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. A limitation of these alignments is 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 of the specificities of the complex matching problem. In this paper, a review 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 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 number of knowledge sources made available on the Linked Open Data (LOD) cloud and their variety of modelling choices, the relationships between entities of these

sources are required to be more expressive. Simple correspondences are not expressive enough to fully overcome conceptual heterogeneity. However, currently, few complex alignments are available and published on the LOD cloud even if the need for these alignments has become more and more present in various application fields. For example, in the cultural heritage domain, the need for complex correspondences has been identified for data integration or data translation applications [2–5]. To tackle the issue, complex matching systems are used [2], or complex correspondences are manually created [4, 5]. In the agronomic domain, complex alignments help 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]. Complex alignments between medical ontologies have also been published [8, 9].

Different complex matching approacheswhich adopt a diversity of strategies and deal with different knowledge representation models, have emerged in the literature. Nevertheless, *complex matching* remains a challenge. In [10], ontology matching researchers were surveyed about future challenges in the field and they

---

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

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 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 a review of the complex matching approaches dealing with different kinds of knowledge representation models such as taxonomies, XML schemata, database schemata, formal ontologies, etc. A classification of the approaches based on the specificities of complex alignments is proposed and the evaluation aspects of these approaches reviewed. The limitations of both the 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 cloud.

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

2. Background

This section defines the scope of this study and provides the definitions related to alignments. The different knowledge representation models considered are presented and the notions of alignment and correspondence introduced. The ontology fragments used for the examples in the paper are presented in Figure 1.

2.1. Knowledge representation models

In the literature, different knowledge representation models have been so far referred to as "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 more specific qualification of the knowledge representation models is given in the description of the approaches when possible.
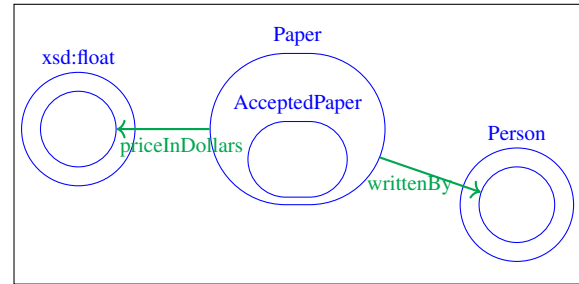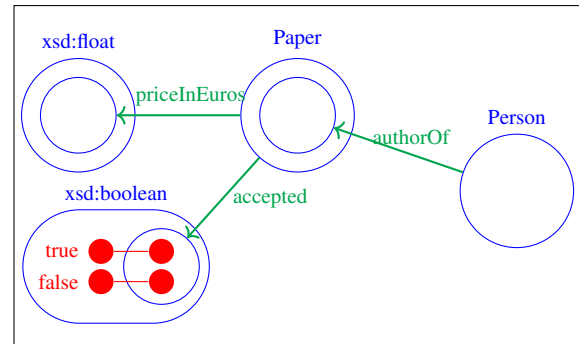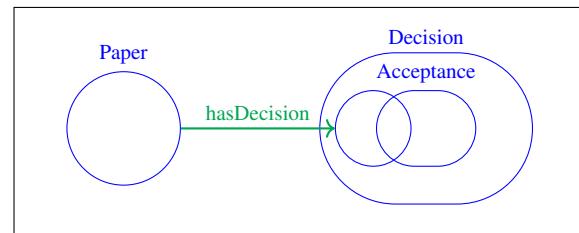

(a) o_1

(b) o_2

(c) o_3

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

In this survey, we consider the knowledge representation models defined as follows.

Table schemata   A table schema is a flat schema instantiated as tabular data. The table schema refers to the name of the table columns (also called attributes).

Relational database schemata (RDB)   Relational database schemata require the data to be organised as relations implemented by tables. The name of each relation is given, as well as the names and types of the relations' attributes. This model includes the notions of primary key and foreign key providing the links between the relations.

Document-oriented schemata (DOS)   DTDs, XML schemata and JSON schemata define the structure

of documents (XML or JSON documents). These document-oriented schemata include elements, attributes and types. Elements can be either complex when specifying nested sub-elements, or simple when specifying built-in data types, such as string, for an element or attribute.

Conceptual models (CM) Conceptual models include entity-relationship models, used to abstract a relational database schema, and UML models, used to abstract object-oriented programs and databases. The entities of these models describe the internal structure of domain objects. The entities can be organised as a hierarchy. Moreover, these models can also express relations (associations) with a multiplicity of constraints between the entities.

Formal ontologies Formal ontologies are axiomatised theories. Their entities are most often classes, object properties, data properties, instances and values. The expressiveness of the ontology's axioms is limited to the fragment of logic they implement (e.g., $\mathcal{SHIN}$, $\mathcal{SROIQ}$, two-variable first-order logic). Even though various ontology languages have been proposed in the past, OWL the W3C standard [18] is now widely used. The variants of OWL implement different logic fragments such as $\mathcal{SHIF}$, $\mathcal{SHOIN}$ or $\mathcal{SROIQ}$. Other ontology languages such as DAML+OIL [19] or CML [20] also implement a fragment of description logic (DL).

2.2. Expressions

The correspondences and alignments rely on the definition of expressions.

A **simple expression** is composed of a single entity represented by its unique identifier (e.g., an IRI for OWL 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, $\exists o_2$:*accepted*.$\{true\}$ is a complex expression which represents all the papers with the value *true* for the o_2:*accepted* property. The constructor used here is a value restriction constructor. A **constructor** is a logic constructor (union, intersection, inverse, etc.) or a restriction constructor (cardinality restriction, type restriction, value restriction, etc.). We introduce the *dom* and *range* functions which represent object property domain and range restrictions over a class. They are interpreted as $(dom(C))^{\mathcal{I}} = C^{\mathcal{I}} \times \mathcal{D}$ and $(range(C))^{\mathcal{I}} = \mathcal{D} \times C^{\mathcal{I}}$, where C is

a class, $\mathcal{I}$ an interpretation function over the domain $(\mathcal{D} \cup \mathcal{D} \times \mathcal{D})$ and $\mathcal{D}$ the object interpretation domain.

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), a conversion function (e.g., metric conversion), etc.

2.3. Alignment and correspondence

Ontology matching is the process of generating an ontology alignment between a source and a target ontology [16]. An ontology alignment consists of correspondences. These notions are defined below.

Definition 1. *An **ontology alignment** $A_{o_1 \to o_2}$ is directional between a source ontology o_1 and a target ontology o_2.*
$A_{o_1 \to o_2}$ *is a set of correspondences,* $A_{o_1 \to o_2} = \{c_1, c_2, ..., c_n\}$.

Definition 2. *A **correspondence** c_i is a tuple (e_{o_1}, e_{o_2}, r). e_{o_1} and e_{o_2} are the **members** of the correspondence. They can be simple or complex expressions with entities from respectively o_1 and o_2:*

- *if the correspondence is **simple**, both e_{o_1} and e_{o_2} are simple expressions;*
- *if the correspondence is **complex**, at least one of e_{o_1} or e_{o_2} is a complex expression;*
- *r is a relation, e.g., equivalence ($\equiv$), more general ($\sqsupseteq$), more specific ($\sqsubseteq$), disjointedness ($\perp$) holding between e_{o_1} and e_{o_2}.*
- *alignment systems usually assign a confidence value to each correspondence, such that correspondences are sometimes defined as quadruples (e_{o_1}, e_{o_2}, r, n). We only exemplify correspondences as triples in the rest of the paper.*

The members of the correspondences can be 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, 21] (*1* for *s* and *m* or *n* for *c*). However, they can be misinterpreted as the alignment arity or multiplicity [22].

We provide below some examples of complex correspondences, based on the definitions above and the fragment of the ontologies in Figure 1.

c_1 = *(o_1:Person, o_2:Person, $\equiv$)* is a *(s:s)* simple correspondence.
c_2 = *(o_1:priceInDollars, changeRate(o_2:priceInEuros), $\equiv$)* is a *(s:c)* complex correspondence with a transformation function: *changeRate*.

$c_3 = (\exists o_3{:}hasDecision.o_3{:}Acceptance,$
     $o_1{:}AcceptedPaper, \equiv)$ is a *(c:s)* complex corre-
     spondence with constructors.
$c_4 = (o_1{:}writtenBy, o_2{:}authorOf^-, \equiv)$ is a *(s:c)*
     complex correspondence with the *inversion* con-
     structor.
$c_5 = (\exists o_2{:}accepted.\{true\},$
     $\exists o_3{:}hasDecision.o_3{:}Acceptance, \equiv)$ is a *(c:c)*
     complex correspondence with constructors.

As opposed to a **simple alignment**, a **complex alignment** contains at least one complex correspondence.

The *pairwise* definition of a matching process (between a source and a target ontology) can be extended to cover multiple ontologies. A **holistic** matching process considers more than two ontologies together without a source or target distinction [23, 24]. 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 [25].

Complex ontology matching is the process of generating a complex alignment between ontologies. The approaches for generating such an alignment are discussed in §5.

2.4. Scope clarification

This section presents reflections on the scope of the survey.

2.4.1. Type of matched objects

Complex alignments can also occur between other objects such as business process models [26], strings [27], etc. However, these objects are different from the knowledge representation models studied in this survey. The nature of their elements is not the same as those of the representation model elements (concepts, relations, attributes). For example, business processes are graph-like models of a process, they have a begin 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 [28], 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" [29]. Most approaches dealing with such activities rely on relations considering both the 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 must *conceptual or logical relations* between entities be considered but also (and mainly) *change relations*, which represent what has actually been transformed between the two versions of the ontology [30]. The first kind of relations (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 design time through the tool used to make the ontology evolve (such as Protégé or KAON [31]) or identified a posteriori through ontology "diff tools" [32–35]. 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 [33] 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 involved 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" [36]. The two types of changes are also called low level/high level operations [34], el-

ementary/composite changes [31] or atomic/complex changes [37]. According to [34], 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 [31, 33].

The work in [38] gives another point of view on the link between the two 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 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 representation

In the following, we present first the languages originally designed to describe axioms or rules outside the specific scope of alignment representation. Then, we introduce the dedicated languages. Examples of complex correspondences expressed in some languages are provided.

3.1.1. Generic representations
Rules and axioms

OWL OWL [18] 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. The correspondence c_3 represented in the XML concrete Syntax of OWL is given below.

```
<owl:Class rdf:about="&o1;AcceptedPaper">
<owl:equivalentClass>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:about="&o3;
          hasDecision"/>
```

```
    </owl:onProperty>
    <owl:someValuesFrom>
      <owl:Class rdf:about="&o3;Acceptance"/>
    </owl:someValuesFrom>
  </owl:Restriction>
</owl:equivalentClass>
</owl:Class>
```

Web-PDDL The Web-PDDL [39] is a strongly typed FOL (first-order logic) language. It allows the use of variables, constants, conditions, logical constructors and quantifiers. The predicates and constants take the form of URIs. An example of Web-PDDL for representing the correspondence c_3 is given below.

```
(forall (x)
  (iff (is @o1:AcceptedPaper x)
      (exists( y - @o3:Acceptance) (@o3:
          hasDecision x y))))
```

SWRL The Semantic Web Rule Language (SWRL) [40] helps to define rules, in the form of FOL Horn-rules, 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. An example of SWRL representing the correspondence c_3 is given below.

```
<ruleml:imp>
  <ruleml:_rlab ruleml:href="#c3"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom  swrlx:
        property="&o3;hasDecision">
      <ruleml:var>x</ruleml:var>
      <ruleml:var>y</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:classAtom>
      <owlx:Class owlx:name="&o3;Acceptance
          "/>
      <ruleml:var>y</ruleml:var>
    </swrlx:classAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:classAtom>
      <owlx:Class owlx:name="&o1;
          AcceptedPaper"/>
      <ruleml:var>x</ruleml:var>
    </swrlx:classAtom>
  </ruleml:_head>
</ruleml:imp>
```

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

Query languages

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. The following queries represent c_3 as equivalent SPARQL SELECT queries and a SPARQL CONSTRUCT query.

```
SELECT ?s WHERE { ?s a o1:AcceptedPaper.}
SELECT ?s WHERE { ?s o3:hasDecision ?o.
                  ?o a o3:Acceptance. }

CONSTRUCT{?s a o1:AcceptedPaper.}
    WHERE{?s o3:hasDecision ?o.
          ?o a o3:Acceptance.}
```

XSLT, XPath XML to XML, Logic, Transformation XSLT (eXtensible Stylesheet Language Tansformations) [41] is a language with an XML concrete syntax. 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 reuses 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.

3.1.2. Dedicated alignment representations

Various formats have been proposed to represent alignments between two different knowledge representation models. A survey on ontology alignment formats is presented in [42].

EDOAL EDOAL [43] is an extension of the Alignment format to represent the complex correspondences between OWL ontologies. This language is based on correspondence patterns [42] and can be processed by the Alignment API [44]. The Alignment format can be extended by other languages to express complex correspondences. c_3 is represented in EDOAL as follows.

```
<map>
  <Cell>
    <entity1>
      <edoal:Class rdf:about="&o1;
          AcceptedPaper" />
    </entity1>
    <entity2>
      <edoal:AttributeDomainRestriction>
        <edoal:onAttribute>
          <edoal:Relation rdf:about="&o3;
              hasDecision" />
        </edoal:onAttribute>
        <edoal:exists>
          <edoal:Class rdf:about="&o3;
              Acceptance" />
        </edoal:exists>
      </edoal:AttributeDomainRestriction>
    </entity2>
    <measure rdf:datatype="&xsd;float">1.0</
        measure>
    <relation>Equivalence</relation>
  </Cell>
</map>
```

XeOML XeOML [45] 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 [46, 47] 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 [48]. It allows for the description of data transformation between ontologies. These transformations include logic rules (based on OWL axioms) and value transformation functions.

In the area of OBDA (Ontology-Based Data Access) [49], different formats to express correspondences between relational databases and RDF datasets have been proposed in the literature. A comprehensive review of different formats can be found in [50]. Here, the W3C R2RML format and some of its extensions are briefly introduced.

R2RML The R2RML is a W3C format [51] used to represent correspondences between relational databases and RDF datasets. 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.

```
@prefix rr:<http://www.w3c.org/ns/r2rml#> .
<TriplesMap1>
    a rr:TriplesMap
    rr:logicalTable [
        rr:sqlQuery
            "'"SELECT paperID
            FROM Table_O3
            WHERE hasDecision = "Acceptance"
            "'"
    ];

    rr:subjectMap [
            rr:template "o1:Paper/{PaperID}";
            rr:class o1:AcceptedPaper
    ];
```

RML The RML language [52] extends the R2RML format by allowing other kinds of data sources such as XML schema, JSON, or tabular data (CSV). The FnO ontology [53] can be used in RML to describe transformation functions in the correspondences.

xR2RML The xR2RML language [54] 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 [55] is based on R2RML and RML, allowing conditional case statements and programming inside the correspondences.

3.1.3. Summary table

Table 1 gives a summary of the complex alignment formats presented in this section with their context of application. For instance, alignments represented in *OWL* are usually used for the task of ontology merging. 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].

As discussed below (§5.6), despite the work on the different representation languages, complex matchers still fail on using those proposals. Many of them output FOL or DL correspondences in a simple text format, use their own specific syntax, or are not strict to EDOAL syntax.

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 below. 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 [56]. 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 [57] 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 [58] or KARMA [59], Ontologies Alignment Tool (OAT) [60] provide a user-interface for complex alignment edition and correction.

Dedicated complex alignment editors use different strategies for the visualisation of the correspondences. MAFRA [46] is an edition and visualisation framework which allows for complex alignment representation as an instantiation of their Semantic Bridge Ontology (SBO). Klint [61] provides a graph-based visual interface for integration rule (correspondence) validation and edition. The correspondences are represented as labelled graphs involving variables. OntoStudio [62] is a suite of software for ontology engineering. Its OntoMap plugin [63] 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 using different strategies to represent the correspondences: block metaphor [64, 65], graph-like [66–69] or tree-like [59]. Wrangler [70] proposes a graphical interface for the edition of value transformation functions such as scripts between tables.

4. Classification of complex matchers

Ontology matching approaches have been classified in various surveys [1, 10–16]. These classifications

Table 1

Complex alignment formats, "Logic" shows whether the format can represent logic constructors, "Transformations" shows whether the format can represent value transformation functions

Format	Type of knowledge representation models	Logic	Transformation	Alignment context application
OWL	OWL onto to OWL onto	√		Ontology merging
Web-PDDL	FOL onto to FOL onto	√		Data integration
SWRL	OWL onto to OWL onto	√	√	Data integration
SQL	RDB to RDB	√	√	Querying, Data transformation
SPARQL	OWL onto to OWL onto	√	√	Querying, Data transformation
XQuery	XML to XML	√	√	Querying, Data transformation
XSLT	XML to XML	√	√	Data transformation
EDOAL	OWL onto to OWL onto	√	√	Generic
XeOML	OWL onto to OWL onto	√	√	Generic
SBO	DAML+OIL onto to DAML+OIL onto	√	√	Data transformation
SPIMBench	RDF to RDF	√	√	Data transformation
R2RML	RDB to RDF	√	√	Data transformation
RML	CSV, XML, JSON to RDF	√	√	Data transformation
xR2RML	mixed formats(CSV, RDB, XML, JSON) to RDF	√	√	Data transformation
D2RML	RDB, CSV, XML, JSON to RDF	√	√	Data transformation

Table 2

Complex alignment visualisation and edition tools

Tool	Visualisation strategy	Align. langage
Protégé [56]	Manchester syntax	OWL
Axiomé [57]	Tree, Natural language	SWRL
MAFRA [46]	Tree	SBO
OAT [60]	Pattern instantiations	EDOAL, OWL
Klint [61]	Graph	SPARQL
OntoMap [63]	Edges	proprietary
Clio [71]	Edges	SQL
Juma [64, 65]	Block metaphor	R2RML
KARMA [59]	Tree	R2RML
Map-On [66]	Graph	R2RML
RMLEditor [67]	Graph	RML/R2RML
SquaRE [69]	Graph	R2RML
Lembo2014 [68]	Graph	R2RML
Wrangler [70]	Scripts	proprietary

however do not address the specificities of the complex approaches. After presenting the main existing classifications of ontology matching approaches (§4.1), we introduce axes for the classification of complex matching approaches (§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 the guiding thread to present the classifications below.

Most classifications so far focused on 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 $\mathcal{TBox}$ and those which deal with the $\mathcal{ABox}$. 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.

This classification [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-way taxonomy ends in basic approach strategies (e.g., string-based, model-based, formal resource-based).

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, the matching process is guided by a top-level ontology from which the

source and target ontologies derive; in the second, the matching process uses heuristics or machine-learning techniques.

Regarding the output dimension of the matching approaches, Rahm *et al.* [14] consider the output alignment arity as a characteristic of the approaches which could be integrated into its taxonomy.

In sum, among the ontology matching classifications so far, that of 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 rarely 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 in which 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, consider the following correspondences:

1. *(o_1:AcceptedPaper, $\exists o_2$:accepted.{true}, $\equiv$)*
2. *(o_1:priceInDollars, changeRate(o_2:priceInEuros), $\equiv$)*
3. *({o_1:Paper,o_1:Person}, {o_2:Paper,o_2:Person}, $\equiv$)*

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. It cannot therefore 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 a kind of correspondence was found. However, some approaches are able to generate both types independent of each other. An example of this correspondence expressed would be: *(o_1:Paper $\sqcap$ o_1:priceInDollars, o_2:Paper $\sqcap$ changeRate(o_2:priceInEuros), $\equiv$).*

Guiding structures. These categories aim at classifying the (complex) matching approaches based on their process dimension. 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 instantiation of an atomic pattern, such as those defined by Scharffe [42]. 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: *(o_1:A, $\exists o_2$:b.o_2:C, $\equiv$)* 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, $\equiv$)* 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 relation and transformation function correspondence patterns. For example, an approach looking for correspondences of the form *(o_1:A, o_2:B $\sqcup$ o_2:C $\sqcup$ o_2:D $\sqcup$ ..., $\equiv$)* , where o_1:A, o_2:B, o_2:C, o_2:D, *etc.* are classes and the number of unions in the target member of correspondences is not *a-priori* defined by the approach, falls into this category. Correspondences representing string concatenation of an unlimited number of properties also fall into this category and in the *transformation function* type of correspondence.
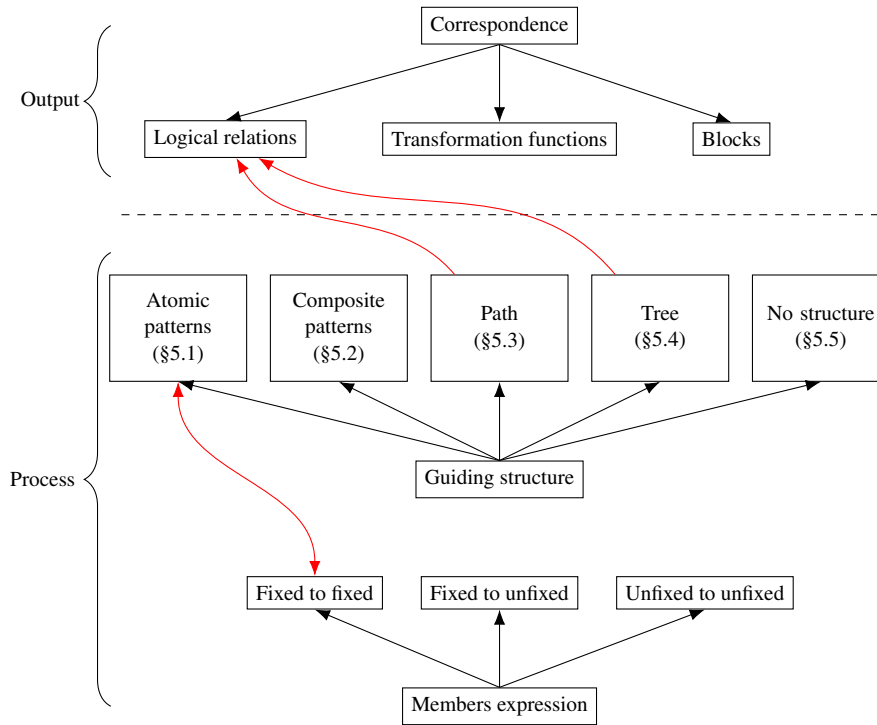
Fig. 2. Two axes to characterise the complex matching approaches: output and process. The correlation between the categories are represented with red arrows.

– **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 inside the ontologies for correspondence detection. The ontologies are either considered as a tree or a tree-like structure is sought in an ontology graph. For example, when an XML schema is considered as a tree and the approach consists in finding the smallest equivalent tree in an ontology.

– **No structure** Contrary 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, and therefore impact the structure of the output correspondences. However, a given correspondence, for example *(o_1:AcceptedPaper, $\exists o_2$:acceptedBy.$\top$, $\equiv$)*, could be obtained by an approach based on atomic patterns with the pattern *(A, $\exists b.\top$, $\equiv$)*, by an approach based on composite patterns such as *(A, $\exists b.\top \sqcup \exists c.\top \sqcup \ldots$, $\equiv$)* or by an approach with no guiding structure.

The *member expression pre-definition* specifies whether 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 member expressions. Atomic pattern-based approaches generate fixed to fixed correspondences as both members' expressions are defined by the pattern. As shown in Figure 2, this category is strongly correlated to the Atomic-pattern guiding structure category.

– The **fixed to unfixed** member 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 members will always be one property while the other will be a path of a-priori undefined length.

– The **unfixed to unfixed** member expression category includes the approaches that output correspondences whose members have an undefined expression beforehand. 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 2: a path or tree-based approach will only output logical correspondences. There is also 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 specific to complex matching. Not only do they guide the matching process, but the correspondence 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 entity (concepts, properties, etc.) dealt with by the matchers but this was not specific to complex alignment.

In some way, the structure-based classification can be considered as a specialisation of the graph-based techniques category in 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 2), a reference to the kind of output and member expression pre-definition is made in the text. The approaches are detailed in paragraphs with titles following 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. Table 3 presents several 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 [75], other approaches have their own pattern library [60, 72, 74, 76–79]. Two main detection techniques appear: structuro-linguistic conditions (called *matching patterns* defined in [80]) [60, 72–75, 79], and statistical measures [76–78]. These approaches are detailed below.

Ritze et al. [72, 73] OWL Ontology to OWL Ontologyy, (s:c) In [72, 73], 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 [42] (cf. 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[1] [72] 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[2] of the matching conditions [73] 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

---

[1]http://dominique-ritze.de/complex-mappings/
[2]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)	$(o_1{:}A, \exists o_2{:}b.o_2{:}C, \equiv)$	$(o_1{:}AcceptedPaper, \exists o_2{:}hasDecision.o_2{:}Acceptance, \equiv)$
Class by attribute inverse type (CIAT)	$(o_1{:}A, o_2{:}C \sqcap \exists o_2{:}b.\top, \equiv)$	$(o_1{:}AcceptedPaper, o_2{:}Paper \sqcap \exists o_2{:}hasAcceptance.\top, \equiv)$
Class by attribute value (CAV)	$(o_1{:}A, \exists o_2{:}b.\{V\}, \equiv)$	$(o_1{:}AcceptedPaper, \exists o_2{:}accepted.\{true\}, \equiv)$
Class by attribute existence (CAE)	$(o_1{:}A, \exists o_2{:}b.\top, \equiv)$	$(o_1{:}AcceptedPaper, \exists o_2{:}acceptedBy.\top, \equiv)$
Property chain (PC)	$(o_1{:}a, o_2{:}b \circ o_2{:}c, \equiv)$	$(o_1{:}reviewedBy, o_2{:}hasReview \circ o_2{:}reviewWrittenBy, \equiv)$
Inverse Property (IP)	$(o_1{:}a^-, o_2{:}b, \sqsubseteq)$	$(o_1{:}write^-, o_2{:}writtenBy, \sqsubseteq)$
Class Intersection	$(o_1{:}A, o_2{:}B \sqcap o_2{:}C, \equiv)$	$(o_1{:}AuthorAndReviewer, o_2{:}Author \sqcap o_2{:}Reviewer, \equiv)$

Table 4

Atomic patterns per approach

Work	Patterns
Ritze2009 [72]	Class by Attribute Type, Class by Inverse Attribute Type, Class by Attribute Value, Property Chain
Ritze2010 [73]	Inverse Property, Class by Attribute Type, Class by Inverse Attribute Type, Class by Attribute Value
AMLC [74]	Class by Attribute Type, Class by Attribute Existence
Oliveira2018 [25]	Class Intersection
Rouces2016 [75]	Linguistic patterns of FrameBase
Walshe2016 (Bayes-ReCCE) [76]	Class by Attribute Value
OAT [60]	Combinations of predefined expressions
Jiang2016 (KAOM) [77]	Linear Regression
Dhamankar2004 (iMAP) [78]	Conversion functions predefined, basic arithmetic properties
Jimenez2015 (BootOX) [79]	RDB schema properties to OWL axioms

simplified matching conditions to detect inverse property states that if the verb phrase of the label of a source property $o_1{:}a$ is the active voice of the verb phrase of a label of a target property $o_1{:}b$, then $(o_1{:}a^-, o_2{:}b, \sqsubseteq)$ is a probable correspondence.

Example 1. *Conditions: $(o_1{:}a^-, o_2{:}b, \sqsubseteq)$ iff verb$(o_1{:}a)$ = active-voice (verb$(o_2{:}b)$)*
Correspondence: $(o_1{:}writePaper^-, o_2{:}writtenBy, \sqsubseteq)$ 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 $o_1{:}a$ (e.g., $o_1{:}write$) and $o_2{:}b$ (e.g., $o_2{:}writtenBy$) : the domain of $o_1{:}a$ (e.g., $o_1{:}Person$) should be subsumed by the range of $o_2{:}b$ (e.g., $o_2{:}Person$) and the range of $o_1{:}a$

(e.g., $o_1{:}Paper$) should be subsumed by the domain of $o_2{:}b$ (e.g., $o_2{:}Document$). The subsumption between ranges and domains of the two properties can be detected by inference on the ontologies' structure linked by the simple reference alignment or by a hypernymy relation between the labels. In the example, the necessary subsumptions are: $(o_1{:}Person, o_2{:}Person, \sqsubseteq)$ and $(o_1{:}Paper, o_2{:}Document, \sqsubseteq)$.

AMLC [74] OWL ontology to OWL ontology, (s:c) AMLC (Complex AgreementMakerLight) is the complex version of the AML (AgreementMakerLight) system. It relies on lexical similarity and structural conditions to detect correspondence patterns. This approach is very similar to that in [72]. Two types of patterns are sought: Class by Attribute Existence and Class by Attribute Type (cf. Table 3).

Oliveira and Pesquita [25] OWL ontology to OWL ontology, (s:c) The approach proposed in [25] looks for *compound* correspondences which in their target member involve entities from more than one ontology. The sought correspondences follow the pattern $(o_1{:}A, o_2{:}B \sqcap o_3{:}C, \equiv)$ 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 target 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. 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: (o_1:AuthorAndReviewer, o_2:Author ⊓ o_3:Reviewer, ≡)*

Rouces et al. [75]   OWL ontology to the FrameBase ontology (OWL), (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 [61] which provides a graphical interface for correspondence edition.

Example 3. *Created property: frame:hasBirthDate(s,o)*
Pattern:frame:BirthEvent(e) ∧ frame:hasSubject(e,s) ∧ frame:hasDate(e,o)
Source property preprocessing:
  o_1:*birthDate → o_1:hasBirthDate*
Simple correspondence:
  (o_1:hasBirthDate, frame:hasBirthDate, ≡)
Correspondence: (o_1:birthDate, frame:hasSubject⁻ ∘ (dom(frame:BirthEvent) ⊓ frame:hasDate), ≡)

Bayes-ReCCE [76]   OWL ontology to OWL ontology, (s:c) This approach detects *Class Attribute Value Restrictions* correspondences. Bayes-ReCCE uses the properties of matched instances of two classes o_1:*AcceptedPaper* and o_2:*Paper*, with *(o_1:AcceptedPaper, o_2:Paper, ⊑)* in a reference alignment. The match-

ing 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.

Example 4. *A reference alignment between o_1 and o_2 contains the correspondence (o_1:AcceptedPaper, o_2:Paper, ⊑). The common instances of o_1 and o_2 described by o_1:AcceptedPaper in o_1 and o_2:Paper in o_2 are retrieved. The set of attribute-value pairs of each common instance is retrieved and becomes a feature in the feature-selection algorithm. If the attribute-value pair: (o_2:accepted,true) is selected by the algorithm, the correspondence (o_1:AcceptedPaper, ∃o_2:accepted.{true}, ≡) is output.*

Ontologies Alignment Tool (OAT) [60], OWL Ontology to OWL Ontology, (s:c), (c:s), (c:c)   The Ontologies Alignment Tool (OAT) presented in [60] presents a semi-automatic complex matcher. The user can input correspondences through a graphical interface by instantiating correspondence patterns. For each of the two ontologies, the automatic matcher creates a set of expressions following a list of patterns (object property range restriction, inverse property, etc.). These expressions from the two ontologies are then compared by their entities' labels. If the similarity between two expressions is above a threshold, a correspondence putting these two expressions together is suggested to the user who can validate or invalidate it. The confidence of the correspondence is then set to respectively 1 or 0 and propagated to the other correspondences. For example, the system finds that the domain restriction $dom(o_1$:Paper$) ⊓ o_1$:hasAuthor is similar to the single property o_2:writtenBy, the following correspondence is output: *($dom(o_1$:Paper$) ⊓ o_1$:hasAuthor, o_2:writtenBy, ≡)*. The system confidence associated to this correspondence is a weighted average of the similarity (or system confidence value) of the properties (o_1:hasAuthor,o_2:writtenBy), of their respective domains (or domain restrictions) (o_1:Paper, o_2:Paper) and ranges (o_1:Author, o_2:Paper_Author). For ex-

ample, the initial system confidence of *(o$_1$:Author, o$_2$:Paper_Author, ≡)* is 0.6. If the user validates this correspondence, it becomes 1. Then, the system confidence of *(dom(o$_1$:Paper)⊓ o$_1$:hasAuthor, o$_2$:writtenBy, ≡)* is updated to take into account this new range confidence value. The user can also manually add new correspondences.

iMAP, Dhamankar et al. [78]   Relational database schema to relational database schema, (c:s) The iMAP system [78] uses a set of *searchers* to discover simple and complex correspondences between relational database schemata. The validity of each correspondence is then checked by a similarity estimator based on the columns' 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 5. 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 5. *Category searcher correspondence between schemata describing papers and their acceptance status:*
(∃ s$_1$:accepted.{true}, ∃ s$_2$:accepted.{1}, ≡)
Schema mismatch correspondence between schemata describing a conference participant status:
(∃s$_1$:actions.{early-registration},   ∃s$_2$:early-registration.{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. [77]   OWL ontology to OWL ontology, (s:c) (c:s) (c:c) KAOM generates transformation function correspondences and logical relation correspondences. As the iMap's system [78], KAOM implements different matching strategies: one for detecting transformation function correspondences, the other for logical relation correspondences. Here we present 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.

BootOX, Jimenez-Ruiz et al. [79]   Relational database schema to OWL ontology, (c:s) The BootOX approach [79] produces correspondences between a relational database schema and a target ontology via the creation of a "bootstrapped" ontology. The approach proceeds in two phases. 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 [81] 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 [82, 83] but their aim is not to align the schema to an existing ontology. They are therefore 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 patterns. Table 5 presents the different composite patterns detected by the approaches.

Some approaches iteratively construct the member(s) of the correspondence [78, 84, 86, 87, 91] (text searcher of iMap). Others first discover atomic pat-

Table 5

Composite patterns per approach. A, B, C are classes, a, b, c, d are properties, v_1, v_2, v_3, v_4 are values (instances or literals)

Work	Composite pattern	Pattern form
Parundekar2012 [84]	Disjunction of attribute-value pairs	$(\exists o_1{:}a.\{v_1\}, \exists o_2{:}b.\{v_2,v_3,\dots\}, \equiv)$
Parundekar2010 [85]	Conjunction of attribute-value pairs	$(\exists o_1{:}a.\{v_1\} \sqcap \exists o_1{:}b.\{v_2\} \sqcap \dots, \exists o_2{:}c.\{v_3\} \sqcap \exists o_2{:}d.\{v_4\} \sqcap \dots, \equiv)$
Doan2003 (CGLUE) [86]	Class unions	$(o_1{:}A, o_2{:}B \sqcup o_2{:}C \sqcup \dots, \equiv)$
Kaabi2012 (ARCMA) [87]	Class intersection	$(o_1{:}A, o_2{:}B \sqcap o_2{:}C \sqcap \dots, \sqsubseteq)$
Boukottaya2005 [88]	String concatenation	$(o_1{:}a, concatenation(o_2{:}b, o_2{:}c, \dots), \equiv)$
	Subset merging	$(o_1{:}a, o_2{:}b \sqcup o_2{:}c \sqcup \dots, \equiv)$
Dhamankar2004 (iMAP) [78]	String concatenation	$(o_1{:}a, concatenation(o_2{:}b, o_2{:}c, \dots), \equiv)$
Xu2003 [89]	String concatenation	$(o_1{:}a, concatenation(o_2{:}b, o_2{:}c, \dots), \equiv)$
	Subset merging	$(o_1{:}a, o_2{:}b \sqcup o_2{:}c \sqcup \dots, \equiv)$
Xu2006 [90]	String concatenation	$(o_1{:}a, concatenation(o_2{:}b, o_2{:}c, \dots), \equiv)$
	Subset merging	$(o_1{:}a, o_2{:}b \sqcup o_2{:}c \sqcup \dots, \equiv)$
Warren2006 [91]	String concatenation of attribute substrings	$(o_1{:}a, concatenation(substr(o_2{:}b), substr(o_2{:}c), \dots), \equiv)$
Arnold2013 (COMA++) [92]	String concatenation	$(o_1{:}a, concatenation(o_2{:}b, o_2{:}c, \dots), \equiv)$
Wu2004 [93]	Annotated sets of properties (is-a or aggregate)	$(\{o_1{:}a\}, is\text{-}a\{o_2{:}b, o_2{:}c, \dots\}, \equiv)$; $(\{o_1{:}a\}, aggregate\{o_2{:}b, o_2{:}c, \dots\}, \equiv)$
Svab2009 [94]	N-ary to N-ary	Structure matching, see Fig. 3
	N-ary to object property	
Saleem2008 (PORSCHE) [95]	Bag of properties/blocks	$(\{o_1{:}a\}, \{o_2{:}b, o_2{:}c, \dots\}, \equiv)$
He2004 (DCM) [96]	Bag of properties/blocks	$(\{o_1{:}a\}, \{o_2{:}b, o_2{:}c, \dots\}, \equiv)$
Su2006 (HSM) [97]	Bag of properties/blocks	$(\{o_1{:}a, o_1{:}b, \dots\}, \{o_2{:}c, o_2{:}d, \dots\}, \equiv)$

tern correspondences and merge them in a final (non-iterative) step [85, 92]. Approaches use graph-pattern matching either as detection conditions [88, 93–95] or over the properties of a mediating ontology [78, 89, 90] (iMap's date searcher). Finally, [96, 97] start by grouping schema attributes before matching the groups. Even though the holistic approaches [96, 97] 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 [88–90, 95]. These approaches could also be classified into the tree-based category. However, as their matching process relies on the identification of a composite pattern in those trees, they were classified in this category. In [94], the approach detects and matches N-ary relation reifications between ontologies. The N-ary relation contains a repetitive pattern, therefore [94] was classified in this category.

Parundekar et al. [84]   OWL ontology to OWL ontology, (s:c) (c:s) In this approach proposed by Parundekar *et al.* [84], the type of correspondences sought is an *attribute-value* pair matched with an attribute

and a union of its acceptable values. In the first step, the approach finds correspondences between *attribute-value* pairs from the linked instances of the two ontologies (instances linked with *owl:sameAs* predicate). The number of instances sharing both attribute-value pairs defines whether the correspondence has a subsumption or equivalence relation. The second step of 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 following example shows the two-step approach.

Example 6. *First step output:*
$(\exists o_1{:}accepted.\{true\}, \exists o_2{:}hasStatus.\{accepted\}, \sqsupseteq)$
$(\exists o_1{:}accepted.\{true\}, \exists o_2{:}hasStatus.\{camera\text{-}ready\}, \sqsupseteq)$
Second step output:
$(\exists o_1{:}accepted.\{true\}, \exists o_2{:}hasStatus.\{accepted, camera\text{-}ready\}, \equiv)$

Parundekar et al. [85]   OWL ontology to OWL ontology, (s:c) (c:s) (c:c) Parundekar *et al.* [85] look for conjunctions of *attribute-value* pairs, for instance correspondences of the form $(\exists o_1{:}a.\{v_1\} \sqcap \exists o_1{:}b.\{v_2\}$

$\sqcap \ldots, \exists o_2:c.\{v_3\} \sqcap \exists o_2:d.\{v_4\} \sqcap \ldots, \equiv$) 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 $(\exists o_1:hasDecision.\{accept\}, \exists o_2:accepted.\{true\}, \equiv)$. Starting from these seed hypotheses, the approach implements a heuristic in-depth-first exploration of the search space (all the possible conjunctions of attribute-value pairs). 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 (e.g., $\exists o_1:submitted.\{true\}$ has been added to the source member of the seed hypothesis: $(\exists o_1:hasDecision.\{accept\} \sqcap \exists o_1:submitted.\{true\}, \exists o_2:accepted.\{true\}, \equiv)$. 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. [86]   DL ontology to DL ontology, (s:c) The GLUE system [86] is specialised in detecting (s:s) correspondences between ontologies' classes using machine learning techniques such as joint probability distribution. CGLUE, also presented in [86], is an extension of the GLUE system. It can detect (s:c) class unions in class hierarchies such as $(o_1:Document, o_2:Paper \sqcup o_2:Poster \sqcup \ldots, \equiv)$. To detect these unions, the authors make the assumption that the subclasses of a class represent a partition of this class. To find a correspondence to a source class $o_1:Document$, 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 obtained on the similarity score.

ARCMA, Kaabi et Gargouri [87]   OWL ontology to OWL ontology, (s:c) Kaabi *et* Gargouri [87] propose ARCMA (Association Rules Complex Matching Approach) to find correspondences of the form $(o_1:A, o_2:B \sqcap o_2:C \sqcap \ldots, \sqsubseteq)$. 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 ($o_2:B$, $o_2:C$, ...) 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 7. *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:*
- *($o_1:Author$, $o_2:Author$, $\equiv$)*
- *($o_1:Reviewer$, $o_2:Reviewer$, $\equiv$)*

With the overlap of terms associated with $o_1:AuthorAndReviewer$ and the terms of respectively $o_2:Author$ and $o_2:Reviewer$, the following correspondence can be output: ($o_1:AuthorAndReviewer$, $o_2:Author \sqcap o_2:Reviewer$, $\sqsubseteq$)

Boukottaya and Vanoirbeek [88]   XML schema to XML schema, (s:c) (c:s) (c:c) Boukottaya *et* Vanoirbeek [88] propose an XML schema matching approach based on the schema tree and linguistic layer of the schema. This approach finds simple and complex correspondences. The complex correspondences 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 8. *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 con-*

catenation of the children nodes can be matched to the target node:

(concatenation(o_1:street, o_1:city), o_2:address, $\equiv$)

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:

(o_1:Journal-Article $\sqcup$ o_1:Conference-Article, o_2:Article, $\equiv$)

COMA++, Arnold [92] Document-oriented schema to document-oriented schema, (s:c) As an improvement to the COMA system [98], Arnold [92] discusses a solution based on a lexical strategy on the schemata attribute names: *several (s:s)* attribute correspondences with the same attribute as target (or source), could be merged into a complex one. The initial approach generates simple correspondences with expressive relations such as meronymy *part-of* or holonymy *has-a* besides usual relations ($\sqsupseteq$, $\sqsubseteq$, $\equiv$). The extension for transforming the simple correspondences into a complex one can take into account the type of attribute (e.g., concatenation for a string attribute or sum for a numeric attribute). The following example shows a complex correspondence inferred from simple correspondences.

Example 9. *Part-of correspondences with same target member:*
  – *(s_1:firstName, s_2:fullName, part-of)*
  – *(s_1:lastName, s_2:fullName, part-of)*
Aggregation in a new correspondence:
(concatenation(s_1:firstName, s_1:lastName), s_2:fullName, $\equiv$)

iMAP, Dhamankar et al. [78] Relational database schema to relational database schema, (c:s) As seen in the previous section, the iMAP system [78] uses a set of *searchers* to discover simple and complex correspondences between relational 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 this, 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 corre-

spondences are selected, the process is reiterated but with combinations of concatenations of the selected source attribute and other source attributes as base correspondences. These new correspondences are scored, selected, 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) and the relations between them (e.g., concatenation, subset). The attributes of each schema are matched to the date ontology's 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 [89, 90] Conceptual model to conceptual model, (s:c) (c:s) Xu and Embley [89] propose a similar approach to iMap's date matcher. It uses a user-specified domain ontology as mediator between the two conceptual models to be aligned. 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 value* techniques between the source conceptual model (resp. target) entities and the ontology's concepts. These simple correspondences are kept for the next phase if the number of common values between the conceptual model entity and the ontology concept are above a threshold.

The relation between the ontology concepts in simple correspondences will become the transformation functions between the attributes they are linked to. For example, *s:street s:city* are two entities from the source conceptual model and *t:address* is an entity from the target conceptual model. In the first matching phase, simple correspondences are drawn with concepts from the mediating ontology *o*:
  – *(o:Address, t:address, $\equiv$)*
  – *(o:Street, t:street, $\equiv$)*
  – *(o:City, t:city, $\equiv$)*
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 [90] 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 [91] Table schema to table schema, (c:s) Warren and Tompa [91] focus on finding correspondences between string columns of tabular data. They deal with correspondences that translate a concatenation of column sub-strings. The approach starts by ranking the source columns according to the *q-grams* (sequence of *q* characters) of its values found in the target column. Then it looks for matched instances (rows) according to a *tf-idf* formula on co-occurring q-grams. The source 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 source columns.

Example 10. *A correspondence output by this approach could be:*
(concatenation(substr(o_1:firstName,1),
substr(o_1:lastName,6)), o_2:username, ≡), with substr(x,n) a function giving the first n characters of the string x.

Šváb-Zamazal and Svátek [94] OWL ontology to OWL ontology, (s:c),(c:s),(c:c) This approach is based on structural and naming conditions to detect N-ary relation patterns as defined by the Semantic Web Best Practice (SWBP)[3] in the aligned ontologies. First, reified N-ary relations are sought in the ontologies with the help of a lexico-structural pattern. The fragment of ontology represented in Figure 3a 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 label similarities of the concepts in the N-ary relations. 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 3 shows an example of an N-ary relation (3a) and corresponding object properties (3b). The structure to structure correspondences cannot be interpreted.
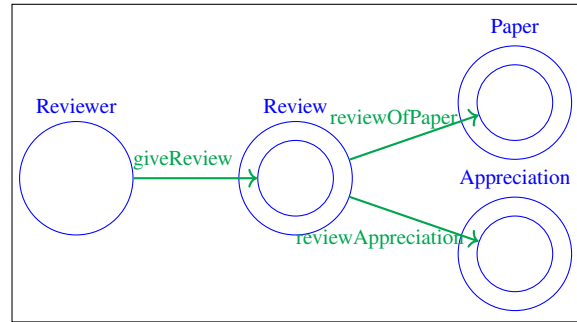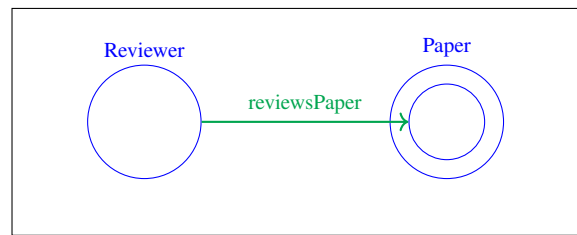
---

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

(a) o_1

(b) o_2

Fig. 3. N-ary relation pattern

PORSCHE, Saleem et al. [95] XML schema to XML schema, (s:c) (c:s) PORSCHE (Performance ORiented SCHEma Matching) [95] matches a set of XML *schema trees* (schemata with a single root) simultaneously. It is a holistic approach. This approach outputs a mediating schema (all the schemata 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 correspondences produced 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 following two approaches are also holistic: they match many schemata simultaneously. They rely on web query interfaces.

DCM, He et al. [96]   Table schema to Table schema, (s:c) (c:s) (c:c) DCM (Dual Correlation Mining) [96] is a holistic schema matching system. It aligns attribute names of Web Forms. 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., $\{s_1\text{:firstName}, s_1\text{:lastName}\}$). In the second step, each set of co-occurring attributes (e.g., $\{s_1\text{:firstName}, s_1\text{:lastName}\}$) is put in correspondence with sets of attributes which do not often co-occur with them (e.g., $\{s_2\text{:author}\}$). The correspondences are then filtered according to their confidence (negative co-occurrence) value, or aggregated if they have a common attribute: if $(\{s_1\text{:firstName}, s_1\text{:lastName}\}, \{s_2\text{:author}\}, \equiv)$ and $(\{s_2\text{:author}\}, \{s_3\text{:writer}\}, \equiv)$, then $(\{s_1\text{:firstName}, s_1\text{:lastName}\}, \{s_2\text{:author}\}, \{s_3\text{:writer}\}, \equiv)$. 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 correct correlations mined. However, only the attributes present on the web query interfaces can be involved in the correspondences.

HSM, Su et al. [97]   Table schema to Table schema, (s:c) (c:s) (c:c) HSM (Holistic Schema Matching) [97] is very similar to DCM [96] 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 already involved in a correspondence, it may be added to an existing group of attributes according to its grouping score with them.

Example 11. *The approach explores the synonyms: s_1:firstName is found to be a synonym of s_2:author.*

The following groups are formed:
$(\{s_1\text{:firstName}\}, \{s_2\text{:author}\}, \equiv)$
Then, s_1:lastName is found to be a synonym of s_2:author. Because s_1:lastName and s_1:firstName have a good grouping score, s_1:lastName is added to the correspondence as follows:
$(\{s_1\text{:firstName}, s_1\text{:lastName}\}, \{s_2\text{:author}\}, \equiv)$

Wu et al. [93]   Table schema to table schema, (s:c) (c:s) Wu *et al.* [93] propose a clustering approach to find attribute correspondences based on web query interfaces. It considers the hierarchical structure of an HTML form. It also considers the values taken in the table 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: $(\{date\}, aggregate\{day,month,year\}, \equiv)$. A *is-a* correspondence shows a union, sum, etc. of these values: $(\{passengers\}, is\text{-}a\{adults,children,seniors\}, \equiv)$. The detection conditions of these correspondences are based on the hierarchy of the web form attributes: the label of the parent node of the grouped attributes must be similar to that of the singleton attribute. For *is-a*, the grouped attributes' domains must be similar to the singleton's, whereas for *aggregate*, the domain of each grouped attribute must be similar to a subset of the singleton attribute's domain.

A clustering technique then 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 interface to interface. Thus, the output correspondences are schema to 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 ontology level). Some of them discover these simple correspondences themselves as a preliminary step [99, 100], others take them as input [58, 71, 101, 102]. Most approaches perform the path search on the graph-like or tree-like structure of the schemata/ontologies directly whereas [102] creates a *mapping graph* on which the search will be performed.

An et al. [101]   Document-oriented schema to CML ontology, (s:c) An *et al.* [101] map a web query form to a CML ontology. The attribute and fieldset names of the form are transformed into a *form tree* (derived from HTML), similar to an XML 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 of the form tree between two nodes, all sub-graphs G_i (as minimum spanning Steiner trees) in the ontology (e.g., in a book related web form an edge can link the node *s:book* to its sub-node *s:author*). The sub-graphs are property chains in the target ontology between two nodes (classes) *o:Book* and *o:Author* such that *(s:book, o:Book, ≡)* and *(s:author, o:Author, ≡)* 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. [71], Yan et al. [58]   Relational database schema to relational database schema, (s:c) (c:s) (c:c) Based on structural information of relational database schemata, the Clio system[4] [58, 71] is one of the first systems to consider the creation of complex correspondences between schemata. The user must input *value correspondences*: functions linking one or many attributes (e.g., *(s_1:Parent1.Salary + s_1:Parent2.Salary, s_2:Student.FamilyIncome, ≡))*. 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 with potential *query graphs* between attributes:

---

trees from the data source schema structure. Clio helps the user find simple, path relation and value transformation 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. [99], Dou et al. [100]   OWL ontology to OWL ontology, (c:c) OntoGrate [99] is a framework that mines *frequent queries* and outputs them as conjunctive first-order logic formulae. The system can deal with ontology matching [99] and was adapted to relational database schema matching in [100] by transforming the relational 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 correspondence generation. The algorithm iterates on both steps (simple correspondence 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 the source path and one in the target path, are found which 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 fre-

quent queries are then refined and formalised into first-order logic formulae.

Example 12. *The simple matching phase computed:*
- *(o_1:Person, o_2:Person, $\equiv$)*
- *(o_1:email, o_2:contactEmail, $\equiv$)*

However, the last correspondence is wrong as it is and considered incomplete because
- *o_1:Person is the domain of o_1:email*
- *o_2:Paper is the domain of 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:
(dom(o_1:Person) $\sqcap$ o_1:email , (dom(o_2:Person) $\sqcap$ o_2:writes) ∘ (dom(o_2:Paper) $\sqcap$ o_2:contactEmail), $\equiv$)

An and Song [102] CML ontology to CML ontology, (c:c) An and Song [102] introduce the concept of mapping graph between two ontologies in CML language. This process relies on a simple alignment between the concepts of the ontologies. The first step of the approach is to generate the mapping graph between the ontologies. The nodes of a mapping graph represent pairs of concepts from the two ontologies. For example *(o_1:Reviewer, o_2:Reviewer)* and *(o_1:Paper, o_2:Paper)* are two nodes of the mapping graph. 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 *(o_1:Reviewer, o_2:Reviewer, $\equiv$)* and *(o_1:Paper, o_2:Paper, $\equiv$)*, then the approach will look for a path between *(o_1:Reviewer, o_2:Reviewer)* and *(o_1:Paper,o_2:Paper)*.

Example 13. *If (o_1:Reviewer, o_2:Reviewer, $\equiv$) and (o_1:Paper, o_2:Paper, $\equiv$) 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 (dom(o_1:Reviewer) $\sqcap$ o_1:reviewerOf $\sqcap$ range(o_1:Paper), (dom(o_2:Reviewer) $\sqcap$ o_2:writesReview $\sqcap$ range(o_2:Review)) ∘ (o_2:reviewOf $\sqcap$ range(o_2:Paper)), $\equiv$).

5.4. Tree

While some approaches [59, 103, 104] 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 [105, 106] 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 [107, 108] use tree-based algorithms such as genetic programming. However they do not consider the schemata or ontologies as trees and therefore are not classified in this category.

MapOnto, An et al. [103, 104] Relational database schema to CML ontology [103], XML schema to OWL ontology [104], (c:c) MapOnto[5] [103, 104], a work of An *et al.* is inspired from Clio in terms of path finding and tree construction. The approaches focus on aligning a source schema to a target ontology. Two approaches were proposed: a relational database schema to ontology [103] and an XML schema to ontology [104]. Both 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 [103]) differs between the two approaches because of the different nature of the schemata. However, both approaches look for the smallest tree spanning all the attributes of the schema. A set of the most "reasonable" alignments are output

---

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

for the user to choose among. These techniques output (c:c) correspondences because a whole table (or *element tree*) is transformed in each correspondence.

Example 14. *Let PAPERS(id,title,accepted) be a table from a relational database schema. The following translation rule to map the PAPERS table to an ontology o can be output by this approach:*

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. [59, 109] Table schema, Relational database schema, XML schema, JSON schema to OWL ontology, (s:c),(c:s),(c:c) KARMA[6] [59, 109] is a semi-automatic relational database schema to ontology matching system. Other types of structured data such as JSON or XML files can be processed by KARMA: they are transformed into a relational data model in a first step following a few rules. KARMA has two parts: a structured data to ontology matching part presented in [59] and a programming-by-example algorithm [109] to create data transformation functions which falls in the *No structure* category. The structured data to ontology approach is similar to those of An *et al.* [103, 104] as it is based on a Steiner-tree algorithm and outputs FOL-like formula as alignments (as in example 14). 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. The user can also generate multiple instances

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

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 (as a translation rule). The translation rule from Example 14 could be output by KARMA.

5.5. No structure

The approaches described in this section do not follow any of the above structures. While [110] is based on Inductive Logic Programming and builds its correspondences in an *ad hoc* manner, [111] relies on competency questions and common instance predicates, [77] uses Markov Logic Networks for combinatorial exploration, [112] uses classifying techniques to generate block correspondences, [78] uses a numeric searcher using context-free grammar for equation discovery, [109] applies a user-driven programming-by-example strategy and finally [107, 108] use genetic programming to combine data value transformation functions.

Hu et al. [110] OWL ontology to OWL ontology, (s:c) The approach proposed by Hu *et al.* [110] uses Inductive Logic Programming (ILP) techniques to discover complex alignments. This technique is inspired by Stuckenschmidt *et al.* [113]. The approach is based on the common instances of a source and a target ontology. It outputs Horn-rules of the form $A \land B \land C \land ... \to D$ with $A, B, C...$ source entities represented as first-order predicates and D a target entity as a first-order predicate. The Horn-rule contains two parts: the body on the left side of the implication and the head on the right side. Three phases compose the approach. In the first, the instances of the two ontologies are matched. In the second 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 with 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 15. *At the first iteration of the process, the head of the Horn-rule is set to a predicate (unary or binary) and the body of the Horn-rule is empty. Let us consider the case when the Horn-rule head is a binary predicate:*

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

All possible pairs of common instances are classified as positive binding or negative binding with regards to whether they instantiate 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 body of the Horn-rule:

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

and in the next iteration:

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

and so on until no more positive binding is left to find or the number of predicates in the Horn-rule body has reached a threshold:

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

which translates as the correspondence:

(o_1:writesReview ∘ (o_1:reviewOfPaper ⊓ range(o_1:Paper)), o_2:reviewerOf, ⊑)

Thiéblin et al. [114] *OWL ontology to OWL ontology, (s:c) (c:s) (c:c)* In [114], 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 one 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 16. *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:paper2,o_2:aut7) : o_2:hasAuthor*
  – *o_2:aut7 : o_2:Author*
  – *(o_2:paper2, "accepted") : o_2:decision*

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

(o_1:AcceptedPaper, ∃o_2:decision.{"accepted"}, ≡)

KAOM, Jiang et al. [77] *OWL ontology to OWL ontology, (s:c) (c:s) (c:c)* KAOM (Knowledge Aware Ontology Matching) is a system proposed by Jiang *et al.* [77]. 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.* [72, 73] 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 17. *A knowledge rule could be "Many reviewers are also authors of paper", which would be in o_1 (⤳ seen as a "is often true" relation): o_1:Reviewer ⤳ ∃o_1:authorOf.o_1:Paper. The same knowledge rule expressed in o_2 would be: ∃o_2:reviewerOf.⊤ ⤳ o_2:Author. Based on these knowledge rules, two candidate corre-*

spondences can be: (o_1:Reviewer, $\exists o_2$:reviewerOf.$\top$,$\equiv$) and ($\exists o_1$:authorOf.o_1:Paper, o_2:Author, $\equiv$).

iMAP, Dhamankar et al. [78]   Relational database schema to relational database schema, (c:s) As seen previously, the iMAP system [78] 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. [107]   OWL ontology to OWL ontology, (c:s) Genetic programming can be used for finding complex correspondences between data-properties. It can combine and transform the data-properties of an ontology to match a property of another ontology. Nunes *et al.* [107] propose a genetic programming approach for numerical and literal data property 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. [108]   Table schema to table schema, (s:c) (c:s) (c:c) De Carvalho *et al.* [108] apply a 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.* [107]) 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 func-*

tions 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 similarity of matched instances (the data must be overlapping) or *value-oriented* with the similarity of all transformed source instances and target instances. The similarity metric for each correspondence is chosen by an expert. Compared to the approach of Nunes et al. [107] it can detect *(c:c)* correspondences thanks to its internal modelling. However the process may require more iterations than [107].

KARMA, Knoblock et al. [59, 109]   Table schema, Relational database schema, XML schema, JSON schema to OWL ontology, (s:c),(c:s),(c:c) The programming-by-example algorithm of KARMA (approach presented in the *Tree* category) creates data transformation functions. It considers the transformation functions as programs divided into subprograms 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 result. This first pair of values constitutes an example and a program (transformation function) is then synthesised and applied to the other instances of the data. The user iteratively corrects the wrongly 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 (substring, 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 18. *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 synthesised. For example, out of all the possible programs (called hypotheses) 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 occurrence and N denotes the n-th occurrence. START is the beginning of the value, END its end. WORD represents a ([A-Za-z]+) string, NUM a number, BNK a white-space. This program is then applied to the other instances of the data. The user iteratively corrects the wrongly translated data, giving new examples from which the process refines its program (the hypothesis space will be reduced).

BMO, Hu et al. [112]   OWL ontology to OWL 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 vectorised virtual document. In the third step, the *relatedness matrix* is used to apply a partitioning algorithm: this algorithm is recursively applied to the set of ontology entities. At the end of this algorithm, the similar entities are together in the 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 obtain 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 present the approaches in the order in which they first appear in the survey.

Table 6 summarises the type of knowledge representation models aligned by the approaches and the additional input. Most approaches require external output such as matched instances or a simple alignment. This table shows the variety of knowledge representation models for which complex matching approaches have been proposed. Table 7 presents the output by the approaches: the correspondence members form, the type of correspondence and the output format. Few approaches can generate both logic construction and transformation function correspon-

dences. Most approaches output correspondences as FOL rules, without following a particular format. The latest approaches ([74, 114] published in 2018) output correspondences in EDOAL, which coincides with their participation in the OAEI complex track (cf. §6).

Table 8 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 member expression (fixed to fixed, unfixed to unfixed, etc.) and the (s:c), (c:s) kind of correspondence.

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

- *Formal resource-based*: rely on formal evidence: upper-level ontology, domain-specific ontology, linked data, linguistic frames, alignment
- *Informal resource-based*: rely on informal evidence: directory, annotated resources, web forms
- *String-based*: use string similarity: name similarity, description similarity, global namespace
- *Language-based*: use linguistic techniques: tokenisation, lemmatisation, thesauri, lexicon, morphology
- *Constraint-based*: use 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*: compare sets of individuals: data analysis, statistics
- *Model-based*: use the semantic interpretation: SAT solvers, DL reasoners

The complex matching approaches are described according to this classification in Table 9. The majority 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-level or instance-level), as done in different surveys in the field. This classification was applied in the last 2 columns of Table 8. Most approaches use the ontology-level information as evidence. The

Table 6

Input of the approaches: type of aligned knowledge representation model and type of additional input information

Approach	Type of Knowledge Representation Model	Additional Input
Ritze2009 [72]	OWL ontology to OWL ontology	simple alignment
Ritze2010 [73]	OWL ontology to OWL ontology	simple alignment (opt.)
Faria2018 (AMLC) [74]	OWL ontology to OWL ontology	simple alignment
Oliveira2018 [25]	OWL ontology to OWL ontology	
Rouces2016 [75]	OWL ontology to OWL ontology	
Walshe2016 (Bayes-ReCCE) [76]	OWL ontology to OWL ontology	matched instances
Chondrogiannis2014 (OAT) [60]	OWL ontology to OWL ontology	
Dhamankar2004 (iMAP) [78]	RDB schema to RDB schema	domain constraints and value distribution
Jiang2016 (KAOM) [77]	OWL ontology to OWL ontology	knowledge rules
Jimenez2015 (BootOX) [79]	RDB schema to OWL ontology	
Parundekar2012 [84]	OWL ontology to OWL ontology	matched instances
Parundekar2010 [85]	OWL ontology to OWL ontology	matched instances
Doan2003 (CGLUE) [86]	DL ontology to DL ontology	
Kaabi2012 (ARCMA) [87]	OWL ontology to OWL ontology	simple alignment
Boukottaya2005 [88]	XML schema to XML schema	
Arnold2013 (COMA++) [92]	Taxonomy to Taxonomy	
Xu2003 [89]	Conceptual Model to Conceptual Model	domain ontology
Xu2006 [90]	Conceptual Model to Conceptual Model	domain ontology
Warren2006 [91]	Table schema to Table schema	
Svab2009 [94]	OWL ontology to OWL ontology	
Saleem2008 (PORSCHE) [95]	XML schema to XML schema	abbreviation Table schema
He2004 (DCM) [96]	Table schema to Table schema	web query interfaces
Su2006 (HSM) [97]	Table schema to Table schema	web query interfaces
Wu2004 [93]	Table schema to Table schema	web query interfaces
An2012 [101]	Document-oriented schema to CML ontology	web query interfaces, simple correspondences web form-onto
Yan2001 (Clio) [58, 71]	RDB schema to RDB schema	value correspondences
Qin2007 (OntoGrate) [99, 100]	OWL ontology to OWL ontology	
An2008 [102]	CML ontology to CML ontology	simple alignment
An2005 (MapOnto) [103]	RDB schema to CML ontology	attribute-data properties correspondences
An2005b (MapOnto) [104]	XML schema to OWL ontology	attribute-data properties correspondences
Knoblock2012 (KARMA) [59, 109]	Table, RDB, XML, JSON schema to OWL ontology	examples for data transformation functions
Hu2011 [110]	OWL ontology to OWL ontology	
Thieblin2018 [114]	OWL ontology to OWL ontology	competency questions for alignment as SPARQL queries
Nunes2011 [107]	OWL ontology to OWL ontology	
deCarvalho2013 [108]	Table schema to Table schema	
Hu2006 (BMO) [112]	OWL ontology to OWL ontology	

Table 7

Output of the approaches: correspondence members form, types of correspondences and correspondence format

Approach	(s:c)	(c:s)	(c:c)	Logic	Transfo	Block	Correspondence format
Ritze2009 [72]	•			•			pseudo-DL
Ritze2010 [73]	•			•			EDOAL
Faria2018 (AMLC) [74]	•			•			EDOAL
Oliveira2018 [25]	•			•			Not specified
Rouces2016 [75]	•			•			SPARQL construct
Walshe2016 (Bayes-ReCCE) [76]	•	•		•			EDOAL
Chondrogiannis2014 (OAT) [60]	•	•	•	•			OWL, EDOAL
Dhamankar2004 (iMAP) [78]		•			•		equations
Jiang2016 (KAOM) [77]	•	•	•	•	•		pseudo-DL
Jimenez2015 (BootOX) [79]		•		•			R2RML
Parundekar2012 [84]	•	•		•			pseudo-DL
Parundekar2010 [85]	•	•	•	•			pseudo-DL
Doan2003 (CGLUE) [86]	•			•			Not specified
Kaabi2012 (ARCMA) [87]	•			•			DL
Boukottaya2005 [88]	•	•	•	•	•		XSLT
Arnold2013 (COMA++) [92]	•	•			•		Not specified
Xu2003 [89]	•	•		•	•		Not specified
Xu2006 [90]	•	•	•	•	•		Not specified
Warren2006 [91]		•			•		SQL queries
Svab2009 [94]	•	•	•			•	Not specified
Saleem2008 (PORSCHE) [95]	•	•				•	Not specified
He2004 (DCM) [96]	•	•	•			•	sets
Su2006 (HSM) [97]	•	•	•			•	sets
Wu2004 [93]	•	•	•			•	sets
An2012 [101]	•			•			Not specified
Yan2001 (Clio) [58, 71]	•	•	•	•			SQL views
Qin2007 (OntoGrate) [99, 100]			•	•			DataLog, SWRL, Web-PDDL
An2008 [102]			•	•			FOL or SPARQL
An2005 (MapOnto) [103]			•	•			FOL
An2005b (MapOnto) [104]			•	•			FOL
Knoblock2012 (KARMA) [59, 109]	•	•	•	•	•		FOL
Hu2011 [110]		•		•			FOL
Thieblin2018 [114]	•	•	•	•			EDOAL
Nunes2011 [107]		•			•		equations
deCarvalho2013 [108]	•	•	•		•		equations
Hu2006 (BMO) [112]	•	•	•			•	sets

Table 8

Process characteristics of the approaches based on the proposed classification

Approach	Guiding structure	fixed to fixed	fixed to unfixed	unfixed to unfixed	Ontology-level evidence	Instance-level evidence	Other
Ritze2009 [72]	Atomic patterns	●			●		
Ritze2010 [73]	Atomic patterns	●			●		
Faria2018 (AMLC) [74]	Atomic patterns	●			●		
Oliveira2018 [25]	Atomic patterns	●			●		Compound
Rouces2016 [75]	Atomic patterns	●			●		
Walshe2016 (Bayes-ReCCE) [76]	Atomic patterns	●				●	
Chondrogiannis2014 (OAT) [60]	Atomic patterns	●			●		
Dhamankar2004 (iMAP) [78]	Atomic patterns, Composite patterns, No structure	●	●		●		
Jiang2016 (KAOM) [77]	Atomic patterns, No structure	●		●	●	●	
Jimenez2015 (BootOX) [79]	Atomic patterns	●			●		
Parundekar2012 [84]	Composite patterns		●		●		
Parundekar2010 [85]	Composite patterns			●	●		
Doan2003 (CGLUE) [86]	Composite patterns		●		●		
Kaabi2012 (ARCMA) [87]	Composite patterns		●		●	●	
Boukottaya2005 [88]	Composite patterns		●		●		
Arnold2013 (COMA++) [92]	Composite patterns		●		●		
Xu2003 [89]	Composite patterns	●	●		●	●	
Xu2006 [90]	Composite patterns, Path to path	●	●	●	●	●	
Warren2006 [91]	Composite patterns		●			●	
Svab2009 [94]	Composite patterns		●	●	●		
Saleem2008 (PORSCHE) [95]	Composite patterns		●		●		Holistic
He2004 (DCM) [96]	Composite patterns			●	●		Holistic
Su2006 (HSM) [97]	Composite patterns			●	●		Holistic
Wu2004 [93]	Composite patterns		●		●		
An2012 [101]	Path to Path		●		●		
Yan2001 (Clio) [58, 71]	Path to Path			●	●	●	
Qin2007 (OntoGrate) [99, 100]	Path to Path			●	●	●	
An2008 [102]	Path to Path			●	●		
An2005 (MapOnto) [103]	Tree to tree			●	●		
An2005b (MapOnto) [104]	Tree to tree			●	●		
Knoblock2012 (KARMA) [59, 109]	Tree to tree, No structure		●	●	●	●	
Hu2011 [110]	No structure		●		●	●	
Thieblin2018 [114]	No structure		●		●	●	
Nunes2011 [107]	No structure		●		●		
deCarvalho2013 [108]	No structure			●	●		
Hu2006 (BMO) [112]	No structure			●	●		

Table 9

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
Ritze2009 [72]			•	•	•	•	•		
Ritze2010 [73]			•	•	•	•	•		
Faria2018 (AMLC) [74]			•		•	•	•		
Oliveira2018 [25]			•						
Rouces2016 [75]	•		•	•			•		
Walshe2016 (Bayes-ReCCE) [76]	•						•	•	
Chondrogiannis2014 (OAT) [60]			•		•	•	•		
Dhamankar2004 (iMap) [78]	•		•		•		•	•	
Jiang2016 (KAOM) [77]	•		•		•			•	•
Jimenez2015 (BootOX) [79]				•	•		•		
Parundekar2012 [84]	•						•	•	
Parundekar2010 [85]	•						•	•	
Doan2003 (CGLUE) [86]			•			•	•		
Kaabi2012 (ARCMA) [87]			•			•	•		
Boukottaya2005 [88]				•	•	•	•		
Arnold2013 (COMA++) [92]			•			•			
Xu2003 [89]	•		•				•	•	
Xu2006 [90]	•		•	•			•	•	
Warren2006 [91]								•	
Svab2009 [94]			•				•		
Saleem2008 (PORSCHE) [95]				•		•			
He2004 (DCM) [96]		•							
Su2006 (HSM) [97]		•							
Wu2004 [93]		•	•	•	•	•		•	
An2012 [101]		•					•		
Yan2001 (Clio) [58, 71]					•		•	•	
Qin (OntoGrate) [99, 100]							•	•	
An2008 [102]	•					•	•		
An2005 (MapOnto) [103]	•						•		
An2005b (MapOnto) [104]	•						•		
Knoblock2012 (KARMA) [59, 109]		•	•				•	•	
Hu2011 [110]								•	
Thieblin2018 [114]	•		•				•	•	
Nunes2011 [107]								•	
deCarvalho [108]								•	
Hu2006 (BMO) [112]			•				•		

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. While most of the approaches have been manually evaluated, the few automatic evaluations are often done on approach-tailored datasets (e.g., correspondences following one pattern only). In this section, we do not analyse the evaluation section of each approach individually but we review the initiatives for complex alignment evaluation.

6.1. Complex alignment datasets

The diverse complex matching approaches exploit a variety of knowledge representation models (e.g., XML schemata, ontologies) and resources (e.g., linked instances, web forms). 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 we present the datasets available online with reference to complex alignments and the benchmarks which deal with complex alignments. They are summarised in Table 10.

In the domain of schema matching (database or XML schema), dedicated complex alignment datasets have been constructed to evaluate the approaches dealing with these schemata. In general, these datasets contain mostly transformation functions. For instance, the Illinois semantic integration archive [115] 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 [116] is a repository of schemata and query forms. XBench-Match [117] 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 [118] proposes an evalua-

tion 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 [48] is a benchmark for instance matching 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 obtain 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 [119] among which the GeoLink dataset [21] and a consensus version of [111].

Of all the datasets presented in Table 10, 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.) [76, 84, 85, 110, 114] or custom-made knowledge bases [77, 78, 107, 108].

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 metrics as defined in [96, 97, 112] but those 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 as true positives or false pos-

Table 10

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

itives. 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 is often the only metric computed. The comparison of complex correspondences is a difficult task which is often performed manually. This makes the evaluation time-consuming and experts on the aligned ontologies are required during the evaluation process. The precision and recall metrics have been adapted into weighted precision and recall, relaxed precision and recall, and semantic precision and recall [44]. The *weighted* precision and recall take the confidence value of a correspondence into account. The *relaxed* precision and recall [120] 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 [121] 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 [78, 79, 86, 96, 97, 101, 108, 114] 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 best three 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.

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

All the metrics presented above 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

Until recently, there was little work on complex alignment evaluation. The latest works propose datasets. Only RODI [118] 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, *(o₁:Author, ∃o₂:authorOf.⊤, ≡)* is semantically equivalent to the correspondence *(o₁:Author,*

$\exists o_2$:*writtenBy*$^-$.$\top$*, $\equiv$)*. However, these two correspondences are synthetically different. 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 alignment formats such as EDOAL may lead to other problems. For example, the correspondences *(o_1:AcceptedPaper, $\exists o_2$:acceptedBy.$\top$, $\equiv$)* and *(o_1:AcceptedPaper, $\geqslant$ 1 o_2:acceptedBy, $\equiv$)* are equivalent but expressed using different constructors (respectively an existential restriction or a cardinality restriction over the o_2:*acceptedBy* property). This also complexifies the comparison of the two correspondences. 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 ($\equiv, \sqsupseteq, \sqsubseteq$) 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, or weighted precision and recall.

Finally, measuring the suitability of the output alignment for a given application, as done for the OA4QA track of the OAEI [122], for a library application [123] or the Taxon track of the complex OAEI track [119], could be considered.

7. Discussion

Interest in complex alignment has recently increased in the community. This probably comes from the fact that applications needing interoperability find simple alignment not sufficient.

Since the XML and database fields are older than ontology matching and ontology-based data access (OBDA), they have stable standards such as XSLT, XQuery and SQL. In the OBDA community, R2RML has become the norm, and many extensions to it proposed, 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 (cf. §3.1), the EDOAL vocabulary implemented in the Alignment API can be seen as an upcoming standard. It has indeed been used for the first OAEI track. However, EDOAL has limited expressiveness, as discussed in [21]. Moreover, there is only one edition or visualisation tool for this format (OAT [60]) but it is not available online. This makes EDOAL only usable or browse-able by experts. 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, contrary to what intuition may suggest, matching more expressive knowledge representation models does not imply applying more sophisticated techniques. Most approaches consider knowledge representation models as graphs, trees or pools of annotated data regardless of their expressiveness. These common representations lead to similar techniques over diverse knowledge representation models. The proposed classification tried to capture some of the aspects described above, by focusing on the specificities of complex correspondences on two main axes. The first axis characterises the different types of output (type of correspondences) and the second the structures used in the process to guide the correspondence detection. With respect to this classification, some approaches adopt a *mono-strategy* (atomic patterns, for instance), while others can fall in diverse categories. Classifying some of the approaches into a specific category was not a simple task.

While some approaches rely on existing simple correspondences (at ontology or instance level), others are able to discover complex correspondences without this kind of input. Other resources are used as evidence such as web query interfaces, knowledge rules, competency questions for alignment or linguistic resources such as WordNet. Another aspect of the approach is related to the kind of correspondence relation they can output. As for simple alignments, most works are still limited to generating equivalences. The semantics of the confidence of a correspondence are rarely considered.

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 problem of sparseness or with a specific corpus distribution that leads to incorrect correspondences. For example, if o_1 is populated with most students aged 23, *(o_1:Student, $\exists o_2$:age.$\{23\}$, $\equiv$)* can be a valid correspondence for the instance-based matching algorithms.

Most approaches are limited to pair-wise matching. Holistic and compound complex matching approaches are scarce but may be needed in complex domains,

such as bio-medicine, where several ontologies describing different but related phenomena have to be linked together [124]. As stated in [125], the increase in the matching space and the inherently higher difficulty to compute alignments pose interesting challenges to this task.

On a different matter, we observe that some correspondences are pragmatically coherent but not semantically equivalent. For example, $(o_1$:*Ticket*, o_2:*Adult* $+ o_2$:*Children* $+ o_2$:*Senior*, $\equiv)$ is a practical correspondence for counting the number of passengers. The semantic meaning of this correspondence is however questionable as a ticket and a passenger are not exactly the same thing. This raises questions about the notion alignment context. In [126] "context mappings" define to which extent an alignment is valid.

Moreover, user involvement is under-exploited in complex matching. This aspect, related to the visualisation and edition of complex correspondences [127, 128], is an important issue to be addressed in the future.

Regarding the evaluation of complex alignments, automatic correspondence comparison remains an open issue. 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 [129], could be also an interesting resource. Another direction would be to evaluate the complex alignments over a real-life application such as ontology merging, data translation or query rewriting. The suitability of the alignment for the given task could be automatically computed. 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:10.1007/11603412_5.

[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, 2013, pp. 593–607. doi:10.1007/978-3-642-38288-8_40.

[3] V. de Boer, J. Wielemaker, J. van 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: *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*, E. Simperl, P. Cimiano, A. Polleres, Ó. Corcho and V. Presutti, eds, Lecture Notes in Computer Science, Vol. 7295, Springer, 2012, pp. 733–747. 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, in: *Proceedings of the 2007 International Conference on Dublin Core and Metadata Applications, DC 2007, Singapore, August 27-31, 2007*, S.A. Sutton, A.S. Chaudhry and C.S.G. Khoo, eds, Dublin Core Metadata Initiative, 2007, pp. 128–139.

[5] T. Nurmikko-Fuller, K.R. Page, P. Willcox, J. Jett, C. Maden, T.W. 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-CE Joint Conference on Digital Libraries, Knoxville, TN, USA, June 21-25, 2015*, P.L.B. II, S. Allard, H. Mercer, M. Beck, S.J. Cunningham, D.H. Goh and G. Henry, eds, ACM, 2015, pp. 169–172. doi:10.1145/2756406.2756944.

[6] É. 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: *Metadata and Semantic Research - 11th International Conference, MTSR 2017 Tallinn, Estonia, November 28 - December 1, 2017, Proceedings*, E. Garoufallou, S. Virkus, R. Siatri and D. Koutsomiha, eds, Communications in Computer and Information Science, Vol. 755, Springer, 2017, pp. 25–37. doi:10.1007/978-3-319-70863-8_3.

[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), 6:1–6:12. doi:10.1186/s13326-017-0114-4.

[8] K.W. Fung and J. Xu, Synergism between the Mapping Projects from SNOMED CT to ICD-10 and ICD-10-CM, in: *AMIA 2012, American Medical Informatics Association Annual Symposium, Chicago, Illinois, USA, November 3-7, 2012*, American Medical Informatics Association, 2012.

[9] K. Giannangelo and J. Millar, Mapping SNOMED CT to ICD-10, in: *Quality of Life through Quality of Information - Proceedings of MIE2012, The XXIVth International Congress of the European Federation for Medical Informatics, Pisa, Italy, August 26-29, 2012*, J. Mantas, S.K. Andersen, M.C. Mazzoleni, B. Blobel, S. Quaglini and A. Moen, eds, Studies in Health Technology and Informatics, Vol. 180, IOS Press, 2012, pp. 83–87. doi:10.3233/978-1-61499-101-4-83.

[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. 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. doi:10.1017/S0269888903000651.

[12] N.F. Noy, Semantic integration: a survey of ontology-based approaches, *ACM Sigmod Record* **33**(4) (2004), 65–70. doi:10.1145/1041410.1041421.

[13] J. De Bruijn, M. Ehrig, C. Feier, F. Martín-Recuerda, F. Scharffe and M. Weiten, Ontology mediation, merging and aligning, in: *Semantic Web Technologies: Trends and Research in Ontology-Based Systems*, J. Davies, R. Studer and P. Warren, eds, John Wiley and Sons, 2006. doi:10.1002/047003033X.ch6.

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

[15] A. Doan and A.Y. Halevy, Semantic integration research in the database community: A brief survey, *AI Magazine* **26**(1) (2005), 83–94.

[16] J. Euzenat and P. Shvaiko, *Ontology Matching, Second edition*, Springer Berlin Heidelberg, 2013. ISBN 978-3-642-38720-3.

[17] G. Stapleton, J. Howse, A. Bonnington and J. Burton, A Vision for Diagrammatic Ontology Engineering, in: *Proceedings of the International Workshop on Visualizations and User Interfaces for Knowledge Engineering and Linked Data Analytics co-located with 19th International Conference on Knowledge Engineering and Knowledge Management, VISUAL@EKAW 2014, Linköping, Sweden, November 24, 2014*, V. Ivanova, T. Kauppinen, S. Lohmann, S. Mazumdar, C. Pesquita and K. Xu, eds, CEUR Workshop Proceedings, Vol. 1299, CEUR-WS.org, 2014, pp. 1–13.

[18] D.L. McGuinness and F. Van Harmelen, OWL web ontology language overview, W3C recommendation, W3C, 2004.

[19] I. Horrocks, DAML+OIL: A Description Logic for the Semantic Web, *IEEE Data Engineering Bulletin* **25**(1) (2002), 4–9.

[20] G. Schreiber, B.J. Wielinga, H. Akkermans, W.V. de Velde and A. Anjewierden, CML: The CommonKADS Conceptual Modelling Language, in: *A Future for Knowledge Acquisition, 8th European Knowledge Acquisition Workshop, EKAW'94, Hoegaarden, Belgium, September 26-29, 1994, Proceedings*, L. Steels, G. Schreiber and W.V. de Velde, eds, Lecture Notes in Computer Science, Vol. 867, Springer, 1994, pp. 1–25. doi:10.1007/3-540-58487-0_1.

[21] L. Zhou, M. Cheatham, A. Krisnadhi and P. Hitzler, A Complex Alignment Benchmark: Geolink Dataset, in: *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part II*, D. Vrandecic, K. Bontcheva, M.C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L. Kaffee and E. Simperl, eds, Lecture Notes in Computer Science, Vol. 11137, Springer, 2018, pp. 273–288. doi:10.1007/978-3-030-00668-6_17.

[22] J. Euzenat, Towards composing and benchmarking ontology alignments, in: *Proceedings of the Semantic Integration Workshop Collocated with the Second International Semantic Web Conference (ISWC-03), Sanibel Island, Florida, USA, 2003*, Vol. 82, CEUR-WS.org, 2003, pp. 165–166.

[23] T. Grütze, C. Böhm and F. Naumann, Holistic and Scalable Ontology Alignment for Linked Open Data, in: *WWW2012 Workshop on Linked Data on the Web, Lyon, France, 16 April, 2012*, C. Bizer, T. Heath, T. Berners-Lee and M. Hausenblas, eds, CEUR Workshop Proceedings, Vol. 937, CEUR-WS.org, 2012.

[24] I. Megdiche, O. Teste and C.T. dos Santos, An Extensible Linear Approach for Holistic Ontology Matching, in: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, P.T. Groth, E. Simperl, A.J.G. Gray, M. Sabou, M. Krötzsch, F. Lécué, F. Flöck and Y. Gil, eds, Lecture Notes in Computer Science, Vol. 9981, 2016, pp. 393–410. doi:10.1007/978-3-319-46523-4_24.

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

[26] A. Gater, D. Grigori and M. Bouzeghoub, Complex mapping discovery for semantic process model alignment, in: *iiWAS'2010 - The 12th International Conference on Information Integration and Web-based Applications and Services, 8-10 November 2010, Paris, France*, G. Kotsis, D. Taniar, E. Pardede, I. Saleh and I. Khalil, eds, ACM, 2010, pp. 317–324. doi:10.1145/1967486.1967537.

[27] G. Navarro, A guided tour to approximate string matching, *ACM computing surveys (CSUR)* **33**(1) (2001), 31–88. doi:10.1145/375360.375365.

[28] 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. doi:10.1017/S0269888913000349.

[29] 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*, I.F. Cruz, S. Decker, J. Euzenat and D.L. McGuinness, eds, 2001, pp. 75–91.

[30] M.C.A. Klein, D. Fensel, A. Kiryakov and D. Ognyanov, Ontology Versioning and Change Detection on the Web, in: *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002, Proceedings*, A. Gómez-Pérez and V.R. Benjamins, eds, Lecture Notes in Computer Science, Vol. 2473, Springer, 2002, pp. 197–212. doi:10.1007/3-540-45810-7_20.

[31] L. Stojanovic, A. Maedche, B. Motik and N. Stojanovic, User-Driven Ontology Evolution Management, in: *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002, Proceedings*, A. Gómez-Pérez and V.R. Benjamins, eds, Lecture Notes in Computer Science, Vol. 2473, Springer, 2002, pp. 285–300. doi:10.1007/3-540-45810-7_27.

[32] M. Völkel, C.F. Enguix, S.R. Kruk, A.V. Zhdanova, R. Stevens and Y. Sure, Semversion: A versioning system for RDF and ontologies, Project deliverable, Knowledge Web, 2005.

[33] N.F. Noy and M.A. Musen, PROMPTDIFF: A Fixed-Point Algorithm for Comparing Ontology Versions, in: *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, July 28 - August 1, 2002, Edmonton, Alberta, Canada*, R. Dechter, M.J. Kearns and R.S. Sutton, eds, AAAI Press / The MIT Press, 2002, pp. 744–750.

[34] V. Papavassiliou, G. Flouris, I. Fundulaki, D. Kotzinos and V. Christophides, On Detecting High-Level Changes in RDF/S KBs, in: *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*, A. Bernstein, D.R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta and K. Thirunarayan, eds, Lecture Notes in Computer Science, Vol. 5823, Springer, 2009, pp. 473–488. doi:10.1007/978-3-642-04930-9_30.

[35] D. Zeginis, Y. Tzitzikas and V. Christophides, On the Foundations of Computing Deltas Between RDF Models, in: *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, K. Aberer, K. Choi, N.F. Noy, D. Allemang, K. Lee, L.J.B. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber and P. Cudré-Mauroux, eds, Lecture Notes in Computer Science, Vol. 4825, Springer, 2007, pp. 637–651. doi:10.1007/978-3-540-76298-0_46.

[36] M. Hartung, J.F. Terwilliger and E. Rahm, Recent Advances in Schema and Ontology Evolution, in: *Schema Matching and Mapping*, Z. Bellahsene, A. Bonifati and E. Rahm, eds, Data-Centric Systems and Applications, Springer, 2011, pp. 149–190. doi:10.1007/978-3-642-16518-4_6.

[37] H. Stuckenschmidt and M.C.A. Klein, Integrity and Change in Modular Ontologies, in: *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, G. Gottlob and T. Walsh, eds, Morgan Kaufmann, 2003, pp. 900–908.

[38] 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. doi:10.1016/j.jbi.2013.09.006.

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

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

[41] M. Kay, XSL Transformations (XSLT) Version 3.0, W3C Recommandation, W3C, 2017.

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

[43] J. Euzenat, F. Scharffe and A. Zimmermann, Expressive alignment language and implementation, Project deliverable, Knowledge Web, 2007.

[44] J. David, J. Euzenat, F. Scharffe and C. Trojahn, The Alignment API 4.0, *Semantic Web* **2**(1) (2011), 3–10. doi:10.3233/SW-2011-0028.

[45] M.T. Pazienza, A. Stellato, M. Vindigni and F.M. Zanzotto, XeOML: An XML-based extensible ontology mapping language, in: *Working notes of the ISWC-04 Workshop on Meaning Coordination and Negotiation(MCN-04) held in conjunction with3rd International Semantic Web Conference (ISWC-2004), Hiroshima, Japan*, P. Bouquet and L. Serafini, eds, 2004, pp. 83–94.

[46] A. Maedche, B. Motik, N. Silva and R. Volz, MAFRA - A MApping FRAmework for Distributed Ontologies, in: *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002, Proceedings*, A. Gómez-Pérez and V.R. Benjamins, eds, Lecture Notes in Computer Science, Vol. 2473, Springer, 2002, pp. 235–250. doi:10.1007/3-540-45810-7_23.

[47] N. Silva and J. Rocha, Semantic Web complex ontology mapping, in: *2003 IEEE / WIC International Conference on Web Intelligence, (WI 2003), 13-17 October 2003, Halifax, Canada*, IEEE Computer Society, 2003, pp. 82–88. doi:10.1109/WI.2003.1241177.

[48] T. Saveta, E. Daskalaki, G. Flouris, I. Fundulaki, M. Herschel and A.N. 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 Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*, A. Gangemi, S. Leonardi and A. Panconesi, eds, ACM, 2015, pp. 105–106. doi:10.1145/2740908.2742729.

[49] 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 2018, July 13-19, 2018, Stockholm, Sweden*, J. Lang, ed., ijcai.org, 2018, pp. 5511–5519. doi:10.24963/ijcai.2018/777.

[50] M. Hert, G. Reif and H.C. Gall, A comparison of RDB-to-RDF mapping languages, in: *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, C. Ghidini, A.N. Ngomo, S.N. Lindstaedt and T. Pellegrini, eds, ACM International Conference Proceeding Series, ACM, 2011, pp. 25–32. doi:10.1145/2063518.2063522.

[51] S. Das, S. Sundara and R. Cyganiak, R2RML: RDB to RDF Mapping Language, W3C Recommandation, W3C, 2012.

[52] 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: *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014*, C. Bizer, T. Heath, S. Auer and T. Berners-Lee, eds, CEUR Workshop Proceedings, Vol. 1184, CEUR-WS.org, 2014.

[53] B. De Meester, A. Dimou, R. Verborgh and E. Mannens, An Ontology to Semantically Declare and Describe Functions, in: *The Semantic Web - ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers*, H. Sack, G. Rizzo, N. Steinmetz, D. Mladenić, S. Auer and C. Lange, eds, Lecture Notes in Computer Science, Vol. 9989, Springer International Publishing, 2016, pp. 46–49. doi:10.1007/978-3-319-47602-5_10.

[54] F. Michel, L. Djimenou, C. Faron-Zucker and J. Montagnat, Translation of Relational and Non-relational Databases into RDF with xR2RML, in: *WEBIST 2015 - Proceedings of the 11th International Conference on Web Information Systems and Technologies, Lisbon, Portugal, 20-22 May, 2015*, V. Monfort, K. Krempels, T.A. Majchrzak and Z. Turk, eds, SciTePress, 2015, pp. 443–454. doi:10.5220/0005448304430454.

[55] A. Chortaras and G. Stamou, Mapping Diverse Data to RDF in Practice, in: *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, D. Vrandecic, K. Bontcheva, M.C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L. Kaffee and E. Simperl, eds, Lecture Notes in Computer Science, Vol. 11136, Springer, 2018, pp. 441–457. doi:10.1007/978-3-030-00671-6_26.

[56] M.A. Musen, The Protégé project: a look back and a look forward, *AI matters* **1**(4) (2015), 4–12. doi:10.1145/2757001.2757003.

[57] 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, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part II*, L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral and T. Tudorache, eds, Lecture Notes in Computer Science, Vol. 6089, Springer, 2010, pp. 381–385. doi:10.1007/978-3-642-13489-0_28.

[58] L. Yan, R.J. Miller, L.M. Haas and R. Fagin, Data-Driven Understanding and Refinement of Schema Mappings, in: *Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, USA, May 21-24, 2001*, S. Mehrotra and T.K. Sellis, eds, ACM, 2001, pp. 485–496. doi:10.1145/375663.375729.

[59] C.A. Knoblock, P.A. 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 - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*, E. Simperl, P. Cimiano, A. Polleres, Ó. Corcho and V. Presutti, eds, Lecture Notes in Computer Science, Vol. 7295, Springer, 2012, pp. 375–390. doi:10.1007/978-3-642-30284-8_32.

[60] E. Chondrogiannis, V. Andronikou, E. Karanastasis and T.A. Varvarigou, An Intelligent Ontology Alignment Tool Dealing with Complicated Mismatches, in: *Proceedings of the 7th International Workshop on Semantic Web Applications and Tools for Life Sciences, Berlin, Germany, December 9-11, 2014*, A. Paschke, A. Burger, P. Romano, M.S. Marshall and A. Splendiani, eds, CEUR Workshop Proceedings, Vol. 1320, CEUR-WS.org, 2014.

[61] J. Rouces, G. de Melo and K. Hose, Addressing structural and linguistic heterogeneity in the Web, *AI Communications* **31**(1) (2018), 3–18. doi:10.3233/AIC-170745.

[62] 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. doi:10.1007/978-3-540-88845-1_5.

[63] 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.

[64] A.C. Junior, C. Debruyne and D. O'Sullivan, An Editor that Uses a Block Metaphor for Representing Semantic Mappings in Linked Data, in: *The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers*, A. Gangemi, A.L. Gentile, A.G. Nuzzolese, S. Rudolph, M. Maleshkova, H. Paulheim, J.Z. Pan and M. Alam, eds, Lecture Notes in Computer Science, Vol. 11155, Springer, 2018, pp. 28–33. doi:10.1007/978-3-319-98192-5_6.

[65] 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: *The Semantic Web: ESWC 2017 Satellite Events - ESWC 2017 Satellite Events, Portorož, Slovenia, May 28 - June 1, 2017, Revised Selected Papers*, E. Blomqvist, K. Hose, H. Paulheim, A. Lawrynowicz, F. Ciravegna and O. Hartig, eds, Lecture Notes in Computer Science, Vol. 10577, Springer, 2017, pp. 87–92. doi:10.1007/978-3-319-70407-4_17.

[66] Á. Sicilia, G. Nemirovski and A. Nolle, Map-On: A web-based editor for visual ontology mapping, *Semantic Web* **8**(6) (2017), 969–980. doi:10.3233/SW-160246.

[67] P. Heyvaert, A. Dimou, A. Herregodts, R. Verborgh, D. Schuurman, E. Mannens and R.V. de Walle, RMLEditor: A Graph-Based Mapping Editor for Linked Data Mappings, in: *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, H. Sack, E. Blomqvist, M. d'Aquin, C. Ghidini, S.P. Ponzetto and C. Lange, eds, Lecture Notes in Computer Science, Vol. 9678, Springer, 2016, pp. 709–723. doi:10.1007/978-3-319-34129-3_43.

[68] 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: *Informal Proceedings of the 27th International Workshop on Description Logics, Vienna, Austria, July 17-20, 2014*, M. Bienvenu, M. Ortiz, R. Rosati and M. Simkus, eds, CEUR Workshop Proceedings, Vol. 1193, CEUR-WS.org, 2014, pp. 595–607.

[69] M. Blinkiewicz and J. Bak, SQuaRE: A Visual Approach for Ontology-Based Data Access, in: *Semantic Technology - 6th Joint International Conference, JIST 2016, Singapore, Singapore, November 2-4, 2016, Revised Selected Papers*, Y. Li, W. Hu, J.S. Dong, G. Antoniou, Z. Wang, J. Sun and Y. Liu, eds, Lecture Notes in Computer Science, Vol. 10055, Springer, 2016, pp. 47–55. doi:10.1007/978-3-319-50112-3_4.

[70] S. Kandel, A. Paepcke, J.M. Hellerstein and J. Heer, Wrangler: interactive visual specification of data transformation scripts, in: *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*, D.S. Tan, S. Amershi, B. Begole, W.A. Kellogg and M. Tungare, eds, ACM, 2011, pp. 3363–3372. doi:10.1145/1978942.1979444.

[71] R.J. Miller, L.M. Haas and M.A. Hernández, Schema Mapping as Query Discovery, in: *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, A.E. Abbadi, M.L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter and K. Whang, eds, Morgan Kaufmann, 2000, pp. 77–88.

[72] D. Ritze, C. Meilicke, O. Sváb-Zamazal and H. Stucken-schmidt, A Pattern-based Ontology Matching Approach for Detecting Complex Correspondences, in: *Proceedings of the 4th International Workshop on Ontology Matching (OM-2009) collocated with the 8th International Semantic Web Conference (ISWC-2009) Chantilly, USA, October 25, 2009*, P. Shvaiko, J. Euzenat, F. Giunchiglia, H. Stuckenschmidt, N.F. Noy and A. Rosenthal, eds, CEUR Workshop Proceedings, Vol. 551, CEUR-WS.org, 2009.

[73] D. Ritze, J. Völker, C. Meilicke and O. Sváb-Zamazal, Linguistic analysis for complex ontology matching, in: *Proceedings of the 5th International Workshop on Ontology Matching (OM-2010), Shanghai, China, November 7, 2010*, P. Shvaiko, J. Euzenat, F. Giunchiglia, H. Stuckenschmidt, M. Mao and I.F. Cruz, eds, CEUR Workshop Proceedings, Vol. 689, CEUR-WS.org, 2010.

[74] D. Faria, C. Pesquita, B.S. Balasubramani, T. Tervo, D. Carriço, R. Garrilha, F.M. Couto and I.F. Cruz, Results of AML participation in OAEI 2018, in: *Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, October 8, 2018*, P. Shvaiko, J. Euzenat, E. Jiménez-Ruiz, M. Cheatham and O. Hassan-zadeh, eds, CEUR Workshop Proceedings, Vol. 2288, CEUR-WS.org, 2018, pp. 125–131.

[75] J. Rouces, G. de Melo and K. Hose, Complex Schema Mapping and Linking Data: Beyond Binary Predicates, in: *Proceedings of the Workshop on Linked Data on the Web, LDOW 2016, co-located with 25th International World Wide Web Conference (WWW 2016)*, S. Auer, T. Berners-Lee, C. Bizer and T. Heath, eds, CEUR Workshop Proceedings, Vol. 1593, CEUR-WS.org, 2016.

[76] B. Walshe, R. Brennan and D. O'Sullivan, Bayes-ReCCE: A Bayesian Model for Detecting Restriction Class Correspondences in Linked Open Data Knowledge Bases, *International Journal on Semantic Web and Information Systems (IJSWIS)* **12**(2) (2016), 25–52. doi:10.4018/IJSWIS.2016040102.

[77] 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*, Vol. 28, Springer, 2016, pp. 75–95. doi:10.1007/978-3-662-53455-7_4.

[78] R. Dhamankar, Y. Lee, A. Doan, A.Y. Halevy and P.M. Domingos, iMAP: Discovering Complex Mappings between Database Schemas, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, G. Weikum, A.C. König and S. Deßloch, eds, ACM, 2004, pp. 383–394. doi:10.1145/1007568.1007612.

[79] 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: *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*, M. Arenas, Ó. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P.T. Groth, M. Dumontier, J. Heflin, K. Thirunarayan and S. Staab, eds, Lecture Notes in Computer Science, Vol. 9367, Springer, 2015, pp. 113–132, Springer. doi:10.1007/978-3-319-25010-6_7.

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

[81] E. Jiménez-Ruiz and B.C. Grau, LogMap: Logic-Based and Scalable Ontology Matching, in: *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N.F. Noy and E. Blomqvist, eds, Lecture Notes in Computer Science, Vol. 7031, Springer, 2011, pp. 273–288. doi:10.1007/978-3-642-25073-6_18.

[82] L.F. de Medeiros, F. Priyatna and Ó. Corcho, MIRROR: Automatic R2RML Mapping Generation from Relational Databases, in: *Engineering the Web in the Big Data Era - 15th International Conference, ICWE 2015, Rotterdam, The Netherlands, June 23-26, 2015, Proceedings*, P. Cimiano, F. Frasincar, G. Houben and D. Schwabe, eds, Lecture Notes in Computer Science, Vol. 9114, Springer, 2015, pp. 326–343. doi:10.1007/978-3-319-19890-3_21.

[83] 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. doi:10.3233/SW-160217.

[84] R. Parundekar, C.A. Knoblock and J.L. Ambite, Discovering Concept Coverings in Ontologies of Linked Data Sources, in: *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I*, P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J.X. Parreira, J. Hendler, G. Schreiber, A. Bernstein and E. Blomqvist, eds, Lecture Notes in Computer Science, Vol. 7649, Springer, 2012, pp. 427–443. doi:10.1007/978-3-642-35176-1_27.

[85] R. Parundekar, C.A. Knoblock and J.L. Ambite, Linking and Building Ontologies of Linked Data, in: *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, P.F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J.Z. Pan, I. Horrocks and B. Glimm, eds, Lecture Notes in Computer Science, Vol. 6496, Springer, 2010, pp. 598–614. doi:10.1007/978-3-642-17746-0_38.

[86] 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.

[87] F. Kaabi and F. Gargouri, A new approach to discover the complex mappings between ontologies, *International Journal of Web Science* **1**(3) (2012), 242–256. doi:10.1504/IJWS.2012.045814.

[88] A. Boukottaya and C. Vanoirbeek, Schema matching for transforming structured documents, in: *Proceedings of the 2005 ACM Symposium on Document Engineering, Bristol, UK, November 2-4, 2005*, A. Wiley and P.R. King, eds, ACM, 2005, pp. 101–110. doi:10.1145/1096601.1096629.

[89] L. Xu and D.W. Embley, Using domain ontologies to discover direct and indirect matches for schema elements, in: *Semantic Integration Workshop (SI-2003) collocated with the Second International Semantic Web Conference, October 20, 2003, Sanibel Island, Florida, USA*, A. Doan, A. Halevy and N. Noy, eds, 2003.

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

[91] 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, Seoul, Korea, September 12-15, 2006*, U. Dayal, K. Whang, D.B. Lomet, G. Alonso, G.M. Lohman, M.L. Kersten, S.K. Cha and Y. Kim, eds, ACM, 2006, pp. 331–342.

[92] P. Arnold, Semantic Enrichment of Ontology Mappings: Detecting Relation Types and Complex Correspondences, in: *Proceedings of the 25th GI-Workshop "Grundlagen von Datenbanken 2013", Ilmenau, Germany, May 28 - 31, 2013*, K. Sattler, S. Baumann, F. Beier, H. Betz, F. Gropengießer and S. Hagedorn, eds, CEUR Workshop Proceedings, Vol. 1020, CEUR-WS.org, 2013, pp. 34–39.

[93] W. Wu, C.T. Yu, A. Doan and W. Meng, An Interactive Clustering-based Approach to Integrating Source Query interfaces on the Deep Web, in: *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, G. Weikum, A.C. König and S. Deßloch, eds, ACM, 2004, pp. 95–106. doi:10.1145/1007568.1007582.

[94] 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.

[95] K. Saleem, Z. Bellahsene and E. Hunt, Porsche: Performance oriented schema mediation, *Information Systems* **33**(7) (2008), 637–657. doi:10.1016/j.is.2008.01.010.

[96] B. He, K.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, Seattle, Washington, USA, August 22-25, 2004*, W. Kim, R. Kohavi, J. Gehrke and W. DuMouchel, eds, ACM, 2004, pp. 148–157. doi:10.1145/1014052.1014071.

[97] W. Su, J. Wang and F.H. Lochovsky, Holistic Schema Matching for Web Query Interfaces, in: *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*, Y.E. Ioannidis, M.H. Scholl, J.W. Schmidt, F. Matthes, M. Hatzopoulos, K. Böhm, A. Kemper, T. Grust and C. Böhm, eds, Lecture Notes in Computer Science, Vol. 3896, Springer, 2006, pp. 77–94. doi:10.1007/11687238_8.

[98] S. Maßmann, S. Raunich, D. Aumüller, P. Arnold and E. Rahm, Evolution of the COMA match system, in: *Proceedings of the 6th International Workshop on Ontology Matching, Bonn, Germany, October 24, 2011*, P. Shvaiko, J. Euzenat, T. Heath, C. Quix, M. Mao and I.F. Cruz, eds, CEUR Workshop Proceedings, Vol. 814, CEUR-WS.org, 2011.

[99] H. Qin, D. Dou and P. LePendu, Discovering Executable Semantic Mappings Between Ontologies, in: *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, OTM Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS 2007, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part I*, R. Meersman and Z. Tari, eds, Lecture Notes in Computer Science, Vol. 4803, Springer, 2007, pp. 832–849. doi:10.1007/978-3-540-76848-7_56.

[100] D. Dou, H. Qin and P. Lependu, Ontograte: towards Automatic Integration for Relational Databases and the Semantic Web through an Ontology-Based Framework, *Interna-*

[101] *tional Journal of Semantic Computing* **4**(1) (2010), 123–151. doi:10.1142/S1793351X10000961.

Y. An, X. Hu and I. Song, Learning to discover complex mappings from web forms to ontologies, in: *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, X. Chen, G. Lebanon, H. Wang and M.J. Zaki, eds, ACM, 2012, pp. 1253–1262. doi:10.1145/2396761.2398427.

[102] Y. An and I.-Y. Song, Discovering semantically similar associations (SeSA) for complex mappings between conceptual models, in: *Conceptual Modeling - ER 2008, 27th International Conference on Conceptual Modeling, Barcelona, Spain, October 20-24, 2008. Proceedings*, Q. Li, S. Spaccapietra, E.S.K. Yu and A. Olivé, eds, Lecture Notes in Computer Science, Vol. 5231, Springer, 2008, pp. 369–382. doi:10.1007/978-3-540-87877-3_27.

[103] Y. An, A. Borgida and J. Mylopoulos, Inferring Complex Semantic Mappings Between Relational Tables and Ontologies from Simple Correspondences, in: *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 31 - November 4, 2005, Proceedings, Part II*, R. Meersman, Z. Tari, M. Hacid, J. Mylopoulos, B. Pernici, Ö. Babaoglu, H. Jacobsen, J.P. Loyall, M. Kifer and S. Spaccapietra, eds, Lecture Notes in Computer Science, Vol. 3761, Springer, 2005, pp. 1152–1169. doi:10.1007/11575801_15.

[104] Y. An, A. Borgida and J. Mylopoulos, Constructing Complex Semantic Mappings Between XML Data and Ontologies, in: *The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*, Y. Gil, E. Motta, V.R. Benjamins and M.A. Musen, eds, Lecture Notes in Computer Science, Vol. 3729, Springer, 2005, pp. 6–20. doi:10.1007/11574620_4.

[105] 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*, Vol. 14, S. Spaccapietra and L. Delcambre, eds, Springer, 2009, pp. 37–73. doi:10.1007/978-3-642-10562-3_2.

[106] 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. doi:10.1016/j.jbi.2012.04.009.

[107] B.P. Nunes, A.A.M. Caraballo, M.A. Casanova, K.K. Breitman and L.A.P.P. Leme, Complex matching of RDF datatype properties, in: *Proceedings of the 6th International Workshop on Ontology Matching, Bonn, Germany, October 24, 2011*, P. Shvaiko, J. Euzenat, T. Heath, C. Quix, M. Mao and I.F. Cruz, eds, CEUR Workshop Proceedings, Vol. 814, CEUR-WS.org, 2011.

[108] 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. doi:10.1016/j.is.2012.10.002.

[109] 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, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, Q. Yang and M.J. Wooldridge, eds, AAAI Press, 2015, pp. 1726–1732.

[110] W. Hu, J. Chen, H. Zhang and Y. Qu, Learning complex mappings between ontologies, in: *The Semantic Web - Joint International Semantic Technology Conference, JIST 2011, Hangzhou, China, December 4-7, 2011. Proceedings*, J.Z. Pan, H. Chen, H. Kim, J. Li, Z. Wu, I. Horrocks, R. Mizoguchi and Z. Wu, eds, Lecture Notes in Computer Science, Vol. 7185, Springer, 2011, pp. 350–357. doi:10.1007/978-3-642-29923-0_24.

[111] É. Thiéblin, O. Haemmerlé, N. Hernandez and C. Trojahn, Task-Oriented Complex Ontology Alignment: Two Alignment Evaluation Sets, in: *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, A. Gangemi, R. Navigli, M. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai and M. Alam, eds, Lecture Notes in Computer Science, Vol. 10843, Springer, 2018, pp. 655–670. doi:10.1007/978-3-319-93417-4_42.

[112] W. Hu and Y. Qu, Block Matching for Ontologies, in: *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, I.F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold and L. Aroyo, eds, Lecture Notes in Computer Science, Vol. 4273, Springer, pp. 300–313. doi:10.1007/11926078_22.

[113] 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, ILP 2008 Prague, Czech Republic, September 10-12, 2008*, F. Železný and N. Lavrač, eds, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 105–111.

[114] E. Thiéblin, O. Haemmerlé and C. Trojahn, Complex matching based on competency questions for alignment: a first sketch, in: *Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, October 8, 2018*, CEUR Workshop Proceedings, Vol. 2288, CEUR-WS.org, 2018, pp. 66–70.

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

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

[117] 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, University of Vienna, Austria, September 23-27, 2007*, C. Koch, J. Gehrke, M.N. Garofalakis, D. Srivastava, K. Aberer, A. Deshpande, D. Florescu, C.Y. Chan, V. Ganti, C. Kanne, W. Klas and E.J. Neuhold, eds, ACM, 2007, pp. 1318–1321.

[118] 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. doi:10.3233/SW-170268.

[119] E. Thiéblin, M. Cheatham, C. Trojahn, O. Zamazal and L. Zhou, The First Version of the OAEI Complex Alignment Benchmark, in: *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018*, M. van Erp, M. Atre, V. López, K. Srinivas and C. Fortuna, eds, CEUR Workshop Proceedings, Vol. 2180, CEUR-WS.org, 2018.

[120] M. Ehrig and J. Euzenat, Relaxed precision and recall for ontology matching, in: *Integrating Ontologies '05, Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies, Banff, Canada, October 2, 2005*, CEUR Workshop Proceedings, Vol. 156, CEUR-WS.org, 2005, pp. 25–32.

[121] J. Euzenat, Semantic Precision and Recall for Ontology Alignment Evaluation, in: *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, M.M. Veloso, ed., 2007, pp. 348–353.

[122] A. Solimando, E. Jiménez-Ruiz and C. Pinkel, Evaluating ontology alignment systems in query answering tasks, in: *Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC 2014, Riva del Garda, Italy, October 21, 2014*, M. Horridge, M. Rospocher and J. van Ossenbruggen, eds, CEUR Workshop Proceedings, Vol. 1272, CEUR-WS.org, 2014, pp. 301–304.

[123] 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: *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, S. Bechhofer, M. Hauswirth, J. Hoffmann and M. Koubarakis, eds, Lecture Notes in Computer Science, Vol. 5021, Springer, 2008, pp. 402–417. doi:10.1007/978-3-540-68234-9_31.

[124] D. Oliveira and C. Pesquita, Compound matching of biomedical ontologies, in: *Proceedings of the International Conference on Biomedical Ontology, ICBO 2015, Lisbon, Portugal, July 27-30, 2015*, F.M. Couto and J. Hastings, eds, CEUR Workshop Proceedings, Vol. 1515, CEUR-WS.org, 2015.

[125] C. Pesquita, M. Cheatham, D. Faria, J. Barros, E. Santos and F.M. Couto, Building reference alignments for compound matching of multiple ontologies using OBO cross-products, in: *Proceedings of the 9th International Workshop on Ontology Matching collocated with the 13th International Semantic Web Conference (ISWC 2014), Riva del Garda, Trentino, Italy, October 20, 2014*, P. Shvaiko, J. Euzenat, M. Mao, E. Jiménez-Ruiz, J. Li and A. Ngonga, eds, CEUR Workshop Proceedings, Vol. 1317, CEUR-WS.org, 2014, pp. 172–173.

[126] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini and H. Stuckenschmidt, C-OWL: Contextualizing Ontologies, in: *The Semantic Web - ISWC 2003, Second International Semantic Web Conference, Sanibel Island, FL, USA, October 20-23, 2003, Proceedings*, D. Fensel, K.P. Sycara and J. Mylopoulos, eds, Lecture Notes in Computer Science, Vol. 2870, Springer, 2003, pp. 164–179. doi:10.1007/978-3-540-39718-2_11.

[127] 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), 983–1024. doi:10.1016/j.ijhcs.2003.08.002.

[128] Z. Dragisic, V. Ivanova, P. Lambrix, D. Faria, E. Jiménez-Ruiz and C. Pesquita, User Validation in Ontology Alignment, in: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October*

17-21, 2016, Proceedings, Part I, P.T. Groth, E. Simperl, A.J.G. Gray, M. Sabou, M. Krötzsch, F. Lécué, F. Flöck and Y. Gil, eds, Lecture Notes in Computer Science, Vol. 9981, 2016, pp. 200–217. doi:10.1007/978-3-319-46523-4_13.

[129] M. Cheatham and P. Hitzler, Conference v2.0: An Uncertain Version of the OAEI Conference Benchmark, in: *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II*, P. Mika, T. Tudorache, A. Bernstein, C. Welty, C.A. Knoblock, D. Vrandecic, P.T. Groth, N.F. Noy, K. Janowicz and C.A. Goble, eds, Lecture Notes in Computer Science, Vol. 8797, Springer, 2014, pp. 33–48. doi:10.1007/978-3-319-11915-1_3.