Semantic Web 0 (0) 1
IOS Press

Survey on complex ontology matching

Elodie 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 about the specificities of the complex matching problem. In this paper, an overview of the different complex matching approaches is provided. It 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. Broadly 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). Two types of correspondences can be distinguished. 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 more complex relationships between entities from different ontologies.

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 conceptual heterogeneity. However, currently, few complex alignments are available and published on the LOD, even if the need for these alignments becomes more and more present in various application fields. For example, in the cultural heritage domain, data integration or data translation applications needed complex correspondences [2–5]. To tackle the issue, complex matching system were used [2], or complex correspondences were manually created [4, 5]. In the agronomic domain, complex alignments have been used to cross-query linked open data repositories [6]. In the biomedical domain, complex alignments have also been used to build a consensual model from heterogeneous terminologies [7]. Moreover, complex alignments between medical ontologies have been published [8, 9].

Different complex matching approaches have emerged in the literature, adopting a diversity of strategies and dealing with different knowledge representation models (from database schemata, taxonomies to heavy ontologies). Nevertheless, *complex matching* remains a challenge. In [10], ontology matching researchers were surveyed about future challenges in the field and they agree that "automatically discovering complex relations, instead of 1:1" is one of them.

Diverse surveys in the literature have focused on the different aspects of the schema and ontology matching [1, 10–16] without paying attention to the specificities of complex matching (underlying strategy, structure of complex correspondences, etc.). The aim of this survey is to provide an overview of the complex

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

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 specificities of complex alignments is proposed. The evaluation aspects of these approaches are also over-viewed. Limitations of 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.

The rest of this paper is organised as follows. After background definitions (§2), complex alignment languages and visualisation and edition tools are presented (§3). Then a classification of complex matching approaches is proposed (§4), followed by a description of state-of-the-art approaches (§5). Finally, the works on complex alignment evaluation are surveyed (§6) and perspectives for the field are discussed (§7).

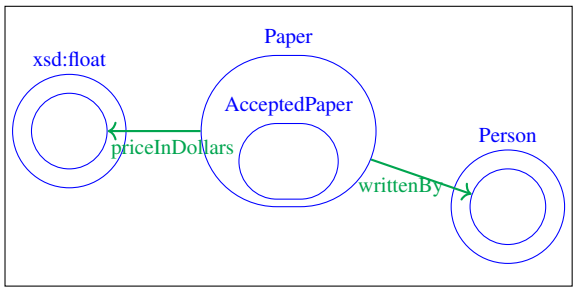
2. Background

In this section the scope of this study is specified and the definitions related to alignments are given. First, the different knowledge models considered in this paper are presented, then the notions of alignment and correspondence are defined. The ontology fragments used for the examples of this section are presented in Figure 1.

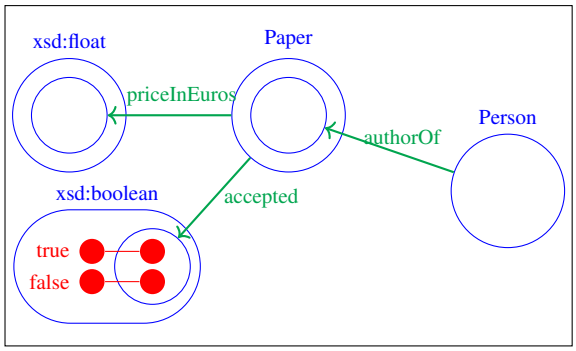
2.1. Knowledge representation models

In the literature, different knowledge representation models are called "ontologies". As stated in [16], *"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."*. In this survey, the term "ontology" is used in a broad sense for the definitions, and a distinction between database schemata, XML schemata, taxonomies, ontologies, etc., is made for the description of the approaches when possible. Figure 2 presents the different kinds of knowledge representation models sorted by expressiveness which are considered in this survey.

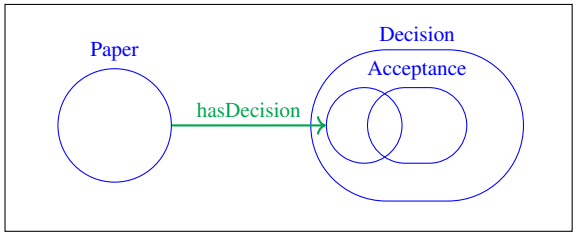
A matching approach can match two models of the same kind (e.g., XML schema to XML schema) or of different kinds (e.g., Database schema to an ontology).



(a) o_1



(b) o_2



(c) o_3

Fig. 1. Example ontologies. The format used to represent the ontologies is described in [17]

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, inter-

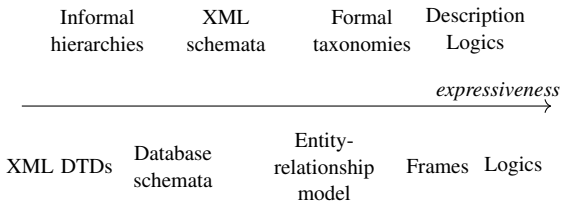


Fig. 2. Kinds of knowledge representation models sorted by expressiveness (adapted from [16]).

section, inverse, 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.

2.3. Alignment and correspondence

Definition 1. *A **correspondence** c_i is composed of a relation r between two **members**: e_{o_1} and e_{o_2}. The members can be simple or complex expressions with entities from respectively the source ontology o_1 and the target ontology o_2:*

- *if the correspondence 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 ($\geqslant$), more specific ($\leqslant$), disjointedness ($\perp$), implication ($\rightarrow$), etc. holding between e_{o_1} and e_{o_2}.*

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 [14, 18] (*1* for *s* and *m* or *n* for *c*). However they can be misinterpreted as the alignment arity or multiplicity [19]. In the following, are provided some examples of complex correspondences based on the definitions above and the motivating example ontologies (Figure 1).

c_1 = $\forall x$, o_1*:Person(x)* $\equiv$ o_2*:Person(x)* is a *(s:s)* simple correspondence.

c_2 = o_1*:priceInDollars* = *changeRate* $\times$ o_2*:priceInEuros* is a *(s:c)* complex correspondence with a transformation function.

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[1].

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.

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.

Definition 2. *An **ontology alignment** A is directional between a source ontology o_1 and a target ontology o_2, denoted $A_{o_1 \rightarrow o_2}$. $A_{o_1 \rightarrow o_2}$ is a set of correspondences $A_{o_1 \rightarrow o_2} = \{c_1, c_2, ..., c_n\}$.*

In opposition to a **simple alignment**, a **complex alignment** contains at least one complex correspondence.

Definition 3. *As defined in [16], **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.*

This definition of *pairwise* matching process can be however extended to cover multiple ontologies. A **holistic** matching process considers more than two ontologies together without a source or target distinction [20, 21]. On the other hand, **compound** matching is the process of matching one or more source ontologies and one or more target ontologies. This process is *pairwise* between the union of the source ontologies and the union of the target ontologies [22].

Complex ontology matching is the process of generating a complex alignment between ontologies. The approaches for generating such a kind of alignment are discussed in the next section.

2.4. Scope definition

This section presents reflections on the scope of the survey.

[1]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.

2.4.1. Type of matched objects

Complex alignments can also occur between other objects such as business process models [23], strings [24], etc. However, these objects are different in various ways from the knowledge representation models studied in this survey. First of all, these objects do not model knowledge related to a particular domain nor define a particular vocabulary. Then, the nature of their elements is different from the representation models elements (concepts, relations, attributes). For example, business processes are graph-like models of a process, they have a beginning and an end node, the nodes of their graph are either connectors or activities which take input and output elements. The strings have no explicit structure. For these reasons, these types of matching are out of the scope of this survey.

2.4.2. Ontology matching and ontology evolution

Some connections can be made between ontology matching and ontology evolution. As defined in the survey presented in [25], ontology evolution is the process which consists in maintaining a resource up to date according to changes occurring in the represented knowledge domain or to new requirements of the application(s) relying on the ontology. Ontology evolution is divided into different tasks: detecting the need for evolution, suggesting changes, validating changes, assessing the impact of the changes and managing changes. The latter includes the activities of change recording and ontology versioning. These activities are defined as "the ability to handle changes in ontologies by creating and managing different variants of it" [26]. Most approaches dealing with such activities rely on relations considering both the two variants, also called versions, of an ontology and the entities within the two representations. Finding these relations can be in some ways similar to ontology matching. When taking a deeper look at what "version relations" express, not only *conceptual or logical relations* between entities must be considered but also (and mainly) *change relations*, which represent what has actually been transformed between the two versions of the ontology [27]. The first kind of relation (conceptual or logical relations) specifies correspondences between the entities of the source and the target ontologies (as defined previously). On the other hand, the second type of relations (change relations) specifies transformations, via a set of change operations, to apply on the source ontology in order to obtain the target ontology (for example adding a new domain to a property, merging two classes, etc.). Such relations are either captured at designing time through the tool used to make the ontology evolve (such as Protege or KAON [28]) or identified a posteriori through ontology "diff tools" [29–32]. Most works in the field have focused on proposing approaches in order to identify the second type of relations, i.e. "change relations". Existing ontology matching approaches are generally reused in this task for finding the initial overlap between the two ontology versions.

As pointed out in [30] the aim of managing changes in ontology evolution is to *highlight differences*, whereas the ontology matching task *concentrates on similarities*.

An analogous classification is made between simple and complex changes according to the entities evolved in the changes : "simple changes refer to the addition, modification or deletion of individual schema constructs, while complex changes refer to multiple such constructs and may be equivalent to multiple simple changes"[33]. The two types of changes are also called low level / high level operations [31], elementary/composite changes [28] or atomic/complex changes [34]. According to [31], high level operations are "intuitive, concise, closer to the intentions of the ontology editors and capture more accurately the semantics of a change" even if the authors point out that it is impossible to define an exhaustive list of such operations. Most languages proposed to represent changes make this distinction [28, 30].

The work in [35] gives another point of view on the link between both tasks, and studies the impacts of evolution changes on existing correspondences between ontologies. Even if the task of change management is complementary to the task of ontology matching, it can strongly benefit from advances in the field of complex matching.

3. Complex alignment representation and visualisation

This section presents the languages and vocabularies used for complex alignment representation as well as works on graphical interfaces for complex alignment visualisation and edition.

3.1. Complex alignment languages and vocabularies

A survey on ontology alignment formats is presented in [36]. Here are presented the known languages and vocabularies to express complex corre-

spondences between knowledge representation models. Many usual alignment languages can only express simple correspondences (DDL, DFOL). The correspondences can be represented as logic rules following a given syntax, described by a dedicated vocabulary or represented as queries. In the area of OBDA (Ontology-Based Data Access) [37], the R2RML format, a W3C standard, has been extended in many different ways.

3.1.1. Logic syntaxes

Correspondences can be expressed as logic formulae.

Web-PDDL The Web-PDDL [38] is a strongly typed first-order language. It allows for the use of variables, constants, conditions, logical constructors and quantifiers. The predicates and constants take the form of URIs.

Other logic syntaxes such as DataLog, RIF, etc. using URIs as predicates can be used to express logic formulae. Even if they are originally meant to express these formulae inside one ontology, they can be used to express correspondences when involving URIs from more than one ontology.

3.1.2. Vocabularies

Correspondences can be described by a dedicated vocabulary or ontology. Ontologies such as OWL or SWRL are originally meant to describe axioms or rules inside one ontology. They can also model correspondences when used between two ontologies. Other vocabularies such as EDOAL or SBO were created to describe the correspondences between a source and a target ontologies.

OWL OWL [39] can represent complex alignments as axioms involving logic constructors and entities from the source and target ontologies. These axioms form a merging ontology. The expressiveness of the correspondences in OWL (taking into account the expressiveness of the aligned ontologies) is restricted to the $\mathcal{SROIQ}$ logic for decidability reasons.

SWRL The Semantic Web Rule Language (SWRL) [40] helps to define rules, in the form of first order Horn-clauses, between OWL ontologies. These rules have no expressiveness restriction and provide flexibility thanks to the use of variables in the definition of the rules. This language comes with an XML Concrete Syntax to express the rules as XML documents. SWRL can be extended by *built-ins* based on the XQuery and XPath built-ins. These built-ins express transformation functions.

EDOAL EDOAL [41] is an extension of the Alignment format to represent the complex correspondences between OWL ontologies. This language is based on correspondence patterns [36] and can be processed by the Alignment API.

XeOML XeOML [42] is a language which represents alignments for ontologies and can be extended to other kinds of knowledge representation models. It is based on an XML schema (*Abstract Mapping* schema) to describe the structure of an alignment and is completed by two other schemata (*Ontology Element Definition* and *Mapping Definition*).

SBO MAFRA [43, 44] is a framework for constructing and editing (DAML+OIL) ontology alignments. The alignment representation part of the framework is based on the Semantic Bridge Ontology (SBO). This (not maintained) ontology provides a vocabulary to express complex correspondences with logical constructors and some transformation functions such as string concatenation.

SPIMBench The SPIMBench vocabulary was defined in an instance matching benchmark [45]. It allows for the description of data transformation between ontologies. These transformations include logic rules (based on OWL axioms) and value transformation functions.

3.1.3. The R2RML family

In the literature, diverse relational database to ontology formats have been defined and extended to other kinds of schemata. The reader can refer to the survey in [46] for a comparison of the existing database to ontology alignment formats. Here are presented the R2RML format, a W3C standard and some of its latest extensions.

R2RML DB to Ontology, Logic, Transformation The R2RML format [47] is used to represent correspondences between relational databases and RDF datasets. This language is a W3C standard. R2RML correspondences are expressed as RDF datasets. A few string operations can be expressed in the correspondences. The R2RML correspondences show how the data from the source schema should be transformed into the target ontology.

RML The RML language [48] extends the R2RML format by allowing other kinds of schemata as source: XML or JSON. The expressions can be specified using XPath or JSONPath. The FnO ontology [49] can be

used in RML to describe tranformation functions in the correspondences.

xR2RML The xR2RML language [50] extends the R2RML format by allowing the description of correspondences of mixed formats in the source schema. For example, if a JSON object is the value of a cell in a relational database.

D2RML The D2RML language [51] is based on R2RML and RML. It allows conditional case statements and programming inside the correspondences.

3.1.4. Query lanquages

Alignments can be directly represented through semantically equivalent queries (or views) of their data. SQL is the language for querying relational databases, XQuery for XML documents and SPARQL for knowledge bases (ontologies). These query languages can use filters (or equivalent) to express transformation functions inside a query.

XSLT, XPath XML to XML, Logic, Transformation XSLT (eXtensible Stylesheet Language Tansformations) [52] is a language under the form of XML documents. This language describes rules to transform a source tree (XML document) into a target tree (XML document). This language is based on transformation patterns and reuse XPath expressions. XPath (XML Path Language) defines expressions with logical operators and transformation functions over XML nodes. The XPath functions are often reused in other alignment languages.

3.1.5. Summary table

Table 1 gives a summary of the complex alignment formats presented in this section with their targeted application. The distinction between data integration and data transformation is that, in a data integration process, there is no transformation of the data. A data integration application can be data querying without loading the data in a central repository [16]. However these two applications are similar and data integration rules can be used for data transformation.

For readability reasons, in this survey, the logic transformations are expressed in FOL, the transformation functions with equations and the blocks with sets.

3.2. Complex alignment visualisation and edition

Few tools allow for complex correspondence visualisation and edition. Some solutions are provided as part of specific standalone matching systems, while others are rather generic solutions, as we describe in the following. Table 2 presents a comparison of the tools.

Axiom and rule editors which allow the import of different ontologies can be used for complex alignment edition, as Protégé [53]. It can be used to edit OWL axioms involving entities from different ontologies. The complex correspondences (as axioms) can be visualised using the Manchester syntax. Another solution is the Axiomé [54] SWRL rule editor. The rules are represented as tree structures and can be paraphrased in English.

Tools as part of existing matchers such as Clio [55] or KARMA [56] provide a user-interface for the complex alignment edition and correction.

Dedicated complex alignment editors use different strategies for the visualisation of the correspondences. MAFRA [43] is an edition and visualisation framework which allows for complex alignment representation as an instanciation of their Semantic Bridge Ontology (SBO). Klint [57] provides a graph-based visual interface for integration rules (correspondence) validation and edition. The correspondences are represented as labelled graphs involving variables. OntoStudio [58] is a suite of software for ontology engineering. Its OntoMap plugin [59] allows for manual edition of complex correspondences (logic and value transformations). OntoMap uses its own internal alignment language which is not public. Many R2RML correspondence editors have emerged in the past years. They use different strategies for the representation of the correspondences: block metaphor [60, 61], graph-like [62–65] or tree-like [56]. Wrangler [66] proposes a graphical interface for the edition of value transformation functions as scripts between tables.

4. Classification of complex matchers

Ontology matching approaches have been classified in various surveys [1, 10–16]. These classifications however do not address the specificities of the complex approaches. After giving an overview of the main ontology matching approaches classifications (§4.1), axes for complex matching approaches classification are presented (§4.2).

4.1. Classifications of ontology matching approaches

Euzenat and Shvaiko [1, 16] define three matching dimensions: input, process and output which will be

Table 1
Complex alignment formats

Format	Type of knowledge representation models	Logic	Transformation	Target application
Web-PDDL	Onto to onto	√		Data integration
OWL	Onto to onto	√		Ontology merging
SWRL	Onto to onto	√	√	Data integration
EDOAL	Onto to onto	√	√	Generic
XeOML	Onto to onto	√	√	Generic
SBO	Onto to onto	√	√	Data transformation
SPIMBench	Onto to onto	√	√	Data transformation
R2RML	DB to onto	√	√	Data transformation
RML	Schema to onto	√	√	Data transformation
xR2RML	Schema to onto	√	√	Data transformation
D2RML	DB to onto	√	√	Data transformation
XSLT	XML to XML	√	√	Data transformation
SQL	DB to DB	√	√	Querying
XQuery	XML to XML	√	√	Querying
SPARQL	Onto to onto	√	√	Querying

Table 2
Complex alignment visualisation and edition tools

Tool	Visualisation strategy	Align. langage
Protégé [53]	Manchester syntax	OWL
Axiomé [54]	Tree, Simple english	SWRL
MAFRA [43]	Tree	SBO
Klint [57]	Graph	SPARQL
OntoMap [59]	Edges	proprietary
Clio [67]	Edges	SQL
Juma [60, 61]	Block metaphor	R2RML
KARMA [56]	Tree	R2RML
Map-On [62]	Graph	R2RML
RMLEditor [63]	Graph	RML/R2RML
SquaRE [65]	Graph	R2RML
Lembo2014 [64]	Graph	R2RML
Wrangler [66]	Scripts	proprietary

the guiding thread to present the classifications. Most of the classifications so far focused on the input and process dimensions [1, 12, 14–16].

Regarding the input dimension, the *instance vs ontology* classification (called *instance vs schema* in [14]) 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.* [14] also consider as input the type of auxiliary information used by the approaches (thesaurii, etc.). For the process dimension, Rahm *et al.* [14] propose classification axes such as *element vs structure*, *linguistic vs constraint-based*. All of these classification axes are put together into a taxonomy.

The classification of Rahm *et al.* [14] has been developed and extended by Euzenat and Shvaiko in [1, 16]. 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.* [15] 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 [12] proposes two main categories of ontology matching approaches: in the first one, the matching process is guided by a top-level ontology from which the source and target ontologies derive; in the second one, the matching process uses heuristics or machine-learning techniques.

Regarding the output dimension of the matching approaches, Rahm *et al.* [14] 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 [16] 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 axes for complex ontology matching approaches.

4.2. Classification for complex matching approaches

The specificities of the complex matching approaches rely on their output and their process. These are the two axes 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 structures).

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*. 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. For example, let's consider the following correspondences:

1. $\forall x,\ o_1{:}AcceptedPaper(x) \equiv o_2{:}accepted(x,true)$
2. $o_1{:}priceInDollars = changeRate \times o_2{:}priceInEuro$
3. $\{o_1{:}Paper, o_1{:}Person\}=\{o_2{:}Paper, o_2{:}Person\}$

Correspondence 1 is a logical relation correspondence, correspondence 2 is a transformation function correspondence and correspondence 3 is a block correspondence. No precise relation is specified between the entities involved in the third correspondence. Therefore, it can not be classified as logical relation or transformation function correspondence. 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. An example of this correspondence expressed in pseudo-FOL would be: $\forall x,\ \exists y,\ o_1{:}Paper(x) \wedge o_1{:}priceInDollars(x,y) \rightarrow o_2{:}Paper(x) \wedge o_2{:}priceInEuros(x,y \div changeRate)$.

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 [36]. 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 $o_1{:}a + o_1{:}b = o_2{:}c$ 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, when a schema is considered as a tree and the approach

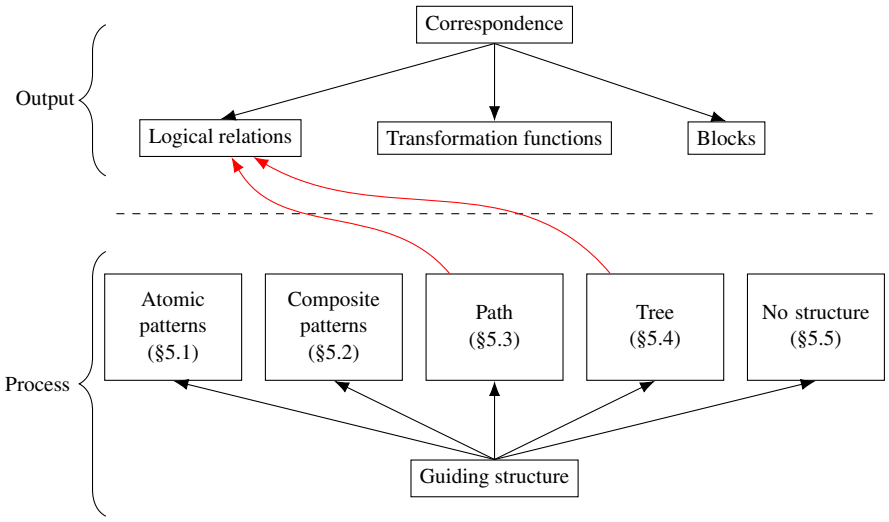


Fig. 3. Two axes to characterise the complex matching approaches: output and process. The correlation between the categories are represented with red arrows.

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 structures are used to guide the matching process, therefore they impact the structure of the output correspondences. However, a given correspondence, for example $\forall x,\ o_1{:}AcceptedPaper(x) \equiv \exists y,\ o_2{:}acceptedBy(x,y)$, could be obtained by an approach based on atomic patterns (with the pattern $\forall\ o_1{:}A(x) \equiv \exists y,\ o_2{:}b(x,y)$), by an approach based on composite patterns ($\forall\ o_1{:}A(x) \equiv \exists y,\ o_2{:}b(x,y) \wedge o_2{:}c(x,y) \wedge \ldots$), or by an approach with no guiding structure.

The *members expressions pre-definition* specifies if one of the members of the correspondence is assigned a fixed structure or not before the process. Three types of pre-definition are possible: fixed to fixed, fixed to unfixed and unfixed to unfixed.

– 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. As shown in 3, this category is strongly correlated to the Atomic-pattern guiding structure category.
– 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 finding for each property of an ontology a corresponding property path in the other ontology 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 strategies to find complex correspondences. In the following, the matching strategies are classified on their guiding structure. Therefore, the same approach can appear in multiple sections.

Some correlations can be noted as depicted in Figure 3: a path or tree based approach will only output logical correspondences. There also is an equivalence between the *fixed to fixed* category and the *atomic pattern* category.

The choice of this guiding structure based classification was made because guiding structures are a specificity of complex matching techniques. Not only do they guide the matching process, but the correspon-

dence structure derives directly from them. Other classifications were considered before this choice:

- A classification per type of knowledge representation model but it would not show the similarities between the matching systems even though they do not deal with the same type of knowledge representation model;
- A classification per type of correspondence output but this was not structuring enough;
- The classification from [16] but most complex matching approaches combine many of those basic matching techniques;
- A classification per type of entities (concepts, properties, etc.) dealt with by the matchers but this was not specific to complex alignment.

This choice of the structure-oriented classification was made because structures are used specific to the complex matching task (whatever may the knowledge representation models be). In some way, the structure can either be seen as a specialisation of the graph-based techniques of the classification of [16].

5. 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)].*

5.1. Atomic patterns

Atomic patterns are used in approaches to detect logical relations as well as transformation functions relations. Table 3 presents a few atomic correspondence patterns. Table 4 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 [70], others approaches have their own pattern library [68, 71–75]. Two main detection techniques appear: structuro-linguistic conditions (called *matching patterns* defined in [76]) [68–70, 74, 75], and statistical measures [71–73]. These approaches are detailed in the following.

Ritze et al. [68, 69] Ontology to Ontology, (s:c), In [68, 69], 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 [36] (c.f. Table 3). The conditions are based on the labels of the ontology 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[2] [68] 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 second version[3] of the matching conditions [69] 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 label of an other property p_2, then $\forall x,y,\ o_1{:}p_1(y,x) \leqslant o_2{:}p_2(x,y)$ is a probable correspondence.

Example 1. *Conditions:* $\forall x,y\ o_1{:}p_1(y,x) \leqslant o_2{:}p_2(x,y)$ *iff* $verb(p_1) = active\text{-}voice\ (verb(p_2))$
Correspondence:
$\forall x,y\ o_1{:}writePaper(y,x) \leqslant o_2{:}writtenBy(x,y)$
because write is the active-form of 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 : the domain of p_1 should be subsumed by of the range of p_2 and the range of p_1 should be subsumed by the domain of 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.

Oliveira and Pesquita [22] Ontology to ontology, (s:c) The approach proposed in [22] looks for *compound* correspondences which in their target mem-

[2]http://dominique-ritze.de/complex-mappings/
[3]https://code.google.com/archive/p/generatingcomplexalignments/downloads/

Table 3
Atomic patterns used in the presented approaches. A, C are classes, a, b, c are properties, V is a value (instance or literal)

Name	Form	Example
Class by attribute type (CAT)	$\forall x, A(x) \equiv \exists y, b(x,y) \wedge C(y)$	$\forall x, o_1{:}AcceptedPaper(x) \equiv \exists y, o_2{:}hasDecision(x,y) \wedge o_2{:}Acceptance(y)$
Class by attribute inverse type (CIAT)	$\forall x, A(x) \equiv \exists y, b(x,y) \wedge C(x)$	$\forall x, o_1{:}AcceptedPaper(x) \equiv \exists y, o_2{:}hasAcceptance(x,y) \wedge o_2{:}Paper(x)$
Class by attribute value (CAV)	$\forall x, A(x) \equiv b(x,V)$	$\forall x, o_1{:}AcceptedPaper(x) \equiv o_2{:}accepted(x,True)$
Property chain (PC)	$\forall x,y, a(x,y) \equiv \exists z, b(x,z) \wedge c(z,y)$	$\forall x,y, o_1{:}reviewedBy(x,y) \equiv \exists z, o_2{:}hasReview(x,z) \wedge o_2{:}reviewWrittenBy(y,z)$
Inverse Property (IP)	$\forall x,y, a(x,y) \equiv b(y,x)$	$\forall x, o_1{:}writtenBy(x,y) \equiv o_2{:}authorOf(y,x)$
Class Intersection	$\forall x, A(x) \equiv B(x) \wedge C(x)$	$\forall x, o_1{:}AuthorAndReviewer(x) \equiv o_2{:}Author(x) \wedge o_2{:}Reviewer(x)$

Table 4
Atomic patterns per approach

Work	*Patterns*
Ritze2009 [68]	Class by Attribute Type, Class by Inverse Attribute Type, Class by Attribute Value, Property Chain
Ritze2010 [69]	Inverse property, Class by Attribute Type, Class by Inverse Attribute Type, Class by Attribute Value
Oliveira2018 [22]	Class Intersection
Rouces2016 [70]	Linguistic patterns of FrameBase
Walshe2016 (Bayes-ReCCE) [71]	Class by Attribute Value
Jiang2016 (KAOM) [72]	Linear Regression
Dhamankar2004 (iMAP) [73]	Conversion functions predefined, basic arithmetic properties
Jimenez2015 (BootOX) [74]	RDB schema properties to OWL axioms

ber involve entities from more than one ontology. The sought correspondences follow the pattern $\forall x, o_1{:}A(x) \equiv o_2{:}B(x) \wedge o_3{:}C(x)$ in which $o_1{:}A$, $o_2{:}B$ and $o_3{:}C$ are classes from a source ontology o_1, and two target ontologies o_2 and o_3. The approach is based on a similarity measure between the labels of the source and targets classes. In a first step, the source classes are aligned to the classes of a first target ontology (e.g., o_2). Each of these correspondences is given a similarity score based on how the labels of the target classes overlap with the label of the source class. The correspondences are filtered over this similarity. The labels of the source class are reduced to the difference between the source and target classes' labels from the previously obtained correspondence. Finally, the source reduced labels are matched with those of the second target ontology (e.g., o_3) based on how this new label allows for the covering of the total source label.

Example 2. *A source class o_1:AuthorAndReviewer with the label "author and reviewer" is first aligned to o_2:Author which has the "author" label. The label of the source class is then reduced to "and reviewer" because of the correspondence in the previous step. Finally, in the last step, o_3:Reviewer with the label "reviewer" is added to the correspondence because its label provides a good coverage of the reduced label "and reviewer". The output correspondence is: $\forall x, o_1{:}AuthorAndReviewer(x) \equiv o_2{:}Author(x) \wedge o_3{:}Reviewer(x)$*

Rouces et al. [70] 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 3). 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). This approach is used in the Klint tool [57] which provides a graphical interface for correspondence edition.

Example 3. *Created property: frame:hasBirthDate(s,o)*
Pattern: frame:BirthEvent(e) $\wedge$ *frame:hasSubject(e,s)* $\wedge$ *frame:hasDate(e,o)*
Source property preprocessing:
o_1*:birthDate* $\rightarrow$ o_1*:hasBirthDate*
Simple correspondence:
$\forall$*x,y,* o_1*:hasBirthDate(x,y)* $\equiv$ *frame:hasBirthDate(x,y)*
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 [71] 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)* $\leqslant$ 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. [73] Relational database schema to relational database schema, (c:s) The iMAP system [73] uses a set 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. The patterns of the numeric searcher are equation templates given by the user or from previous matches. The category correspondence looks for equivalent attribute-value pairs for attributes having a small set of possible values. The schema mismatch searcher looks for correspondences in which an attribute of the source schema has a *true* value if it appears in a list of attributes in the target schema. Examples of category and schema mismatch correspondences are presented in Example 4. 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.

Example 4. *Category searcher correspondence between schemata describing papers and their acceptance status:*
$\forall$*x,* s_1*:accepted(x,"true")* $\equiv$ s_2*:accepted(x,"1")*
Schema mismatch correspondence between schemata describing a conference participant status:
$\forall$*x,* s_1*:actions(x,"early-registration")* $\equiv$ s_2*:early-registration(x,"true")*
This correspondence means that the target attribute s_2*:early-registration is assigned a "true" value if "early-registration" appears in the list of the participant's actions from the source schema.*

KAOM, Jiang et al. [72] Ontology to ontology, (s:c) (c:s) (c:c) KAOM generates transformation function correspondences and logical relation correspondences. As the iMap's system [73], 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 §5.5. The atomic pattern used is a positive linear transformation function between numerical data properties o_1*:a* and o_2*:b* of respectively o_1 and o_2. A Kullback-Leibler divergence measure on the data values is used to define the coefficient $coeff$ of the linear transformation: o_1*:a* = *coeff* $\times$ o_2*:b*.

BootOX, Jimenez-Ruiz et al. [74] Database schema to ontology, (c:s) The BootOX approach [74] 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 (created/extracted) 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 [77] matcher to the target ontology. LogMap relies on linguistic and structural information to perform the matching. Put together, the transformation rules from RDB to ontology and the Logmap ontology alignment form a complex alignment between the RDB and the target ontology.

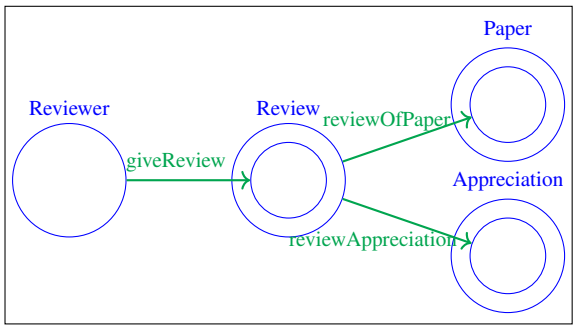
Other systems can bootstrap ontologies from relational database schemata [78, 79] but their aim is not to align the schema to an existing ontology. Therefore, they are out of the scope of this study. In this survey, BootOX is considered with its LogMap extension.

5.2. Composite patterns

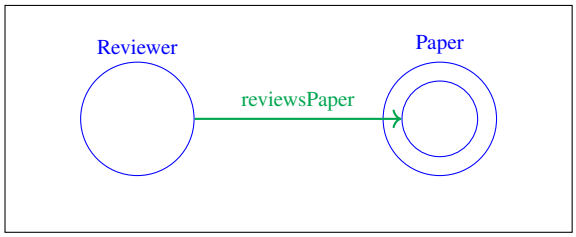
Composite pattern-based approaches often focus on one or two types of patterns. Table 5 presents the different composite pattern types detected by the approaches.

Some approaches iteratively construct the member(s) of the correspondence [73, 80, 82, 83, 87] (text searcher of iMap). Others first discover atomic pattern correspondences and merge them in a final (non-iterative) step [81, 88]. Approaches use graph-pattern matching either as detection conditions [75, 84, 89, 90] or over the properties of a mediating ontology [73, 85, 86] (iMap's date searcher). Finally, [91, 92] start by grouping schema attributes before matching the groups. Even though the holistic approaches [91, 92] 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 [84–86, 90]. 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.

Šváb-Zamazal and Svátek [75] Ontology to ontology, (s:c),(c:s),(c:c) This approach is based on structural and naming conditions to detect N-ary relations as defined by the Semantic Web Best Practice (SWBP)[4] in the aligned ontologies. First, reified N-ary relations are sought in the ontologies by with the help a lexico-structural pattern. The fragment of ontology



(a) o_1



(b) o_2

Fig. 4. N-ary relation pattern

represented in Figure 4a shows an N-ary relation between a reviewer, a paper and its review appreciation. This pattern consists in an intermediate concept (here o_1*:Review*) representing the relation between a domain o_1*:Reviewer* and N ranges o_1*:Appreciation*, o_1*:Paper*. Once the N-ary relations are detected in the source and target ontologies, a similarity measure is computed between the source and target patterns. This similarity is an aggregation of the N-ary relations concepts labels similarity. If the similarity is above a threshold, a structure to structure correspondence is created. The N-ary relations are also matched to object properties by comparing their labels and domain/range compatibility. Figure 4 shows an example of an N-ary relation (4a) and corresponding object properties (4b). The N-ary relation to object property correspondence is heterogeneous: the semantics of this relation are not clear. A N-ary to object property correspondence in FOL could be: $\forall x,y,z_1,z_2,\ o_1{:}giveReview(x,y) \wedge o_1{:}reviewOfPaper(y,z_1) \wedge o_1{:}reviewAppreciation(y,z_2) \equiv o_2{:}reviewsPaper(x,z_1)$

Parundekar et al. [80] Ontology to ontology, (s:c) (c:s) In this approach proposed by Parundekar *et al.* [80], the type of correspondences sought is an *attribute-value* pair matched with an attribute and a union of its acceptable values. In a first step, the approach finds correspondences between *attribute-value*

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

Table 5
Composite patterns per approach. *A, B, C* are classes, *a, b, c, d* are properties, v, v_1, v_2, v_3 are values (instances or literals)

Work	*Composite pattern*	*Pattern form*
Svab2009 [75]	N-ary to N-ary N-ary to object property	Structure matching, see Fig. 4
Parundekar2012 [80]	Disjunction of attribute-value pairs	$\forall x,\ o_1{:}a(x,v) \equiv o_2{:}b(x,v_2) \vee o_2{:}b(x,v_3) \vee \ldots$
Parundekar2010 [81]	Conjunction of attribute-value pairs	$\forall x,\ o_1{:}a(x,v) \wedge o_1{:}b(x,v_1) \wedge \ldots \equiv o_2{:}c(x,v_2) \wedge o_2{:}d(x,v_3) \wedge \ldots$
Doan2003 (CGLUE) [82]	Class unions	$\forall x,\ o_1{:}A(x) \equiv o_2{:}B(x) \vee o_2{:}C(x) \vee \ldots$
Kaabi2012 (ARCMA) [83]	Class intersection	$\forall x,\ o_1{:}A(x) \equiv o_2{:}B(x) \wedge o_2{:}C(x) \ldots$
Boukottaya2005 [84]	String concatenation Subset merging	$o_1{:}a = concatenation(o_2{:}b,\ o_2{:}c, \ldots)$ $\forall x,y,\ o_1{:}a(x,y) \equiv o_2{:}b(x,y) \vee o_2{:}c(x,y), \ldots$
Dhamankar2004 (iMAP) [73]	String concatenation	$o_1{:}a = concatenation(o_2{:}b,\ o_2{:}c, \ldots)$
Xu2003 [85]	String concatenation Subset merging	$o_1{:}a = concatenation(o_2{:}b,\ o_2{:}c, \ldots)$ $\forall x,y,\ o_1{:}a(x,y) \equiv o_2{:}b(x,y) \vee o_2{:}c(x,y), \ldots$
Xu2006 [86]	String concatenation Subset merging	$o_1{:}a = concatenation(o_2{:}b,\ o_2{:}c, \ldots)$ $\forall x,y,\ o_1{:}a(x,y) \equiv o_2{:}b(x,y) \vee o_2{:}c(x,y), \ldots$
Warren2006 [87]	String concatenation of attribute substrings	$o_1{:}a = concatenation(substr(o_2{:}b),\ substr(o_2{:}c), \ldots)$
Arnold2013 (COMA++) [88]	String concatenation	$o_1{:}a = concatenation(o_2{:}b,\ o_2{:}c, \ldots)$
Wu2004 [89]	Annotated sets of properties (is-a or aggregate)	$\{o_1{:}a\}=is\text{-}a\{o_2{:}b,\ o_2{:}c, \ldots\}$; $\{o_1{:}a\}=aggregate\{o_2{:}b,\ o_2{:}c, \ldots\}$
Saleem2008 (PORSCHE) [90]	Bag of properties/blocks	$\{o_1{:}a\}=\{o_2{:}b,\ o_2{:}c, \ldots\}$
He2004 (DCM) [91]	Bag of properties/blocks	$\{o_1{:}a\}=\{o_2{:}b,\ o_2{:}c, \ldots\}$
Su2006 (HSM) [92]	Bag of properties/blocks	$\{o_1{:}a,\ o_1{:}b, \ldots\}=\{o_2{:}c,\ o_2{:}d, \ldots\}$

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: $\forall x,\ o_1{:}a(x,v_1) \geqslant o_2{:}b(x,v_2)$. 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 $\forall x,\ o_1{:}a(x,v_1) \equiv o_2{:}b(x,v_2) \vee o_2{:}b(x,v_3) \vee \ldots$ with $o_1{:}a$, $o_2{:}b$ properties and v_1, v_2, v_3 constant values: instances or literals. The following example shows the two-step approach.

Example 5. *First step output:*
$\forall x,\ o_1{:}accepted(x,True) \geqslant o_2{:}hasStatus(x,\ \text{“accepted”})$
$\forall x,\ o_1{:}accepted(x,True) \geqslant o_2{:}hasStatus(x,\ \text{“camera-ready”})$
Second step output:
$\forall x,\ o_1{:}accepted(x,True) \equiv o_2{:}hasStatus(x,\ \text{“accepted”}) \vee o_2{:}hasStatus(x,\ \text{“camera-ready”})$

Parundekar et al. [81] Ontology to ontology, (s:c) (c:s) (c:c) Parundekar *et al.* [81] look for conjunctions of *attribute-value* pairs, for instance correspondences of the form $\forall x,\ o_1{:}a(x,v_1) \wedge o_1{:}b(x,v_2) \wedge \ldots \equiv o_2{:}c(x,v_3) \wedge o_2{:}d(x,v_4) \wedge \ldots$ with $o_1{:}a$, $o_1{:}b$, $o_2{:}c$, $o_2{:}d$ properties and the $v_{i \in n}$ constant values: instances or literals. 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 manually removed (i.e., properties from a different domain than the common scope of the ontologies and inverse functional properties). A set of first correspondences (the seed hypotheses) are created between *attribute-value* pairs. An example of a seed hypothesis is $\forall x,\ o_1{:}a(x,v_1) \equiv o_2{:}b(x,v_2)$. 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.

CGLUE, Doan et al. [82] Taxonomy to taxonomy, (s:c) The GLUE system [82] is specialised in detecting (s:s) correspondences between taxonomies using machine learning techniques such as joint probability distribution. CGLUE, also presented in [82], 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.

ARCMA, Kaabi et Gargouri [83] Ontology to ontology, (s:c) Kaabi *et* Gargouri [83] propose ARCMA (Association Rules Complex Matching Approach) to find correspondences of the form $\forall x,\ o_1{:}A(x) \leqslant o_2{:}B(x) \wedge o_2{:}C(x) \wedge \ldots$. 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. The following example presents how a correspondence is detected by this approach.

Example 6. *Let o_1:AuthorAndReviewer be a subclass of o_1:Author and o_1:Reviewer. Simple correspondences, between $o_1$and o_2 are given:*

- $\forall x,\ o_1{:}Author(x) \equiv o_2{:}Author(x)$
- $\forall x,\ o_1{:}Reviewer(x) \equiv o_2{:}Reviewer(x)$

With the overlap of terms associated to o_1:AuthorAndReviewer and the terms of respectively o_2:Author and o_2:Reviewer, the following correspondence can be output: $\forall x,\ o_1{:}AuthorAndReviewer(x) \equiv o_2{:}Author(x) \wedge o_2{:}Reviewer(x)$

Boukottaya and Vanoirbeek [84] XML schema to XML schema, (s:c) (c:s) (c:c) Boukottaya *et* Vanoirbeek [84] 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. Union/selection correspondences are detected when nodes have a common abstract type (based on their WordNet similarity) which matches a node from the other schema. Merge/split are computed when a leaf node matches a non-leaf 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.

Example 7. *If a node o_1:address of the source schema with children leaf nodes (o_1:street, o_1:city) matches a leaf node o_2:address of the target schema, then a concatenation of the children nodes can be matched to the target node:*
concatenation(o_1:street,o_1:city)=o_2:address

If two nodes o_1:Journal-Article and o_1:Conference-Article from the source ontology have a common abstract super node (computed from WordNet): Article and that o_2:Article matches this super node, a union pattern is detected:
$\forall x,\ o_1{:}Journal\text{-}Article(x) \vee o_1{:}Conference\text{-}Article(x) \equiv o_2{:}Article(x)$

COMA++, Arnold [88] Ontology to ontology, (s:c) As an improvement of the COMA system [93], Arnold [88] 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 ($\geqslant$, $\leqslant$, $\equiv$). 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 8. *Part-of correspondences with same target member:*

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

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

iMAP, Dhamankar et al. [73] Relational database schema to relational database schema, (c:s) As seen in the previous section, the iMAP system [73] uses a set 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 combinations of concatenations of the picked source attribute and 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.

Xu and Embley [85, 86] Schema to schema, (s:c) (c:s) Xu and Embley [85] propose a similar approach to iMap's date matcher. It uses a user-specified domain ontology as mediator between the two schemata to align. The papers do not specify the kind of schemata (XML or DB) aligned to the ontology. 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 [86] 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 [87] Database schema to database schema, (c:s) Warren and Tompa [87] 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.

Example 9. *A correspondence which can be output by this approach would be:*
o_2*:username = concatenation(substr(*o_1*:firstName,1), substr(*o_1*:lastName,6)), with substr(x,n) a function giving the first n characters of the string x.*

PORSCHE, Saleem et al. [90] XML schema to XML schema, (s:c) (c:s) PORSCHE (Performance ORiented SCHEma Matching) [90] 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 the 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 (e.g.,s_1*:address*) is similar to a leaf node (e.g.,s_2*:address*), a (c:s) correspondence is created between the leaf node s_2*:address* and the leaf nodes descending from s_1*:address* (e.g., s_1*:street*, s_1*:city*). 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 un-annotated sets of properties. For example, *{*s_1*:street,*s_1*:city}={*s_2*:address}* could be an output correspondence.

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. [91] Database schema to database schema, (s:c) (c:s) (c:c) DCM (Dual Correlation Mining) [91] is a holistic schema matching system. It aligns 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. [92] Database schema to database schema, (s:c) (c:s) (c:c) HSM (Holistic Schema Matching) [92] is very similar to DCM [91] 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.

Wu et al. [89] Database schema to database schema, (s:c) (c:s) Wu *et al.* [89] 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, or else of these values: *{passengers}=is-a{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.

5.3. 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 [94, 95], others take them as input [55, 67, 96, 97]. Most approaches perform the path search on the graph-like or tree-like structure of the schemata/ontologies directly whereas [97] creates a *mapping graph* on which the search will be performed.

An et al. [96] Database schema to ontology, (s:c) An *et al.* [96] 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 $u \equiv s$ and $v \equiv t$ 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. [67], Yan et al. [55] Relational database schema to relational database schema, (s:c) (c:s) (c:c) Based on structural information of relational databases schemata, the Clio system[5] [55, 67] 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* = *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 him or 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. The Clio system is user-oriented: the user intervenes at every step of the matching process. What Clio does automatically is find the path between the attributes and tables to complete the input *value correspondences*. It also automatically transforms the correspondences into SQL queries.

Ontograte, Qin et al. [94], Dou et al. [95] Ontology to ontology, (c:c) OntoGrate [94] is a framework that mines *frequent queries* and outputs them as conjunctive first-order logic formulas. The system can deal with ontology matching [94] and was adapted to relational database schema matching in [95] 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 correspondence or simple correspondence is discovered. Once the simple correspondences are found, a *group generator* process generates groups of entities closely related to a source property. 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

[5] http://www.almaden.ibm.com/cs/projects/criollo/

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.

Example 10. *The simple matching phase computed:*
- $\forall x$, o_1*:Person(x)* $\equiv$ o_2*:Person(x)*
- $\forall x,y$, o_1*:email(x,y)* $\equiv$ o_2*:contactEmail(x,y)*

However, the last correspondence is wrong as it is and considered incomplete because
- o_1*:Person is the domain* o_1*:email*
- o_2*:Paper is the domain* o_2*:contactEmail*

The group entity algorithm starts with the following entity groups:
- *source: {*o_1*:Person,*o_1*:email}*
- *target: {*o_2*:Paper,* o_2*:contactEmail}*

The process searches both ontologies so that two equivalent classes can be found in the groups: the o_2*:writes property and its domain* o_2*:Paper are added to the target group:*
- *source: {*o_1*:Person,*o_1*:email}*
- *target: {*o_2*:Person,*o_2*:writes,*o_2*:Paper,* o_2*:contactEmail}*

If the matched instances give the entity groups enough support, the following correspondence is output:
$\forall x,y$, o_1*:Person(x)* $\wedge$ o_2*:email(x,y)* $\equiv$ $\exists z$, o_2*:Person(x)* $\wedge$ o_2*:writes(x,z)* $\wedge$ o_2*:Paper(z)* $\wedge$ o_2*:contactEmail(z,y)*

An and Song [97] Ontology to ontology, (c:c) An and Song [97] introduce the concept of mapping graph between two conceptual models (defined as graph-like structures involving concepts, relations and attributes, e.g., entity-relationship diagrams, UML class diagrams, OWL ontologies). This process relies on a simple alignment between the concepts of the conceptual models. The first step of the approach is to generate the mapping graph between the conceptual models. The nodes of a mapping graph represent pairs of concepts from the two conceptual models. For example *(*o_1*:A,* o_2*:B)* and *(*o_1*:C,*o_2*:D)* are two nodes of the mapping graph, o_1*:A* and o_1*:C* being classes (concepts) of a source ontology and o_2*:B* and o_2*:D* 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. If the simple alignment states that $\forall x$, o_1*:A(x)* $\equiv$ o_2*:B(x)* and $\forall x$, o_1*:C(x)* $\equiv$ o_2*:D(x)*, then the approach will look for a path between *(*o_1*:A,* o_2*:B)* and *(*o_1*:C,*o_2*:D)*.

Example 11. *If* $\forall x$, o_1*:Reviewer(x)* $\equiv$ o_2*:Reviewer(x) and* $\forall x$, o_1*:Paper(x)* $\equiv$ o_2*:Paper(x) are two correspondences in an input alignment, a path between the nodes (*o_1*:Reviewer,*o_2*:Reviewer) and (*o_1*:Paper,*o_2*:Paper) of the mapping graph will be sought. The mapping graph edges are products of the source and target relations, as well as identity, subclass-of, part-of properties. A path in the mapping graph could be as follows, where the nodes are marked between parenthesis () and the edges between brackets* `- -[]- ->`.

```
(o1:Reviewer,o2:Reviewer)
  --[o1:reviewerOf,o2:writesReview]-->
     (o1:Paper,o2:Review)
       --[Identity,o2:reviewOf]-->
              (o1:Paper,o2:Paper)
```

The correspondence translating this path is $\forall x,y$, o_1*:Reviewer(x)* $\wedge$ o_1*:reviewerOf(x,y)* $\wedge$ o_1*:Paper(y)* $\equiv$ $\exists z$, o_2*:Reviewer(x)* $\wedge$ o_2*:writesReview(x,z)* $\wedge$ o_2*:Review(z)* $\wedge$ o_2*:reviewOf(x,y)* $\wedge$ o_2*:Paper(y)*

5.4. Tree

While some approaches [56, 98, 99] rely on a *semantic tree* derived from the schema The approaches focusing on structural transformations between two trees (addition of a node, deletion of an attribute, etc.) such as [100, 101] often rely on tree-structure. However, they are out of the scope of this study as they are part of the ontology evolution field. Other approaches such as [102, 103] use tree-based algorithms such as genetic programming. However they do not consider the schemas or ontologies as trees and are therefore not classified in this category.

MapOnto, An et al. [98, 99] Relational database schema to ontology [98], XML schema to ontology [99], (c:c) MapOnto[6] [98, 99], 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 [98] and an XML schema to ontology [99]. Both

[6] http://www.cs.torOnto.edu/semanticweb/mapOnto/

approaches take simple correspondences between the schema attributes and the ontology 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 [98]) 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.

Example 12. *Let PAPERS(id,title,accepted) be a table from a relational database schema. The following correspondence with an ontology o can be obtained by this approach, note that the correspondence is expressed in the format used by the authors:*
PAPERS(id,title,author) :- o:Paper(x) ∧ o:paperId(x,id) ∧ o:title(x,title) ∧ o:Author(y) ∧ o:authorOf(x,y) ∧ o:name(y,author)

KARMA, Knoblock et al. [56, 104] Relational database schema to ontology, (s:c),(c:s),(c:c) KARMA[7] [56, 104] is a semi-automatic database to ontology matching system. Other types of structured data such as JSON or XML files can be processed by KARMA: they are transformed into relational database in a first step following a few rules. KARMA has two parts: a database to ontology matching part presented in [56] and a programming-by-example algorithm [104] to create data transformation functions which falls in the *No structure* category. The database to ontology approach is similar to those of An *et al.* [98, 99] as it is based on a Steiner-tree algorithm and outputs FOL-like formula as alignments (as in example 12). It can be categorised as *Tree* based and will output *(c:c)* correspondences. The matching process is articulated in 4 steps during which the user can intervene to correct or refine the correspondences. The first step consists in finding correspondences between the columns of one of the source database tables and the target ontology. The ontology member of the correspondence can be a class or a pair of property-domain or subclass of domain. These correspondences are found using a conditional random field trained with labelled data (column names, values and associated ontology entity). The training labelled data can be obtained from previous user assignments or generated using feature vectors based on the names and values of the columns. The second step consists in constructing a graph linking the ontology entities from the previous step together by using object properties and hierarchical relations of the ontology. The reachable classes from the ontology are added as nodes of the graph. The user can edit the graph by changing the correspondences with the ontology, edges of the graphs, generate multiple instances of a class. In the third step, a Steiner-tree algorithm looks for the minimum-weight tree in the graph that spans all nodes. Finally, the computed Steiner-tree is transformed into a FOL-like formula as target member of the correspondence. Example 13 shows a correspondence output by KARMA.

Example 13. *Let PAPERS(id,title,accepted) be a table from a relational database schema. The pseudo-FOL correspondence with an ontology o can be obtained by KARMA, the uri function builds URIs for class instances from an id or foreign key id, the correspondence is written using the authors' format:*
PAPERS(id,title,authorId) → o:Paper(uri(id)) ∧
o:Author(uri(authorId)) ∧
o:authorOf(uri(id),uri(authorId)) ∧
o:title(uri(id),title)

5.5. No structure

The approaches described in this section do not follow any of the above structures. While [105] is based on Inductive Logic Programming and builds its correspondences in a *ad. hoc* manner, [72] uses Markov Logic Networks for combinatorial exploration, [106] uses classifying techniques to generate block correspondences, [73] overlaps numeric searcher using context-free grammar for equation discovery, and finally [102, 103] use genetic programming to combine data value transformation functions.

Hu et al. [105] Ontology to ontology, (s:c) The approach proposed by Hu *et al.* [105] uses Inductive Logic Programming (ILP) techniques to discover complex alignments. This technique is inspired by Stuckenschmidt *et al.* [107]. 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

[7] https://github.com/usc-isi-i2/Web-Karma

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 information 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 information gain metric involved in the process is based on the number of facts (instances or instance pairs) which support the correspondence or not.

Example 14. *At the first iteration of the process, only the target predicate is given. Let us consider the case of an object property as target predicate:*

$\forall x,y, \rightarrow o_2{:}reviewerOf(x,y)$

All possible pairs of common instances are classified as positive binding or negative binding with regards to whether they instanciate o_2:reviewerOf or not. The predicate with the biggest information gain (calculated from the positive and negative bindings) over the instance pairs is added to the correspondence:

$\forall x,y, \exists z, o_1{:}writesReview(x,z) \rightarrow o_2{:}reviewerOf(x,y)$

The process is iterated until no more positive binding is left to find or the number of predicates in the correspondence has reached a threshold.

Thiéblin et al. [108] Ontology to ontology, (s:c) (c:s) (c:c) In [108], only class expression correspondences are sought. The approach takes as input a set of SPARQL queries over the source ontology defined as *Competency Questions for Alignment* (CQAs). These CQAs guide the matching process: the answers to each CQA are matched to instances of the target ontology. Then, the surroundings of these target instances are lexically compared to the CQA. The surroundings include the triples in which the target instance appears and the type of the objects or subjects of these triples which are not the target instance. The labels of the CQA used for comparison in the matching process are those of the entities which appear in the CQA. To find a correspondence, the two ontologies must have at least a common instance per CQA. The instance matching process uses existing links or the sharing of a label. The SPARQL query (CQA) is turned into a DL formula to become the source member of the correspondence. The most similar surroundings of the target instances (triple with or without object/subject type) are turned into DL formula to become the target member of the correspondence. The form of the correspondence depends on the structure of the CQA and the most similar surroundings of the target instances.

Example 15. *Let a CQA over the source ontology o_1 be "Which are the accepted papers ?" which in SPARQL gives:*

```
SELECT ?x WHERE{?x a o1:AcceptedPaper}.
```

The CQA labels are those of o_1:AcceptedPaper: "accepted paper".

An answer to the CQA is o_1:paper1. o_1:paper1 has an existing owl:sameAs link to a target instance o_2:paper2. The approach considers the surroundings of o_2:paper2:

- $o_2{:}hasAuthor(o_2{:}paper2,o_2{:}aut7) \wedge o_2{:}Author(o_2{:}aut7)$
- $o_2{:}decision(o_2{:}paper2, \text{"accepted"})$

If the label of the object/subject is more similar to the CQA than its type, only its value is kept.

The triple o_2:decision(o_2:paper2,"accepted") has the highest similarity to the CQA labels. The following correspondence is created:

$\forall x, o_1{:}AcceptedPaper(x) \equiv o_2{:}decision(x,\text{"accepted"})$

KAOM, Jiang et al. [72] Ontology to ontology, (s:c) (c:s) (c:c) KAOM (Knowledge Aware Ontology Matching) is a system proposed by Jiang *et al.* [72]. 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 do not have to be semantically exact. 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.* [68, 69] 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 a priori algorithm or manually written rules were translated as *knowledge rules* for three different test cases.

Example 16. *A knowledge rule could be"Many reviewers are also authors of paper", which would be in*

pseudo-FOL (⇝ seen as a "is often true" relation): ∀x, o_1:Reviewer(x) ⇝ ∃y, o_2:authorOf(x,y) ∧ o_2:Paper(y).

iMAP, Dhamankar et al. [73] Relational database schema to relational database schema, (c:s) As seen previously, the iMAP system [73] uses a set 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.

Nunes et al. [102] 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.* [102] 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. [103] Schema to schema (relational database or XML), (s:c) (c:s) (c:c) De Carvalho *et al.* [103] 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.* [102]) 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. [102] it can detect *(c:c)* correspondences thanks to its modelling. However the process may require more iterations than [102].

KARMA, Knoblock et al. [56, 104] Relational database schema to ontology, (s:c),(c:s),(c:c) KARMA [56, 104] is a semi-automatic database to ontology matching system. Other types of structured data such as JSON or XML files can be processed by KARMA: they are transformed into relational database in a first step following a few rules. KARMA has two parts: a database to ontology matching part presented in [56] (this approach is described in the *Tree* category) and a programming-by-example algorithm [104] to create data transformation functions which falls in the *No structure* category. The latter part considers the transformation functions as programs divided into subprograms which are to be applied to the data to transform it. At the beginning of the process, an example of source data (a table cell or row value) is given to the user and he or she gives what he or she expects as a results. This first pair of values constitutes an example and a program (transformation function) is then synthesized and applied to the other instances of the data. The user iteratively corrects the wrong translated data, giving new examples from which the process refines its program by detecting and changing incorrect subprograms. The basic operations (or segments) of a program or subprogram are string operations (subtring, concatenation, recognizing a number, etc.). As the input and the output of the process can cover one or many columns of the source and target tables, this part of KARMA can output *(s:c), (c:s)* or *(c:c)* correspondences.

Example 17. *A first example "PaperABC written by AuthorTT strong accept 2016" from the source database is given to the user. The user gives the expected value "PaperABC (2016)". This first pair of values constitutes an example and a program (transformation function) is synthesized. For example, out of all*

the possible programs (called hypothesis) one could be:

```
transform(val):
pos1=val.indexOf(START,WORD,1)
pos2=val.indexOf(WORD,BNK,1)
pos3=val.indexOf(BNK,NUM,1)
pos4=val.indexOf(NUM,END,1)
output= val.substr(pos1,pos2)+' ('+val.subtr(
    pos3,pos4)+')'
return output
```

where indexOf(LEFT, RIGHT, N) takes the left and right context of the occurence and N precises the n-th occurence. START is the beginning of the value, END its end. WORD represents a ([A-Za-z]+) string, NUM a number, BNK a whitespace. This program is then applied to the other instances of the data. The user iteratively corrects the wrong translated data, giving new examples from which the process refines its program (the hypothesis space will be reduced).

BMO, Hu et al. [106] 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.

5.6. Summary

The proposed classification is based on two main axes, the output (type of correspondence) and process (guiding structure) dimensions of the approaches. The following tables are organised as the types of correspondences: logic relations, transformation functions and blocks. The order of the approach for each type of correspondence follows the one in the survey (atomic pattern, composite pattern, path, tree, no structure).

Table 6 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 input (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 the Semantic Web.

Table 7 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.

In the Ontology Matching book [16], the basic matching techniques are classified as follow:

- *Formal resource-based*: relying on formal evidence: upper-level ontology, domain-specific ontology, linked data, linguistic frames, alignment
- *Informal resource-based*: relying on informal evidence: directory, annotated resources, web forms
- *String-based*: use of string similarity: name similarity, description similarity, global namespace
- *Language-based*: use of linguistic techniques: tokenisation, lemmatisation, thesauri, lexicon, morphology
- *Constraint-based*: use of internal ontology constraints: types, key properties
- *Taxonomy-based*: consider the specialisation relation of the ontologies: taxonomy, structure
- *Graph-based*: consider the ontologies as graphs: graph homomorphism, path, children, leaves, correspondence patterns
- *Instance-based*: comparison of sets of individuals: data analysis, statistics
- *Model-based*: based on the semantic interpretation: SAT solvers, DL reasoners

The complex matching approaches are described according to this classification in Table 8. The majority of them combine different matching techniques.

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 to the kind of evidence they exploit (ontology-

Table 6

Input (type of aligned knowledge representation model and type of additional input information) and output (correspondences members form, format) of the approaches. *KRM stands for Knowledge Representation Model.*

	Work	*Type of KRM*	*Additional Input*	*Kind of correspondence*	*Output format*
Logical relations	Ritze2009 [68]	Onto to Onto	simple alignment	(s:c)	(DL)
	Ritze2010 [69]	Onto to Onto	simple alignment (opt.)	(s:c)	EDOAL
	Oliveira2018 [22]	Onto to Onto		(s:c)	Not specified
	Rouces2016 [70]	Onto to Onto		(s:c), (c:s)	SPARQL construct
	Walshe2016 (Bayes-ReCCE) [71]	Onto to Onto	matched instances	(s:c), (c:s)	EDOAL
	Jimenez2015 (BootOX) [74]	DB to Onto		(c:s)	R2RML
	Svab2009 [75]	Onto to Onto	simple alignment (opt.)	(s:c),(c:s),(c:c)	Not specified
	Parundekar2012 [80]	Onto to Onto	matched instances	(s:c), (c:s)	pseudo-DL
	Doan2003 (CGLUE) [82]	Taxo to Taxo		(s:c)	Not specified
	Parundekar2010 [81]	Onto to Onto	matched instances	(s:c), (c:s), (c:c)	pseudo-DL
	Kaabi2012 (ARCMA) [83]	Onto to Onto		(s:c)	DL
	Yan2001 (Clio) [55, 67]	DB to DB	matched instances	(s:c), (c:s),(c:c)	SQL views
	Qin2007 (OntoGrate) [94, 95]	Onto to Onto	matched instances	(c:c)	DataLog, SWRL, Web-PDDL
	An2008 [97]	Onto to Onto		(c:c)	FOL or SPARQL
	An2012 [96]	DB to Onto	web query interfaces, simple correspondences web form-onto	(s:c)	Not specified
	An2005 (MapOnto) [98]	DB to Onto	attribute-data properties correspondences	(c:c)	FOL
	An2005b (MapOnto) [99]	XML to Onto	attribute-data properties correspondences	(c:c)	FOL
	Hu2011 [105]	Onto to Onto		(c:s)	FOL
	Thieblin2018 [108]	Onto to Onto	competency questions for alignment as SPARQL queries	(s:c),(c:s),(c:c)	EDOAL
Logic/Transfo	Jiang2016 (KAOM) [72]	Onto to Onto	knowledge rules	(s:c), (c:s), (c:c)	Not specified (DL)
	Boukottaya2005 [84]	XML to XML		(s:c), (c:s), (c:c)	XSLT
	Xu2003 [85]	Schema to Schema	domain ontology	(c:s), (s:c)	Not specified
	Xu2006 [86]	Schema to Schema	domain ontology	(c:s), (s:c), (c:c)	Not specified
	Knoblock2012 (KARMA) [56, 104]	DB to Onto	examples for data transformation functions	(s:c),(c:s),(c:c)	FOL
Transfo. functions	Dhamankar2004 (iMAP) [73]	DB to DB	domain constraints and value distribution	(c:s)	equations
	Arnold2013 (COMA++) [88]	Onto to Onto		(s:c)	Not specified
	Warren2006 [87]	DB to DB		(c:s)	SQL queries
	Nunes2011 [102]	Onto to Onto		(c:s)	equations
	deCarvalho2013 [103]	Schema to Schema		(c:s), (s:c), (c:c)	equations
Blocks	Saleem2008 (PORSCHE) [90]	XML to XML	abbreviation table	(c:s), (s:c)	Not specified
	He2004 (DCM) [91]	DB to DB	web query interfaces	(s:c), (c:s), (c:c)	sets
	Su2006 (HSM) [92]	DB to DB	web query interfaces	(s:c), (c:s), (c:c)	sets
	Wu2004 [89]	DB to DB	web query interfaces	(s:c), (c:s)	sets
	Hu2006 (BMO) [106]	Onto to Onto		(s:c), (c:s), (c:c)	sets

Table 7
Process characteristics of the approaches based on the proposed classification

	Article	Guiding structure	fixed to fixed	fixed to unfixed	unfixed to unfixed	Ontology-level evidence	Instance-level evidence	Other
Logical relations	Ritze2009 [68]	Atomic patterns	•			•		
	Ritze2010 [69]	Atomic patterns	•			•		
	Oliveira2018 [22]	Atomic patterns	•			•		Compound
	Rouces2016 [70]	Atomic patterns	•			•		
	Walshe2016 (Bayes-ReCCE) [71]	Atomic patterns	•				•	
	Jimenez2015 (BootOX) [74]	Atomic patterns	•			•		
	Svab2009 [75]	Composite patterns		•	•	•		
	Parundekar2012 [80]	Composite patterns		•			•	
	Parundekar2010 [81]	Composite patterns			•		•	
	Doan2003 (CGLUE) [82]	Composite patterns		•			•	
	Kaabi2012 (ARCMA) [83]	Composite patterns		•		•	•	
	Yan2001 (Clio) [55, 67]	Path to Path			•	•	•	
	Qin2007 (OntoGrate) [94, 95]	Path to Path			•	•	•	
	An2008 [97]	Path to Path			•	•		
	An2012 [96]	Path to Path		•		•		
	An2005 (MapOnto) [98]	Tree to tree			•	•		
	An2005b (MapOnto) [99]	Tree to tree			•	•		
	Hu2011 [105]	No structure		•		•	•	
	Thieblin2018 [108]	No structure		•		•	•	
Logic/Transfo	Jiang2016 (KAOM) [72]	Atomic patterns, No structure	•		•	•	•	
	Boukottaya2005 [84]	Composite patterns		•		•		
	Xu2003 [85]	Composite patterns	•	•		•	•	
	Xu2006 [86]	Composite patterns, Path to path	•	•	•	•	•	
	Knoblock2012 (KARMA) [56, 104]	Tree to tree, No structure		•	•	•	•	
Transfo. functions	Dhamankar2004 (iMAP) [73]	Atomic patterns, Composite patterns, No structure	•	•			•	
	Arnold2013 (COMA++) [88]	Composite patterns		•		•		
	Warren2006 [87]	Composite patterns		•			•	
	Nunes2011 [102]	No structure		•			•	
	deCarvalho2013 [103]	No structure			•		•	
Blocks	Saleem2008 (PORSCHE) [90]	Composite patterns		•		•		Holistic
	He2004 (DCM) [91]	Composite patterns			•	•		Holistic
	Su2006 (HSM) [92]	Composite patterns			•	•		Holistic
	Wu2004 [89]	Composite patterns		•		•		
	Hu2006 (BMO) [106]	No structure			•	•		

Table 8
Classification of the complex matchers on [16]'s basic techniques

	Approach	Formal resource-based	Informal resource-based	String-based	Language-based	Constraint-based	Taxonomy-based	Graph-based	Instance-based	Model-based
Logical relations	Ritze2009 [68]			•	•	•	•	•		
	Ritze2010 [69]			•	•	•	•	•		
	Oliveira2018 [22]			•						
	Rouces2016 [70]	•		•	•			•		
	Walshe2016 (Bayes-ReCCE) [71]	•						•	•	
	Jimenez2015 (BootOX) [74]				•	•		•		
	Svab2009 [75]			•				•		
	Parundekar2012 [80]	•						•	•	
	Parundekar2010 [81]	•						•	•	
	Doan2003 (CGLUE) [82]			•			•		•	
	Kaabi2012 (ARCMA) [83]			•			•		•	
	Yan2001 (Clio) [55, 67]					•		•	•	
	Qin (OntoGrate) [94, 95]							•	•	
	An2008 [97]	•					•	•		
	An2012 [96]		•					•		
	An2005 (MapOnto) [98]	•						•		
	An2005b (MapOnto) [99]	•						•		
	Hu2011 [105]								•	
	Thieblin2018 [108]	•		•				•	•	
Logic/Transfo	Jiang2016 (KAOM) [72]	•		•		•			•	•
	Boukottaya2005 [84]				•	•	•	•		
	Xu2003 [85]	•		•				•	•	
	Xu2006 [86]	•		•	•			•	•	
	Knoblock2012 (KARMA) [56, 104]		•	•				•	•	
Transfo. functions	Dhamankar2004 (iMap) [73]	•		•		•		•	•	
	Arnold2013 (COMA++) [88]			•			•			
	Warren2006 [87]								•	
	Nunes2011 [102]								•	
	deCarvalho [103]								•	
Blocks	Saleem2008 (PORSCHE) [90]				•		•			
	He2004 (DCM) [91]		•							
	Su2006 (HSM) [92]		•							
	Wu2004 [89]		•	•	•	•	•		•	
	Hu2006 (BMO) [106]			•				•		

level or instance-level), as done in different surveys in the field. This classification was applied in the last 2 columns of Table 7. Most approaches use the ontology-level information as evidence. The approaches which output transformation functions mostly rely on instance-level information.

6. Evaluation of complex matchers

This section discusses the evaluation of complex alignments regarding the datasets and metrics. Most of the approaches were manually evaluated. The only automatic evaluations of complex alignments were often done on approach-tailored datasets (e.g., one kind of correspondence only).

6.1. Complex alignment datasets

The diverse approaches discussed in this paper exploit a variety of knowledge representation models (e.g., XML schemata, ontologies) and resources (e.g., linked instances, web forms, etc.). They also generate different types of correspondences. This makes their evaluation difficult and heterogeneous. The approaches were mostly evaluated on pairs of knowledge representation models over a wide range of domains (geography, biomedicine, conference organisation, sports, companies, libraries, etc.). Few systems were evaluated by comparison to a reference alignment, and even fewer of these reference alignments were made available online. In this section are presented the datasets available online with reference complex alignments and the benchmarks which deal with complex alignments. They are summarized in Table 9.

In the domain of schema matching (database or XML schema), dedicated complex alignment datasets have been constructed for evaluating the approaches dealing with these schemata. In general, these datasets contain mostly transformation functions. For instance, the Illinois semantic integration archive [109] is a dataset of complex correspondences on value transformations (e.g. string concatenation) in the inventory and real estate domain. This dataset only contains correspondences between schemata with transformation functions. The UIUC Web integration Repository [110] has also been reused by various schema matching approaches for their evaluation. It is a repository of schemata and query forms. Other repositories such as the UCI Machine Learning Repository [111] have been adapted in some matcher evaluation for the purpose of evaluating schema matching approaches. However, the resulting adaptation was not published. XBenchMatch [112] is a benchmark for XML schema matching. The reference alignments of the *person* dataset contains correspondences with string concatenation.

For the purpose of evaluating matching hybrid structures, the RODI Benchmark [113] 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, Geodata ontology, Oil and gas ontology. The schemata are either derived from the ontologies themselves or curated on the Web. The RODI deals with R2RML alignment and uses reference SPARQL and SQL queries to assess the quality of the alignment.

SPIMBench [45] is an instance matching benchmark based but it could be used for complex ontology alignment evaluation. A set of transformations were applied to the BBC core and other domain ontologies in order to get derived ontologies with the same instances. The transformation rules can be considered as correspondences. Some transformation rules are even complex correspondences (either logic relations or value transformation functions). Each set of transformation rules between two ontologies was documented in the SPIMBench vocabulary and constitutes a reference complex alignment. However, the reference alignment is not considered in the evaluation process of the benchmark, which only focuses on instance matching.

This year, the first complex track of the Ontology Alignment Evaluation Initiative was conducted. It included four datasets [114] among which the GeoLink dataset [18] and a consensus version of [115], having a reference alignment were used.

Of all the datasets presented in Table 9, only SPIMBench contains common or matched instances. However, the derived datasets are synthetic. Because they needed common instances, some matchers have been evaluated on LOD repositories (DBpedia, Yago, Agrovoc, Geospecies, etc.) [71, 80, 81, 105, 108] or custom-made knowledge bases [72, 73, 102, 103].

6.2. Evaluation metrics

Complex matching evaluation can be performed under various dimensions such as time execution or the quality of the output alignment. In this section, the complex alignment quality metrics are presented. These metrics do not include approach-specific met-

Table 9
Datasets for complex alignment evaluation. KRM stands for Knowledge Representation Model.

Dataset	type of KRM	Logic	Transf.	Alignment format	URL
Conference	Ontologies	√	√	EDOAL	http://doi.org/10.6084/m9.figshare.4986368
GeoLink	Ontologies	√		EDOAL	http://doi.org/10.6084/m9.figshare.5907172
Hydrography	Ontologies	√		EDOAL	http://oaei.ontologymatching.org/2018/complex/#hydrography
SPIMBench	Ontologies	√	√	SPIMBench	https://github.com/jsaveta/SPIMBench
Real estate I	XML		√	XML schema	http://pages.cs.wisc.edu/~anhai/wisc-si-archive/domains/real_estate1.html
Real estate II	XML		√	own syntax	http://pages.cs.wisc.edu/~anhai/wisc-si-archive/domains/real_estate2.html
Inventory	XML		√	own syntax	http://pages.cs.wisc.edu/~anhai/wisc-si-archive/domains/inventory.html
XBenchMatch	XML		√	own syntax	https://perso.liris.cnrs.fr/fabien.duchateau/research/tools/xbenchmatch/
RODI	DB to onto	√	√	SQL/SPARQL	https://github.com/chrpin/rodi

rics as defined in [91, 92, 106] but the ones that can be generalised to all complex alignments.

The most usual metrics, adapted from information retrieval, used for evaluating the quality of alignments with respect to a reference one are *precision* and *recall*, combined into *F-measure*. The calculation of the recall and F-measure requires a reference alignment whereas the precision alone can be assessed by classifying correspondences are true positives or false positives. The usual precision and recall are the metrics which were the most used in the evaluation of the approaches. However, as reference alignments are not always available, the precision was often the only metric computed. The evaluation was also often manually performed which makes it time-consuming task requiring experts about the aligned ontologies. The precision and recall metrics were adapted into weighted precision and recall, relaxed precision and recall, and semantic precision and recall. The *weighted* precision and recall take the confidence value of a correspondence into account. The *relaxed* precision and recall [116] take the subsumption relations into account: a correspondence is not discarded if it states an equivalence instead of a subsumption, its "score value" will be 0.5 instead of 1 for example. The *semantic* precision and recall [117] considers the alignments as sets of axioms and is measured by comparing their deductive closure, i.e., the set of axioms which can be derived from the alignment together with the ontologies.

The metrics of *accuracy* or *top−x accuracy* have been used in various evaluations [73, 74, 82, 91, 92, 96, 103, 108] 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 predefined questions having a correct answer. A "question" in this context could be a source entity to be matched and the "answers" the correspondences having this entity as source member. Some approaches output various answers for each question, e.g., a ranked list of correspondences for each source 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 source entities for which the correct correspondence is in the three best correspondences output by the system.

During this year's OAEI complex track, the evaluation was mostly manual. The usual precision and recall metrics were reused for the Conference dataset. For the Hydrography and GeoLink dataset, three tasks were defined, but the matchers could be evaluated on the first one only using precision and recall:

- Finding the entities which belong together in a correspondence, regardless to the correspondence structure;
- Finding the correct correspondence structure given the set of entities to match;
- Finding the correspondences from scratch.

Semantic precision and recall were regarded as perspectives. Finally, the Taxon dataset was manually evaluated with a usual precision metric and within a query rewriting scenario. The *accuracy*, as the percentage of queries well rewritten, was also computed.

All the metrics presented before need either a reference alignment or a manual evaluation. Even with a reference alignment, the evaluation is not straightforward due to the difficulty of comparing two complex correspondences.

6.3. Summary

The different approaches discussed in this survey have been evaluated on a variety of datasets.

Until recently, there was very little effort on complex alignment evaluation. The latest works propose datasets. Only RODI [113] is an automated benchmark. Most of the OAEI complex track evaluation is still manually performed. This comes from the difficulty of comparing two complex correspondences. For example, $\forall$ o_1*:Author(x)* $\equiv$ $\exists$*y,* o_2*:authorOf(x,y)* is the same as $\forall$ o_1*:Author(x)* $\equiv$ $\exists$*y,* o_2*:writtenBy(y,x)*. Given this example, semantic precision and recall could integrate the fact that the two example expressions mean the same thing given that o_2*:authorOf* is the inverse property of o_2*:writtenBy*. However, pattern-based ontology formats such as EDOAL (or OWL) may lead to other problems. For example, the correspondence $\forall$*x,* o_1*:AcceptedPaper(x)* $\equiv$ $\exists$ o_2*:acceptedBy(x,y)* can be expressed using an occurrence restriction on the o_2*:acceptedBy* property (a paper that was accepted at least once) or by using a type restriction to the *owl:Thing* class (a paper that was accepted by some *Thing*). This would prevent the calculation of semantic precision and recall as the axioms alone would not be comparable (*owl:minCardinality* versus *owl:someValuesFrom*). An alternative would be to assess and compare the interpretation of the correspondences (at instance level) but this would require consistently populated ontologies which might not be the case on the LOD (e.g., DBpedia contains inconsistencies) nor on the usual OAEI datasets (e.g., conference, anatomy are not populated).

The relation of the correspondences ($\equiv, \geqslant, \leqslant$) is also not taken into account in the evaluation process as most matchers only consider equivalence. The confidence given to a correspondence is taken into account when dealing with top-x accuracy. The weighted or relaxed precision and recall metrics [116] could be adapted to deal with it.

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 [118], for a library application [119] or the Taxon track of the complex OAEI track [114]. The metrics would then take into account the suitability of the output alignment to the given task.

7. Discussion

Recently, the community has gained interest over complex alignments. This comes probably from the fact that applications needing interoperability find simple alignment not sufficient.

The XML and database fields are older than ontology matching and ODBA, therefore, they have stable standards such as XSLT, XQuery and SQL. In the ODBA community, R2RML has become the norm, many extensions to R2RML are proposed and there is a proliferation of edition and visualisation tools, as well as matching approaches. In the ontology matching community, this proliferation is not so marked. Event if various alignment formats have been proposed (c.f. §3.1), the EDOAL vocabulary implemented in the Alignment API can be seen as an up-coming standard. It has indeed been used for the first OAEI track. However, EDOAL has already been faced with expressiveness limitations, as discussed in [18]. Moreover, there is no edition or visualisation tool for this format, which makes it usable or browse-able by experts only. SPARQL could be an alternative to EDOAL as it does not suffer from such limitations and can be directly executed.

The study of the approaches in this survey shows that, unlike what intuition may suggest, matching more expressive knowledge representation models does not imply applying more sophisticated techniques. Most approaches consider the knowledge representation models as graphs, trees or pools of annotated data regardless their expressiveness. These common representations leads to similar techniques over diverse knowledge representation models. While the use of instance data evidence is valuable for the matching process, statistical approaches are directly impacted by the quality of this data. They can be faced with the sparseness problem or with a specific corpus distribution that leads to incorrect correspondences. For example, 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. The notion of contextual alignment [120] could help specify to which extent the alignment is valid.

Another point is related to the user involvement in the complex matching process, which is underexploited in complex matching. However, this aspect, related to the matching process [121, 122], related to the visualisation and edition of complex correspondences, is an important issue to be addressed in the future.

Regarding the evaluation of complex alignments, correspondence comparison remains a problem. The perspective of a benchmark with a reference alignment, real-life ontologies populated with controlled instances and metrics based on these instances, would be a useful resource in the field. As the interpretation of an ontology can vary from user to user, having a consensual benchmark with correspondence confidences reflecting the agreement between annotators, as in [123], could be also an interesting resource. Another direction would be to evaluate the complex alignments over a real-life application. The first OAEI complex track could –hopefully– stimulate new works on complex ontology matching, evaluation, visualisation, etc.

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: 10.1007/116034125. ISBN 978-3-540-31001-3 978-3-540-31447-9.

[2] P. Szekely, C.A. Knoblock, F. Yang, X. Zhu, E.E. Fink, R. Allen and G. Goodlander, Connecting the Smithsonian American Art Museum to the Linked Data Cloud, in: *The Semantic Web: Semantics and Big Data*, Vol. 7882, D. Hutchison, T. Kanade, J. Kittler, J.M. Kleinberg, F. Mattern, J.C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M.Y. Vardi, G. Weikum, P. Cimiano, O. Corcho, V. Presutti, L. Hollink and S. Rudolph, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 593–607. ISBN 978-3-642-38287-1 978-3-642-38288-8. doi:10.1007/978-3-642-38288-8_40. http://link.springer.com/10.1007/978-3-642-38288-8_40.

[3] V. Boer, J. Wielemaker, J. Gent, M. Hildebrand, A. Isaac, J. Van Ossenbruggen and G. Schreiber, Supporting Linked Data Production for Cultural Heritage Institutes: The Amsterdam Museum Case Study, in: *ACM Journal of Experimental Algorithms - JEA*, 2012. doi:10.1007/978-3-642-30284-8_56.

[4] C. Kakali, I. Lourdi, T. Stasinopoulou, L. Bountouri, C. Papatheodorou, M. Doerr and M. Gergatsoulis, Integrating Dublin Core Metadata for Cultural Heritage Collections Using Ontologies, *International Conference on Dublin Core and Metadata Applications* **0**(0) (2007), 128–139, ISSN 1939-1366. http://dcpapers.dublincore.org/pubs/article/view/871.

[5] T. Nurmikko-Fuller, K.R. Page, P. Willcox, J. Jett, C. Maden, T. Cole, C. Fallaw, M. Senseney and J.S. Downie, Building complex research collections in digital libraries: A survey of ontology implications, in: *Proceedings of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries*, ACM, 2015, pp. 169–172.

[6] E. Thiéblin, F. Amarger, N. Hernandez, C. Roussey and C.T. Dos Santos, Cross-querying LOD datasets using complex alignments: an application to agronomic taxa, in: *Research Conference on Metadata and Semantics Research*, Springer, 2017, pp. 25–37.

[7] V. Jouhet, F. Mougin, B. Bréchat and F. Thiessard, Building a model for disease classification integration in oncology, an approach based on the national cancer institute thesaurus, *Journal of biomedical semantics* **8**(1) (2017), 6.

[8] K.W. Fung and J. Xu, Synergism between the Mapping Projects from SNOMED CT to ICD-10 and ICD-10-CM, in: *AMIA Annual Symposium Proceedings*, Vol. 2012, American Medical Informatics Association, 2012, p. 218.

[9] K. Giannangelo and J. Millar, Mapping SNOMED CT to ICD-10., 2012.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] J. Euzenat and P. Shvaiko, *Ontology Matching, Second edition*, 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.

[17] 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.

[18] L. Zhou, M. Cheatham, A. Krisnadhi and P. Hitzler, A complex alignment benchmark: Geolink dataset, in: *International Semantic Web Conference*, Springer, 2018, pp. 273–288.

[19] 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.

[20] T. Gruetze, C. Böhm and F. Naumann, Holistic and Scalable Ontology Alignment for Linked Open Data, in: *Proceedings of the 5th Linked Data on the Web Workshop at the 21th WWW*, 2012.

[21] I. Megdiche, O. Teste and C. Trojahn, An Extensible Linear Approach for Holistic Ontology Matching, in: *Proceedings of the 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016*, 2016, pp. 393–410.

[22] 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. https://jbiomedsem.biomedcentral.com/articles/10.1186/s13326-017-0171-8.

[23] 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.

[24] G. Navarro, A guided tour to approximate string matching, *ACM computing surveys (CSUR)* **33**(1) (2001), 31–88.

[25] F. Zablith, G. Antoniou, M. d'Aquin, G. Flouris, H. Kondylakis, E. Motta, D. Plexousakis and M. Sabou, Ontology evolution: a process-centric survey, *The knowledge engineering review* **30**(1) (2015), 45–75.

[26] M.C.A. Klein and D. Fensel, Ontology versioning on the Semantic Web, in: *Proceedings of SWWS'01, The first Semantic Web Working Symposium, Stanford University, California, USA, July 30 - August 1, 2001*, 2001, pp. 75–91. http://www.semanticweb.org/SWWS/program/full/paper56.pdf.

[27] M. Klein, D. Fensel, A. Kiryakov and D. Ognyanov, Ontology versioning and change detection on the web, in: *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2002, pp. 197–212.

[28] L. Stojanovic, A. Maedche, B. Motik and N. Stojanovic, User-driven ontology evolution management, in: *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2002, pp. 285–300.

[29] M. Volkel, W. Winkler, Y. Sure, S.R. Kruk and M. Synak, Semversion: A versioning system for rdf and ontologies, in: *Proc. of ESWC*, 2005.

[30] N.F. Noy, M.A. Musen et al., Promptdiff: A fixed-point algorithm for comparing ontology versions, *AAAI/IAAI* **2002** (2002), 744–750.

[31] V. Papavassiliou, G. Flouris, I. Fundulaki, D. Kotzinos and V. Christophides, On detecting high-level changes in RDF/S KBs, in: *International Semantic Web Conference*, Springer, 2009, pp. 473–488.

[32] D. Zeginis, Y. Tzitzikas and V. Christophides, On the foundations of computing deltas between RDF models, in: *The Semantic Web*, Springer, 2007, pp. 637–651.

[33] M. Hartung, J. Terwilliger and E. Rahm, Recent advances in schema and ontology evolution, in: *Schema matching and mapping*, Springer, 2011, pp. 149–190.

[34] H. Stuckenschmidt and M. Klein, Integrity and change in modular ontologies, in: *IJCAI*, 2003, pp. 900–908.

[35] J.C. Dos Reis, C. Pruski, M. Da Silveira and C. Reynaud-Delaître, Understanding semantic mapping evolution by observing changes in biomedical ontologies, *Journal of biomedical informatics* **47** (2014), 71–82.

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

[37] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati and M. Zakharyaschev, Ontology-Based Data Access: A Survey, in: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 5511–5519. doi:10.24963/ijcai.2018/777. https://doi.org/10.24963/ijcai.2018/777.

[38] D. Dou, The Formal Syntax and Semantics of Web-PDDL, Technical Report, Technical report, University of Oregon, 2008.

[39] D.L. McGuinness, F. Van Harmelen et al., OWL web ontology language overview, *W3C recommendation* **10**(10) (2004), 2004.

[40] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean et al., SWRL: A semantic web rule language combining OWL and RuleML, *W3C Member submission* **21** (2004), 79.

[41] J. Euzenat, F. Scharffe and A. Zimmermann, Expressive alignment language and implementation (2007). https://hal.inria.fr/hal-00822892/.

[42] M.T. Pazienza, A. Stellato, M. Vindigni and F.M. Zanzotto, XeOML: An XML-based extensible ontology mapping language, in: *WORKSHOP ON MEANING COORDINATION AND NEGOTIATION, IN 3RD INTERNATIONAL SEMANTIC WEB CONFERENCE (ISWC-2004*, Citeseer, 2004.

[43] 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.

[44] N. Silva and J. Rocha, Semantic Web complex ontology mapping, IEEE Comput. Soc, 2003, pp. 82–88. ISBN 978-0-7695-1932-6. doi:10.1109/WI.2003.1241177. http://ieeexplore.ieee.org/document/1241177/.

[45] T. Saveta, E. Daskalaki, G. Flouris, I. Fundulaki, M. Herschel and A.-C. Ngonga Ngomo, Pushing the Limits of Instance Matching Systems: A Semantics-Aware Benchmark for Linked Data, in: *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, ACM, New York, NY, USA, 2015, pp. 105–106. ISBN 978-1-4503-3473-0. doi:10.1145/2740908.2742729. http://doi.acm.org/10.1145/2740908.2742729.

[46] M. Hert, G. Reif and H.C. Gall, A comparison of RDB-to-RDF mapping languages, in: *Proceedings of the 7th International Conference on Semantic Systems - I-Semantics '11*, ACM Press, Graz, Austria, 2011, pp. 25–32. ISBN 978-1-4503-0621-8. doi:10.1145/2063518.2063522. http://dl.acm.org/citation.cfm?doid=2063518.2063522.

[47] S. Das, S. Sundara and R. Cyganiak, R2RML: RDB to RDF Mapping Language, 2012. https://www.w3.org/TR/r2rml/.

[48] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens and R. Van de Walle, RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data., in: *LDOW*, 2014.

[49] B. De Meester, A. Dimou, R. Verborgh and E. Mannens, An Ontology to Semantically Declare and Describe Functions, in: *The Semantic Web*, Vol. 9989, H. Sack, G. Rizzo, N. Steinmetz, D. Mladenić, S. Auer and C. Lange, eds, Springer International Publishing, Cham, 2016, pp. 46–49. ISBN 978-3-319-47601-8 978-3-319-47602-5. doi:10.1007/978-3-319-47602-5_10. http://link.springer.com/10.1007/978-3-319-47602-5_10.

[50] F. Michel, L. Djimenou, C. Faron-Zucker and J. Montagnat, xR2RML: Non-relational databases to RDF mapping, Technical Report, Citeseer, 2015.

[51] A. Chortaras and G. Stamou, Mapping Diverse Data to RDF in Practice, in: *The Semantic Web – ISWC 2018*, Vol. 11136, D. Vrandečić, K. Bontcheva, M.C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L.-A. Kaf-

fee and E. Simperl, eds, Springer International Publishing, Cham, 2018, pp. 441–457. ISBN 978-3-030-00670-9 978-3-030-00671-6. doi:10.1007/978-3-030-00671-6_26. http://link.springer.com/10.1007/978-3-030-00671-6_26.

[52] J. Clark et al., Xsl transformations (xslt), *World Wide Web Consortium (W3C). URL http://www. w3. org/TR/xslt* (1999), 103.

[53] M.A. Musen, The protégé project: a look back and a look forward, *AI matters* **1**(4) (2015), 4–12.

[54] S. Hassanpour, M.J. O'Connor and A.K. Das, A Software Tool for Visualizing, Managing and Eliciting SWRL Rules, in: *The Semantic Web: Research and Applications*, Vol. 6089, D. Hutchison, T. Kanade, J. Kittler, J.M. Kleinberg, F. Mattern, J.C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M.Y. Vardi, G. Weikum, L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral and T. Tudorache, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 381–385. ISBN 978-3-642-13488-3 978-3-642-13489-0. doi:10.1007/978-3-642-13489-0_28. http://link.springer.com/10.1007/978-3-642-13489-0_28.

[55] 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.

[56] C.A. Knoblock, P. Szekely, J.L. Ambite, A. Goel, S. Gupta, K. Lerman, M. Muslea, M. Taheriyan and P. Mallick, Semi-automatically Mapping Structured Sources into the Semantic Web, in: *The Semantic Web: Research and Applications*, Vol. 7295, D. Hutchison, T. Kanade, J. Kittler, J.M. Kleinberg, F. Mattern, J.C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M.Y. Vardi, G. Weikum, E. Simperl, P. Cimiano, A. Polleres, O. Corcho and V. Presutti, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 375–390. ISBN 978-3-642-30283-1 978-3-642-30284-8. doi:10.1007/978-3-642-30284-8_32. http://link.springer.com/10.1007/978-3-642-30284-8_32.

[57] J. Rouces, G. de Melo and K. Hose, Addressing structural and linguistic heterogeneity in the Web1, *AI Communications* **31**(1) (2018), 3–18, ISSN 18758452, 09217126. doi:10.3233/AIC-170745. http://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/AIC-170745.

[58] M. Weiten, OntoSTUDIO® as a Ontology Engineering Environment, in: *Semantic Knowledge Management*, J. Davies, M. Grobelnik and D. Mladenić, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 51–60. ISBN 978-3-540-88844-4 978-3-540-88845-1. doi:10.1007/978-3-540-88845-1_5. http://link.springer.com/10.1007/978-3-540-88845-1_5.

[59] M. Weiten, D. Wenke and M. Meier-Collin, D4. 5.3. Prototype of the Ontology Mediation Software V1, Technical Report, Project IST-2003-506826 SEKT Project Report, 2005.

[60] A.C. Junior, C. Debruyne and D. O'Sullivan, Using a Block Metaphor for Representing R2RML Mappings, in: *ESWC Poster*, 2018.

[61] A.C. Junior, C. Debruyne and D. O'Sullivan, Juma: An editor that uses a block metaphor to facilitate the creation and editing of R2RML mappings, in: *European Semantic Web Conference*, Springer, 2017, pp. 87–92.

[62] Á. Sicilia, G. Nemirovski and A. Nolle, Map-On: A web-based editor for visual ontology mapping, *Semantic Web* **8**(6) (2017), 969–980.

[63] P. Heyvaert, A. Dimou, A.-L. Herregodts, R. Verborgh, D. Schuurman, E. Mannens and R. Van de Walle, RMLEditor: a graph-based mapping editor for linked data mappings, in: *International Semantic Web Conference*, Springer, 2016, pp. 709–723.

[64] D. Lembo, R. Rosati, M. Ruzzi, D.F. Savo and E. Tocci, Visualization and Management of Mappings in Ontology-based Data Access (Progress Report)., in: *Description Logics*, 2014, pp. 595–607.

[65] M. Blinkiewicz and J. Bąk, SQuaRE: A Visual Approach for Ontology-Based Data Access, in: *Semantic Technology*, Y.-F. Li, W. Hu, J.S. Dong, G. Antoniou, Z. Wang, J. Sun and Y. Liu, eds, Springer International Publishing, Cham, 2016, pp. 47–55. ISBN 978-3-319-50112-3.

[66] S. Kandel, A. Paepcke, J. Hellerstein and J. Heer, Wrangler: Interactive Visual Specification of Data Transformation Scripts, in: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM, New York, NY, USA, 2011, pp. 3363–3372. ISBN 978-1-4503-0228-9. doi:10.1145/1978942.1979444. http://doi.acm.org/10.1145/1978942.1979444.

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

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

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

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

[78] 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.

[79] 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.

[80] 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.

[81] 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.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

[86] 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.

[87] 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.

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

[89] 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.

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

[91] 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.

[92] 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.

[93] 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.

[94] 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.

[95] 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.

[96] 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.

[97] 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.

[98] 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.

[99] 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.

[100] 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.

[101] 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.

[102] 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.

[103] 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.

[104] B. Wu and C.A. Knoblock, An Iterative Approach to Synthesize Data Transformation Programs, in: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015, pp. 1726–1732.

[105] 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.

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

[107] 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.

[108] E. Thiéblin, O. Haemmerlé and C. Trojahn, Complex matching based on competency questions for alignment: a first sketch, in: *OM 2018 - 13th ISWC workshop on ontology matching*, 2018.

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

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

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

[112] F. Duchateau, Z. Bellahsene and E. Hunt, XBenchMatch: a benchmark for XML schema matching tools, in: *Proceedings of the 33rd international conference on Very large data bases*, VLDB Endowment, 2007, pp. 1318–1321.

[113] C. Pinkel, C. Binnig, E. Jiménez-Ruiz, E. Kharlamov, W. May, A. Nikolov, A. Sasa Bastinos, M.G. Skjæveland, A. Solimando, M. Taheriyan, C. Heupel and I. Horrocks, RODI: Benchmarking relational-to-ontology mapping generation quality, *Semantic Web* **9**(1) (2017), 25–52, ISSN 22104968, 15700844. doi:10.3233/SW-170268. http://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/SW-170268.

[114] E. Thiéblin, M. Cheatham, C. Trojahn, O. Zamazal and L. Zhou, The First Version of the OAEI Complex Alignment Benchmark, in: *ISWC Posters and Demos*, Springer, 2018.

[115] E. Thiéblin, O. Haemmerlé, N. Hernandez and C. Trojahn, Task-Oriented Complex Ontology Alignment: Two Alignment Evaluation Sets, in: *The Semantic Web*, A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai and M. Alam, eds, Lecture Notes in Computer Science, Springer International Publishing, 2018, pp. 655–670. ISBN 978-3-319-93417-4.

[116] 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.

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

[118] 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.

[119] 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.

[120] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini and H. Stuckenschmidt, C-OWL: Contextualizing Ontologies, in: *The Semantic Web - ISWC 2003*, Vol. 2870, G. Goos, J. Hartmanis, J. van Leeuwen, D. Fensel, K. Sycara and J. Mylopoulos, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 164–179. ISBN 978-3-540-20362-9 978-3-540-39718-2. doi:10.1007/978-3-540-39718-2_11. http://link.springer.com/10.1007/978-3-540-39718-2_11.

[121] N.F. Noy and M.A. Musen, The PROMPT suite: interactive tools for ontology merging and mapping, *International Journal of Human-Computer Studies* **59**(6) (2003).

[122] Z. Dragisic, V. Ivanova, P. Lambrix, D. Faria, E. Jiménez-Ruiz and C. Pesquita, User validation in ontology alignment, in: *International Semantic Web Conference*, Springer, 2016, pp. 200–217.

[123] M. Cheatham and P. Hitzler, Conference v2. 0: An uncertain version of the oaei conference benchmark, in: *International Semantic Web Conference*, Springer, 2014, pp. 33–48.