered based on their confidence (negative co-occurrence) value, or aggregated if they have a common attribute: if *{firstName, lastName}* = *{author}* and *{author}* = *{writer}*, then *{firstName, lastName}* = *{author}* = *{writer}*. As this approach is holistic, the correspondences are not limited to two members.

A holistic approach reduces the bias of one-to-one schema matching as errors can be overcome by the number of right correlations mined. However, only the attributes present on the web query interfaces can be involved in the correspondences.

*HSM, Su et al. [52] Database schema to database schema, (s:c) (c:s) (c:c)* HSM (Holistic Schema Matching) [52] is very similar to DCM [50] as it considers schema matching as a whole. It finds synonyms and grouping attributes based on their co-occurrence frequency and proximity in the web query interfaces. Two scores are computed between attributes : synonym scores (the confidence that two fields may refer to the same concept or *thing*) and grouping scores (confidence that two concepts are complementary to one-another). The algorithm then goes through the synonym scores in decreasing order and adds new correspondences to the alignment. If an attribute is a synonym of an attribute that was already involved in a correspondence, it may be grouped with other attributes according to its grouping score with them.

### 4.3. Path

A specificity of the path-based approaches is that they all rely on simple correspondences (at instance or schemata/ontology level). Some of them discover these simple correspondences themselves as a preliminary step [6, 38], others take them as input [28, 29, 32, 37]. Most approaches perform the path search on the graph-like or tree-like structure of the schemata/ontologies directly whereas [32] creates a *mapping graph* on which the search will be performed.

Most approaches of this survey look for path of undefined length in both ontologies/schema. An exception is [37].

*An et al. [37] Database schema to ontology, (s:c)* An *et al.* [37] use web query interfaces (web forms) to map a deep web database to an ontology. The web query interface must be transformed into a *form tree* (derived from HTML), similar to a schema tree. The algorithm takes the form tree, the ontology and simple correspondences between the form tree and the ontology as input. The first step of the algorithm is to find for each edge $e$ between nodes $u$ and $v$ of the form tree, all sub-graphs $G_i$ (as minimum spanning Steiner trees) in the ontology. The sub-graphs are property chains in the target ontology between two nodes (classes) $s$ and $t$ such that $u \approxeq s$ and $v \approxeq t$ are two simple correspondences given in the input. The goal of the algorithm is to output the most (or k-most) probable sub-graphs for the given form tree. To compute the probability of a sub-graph given a form tree, a model is trained with machine learning techniques. The training corpus is composed of web query interfaces annotated with the target ontology. The model is based on a Naive Bayesian approach and *m-estimate probabilities* to approximate the sub-graph probability given a form tree.

*Clio, Miller et al. [28], Yan et al. [29] Relational database schema to relational database schema, (s:c) (c:s) (c:c)* Based on structural information of relational databases schemata, the Clio system [28, 29] is one of the first system to consider the creation of complex correspondences between schemata. The user must input *value correspondences*: functions linking one or many attributes (e.g. *Parent1.Salary* + *Parent2.Salary* $\rightarrow$ *Student.FamilyIncome* with *Parent1* and *Parent2* in the source schema and *Student* in the target schema). Used for populating target schemata with source data, it provides the user with a framework for alignment creation. Clio discovers formal queries from these *value correspondences*. The formal queries are defined step-by-step with the user by presenting her potential *query graphs* between attributes: trees from the data source schema structure. Clio helps the user find simple, path relations and value transformations correspondences with data visualisation, data walk and data chase. The alignments are automatically transformed into SQL queries. The SQL queries transform the source data into target schema. The user can refine and extend the alignments (queries) with filters and joins.

*Ontograte, Qin et al. [38], Dou et al. [6] Ontology to ontology, (c:c)* OntoGrate [38] is a framework that mines *frequent queries* and outputs them as conjunc-

tive first-order logic formulas. The system can deal with ontology matching [38] and was adapted to relational database schema matching in [6] by transforming the database schema into a *database ontology*. In OntoGrate, the first step of the matching algorithm is to *generate simple correspondences* at ontology level. An *object reconciliation* phase then aligns instances from source and target knowledge bases. The instance correspondences from the object reconciliation fuel the simple correspondences generation. The algorithm iterates on both steps (simple correspondences generation and object reconciliation) until no new instance matching or simple correspondence is discovered. Once the simple correspondences are found, a *group generator* process generates groups of entities so that the groups have the same semantics. The group generation is done by exploring the ontology graph and finding a path between entities (e.g. classes) linked by a simple property/property correspondence (the property/property correspondence can be data-property/data-property or object-property/object-property). The path finding algorithm is an exploration algorithm of the two ontology graphs where classes are the nodes and properties (object properties, data properties, subclass relations and super-class relations) are the edges. The ontology graphs are explored until two nodes, one in the source path and one in the target path, are found and were matched in the first steps of the matching process. The final steps of the matching process is Multi-Relational Data Mining (MRDM) to retrieve *frequent queries* among the matched instances for the given *entity groups*. If the support of a query is above a threshold, the query is considered *frequent* and kept. The frequent queries are then refined and formalised into first-order logic formulae.

*An and Song [32] Conceptual model to conceptual model, (c:c)* An and Song [32] introduce the concept of mapping graph between two conceptual models. The nodes of a mapping graph represent pairs of concepts from the two conceptual models. For example $(x_1, x_2)$ and $(y_1, y_2)$ are two nodes of the mapping graph, $x_1$ and $y_1$ being classes (concepts) of a source ontology and $x_2$ and $y_2$ two classes of a target ontology. The weighted edges of the mapping graph are defined according to the presence and nature of the relations between the concerned concepts in the conceptual models. Once the mapping graph is generated, a Dijkstra algorithm is used to find the smallest path (with maximum weights) between nodes that appear in an input simple alignment. For example if $x_1 \equiv x_2$ and $y_1 \equiv y_2$ are two correspondences in an input alignment, a path between the nodes $(x_1, x_2)$ and $(y_1, y_2)$ of the mapping graph will be sought.

### 4.4. Tree

While some approaches [34, 35] rely on a *semantic tree* derived from the schema, others [48, 49] use trees as base for constructing functions. The approaches focusing on structural transformations between two trees (addition of a node, deletion of an attribute, etc.) such as [7, 19] often rely on tree-structure. However, they are out of the scope of this study as they are part of the ontology evolution field.

*MapOnto, An et al. [34, 35] Relational database schema to ontology [34], XML schema to ontology [35], (c:c)* MapOnto [34, 35], a work of An *et al.* is inspired from Clio in terms of path finding and tree construction. The approaches focuses on aligning a source schema to a target ontology. Two approaches were proposed: a relational database schema to ontology [34] and a XML schema to ontology [35]. Both approaches take simple correspondences between the schema's attributes and the ontology's data-properties as input. These matching techniques construct a conjunctive first-order formula composed of target ontology entities to match a table (relational database) or *element trees* (XML) from the source schema. The production of the logical formula (presented as a *semantic tree* in [34]) differs between the two approaches because of the different nature of the schemata. However, both approaches look for the smallest tree representing the attribute of the schema. A set of the most "reasonable" alignments are output for the user to choose among. These techniques output (c:c) correspondences as a whole table (or *element tree*) is transformed in each correspondence.

*Nunes et al. [48] Ontology to ontology, (c:s)* Genetic programming is a way of finding complex correspondences between data properties. It can combine and transform the data-properties of an ontology to match a property of an other ontology. Nunes *et al.* [48] propose a genetic programming approach for numerical and literal data properties matching. The correspondences generated are (c:s) as *n* data-properties from the source ontology are combined to match a target data-property. The source data-properties are chosen from a calculated estimated mutual information (EMI) matrix. Each "individual" of the genetic algorithm is a tree representing the combination operations

over data properties. The elementary operations used for combination are concatenation or split for literal data-properties and basic arithmetic operations for numerical data-properties (sum, multiplication, etc.). The fitness of a solution is evaluated on the values given by this solution and the values expected (based on matched instances) using a Levenshtein distance.

*De Carvalho et al. [49] Schema to schema (relational database or XML), (s:c) (c:s) (c:c)* De Carvalho *et al.* [49] apply the genetic algorithm to alignments as its "individuals". Each "individual" is a set of correspondences. Each correspondence is a pair of *tree functions* made of elementary operations (as for Nunes *et al.* [48]) and having source (resp. target) attributes as leaves. Constraints over the correspondences have been defined: a schema attribute cannot appear more than once in a correspondence, crossover and mutation can only be applied to attributes of the same data type, the number of correspondences in an alignment is fixed a priori. Mutation and cross-over operations occur at the correspondence's tree-level when parts of two *tree functions* are swapped, or changed. The fitness evaluation function of the schema alignments (individuals) is the sum of the fitness score of its correspondences. The fitness score of a correspondence can be calculated in two ways: *entity-oriented* with the average similarity of matched instances' attributes transformation (matched instances of overlapping data are needed) or *value-oriented* with the similarity of all transformed source instances and target instances. The similarity for each correspondence is chosen by an expert. Compared to the approach of Nunes et al. [48] it can detect *(c:c)* correspondences thanks to its modelling. However the process may require more iterations than [48].

### 4.5. No structure

The approaches described in this section do not follow any of the above structures. While [33] is based on Inductive Logic Programming and builds its correspondences in a *ad. hoc* manner, [43] uses Markov Logic Networks for combinatorial exploration, [54] uses classifying techniques to generate block correspondences, and finally [45] overlaps numeric searcher using context-free grammar for equation discovery.

*Hu et al. [33] Ontology to ontology, (s:c)* The approach proposed by Hu *et al.* [33] uses Inductive Logic Programming (ILP) techniques to discover complex alignments. This technique is inspired by Stuckenschmidt *et al.* [60]. The approach is based on the common instances of a source and a target ontology. It outputs Horn-rules of the form $A \wedge B \wedge C \wedge ... \rightarrow D$ with $A, B, C...$ source entities represented as first-order predicates and D a target entity as a first-order predicate. The Horn-rule contains two parts: the body on the left side of the implication and the head on the right side. Three phases compose the approach. In the first one, the instances of the two ontologies are matched. In the second one called *data-tailoring*, instances and attributes from their context (relations, data-properties, other linked instances, etc.) are chosen for each target entity. The purpose of this phase is to eliminate irrelevant data. The last phase is the *mapping learning* phase. For each target entity, a new Horn-rule is created with this target entity as head predicate. Then iteratively, the predicate having the highest FOIL gain score is added to the body of the Horn-rule. During this process, the variables of the Horn-rule are bound according to the instances and their context. The FOIL gain uses the positive and negative binding of variables.

*BMO, Hu et al. [54] Ontology to ontology, (s:c) (c:s) (c:c)* BMO (Block Matching for Ontologies) focuses on matching sets of entities (classes, relations or instances) called blocks. This approach is articulated into four steps. The first step is the construction of virtual documents for each entity of both ontologies: the annotations and all triples in which an entity occurs are gathered into a "document". The second one computes a *relatedness matrix* by calculating the similarity between each vectorized virtual document. In the third step, the *relatedness matrix* is used to apply a partitionning algorithm: this algorithm is recursively applied to the set of ontology entities. At the end of this algorithm, the similar entities are together in a same block while dissimilar entities are in distinct blocks. The final step consists in finding the optimal alignment given a number of blocks. Ontology entities which are in the same block can be separated into $o_1$ and $o_2$ to get a correspondence. As the blocks can contain any type of entity, it is not considered as a composite pattern.

*KAOM, Jiang et al. [43] Ontology to ontology, (s:c) (c:s) (c:c)* KAOM (Knowledge Aware Ontology Matching) is a system proposed by Jiang *et al.* [43]. It uses Markov Logic Network as a probabilistic framework for ontology matching. The Markov Logic formulae presented in this approach use the entities of the two ontologies (source and target) as *constants*, the relations between entities and the input *knowledge rules* as *evidence*. The *knowledge rules* can be axioms of an

ontology or they can be specified by the user. They can be interpreted as *block matching patterns* [55]. To handle numerical data-properties, KAOM proposes two methods to find positive linear transformations between rules. These methods are based on the values that the data-properties take in a given knowledge base (the distribution of the values or a way to discretise them). The correspondence patterns and conditions presented by Ritze *et al.* [23, 24] can be translated into knowledge rules and therefore used into Markov Logic formulae. The *knowledge rules* can be obtained in various ways as was shown in the experiments where decision trees, association rules obtained from an Apriori algorithm or manually written rules were translated as *knowledge rules* for three different test cases.

*iMAP, Dhamankar et al. [45] Relational database schema to relational database schema, (c:s)* As seen previously, the iMAP system [45] uses a combination of *searchers* to discover simple and complex correspondences between database schemata. The overlap numeric searcher uses the Lagramge algorithm for equation discovery based on overlapping data. This algorithm uses a contex-free grammar to define the search space of the arithmetic equations and executes a beam-search to find a suitable correspondence. The output of this search space is then stored as a pattern for the numeric searcher.

### *4.6. Summary*

The proposed classification is based on two main axis, the output (type of correspondence) and process (guiding structure) dimensions of the approaches. The following tables are organised as the categories of Figure 4: logic relations, transformation functions and blocks.

Table 3 summarises the type of knowledge representation models to be aligned, the needed input and the kind of generated correspondences. Most of the approaches generate (s:c) or (c:s) correspondences and require inputs (simple alignments, matched instances, etc.). This table shows the variety of knowledge representation models for which complex matching approaches have been proposed. This points out that complex matching is not dedicated only to Semantic Web.

Table 4 presents the process of the approaches according to our classification. Most approaches are pattern-based (atomic or composite). Only a few approaches have no guiding structure. There is no direct correlation between the members expression (fixed to fixed, unfixed to unfixed, etc.) and the (s:c), (c:s) kinds of correspondence. Very few approaches are available online.

With respect to the basic techniques defined in [13], presented in Table 5, the majority of approaches combine different matching strategies. By definition, approaches which output logical relation correspondences and which rely on a guiding structures have a graph-based or taxonomy-based component. Few approaches are model-based (no semantic interpretation of the alignment). However, it is important to note that identifying the strategies based on Euzenat and Shvaiko's classification was not always straightforward.

Another way of classifying the approaches is with respect the kind of input they exploit (ontology-level or instance-level), as done in different surveys in the field. Figure 5 presents a classification of the approaches on their input. Only a few approaches rely on both types of information.

## 5. Evaluation of complex matchers

Some surveys [65] have focused the comparison of ontology matching systems evaluation not addressing the complex matching perspective. This section discuss the evaluation of complex alignments regarding both the metrics and datasets. Table 6 provides an overview of the type of evaluation, metrics and datasets used to evaluate the approaches described in the previous section.

### *5.1. Complex alignment datasets*

The diverse approaches discussed in this paper exploit a variety of knowledge representation models and resources (XML, ontologies, linked instances, web forms, etc.) and generates different types of correspondences. This results in a variety of evaluation datasets used for evaluating these approaches and their outputs. In the domain of schema matching (database or XML schema), dedicated complex alignment datasets have been constructed for evaluating the approaches dealing with these schemes. In general, these datasets contains mostly transformation functions. For instance, the Illinois semantic integration archive [62] is a dataset of complex correspondences on value transformations (e.g. string concatenation) in the inventory and real estate domain. This dataset only contains correspon-

| | *Work* | *Type of schemata* | *Input* | *Kind of correspondence* |
|---|---|---|---|---|
| Logical relations | Ritze2009 [23] | Onto to Onto | simple alignment | (s:c) |
| | Ritze2010 [24] | Onto to Onto | simple alignment (opt.) | (s:c) |
| | Svab2009 [31] | Onto to onto | simple alignment (opt.) | (s:c) |
| | Rouces2016 [36] | Onto to Onto | | (s:c), (c:s) |
| | Walshe2014 (Bayes-ReCCE) [25] | Onto to Onto | matched instances | (s:c), (c:s) |
| | Jimenez2015 (BootOX) [40] | DB to Onto | | (c:s) |
| | Doan2003 (CGLUE) [30] | Taxo to Taxo | | (s:c) |
| | Parundekar2012 [27] | Onto to Onto | matched instances | (s:c), (c:s) |
| | Kaabi2012 (ARCMA) [39] | Onto to Onto | | (s:c) |
| | Parundekar2010 [26] | Onto to Onto | matched instances | (s:c), (c:s), (c:c) |
| | Yan2001 (Clio) [28, 29] | DB to DB | matched instances | (s:c), (c:s),(c:c) |
| | Qin2007 (OntoGrate) [6, 38] | Onto to Onto | matched instances | (c:c) |
| | An2008 [32] | CM to CM | | (c:c) |
| | An2012 [37] | DB to Onto | web query interfaces, simple correspondences web form-onto | (s:c) |
| | An2005 (MapOnto) [34] | DB to Onto | attribute-data properties correspondences | (c:c) |
| | An2005b (MapOnto) [35] | XML to Onto | attribute-data properties correspondences | (c:c) |
| | Hu2011 [33] | Onto to Onto | | (c:s) |
| Logic/Transfo | Jiang2016 (KAOM) [43] | Onto to Onto | knowledge rules | (s:c), (c:s), (c:c) |
| | Boukottaya2005 [44] | XML to XML | | (s:c), (c:s), (c:c) |
| | Xu2003 [41] | Schema to Schema | domain ontology | (c:s), (s:c) |
| | Xu2006 [42] | Schema to Schema | domain ontology | (c:s), (s:c), (c:c) |
| Transfo. functions | Dhamankar2004 (iMAP) [45] | DB to DB | domain constraints and value distribution | (c:s) |
| | Arnold2013 (COMA++) [47] | Onto to Onto | | (s:c) |
| | Warren2006 [46] | DB to DB | | (c:s) |
| | Nunes2011 [48] | Onto to Onto | | (c:s) |
| | deCarvalho2013 [49] | Schema to Schema | | (c:s), (s:c), (c:c) |
| Blocks | Saleem2008 (PORSCHE) [51] | XML to XML | abbreviation table | (c:s), (s:c) |
| | He2004 (DCM) [50] | DB to DB | web query interfaces | (s:c), (c:s), (c:c) |
| | Su2006 (HSM) [52] | DB to DB | web query interfaces | (s:c), (c:s), (c:c) |
| | Wu2004 [53] | DB to DB | web query interfaces | (s:c), (c:s) |
| | Hu2006 (BMO) [54] | Onto to Onto | | (s:c), (c:s), (c:c) |

Table 3

Input (type of aligned schemata and type of input information) and output (correspondences members form) of the approaches.

| | Article | Guiding structure | fixed to fixed | fixed to unfixed | unfixed to unfixed | Online |
|---|---|---|---|---|---|---|
| Logical relations | Ritze2009 [23] | Atomic patterns | • | | | yes[3] |
| | Ritze2010 [24] | Atomic patterns | • | | | yes[4] |
| | Svab2009 [31] | Atomic patterns | • | | | |
| | Rouces2016 [36] | Atomic patterns | • | | | |
| | Walshe2016 (Bayes-ReCCE) [25] | Atomic patterns | • | | | |
| | Jimenez2015 (BootOX) [40] | Atomic patterns | • | | | |
| | Doan2003 (CGLUE) [30] | Composite patterns | | • | | |
| | Parundekar2012 [27] | Composite patterns | | • | | |
| | Kaabi2012 (ARCMA) [39] | Composite patterns | | • | | |
| | Parundekar2010 [26] | Composite patterns | | | • | |
| | Yan2001 (Clio) [28, 29] | Path to Path | | | • | |
| | Qin2007 (OntoGrate) [6, 38] | Path to Path | | | • | yes[5] |
| | An2008 [32] | Path to Path | | | • | |
| | An2012 [37] | Path to Path | | • | | |
| | An2005 (MapOnto) [34] | Tree to tree | | | • | yes[6] |
| | An2005b (MapOnto) [35] | Tree to tree | | | • | yes[7] |
| | Hu2011 [33] | No structure | | • | | |
| Logic/Transfo | Jiang2016 (KAOM) [43] | Atomic patterns, No structure | • | | • | |
| | Boukottaya2005 [44] | Composite patterns | | • | | |
| | Xu2003 [41] | Composite patterns | • | • | | |
| | Xu2006 [42] | Composite patterns, Path to path | • | • | • | |
| Transfo. functions | Dhamankar2004 (iMAP) [45] | Atomic patterns, Composite patterns, No structure | • | • | | |
| | Arnold2013 (COMA++) [47] | Composite patterns | | • | | |
| | Warren2006 [46] | Composite patterns | | • | | |
| | Nunes2011 [48] | Tree to tree | | • | | |
| | deCarvalho2013 [49] | Tree to tree | | | • | |
| Blocks | Saleem2008 (PORSCHE) [51] | Composite patterns | | • | | |
| | He2004 (DCM) [50] | Composite patterns | | | • | |
| | Su2006 (HSM) [52] | Composite patterns | | | • | |
| | Wu2004 [53] | Composite patterns | | • | | |
| | Hu2006 (BMO) [54] | No structure | | | • | |

Table 4

Process characteristics of the approaches based on the proposed classification

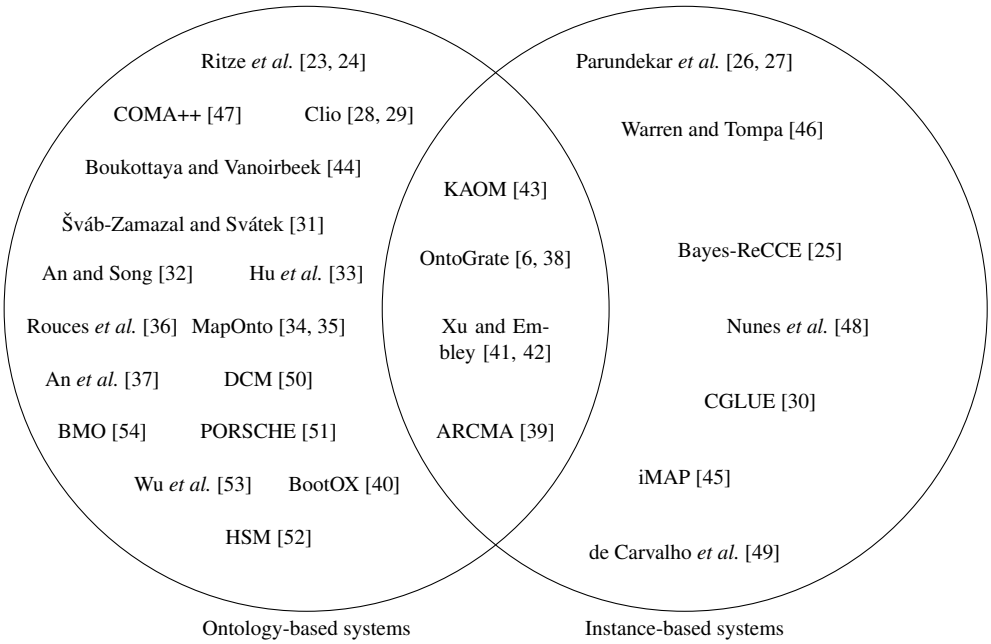



Fig. 5. The complex matching techniques according to the instance vs ontology (schema)-level classification

dences between schemata with transformation functions. The UIUC Web integration Repository [64] has also been reused by various schema matching approaches for their evaluation. It is a schemata and query form repository. Other repositories has been adapted for the purpose of evaluating schema matching approaches. In [61], a data set dedicated to evaluating machine-learning matching-based approaches has been proposed.

For the purpose of evaluating matching hybrid structures, the RODI Benchmark [63] proposes an evaluation over given scenarii, R2RML correspondences between a database schema and an ontology. The benchmark relies on ontologies from the OAEI Conference dataset.

Despite the fact that many approaches have been automatically evaluated, supposing the existence of a reference alignment, with respect to ontology matching, few reference alignment sets are publicly available.

### 5.2. Evaluation metrics

Evaluating complex alignments is usually done manually. However, it is a time-consuming task requiring experts in the ontology domain. Systems can also be evaluated automatically by considering a reference dataset usually composed of schemata or ontologies and reference alignments between them. The difficulty here is to produce such reference alignments. The classical metrics, adapted from information retrieval, used for evaluating the quality of alignments with respect a reference one are precision and recall, combined into F-measure. These metrics in their usual form are used in many evaluations, as shown in Table 6. In particular, the calculation of the recall and F-measure require a reference dataset.

The metrics of *accuracy* or *top-x accuracy* have been used in various evaluations (c.f. Table 6) when the number of correspondences is predefined, e.g., one correspondence for each entity of the target schema/ontology. The accuracy is then the percentage of correct answers on a defined number of questions. Some approaches output various answers for each questions, e.g., a ranked list of correspondences for each target entity. In this case the top-$x$ accuracy is the percentage of questions whose correct answer is in the top-$x$ answers to the question. For example, top-3 accuracy is the fraction of target entities for which the correct correspondence is in the three best correspondences output by the system.

A few systems define their own metrics, more closely related to the type of correspondence they deal with. For example, the approaches dealing with synsets [50, 52] have adapted precision and recall to incorporate the synset similarity. The approach BMO [54] evaluates characteristics of the correspondences

| | Approach | Formal resource-based | Informal resource-based | String-based | Language-based | Constraint-based | Taxonomy-based | Graph-based | Instance-based | Model-based |
|---|---|---|---|---|---|---|---|---|---|---|
| Logical relations | Ritze2009 [23] | | | • | • | • | • | • | | |
| | Ritze2010 [24] | | | • | • | • | • | • | | |
| | Svab2009 [31] | | | • | | | | • | | |
| | Rouces2016 [36] | • | | • | • | | | • | | |
| | Walshe2016 (Bayes-ReCCE) [25] | • | | | | | | • | • | |
| | Jimenez2015 (BootOX) [40] | | | | • | • | | • | | |
| | Doan2003 (CGLUE) [30] | | | • | | | • | | • | |
| | Parundekar2012 [27] | • | | | | | | • | • | |
| | Kaabi2012 (ARCMA) [39] | | | • | | | • | | • | |
| | Parundekar2010 [26] | • | | | | | | • | • | |
| | Yan2001 (Clio) [28, 29] | | | | | • | | • | • | |
| | Qin (OntoGrate) [6, 38] | | | | | | | • | • | |
| | An2008 [32] | • | | | | | • | • | | |
| | An2012 [37] | | • | | | | | • | | |
| | An2005 (MapOnto) [34] | • | | | | | | • | | |
| | An2005b (MapOnto) [35] | • | | | | | | • | | |
| | Hu2011 [33] | | | | | | | | • | |
| Logic/Transfo | Jiang2016 (KAOM) [43] | • | | • | | • | | | • | • |
| | Boukottaya2005 [44] | | | | • | • | • | • | | |
| | Xu2003 [41] | • | | • | | | | • | • | |
| | Xu2006 [42] | • | | • | • | | | • | • | |
| Transfo. functions | Dhamankar2004 (iMap) [45] | • | | • | | • | | • | • | |
| | Arnold2013 (COMA++) [47] | | | • | | | • | | | |
| | Warren2006 [46] | | | | | | | | • | |
| | Nunes2011 [48] | | | | | | | | • | |
| | deCarvalho [49] | | | | | | | | • | |
| Blocks | Saleem2008 (PORSCHE) [51] | | | | • | | • | | | |
| | He2004 (DCM) [50] | | • | | | | | | | |
| | Su2006 (HSM) [52] | | • | | | | | | | |
| | Wu2004 [53] | | • | • | • | • | • | | • | |
| | Hu2006 (BMO) [54] | | | • | | | | • | | |

Table 5
Classification of the complex matchers on [13]'s basic techniques

as blocks such as the entropy and correctness of the blocks.

Some approaches have been evaluated as a comparison with other approaches or on their execution time.

### 5.3. Summary

The different approaches discussed in this survey have been evaluated on a variety of datasets. For many of them, the datasets have been constructed to the specific purpose of evaluating the given approaches. Furthermore, most of these datasets result from extensions of original datasets in order to fit the complex matching. There still is no benchmark on which complex ontology matching approaches can be systematically evaluated and compared. In fact, the evaluation of complex matching approaches has not yet been studied as a whole problem. The usual metrics appear insufficient to capture for example, that the same complex correspondence can be expressed in differ-

Table 6
Evaluation of the complex matching approaches

| *Work* | *Type* of *Evaluation* | *Metrics* | *Dataset* |
|---|---|---|---|
| Walshe2016 (Bayes-ReCCE) [25] | Automatic | Precision, Recall, F-measure (on synthetic dataset) | Complex correspondences between Dbpedia and Yago2 / geonames + synthetic dataset made from dbpedia (on CAV) |
| Jiang2016 (KAOM) [43] | Automatic | Precision, Recall, F-measure | NBA, Census [61], extended OAEI conference dataset with complex correspondences |
| Xu2003 [41] | Automatic | Precision, Recall, F-measure | Schemas transformed into Conceptual model graphs: Real estate dataset [62] |
| Xu2006 [42] | Automatic | Precision, Recall, F-measure | Real Estate, Course schedule and Faculty datasets transformed into CMs [62] |
| Wu2004 [53] | Automatic | Precision, Recall, F-measure (s:c correspondences counted as many s:s correspondences) | 20 web query interfaces from invisibleweb.com and yahoo.com transformed into schema trees |
| Hu2006 (BMO) [54] | Automatic | Precision, Recall, F-measure, Entropy, Correctness | 2 datasets: russia, tourism with reference block alignments |
| Jimenez2015 (BootOX) [40] | Automatic | Accuracy | RODI Benchmark [63] |
| He2004 (DCM) [50], Su2006 (HSM) [52] | Automatic | Target precision, Target recall, Target accuracy | TEL-8 and BAMM [64] |
| Parundekar2012 [27] | Manual / Automatic | 1. Precision on a subset of correspondences 2. Precision, Recall, F-measure on subset of ontologies | See [8] |
| Ritze2009, Ritze2010 [23, 24] | Manual | Precision | OAEI Conference and OAEI benchmark |
| Rouces2016 [36] | Manual | Precision | Dbpedia-Framebase |
| Hu2011 [33] | Manual | Precision | Only 6 complex correspondences found on the 3 datasets : Restaurant (OAEI), SwetoDBLP/ESWC, Mondial/Factbook |
| Warren2006 [46] | Manual | Precision | Small datasets with 1 correspondence (userid, time, name concatenation), Citeseer-DBLP |
| Qin2007 (Ontograte) [38] | Manual | Precision, Recall | UMD People Ontology and CMU Person and Employee Ontology |
| Dou2010 (Ontograte) [6] | Manual | Precision, Recall, Execution time | Store7 database, NBA schemata transformed into ontologies (Official site[9] and Yahoo Sport Site [10]), Mouse and zebrafish gene DB |
| An2008 [32] | Manual | Precision, Recall, F-measure, Execution time | GeneExpress Data Management + Other CM pairs from previous paper |
| An2012 [37] | Manual | Top-1 accuracy, top-k accuracy | Form libraries [64], VSO, bibliographic data ontology, movie ontology, extension of the GoodRelations ontology |
| An2005 (MapOnto) [34] | Manual | Qualitative evaluation, Execution time | See [11] |
| An2005b (MapOnto) [35] | Manual | Precision, Recall, F-measure, Labor Savings | See [12] |
| Doan2003 (CGLUE) [30] | Unknown | Accuracy | Course Catalog I and II (Cornell and Washington) [62] , Company Profile (ontologies from Yahoo.com and TheStandard.com) |
| Boukottaya2005 [44] | Unknown | Precision, Recall, F-measure | Bibliographic data description schemas |
| Nunes2011 [48] | Unknown | Precision, Recall, F-measure | Created target ontology from source ontology |
| deCarvalho2013 [49] | Unknown | Accuracy | Synthetic datasets from SDG (Synthetic Dataset Generator), datasets from [62], Google Fusion Tables |
| Dhamankar2004 (iMAP) [45] | Unknown | Top-1 accuracy, Top-3 accuracy | Real Estate, Inventory, Cricket (cricinfo.com vs cricketbase.com), Financial wizard |
| Parundekar2010 [26] | Unknown | Statistics on the output alignments | See [13] |
| Saleem2008 (PORSCHE) [51] | Unknown | Execution time, Comparison with COMA++ | Purchase order schemas, BOOKS, OAGIS |
| Svab2009 [31], Kaabi2012 (ARCMA) [39], Yan2001 (Clio) [28, 29], Arnold2013 (COMA++) [47] | | | None |

ent ways. The use of weighted, relaxed and semantic precision/recall could be an interesting solution [66]. First, precision and recall metrics do not take into account the confidence and relations of the correspondences nor the equivalence of two correspondences expressed in different ways. For these matters, weighted and relaxed precision and recall [67] consider the relation, and confidence of a correspondence. Semantic precision and recall [66] compare the deductive closure sets of axioms of the ontologies merged with a reference alignment and with the output alignment. Therefore, these metrics integrate the fact that two expressions may mean the same thing (e.g., $Author \equiv \exists authorOf.\top$ is the same as $Author \equiv \exists writtenBy^{-}.\top$). However, these metrics are not used in complex ontology alignment evaluation so far. Finally, one could also consider to measure the suitability of the output alignment for a given application as it was done for the OA4QA track of the OAEI [68] or for a library application [69]. The metrics would then take into account the suitability of the output alignment to the given task.

## 6. Discussion

Earlier works in the field have introduced the need for complex alignments [2]. With the explosion of LOD datasets this need becomes even more evident. Many works have been proposed so far in the literature, in different fields (databases, semantic web). This is a complex task, even manually, that requires to deeply identify the context of ontology entities. This section summarises the findings in this survey and discusses potential directions to fostering the development in the field.

Here, an overview of the approaches for generating complex correspondences has been presented. These works, due to their nature and scope, exploit representation models with different levels of expressiveness (XML, web forms, database schemes, ontologies) and, as a consequence, generate complex alignments mostly "compatible" with their models and query languages (logical constructions for ontology and transformations functions for XML, web queries, etc.). Few works focus on block-based matching. Furthermore, there is a clear distinction between the approaches based on instance-level information and the ones based on ontology-level information (terminological level, schema structure, etc.). Hybrid solutions have been addressed to a lesser extent. One can observe as well the variety of approaches (patterns, statistical-based, genetic programming, etc.) that have been adopted in these proposals.

The field could be organised into two 'classes' of matching approaches, those relying on defined structures such as atomic patterns, composite patterns and paths, and those which discover the correspondences more freely. For the former, mostly classical patterns have been adopted (CAT, CAV, etc.), while automatically learning new patterns is mostly underexploited. For the latter, the quality of the output alignments depends on the quality of the input they rely on (e.g., corpus of instances, corpus of web forms, set of knowledge-rules, etc.). These approaches may be faced with sparseness problem, frequent attribute problem or with specific corpus distribution that may lead to false correspondences (cf. same problem as Example 5):

**Example 5.** *If $o_1$ is populated with most students of age 23,*
*$\forall$ x, $o_1$:Student(x) $\equiv$ $o_2$:age(x, 23) can be a valid correspondence for the instance-based matching algorithms.*

While some approaches rely on simple correspondences as an input resources, others are able to discover complex correspondences without this kind of input. Another aspect refers to the kind of relation of a correspondence generated. As for simple alignments, most works are still limited to generating equivalences. The semantics of the confidence of a correspondence as well are rarely considered.

The proposed classification tried to capture some of the aspects above, by focusing on the specificities of complex correspondences on two main axis. The first axis characterises the different types of output (type of correspondence) and the second one the structures used in the process to guide the correspondence detection. With respect to this classification, while some approaches adopt a *mono-strategy* (atomic patterns, for instance), others can fall in diverse categories. For some of them, the task of classifying them into some specific category was not a simple task. This classification could hence be extended with a "hybrid" category for both axis. However, this would make reading difficult.

The classification on the type of correspondence can also be extended. In particular, correspondences can be of both logical and transformation function types For example, $\forall x, y_1, y_2$ *Person(x)* $\wedge$ *name(x, concatenation($y_1$, " ",$y_2$))* $\equiv$ *firstName(x,$y_1$)* $\wedge$

*lastName(x,y*$_2$*)* contains both logical constructors and transformation functions.

Other characteristics of the alignments could be a base for classification such as the nature of the correspondence relation (subsumption, implication, disjointness, etc.), the formalism of the alignment (query, first-order logic, etc.), the purpose of the alignment (query rewriting, ontology merging, etc.), the user involvment in the matching process, etc. One could also argue that one could dispose the different axis of our classification in a multi-layer classification as in [70]. However, this does not make sense here because our layers are not fully independent of each other, as in [70].

On a different matter, one can observe that (in matching web query interfaces, for instance) the correspondences are pragmatically coherent but not semantically equivalent. For example, *number of tickets* = *adult* + *children* + *senior* is a practical correspondence for counting the number of passengers. The semantic meaning of this correspondence is however questionable as a ticket and a passenger are not quite the same thing. This raises questions about the notion of context of an alignment.

Few approaches here are able to match many schemata at once. They rely on web query interfaces for their matching (schemas and query interfaces). However, in complex domains where several ontologies describing different but related aspects of the domain have to be linked together, matching multiple ontologies simultaneously, known as *holistic matching*, is required. It is typically the case in complex domains, such as biomedicine, where several ontologies describing different but related phenomena have to be linked together [71]. As stated in [72], the increase in the matching space and the inherently higher difficulty to compute alignments pose interesting challenges to this task. Generating complex correspondences in a holistic setting is in the next generation of complex ontology matching.

One can also observe that few approaches involve an interactive task where the user could provide some seeds for the matching process. Moreover, a correspondence can be expressed in various ways, a user might prefer one expression. For now, no approach proposes a framework for visualisation and edition of complex correspondences. This could be a lead for the development of the field and a solution for user-involvement. As most matchers are black boxes to the end users, explanation of complex correspondences could be another interesting direction.

Another important aspect in (complex) ontology matching is the evaluation of approaches. It helps developers to improve their approaches and users to choose the most suitable one for their needs or tasks. While the evaluation of simple ontology matching has developed fully in the last fifteen years in the context of the Ontology Alignment Evaluation Initiative (OAEI) [73], to the best of our knowledge, there is no benchmark for evaluating complex correspondences. One can argue that proposing such kind of data set could help foster the development of the field. While the approaches discussed have been evaluated on different data sets, which mostly have extended raw data sets in order to fit the evaluation of the approaches, they have not been exposed as a benchmark (in fact, few of them are publicly available, this is as well the case for most complex matchers). As stated above, a complex correspondence can be expressed in different ways. Thus, the need for several annotators and a measurement of their agreement in order to generate a consensual benchmark is evident. The ways for computing the confidence can rely on such consensus. Furthermore, representing the different ways a correspondence can be expressed may reduce the bias in the evaluation and avoiding privileging approaches which fit a particular type of correspondence.

Having a benchmark for complex evaluation also requires methods for evaluating the generated alignments on this benchmark. This opens different perspectives in terms of (automatic) evaluation. While the manual evaluation of the generated alignments or of sample of them is a time-consuming task that depends on the interpretation of the expert, the classical metrics of precision and recall are not adapted to complex correspondences. A more suitable version of precision and recall could be extending the semantic precision and recall proposed in [66], which is based on the deductive closure axiom sets of the reference and generated alignments. Evaluating the approaches in a task-oriented way could be also interesting, as for instance ontology merging and query rewriting. For ontology merging, a metric using the complex correspondences could take into account the conservativity, the coherence and the decidability of the merged ontology. For query rewriting, having a populated benchmark and gold standard queries, as for the OA4QA campaign [74], could be exploited for evaluating the correspondences in a query rewriting setting.

As a more general reflection, questions on pragmatic and semiotic heterogeneity [75] are still open. Even if

the output of the approach are useful in practice, one can still wonder whether they are absolute [76].

## 7. Conclusions

Complex alignments are complementary to simple alignments for overcoming conceptual heterogeneities between ontologies. This survey have provided an overview of the different complex matching approaches. This covers approaches deadling with models of different levels of expressiveness (from XML to databases and ontologies). A definition of complex correspondences (and alignments) has been proposed. A classification of these approaches based on their specificities has been proposed. The specificities of the complex matching approaches rely on their output (type of correspondence) and their process (guiding structure). The evaluation of these approaches on different datasets have also been discussed. Finally, some perspectives for the field has been discussed.

As a main perspective, a benchmark for complex evaluation will be proposed in the context of the OAEI campaigns. Another direction is to proposed a task-oriented evaluation of complex alignments, as for query rewriting. Fostering the development of complex approaches depends on the availability of such data sets.

## References


[1] P. Shvaiko and J. Euzenat, A Survey of Schema-Based Matching Approaches, in: *Journal on Data Semantics IV*, S. Spaccapietra, ed., Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, pp. 146–171, DOI: 10.1007/11603412_5. ISBN 978-3-540-31001-3 978-3-540-31447-9.

[2] A. Maedche, B. Motik, N. Silva and R. Volz, MAFRA—A MApping FRAmework for Distributed Ontologies, in: *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2002, pp. 235–250.

[3] L. Otero-Cerdeira, F.J. Rodríguez-Martínez and A. Gómez-Rodríguez, Ontology matching: A literature review, *Expert Systems with Applications* **42**(2) (2015), 949–971, ISSN 09574174. doi:10.1016/j.eswa.2014.08.032.

[4] G. Correndo and N. Shadbolt, Translating expressive ontology mappings into rewriting rules to implement query rewriting, in: *6th Workshop on Ontology Matching*, 2011.

[5] J. David, J. Euzenat, F. Scharffe and C. Trojahn, The Alignment API 4.0, *Semantic Web* **2**(1) (2011), 3–10.

[6] D. Dou, H. Qin and P. Lependu, Ontograte: towards Automatic Integration for Relational Databases and the Semantic Web through an Ontology-Based Framework, *International Journal of Semantic Computing* **04**(01) (2010), 123–151, ISSN 1793-351X, 1793-7108. doi:10.1142/S1793351X10000961.

[7] M. Hartung, A. Groß and E. Rahm, COnto–Diff: generation of complex evolution mappings for life science ontologies, *Journal of Biomedical Informatics* **46**(1) (2013), 15–32, ISSN 15320464. doi:10.1016/j.jbi.2012.04.009.

[8] Y. Kalfoglou and M. Schorlemmer, Ontology mapping: the state of the art, *The Knowledge Engineering Review* **18**(1) (2003), 1–31, ISSN 02698889, 14698005. doi:10.1017/S0269888903000651.

[9] N.F. Noy, Semantic integration: a survey of ontology-based approaches, *ACM Sigmod Record* **33**(4) (2004), 65–70.

[10] J. De Bruijn, M. Ehrig, C. Feier, F. Martín-Recuerda, F. Scharffe and M. Weiten, Ontology mediation, merging and aligning, *Semantic web technologies* (2006), 95–113.

[11] E. Rahm and P.A. Bernstein, A survey of approaches to automatic schema matching, *The VLDB Journal* **10**(4) (2001), 334–350, ISSN 10668888. doi:10.1007/s007780100057.

[12] A. Doan and A.Y. Halevy, Semantic integration research in the database community: A brief survey, *AI Magazine* **26** (2005), 83–94.

[13] J. Euzenat and P. Shvaiko, *Ontology Matching*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, DOI: 10.1007/978-3-642-38721-0. ISBN 978-3-642-38720-3 978-3-642-38721-0.

[14] G. Stapleton, J. Howse, A. Bonnington and J. Burton, A vision for diagrammatic ontology engineering, in: *International Workshop on Visualizations and User Interfaces for Knowledge Engineering and Linked Data Analytics*, 2014.

[15] E. Thiéblin, F. Amarger, O. Haemmerlé, N. Hernandez and C. Trojahn, Rewriting SELECT SPARQL queries from 1: n complex correspondences, in: *Ontology Matching*, 2016, p. 49.

[16] K. Makris, N. Gioldasis, N. Bikakis and S. Christodoulakis, Ontology Mapping and SPARQL Rewriting for Querying Federated RDF Data Sources, in: *OTM Confederated International Conferences*, 2010, pp. 1108–1117.

[17] M. Uschold and M. Gruninger, Ontologies and semantics for seamless connectivity, *ACM SIGMod Record* **33**(4) (2004), 58–64.

[18] A. Gater, D. Grigori and M. Bouzeghoub, Complex mapping discovery for semantic process model alignment, in: *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services*, ACM, 2010, pp. 317–324.

[19] G.H. Fletcher and C.M. Wyss, Towards a general framework for effective solutions to the data mapping problem, in: *Journal on Data Semantics XIV*, Springer, 2009, pp. 37–73.

[20] D. Oliveira and C. Pesquita, Improving the interoperability of biomedical ontologies with compound alignments, *Journal of Biomedical Semantics* **9**(1) (2018), ISSN 2041-1480. doi:10.1186/s13326-017-0171-8.

[21] J. Euzenat, Towards composing and benchmarking ontology alignments, in: *Proc. ISWC-2003 workshop on semantic information integration, Sanibel Island (FL US)*, Citeseer, 2003, pp. 165–166.

[22] F. Scharffe, Correspondence Patterns Representation, PhD thesis, Faculty of Mathematics, Computer Science and University of Innsbruck, 2009.

[23] D. Ritze, C. Meilicke, O. Šváb-Zamazal and H. Stuckenschmidt, A pattern-based ontology matching approach for detecting complex correspondences, in: *Proceedings of the 4th International Conference on Ontology Matching-Volume 551*, CEUR-WS. org, 2009, pp. 25–36.

[24] D. Ritze, J. Völker, C. Meilicke and O. Šváb-Zamazal, Linguistic analysis for complex ontology matching, in: *Proceedings of the 5th International Conference on Ontology Matching-Volume 689*, CEUR-WS. org, 2010, pp. 1–12.

[25] B. Walshe, R. Brennan and D. O'Sullivan, Bayes-ReCCE: A Bayesian Model for Detecting Restriction Class Correspondences in Linked Open Data Knowledge Bases, *Int. J. Semant. Web Inf. Syst.* **12**(2) (2016), 25–52, ISSN 1552-6283. doi:10.4018/IJSWIS.2016040102.

[26] R. Parundekar, C.A. Knoblock and J.L. Ambite, Linking and building ontologies of linked data, in: *International Semantic Web Conference*, Springer, 2010, pp. 598–614.

[27] R. Parundekar, C.A. Knoblock and J.L. Ambite, Discovering concept coverings in ontologies of linked data sources, in: *International Semantic Web Conference*, Springer, 2012, pp. 427–443.

[28] R.J. Miller, L.M. Haas and M.A. Hernández, Schema Mapping as Query Discovery., 2000, pp. 77–88.

[29] L.-L. Yan, R.J. Miller, L.M. Haas and R. Fagin, Data-Driven Understanding and Refinement of Schema Mappings., in: *ResearchGate*, Vol. 30, 2001, pp. 485–496. doi:10.1145/375663.375729.

[30] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos and A. Halevy, Learning to match ontologies on the semantic web, *The VLDB Journal—The International Journal on Very Large Data Bases* **12**(4) (2003), 303–319.

[31] 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.

[32] Y. An and I.-Y. Song, Discovering semantically similar associations (SeSA) for complex mappings between conceptual models, in: *International Conference on Conceptual Modeling*, Springer, 2008, pp. 369–382.

[33] W. Hu, J. Chen, H. Zhang and Y. Qu, Learning complex mappings between ontologies, in: *Joint International Semantic Technology Conference*, Springer, 2011, pp. 350–357.

[34] Y. An, A. Borgida and J. Mylopoulos, Inferring complex semantic mappings between relational tables and ontologies from simple correspondences, in: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, Springer, 2005, pp. 1152–1169.

[35] Y. An, A. Borgida and J. Mylopoulos, Constructing complex semantic mappings between XML data and ontologies, in: *International Semantic Web Conference*, Springer, 2005, pp. 6–20.

[36] J. Rouces, G. de Melo and K. Hose, Complex Schema Mapping and Linking Data: Beyond Binary Predicates, in: *Proceedings of the WWW 2016 Workshop on Linked Data on the Web (LDOW 2016)*, 2016.

[37] Y. An, X. Hu and I.-Y. Song, Learning to discover complex mappings from web forms to ontologies, in: *Proceedings of the 21st ACM international conference on Information and knowledge management*, ACM, 2012, pp. 1253–1262.

[38] H. Qin, D. Dou and P. LePendu, Discovering executable semantic mappings between ontologies, in: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, Springer, 2007, pp. 832–849.

[39] F. Kaabi and F. Gargouri, A new approach to discover the complex mappings between ontologies, *International Journal of Web Science 3* **1**(3) (2012), 242–256.

[40] E. Jiménez-Ruiz, E. Kharlamov, D. Zheleznyakov, I. Horrocks, C. Pinkel, M.G. Skjæveland, E. Thorstensen and J. Mora, BootOX: Practical mapping of RDBs to OWL 2, in: *International Semantic Web Conference*, Springer, 2015, pp. 113–132.

[41] L. Xu and D.W. Embley, Using domain ontologies to discover direct and indirect matches for schema elements, in: *Semantic Integration Workshop (SI-2003)*, 2003, p. 97.

[42] L. Xu and D.W. Embley, A composite approach to automating direct and indirect schema mappings, *Information Systems* **31**(8) (2006), 697–732, ISSN 0306-4379. doi:10.1016/j.is.2005.01.003.

[43] S. Jiang, D. Lowd, S. Kafle and D. Dou, Ontology matching with knowledge rules, in: *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXVIII*, Springer, 2016, pp. 75–95.

[44] A. Boukottaya and C. Vanoirbeek, Schema matching for transforming structured documents, in: *Proceedings of the 2005 ACM symposium on Document engineering*, ACM, 2005, pp. 101–110.

[45] R. Dhamankar, Y. Lee, A. Doan, A. Halevy and P. Domingos, iMAP: discovering complex semantic matches between database schemas, in: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, ACM, 2004, pp. 383–394.

[46] R.H. Warren and F.W. Tompa, Multi-column substring matching for database schema translation, in: *Proceedings of the 32nd international conference on Very large data bases*, VLDB Endowment, 2006, pp. 331–342.

[47] P. Arnold, Semantic Enrichment of Ontology Mappings: Detecting Relation Types and Complex Correspondences., *Grundlagen von Datenbanken* **1020** (2013), 34–39.

[48] B.P. Nunes, A. Mera, M.A. Casanova, K.K. Breitman and L.A.P. Leme, Complex Matching of RDF Datatype Properties, in: *Proceedings of the 6th International Conference on Ontology Matching-Volume 814*, CEUR-WS. org, 2011, pp. 254–255.

[49] M.G. de Carvalho, A.H.F. Laender, M.A. Gonçalves and A.S. da Silva, An evolutionary approach to complex schema matching, *Information Systems* **38**(3) (2013), 302–316, ISSN 0306-4379. doi:10.1016/j.is.2012.10.002.

[50] B. He, K.C.-C. Chang and J. Han, Discovering complex matchings across web query interfaces: a correlation mining approach, in: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM Press, 2004, pp. 148–157. doi:10.1145/1014052.1014071.

[51] K. Saleem, Z. Bellahsene and E. Hunt, Porsche: Performance oriented schema mediation, *Information Systems* **33**(7) (2008), 637–657.

[52] W. Su, J. Wang and F. Lochovsky, Holistic schema matching for web query interfaces, in: *International Conference on Extending Database Technology*, Springer, 2006, pp. 77–94.

[53] W. Wu, C. Yu, A. Doan and W. Meng, An interactive clustering-based approach to integrating source query interfaces on the deep web, in: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, ACM, 2004, pp. 95–106.

[54] W. Hu and Y. Qu, Block matching for ontologies, in: *International Semantic Web Conference*, Springer, 2006, pp. 300–313.

[55] O. Šváb-Zamazal, Pattern-based ontology matching and ontology alignment evaluation, PhD thesis, University of Economics, Prague, 2010.

[56] E. Jiménez-Ruiz and B.C. Grau, Logmap: Logic-based and scalable ontology matching, in: *International Semantic Web Conference*, Springer, 2011, pp. 273–288.

[57] L.F. de Medeiros, F. Priyatna and O. Corcho, MIRROR: Automatic R2RML mapping generation from relational databases, in: *International Conference on Web Engineering*, Springer, 2015, pp. 326–343.

[58] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro and G. Xiao, Ontop: Answering SPARQL queries over relational databases, *Semantic Web* **8**(3) (2017), 471–487.

[59] S. Massmann, S. Raunich, D. Aumüller, P. Arnold and E. Rahm, Evolution of the COMA match system, in: *Proceedings of the 6th International Conference on Ontology Matching-Volume 814*, CEUR-WS. org, 2011, pp. 49–60.

[60] 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.

[61] M. Lichman, UCI Machine Learning Repository, 2013. http://archive.ics.uci.edu/ml.

[62] A. Doan, The Illinois Semantic Integration Archive, 2005. http://pages.cs.wisc.edu/~anhai/wisc-si-archive/.

[63] C. Pinkel, C. Binnig, E. Jiménez-Ruiz, W. May, D. Ritze, M.G. Skjæveland, A. Solimando and E. Kharlamov, RODI: A benchmark for automatic mapping generation in relational-to-ontology data integration, in: *European Semantic Web Conference*, Springer, 2015, pp. 21–37.

[64] K.C.-C. Chang, B. He, C. Li and Z. Zhang, The UIUC Web Integration Repository, 2003. http://metaquerier.cs.uiuc.edu/repository.

[65] H.-H. Do, S. Melnik and E. Rahm, Comparison of Schema Matching Evaluations, in: *Web, Web-Services, and Database Systems*, Vol. 2593, G. Goos, J. Hartmanis, J. van Leeuwen, A.B. Chaudhri, M. Jeckle, E. Rahm and R. Unland, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 221–237, DOI: 10.1007/3-540-36560-5_17. ISBN 978-3-540-00745-6 978-3-540-36560-0.

[66] J. Euzenat, Semantic Precision and Recall for Ontology Alignment Evaluation., in: *IJCAI*, 2007, pp. 348–353.

[67] M. Ehrig and J. Euzenat, Relaxed precision and recall for ontology matching, in: *Proc. K-Cap 2005 workshop on Integrating ontology*, No commercial editor., 2005, pp. 25–32.

[68] A. Solimando, E. Jiménez-Ruiz and C. Pinkel, Evaluating ontology alignment systems in query answering tasks, in: *Proceedings of the 2014 International Conference on Posters & Demonstrations Track-Volume 1272*, CEUR-WS. org, 2014, pp. 301–304.

[69] A. Isaac, H. Matthezing, L. van der Meij, S. Schlobach, S. Wang and C. Zinn, Putting Ontology Alignment in Context: Usage Scenarios, Deployment and Evaluation in a Library Case, in: *5th European Semantic Web Conference*, 2008, pp. 402–417.

[70] J. Euzenat, *Ontology matching*, Springer, Berlin ; New York, 2007, OCLC: ocn124038270. ISBN 978-3-540-49611-3.

[71] D. Oliveira and C. Pesquita, Compound matching of biomedical ontologies, in: *Proceedings of the International Conference on Biomedical Ontology, ICBO 2015, Lisbon, Portugal, July 27-30*, 2015.

[72] C. Pesquita, M. Cheatham, D. Faria, J. Barros, E. Santos and F.M. Couto, Building Reference Alignments for Compound Matching of Multiple Ontologies Using OBO Cross-products, in: *Proceedings of the 9th International Workshop on Ontology Matching*, 2014, pp. 172–173.

[73] M. Achichi, M. Cheatham, Z. Dragisic, J. Euzenat, D. Faria, A. Ferrara, G. Flouris, I. Fundulaki, I. Harrow, V. Ivanova, E. Jiménez-Ruiz, K. Kolthoff, E. Kuss, P. Lambrix, H. Leopold, H. Li, C. Meilicke, M. Mohammadi, S. Montanelli, C. Pesquita, T. Saveta, P. Shvaiko, A. Splendiani, H. Stuckenschmidt, É. Thiéblin, K. Todorov, C. Trojahn dos Santos and O. Zamazal, Results of the Ontology Alignment Evaluation Initiative 2017, in: *OM 2017 - 12th ISWC workshop on ontology matching*, Wien, Austria, 2017, pp. 61–113.

[74] A. Solimando, E. Jiménez-Ruiz and G. Guerrini, Detecting and correcting conservativity principle violations in ontology-to-ontology mappings, in: *International Semantic Web Conference*, Springer, 2014, pp. 1–16.

[75] P. Bouquet, J. Euzenat, E. Franconi, L. Serafini, G. Stamou and S. Tessaris, D2. 2.1 Specification of a common framework for characterizing alignment, Technical Report, University of Trento, 2004.

[76] W. Ceusters, Towards A Realism-Based Metric for Quality Assurance in Ontology Matching., in: *Proceedings of the 4th conference on Formal Ontology in Information Systems*, Vol. 321 – 332, Baltimore, Maryland, USA, 2006.