# A Survey of Schema-based Matching Approaches

Pavel Shvaiko[1] and Jérôme Euzenat[2]

[1] University of Trento, Povo, Trento, Italy,
pavel@dit.unitn.it
[2] INRIA, Rhône-Alpes, France,
Jerome.Euzenat@inrialpes.fr

**Abstract.** Schema/ontology matching is a critical problem in many application domains, such as, semantic web, schema/ontology integration, data warehouses, e-commerce, catalog matching, etc. Many diverse solutions to the matching problem have been proposed so far. In this paper we present a new classification of schema-based matching techniques that builds on the previous work on classifying schema matching approaches. Some innovations are in introducing new criteria which are based on (i) general properties of matching techniques, (ii) interpretation of input information, and (iii) the kind of input information. In particular, we distinguish between heuristic and exact techniques at schema-level; and syntactic, external, and semantic techniques at element- and structure-level. Based on the classification proposed we overview some of the recent schema/ontology matching systems pointing which part of the solution space they cover. The classification proposed provides a common conceptual basis, and hence can be used for comparing different existing schema/ontology matching systems as well as for designing a new one, taking advantages of state of the art solutions.

## 1 Introduction

*Matching* is a critical operation in many well-known application domains, such as, semantic web, schema/ontology integration, data warehouses, e-commerce, XML message mapping, catalog matching, etc. It takes two schemas/ontologies, each consisting of a set of discrete entities (e.g., tables, XML elements, classes, properties, rules, predicates) as input and determines as output the relationships (e.g., equivalence, subsumption) holding between these entities.

Many diverse solutions to the matching problem have been proposed so far, for example [29, 25, 14, 32, 53, 1, 27, 34, 37, 31]. Good surveys through the recent years are provided in [42, 52, 26]. The survey of [26] focuses on current state of the art in ontology alignment. Authors review recent approaches, techniques and tools. The survey of [52] concentrates on approaches to ontology-based information integration and discusses general matching approaches that are used in information integration systems. However, none of the above mentioned surveys provide a comparative review of the existing ontology alignment techniques and systems. On the contrary, the survey of [42] is devoted to a classification of database schema matching approaches and a comparative review of matching systems.

In this paper we focus only on schema-based solutions, i.e., matching systems exploiting only intensional information, not instance data. Although, there is a difference

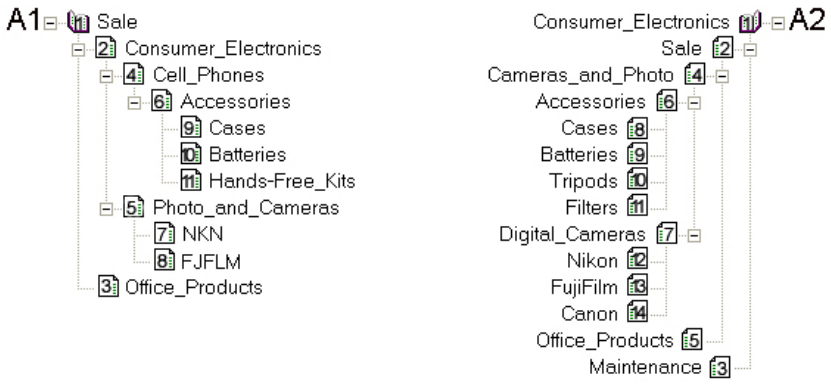


**Fig. 1.** Two XML schemas

between schema and ontology matching (alignment)[1] problems (see next section for details), we believe that techniques developed for each of them can be of a mutual benefit, therefore we discuss schema and ontology matching as the same problem. We bring together and discuss systematically recent approaches and systems developed in schema matching and ontology alignment domains. We present a new classification of schema/ontology matching approaches that builds on the work of [42] augmented in [45, 20] and [18]. Some innovations are in introducing new criteria which are based on (i) general properties of matching techniques, (ii) interpretation of input information, and (iii) the kind of input information. We distinguish between heuristic and exact techniques at schema-level; and syntactic, external, and semantic techniques at element- and structure-level. Based on the classification proposed we provide a comparative review of the recent schema/ontology matching systems pointing which part of the solution space they cover.

The rest of the paper is organized as follows. Section 2 provides, via an example, the basic motivations and definitions to the schema/ontology matching problem. Section 3 discusses possible matching dimensions. Section 4 introduces the classification of elementary automatic schema-based matching techniques and discusses in details possible alternatives. Section 5 provides a vision on classifying matching systems. Section 6 overviews some of the recent schema/ontology matching solutions in light of the classification proposed pointing which part of the solution space they cover. Section 7 reports some conclusions.

## 2 The Matching Problem

### 2.1 Motivating Example

To motivate the matching problem, let us use two simple XML schemas that are shown in Figure 1 and exemplify one of the possible situations which arise, for example, when resolving a schema integration task.

Suppose an e-commerce company A1 needs to finalize a corporate acquisition of another company A2. To complete the acquisition we have to integrate databases of

[1] We use terms matching and alignment interchangeably.

the two companies. The documents of both companies are stored according to XML schemas A1 and A2 respectively. Numbers in boxes are the unique identifiers of the XML elements. A first step in integrating the schemas is to identify candidates to be merged or to have taxonomic relationships under an integrated schema. This step refers to a process of schema matching. For example, the elements with labels $Office\_Products$ in A1 and in A2 are the candidates to be merged, while the element with label $Digital\_Cameras$ in A2 should be subsumed by the element with label $Photo\_and\_Cameras$ in A1.

### 2.2 Schema Matching vs Ontology Alignment

In this paper we discuss the problem of matching schemas and ontologies from the perspective of information which is exploited by matching systems in order to determine correspondences between the input schemas/ontologies. In this respect, ontology alignment differs substantially from schema matching in the following two (among the others, see [36]) areas:

- Database schemas often do not provide explicit semantics for their data. Semantics is usually specified explicitly at design-time, and frequently is not becoming a part of a database specification, therefore it is not available. Ontologies are logical systems that themselves incorporate semantics (intuitive or formal). For example, in the case of formal semantics we can interpret ontology definitions as a set of logical axioms.
- Ontology data models are richer (the number of primitives is higher, and they are more complex) then schema data models. For example, OWL [47] allows defining inverse properties, transitive properties; disjoint classes, new classes as unions or intersections of other classes.

However, ontologies can be viewed as schemas (and vice versa, schemas can be viewed as simple ontologies) for knowledge bases. Having defined classes and slots in the ontology, we populate the knowledge base with instance data [36]. Thus, techniques developed for each separate problem can be of interest to each other.

On the one side, schema matching is usually performed with the help of techniques trying to guess the meaning encoded in the schemas. On the other side, ontology matching systems (primarily) try to exploit knowledge explicitly encoded in the ontologies. In real-world applications, schemas/ontologies usually have both well defined and obscure labels (terms), and contexts they occur, therefore, solutions from both problems would be mutually beneficial.

### 2.3 Problem Statement

A *mapping element* is a 5-tuple: $\langle id, e, e', n, R\rangle$, where

- $id$ is a unique identifier of the given mapping element;
- $e$ and $e'$ are entities (e.g., tables, XML elements, properties, classes, individuals) of the first and the second schema/ontology respectively;

– $R$ is a relation (e.g., *equivalence* (=); *more general* ($\sqsupseteq$); *mismatch* ($\perp$); *overlapping* ($\sqcap$)), that should hold between the entities $e$ and $e'$;
– $n$ is a *confidence measure* in some mathematical structure (typically in the [0,1] range), that holds for the correspondence between the entities $e$ and $e'$.

The above definition is highly motivated by the work of [17, 5] which contain an in-depth discussion on a format for expressing alignments between ontologies.

An *alignment* is a set of mapping elements. The *matching* operation determines an alignment. For instance, in Figure 1, according to some matching algorithm based on linguistic and structure analysis, the confidence measure (for the fact that the equivalence relation holds) between entities with labels $Photo\_and\_Cameras$ in A1 and $Cameras\_and\_Photo$ in A2 could be 0.67. While, the relation between the same pair of entities, according to another matching algorithm which is able to determine that both entities mean the same thing, could be exactly the equivalence relation.

The matching process generates an alignment ($A'$) from a pair of schemas/ontologies ($o$ and $o'$). However, there are various other parameters which can extend the definition of the matching process. These are namely, the use of an input alignment ($A$) which is to be completed by the process, the matching parameters, $p$ (e.g., weights) and some external resources used by the matching process, $r$ (e.g., lexicons). The matching process described above is schematically represented in Figure 2.

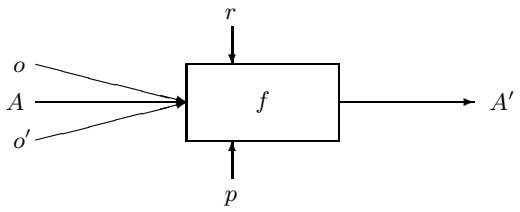


**Fig. 2.** The matching process

### 2.4 Applications

Matching is a critical operation in many well-known application domains, such as semantic web, schema/ontology integration, data warehouses, e-commerce, and semantic query processing, see [42]. More recently, new application domains have emerged, such as agent communication, web services integration, catalog matching, peer-to-peer (P2P) databases, etc. Let us discuss them in a more detail.

**Agent Communication.** Agents are computer entities characterized by autonomy and capacity of interaction. Cognitive agents communicate through speech-act inspired languages which determine the ”envelope” of the messages and enable agents to position them within a particular interaction contexts. The actual content of messages is to be

expressed in knowledge representation languages and often refer to some ontology. As a consequence, when two autonomous and independently designed agents meet, they have the possibility of exchanging messages but little chance to understand each others if they do not share the same content language and ontology. Thus, it is necessary to provide the possibility for these agents to match their ontologies in order to either translate their messages or integrate bridge axioms in their own models, see [51]. One solution to this problem is to have an ontology alignment protocol able to be interleaved with any other agent interaction protocol and which could be triggered upon receiving a message expressed in an alien ontology.

**Web Services Integration.** Web services are processes that expose their interface to the web so that users can invoke them. Semantic web services provide a richer and more precise way to describe the services through the use of knowledge representation languages and ontologies. Web service discovery and integration is the process of finding a web service able to deliver a particular service and composing several services in order to achieve a particular goal, see [39]. However, semantic web services descriptions have no reasons to be expressed by reference to exactly the same ontologies. Henceforth, both for finding the adequate service and for interfacing services it will be necessary to establish the correspondences between the terms of the description. This can be provided either on-line or off-line through matching the corresponding ontologies.

**Catalog Matching.** In B2B applications, trade partners store their products in electronic catalogs. Catalogs are tree-like structures, namely concept hierarchies with attributes. Typical examples of catalogs are product directories of www.amazon.com, www.ebay.com, etc. In order for a private company to participate in the marketplace (e.g., eBay), it is used to determine correspondences between entries of its catalogs and entries of a single catalog of a marketplace. This process of mapping entries among catalogs is referred to the catalog matching problem, see [6]. Having aligned the catalogs, users of a marketplace have a unified access to the products which are on sale.

**P2P Databases.** P2P networks are characterized by an extreme flexibility and dynamics. Peers may appear and disappear on the network, their databases are autonomous in their language, contents, how they can change their schemas, and so on. Since the peer schemas are autonomous, they might use different terminology, even if they refer to the same domain of interest. Thus, in order establish (meaningful) information exchange between peers, one of the steps is to identify and characterize relationships between their schemas. This process is that of schema matching. However, P2P applications pose additional requirements on matching algorithms. In P2P settings an assumption that all peers rely on one global schema, as in data integration, can not be made, because the global schema should be updated any time the system evolves, see [24]. Thus, if in the case of data integration schema matching operation can be performed at design time, in P2P applications peers need coordinating their databases on the fly, therefore requiring a run time schema matching operation.

## 3 The Matching Dimensions

There are many independent dimensions along which algorithms can be classified. Figure 2 suggests possible axis for classifying matching approaches: we may classify them according to (i) input of the algorithms, (ii) characteristics of the matching process, and (iii) output of the algorithms. We use these axis for presenting possible dimensions.

**Input dimensions.** These dimensions concern the kind of input on which algorithms operate. Algorithms can be classified depending on the data models in which ontologies or schemas are expressed. For example, the Artemis [7] system supports the relational, OO, and ER data models; Cupid [29] supports XML and relational data models; QOM [16] supports RDF and OWL models. A second possible dimension depends on the kind of data that the algorithms exploit: different approaches exploit different information of the input data models, some of them rely only on schema-level information (e.g., Cupid [29], COMA[25]), others rely only on instance data (e.g., GLUE [14]), or exploit both, schema- and instance-level information (e.g., QOM [16]). Even with the same data models, matching systems do not always use all available constructs. For example, S-Match [22] so far is limited to class hierarchies, and does not handle attributes.

**Process dimensions.** A classification of the matching process could be based on its general properties, as soon as we restrict ourselves to formal algorithms. In particular, it depends on the *heuristic or exact* nature of its computation. Heuristics algorithms sacrifice exactness to performance (e.g., [16]). All the techniques discussed in the remainder of the paper can be either heuristic or exact. Another dimension for analyzing the matching algorithms is based on the way they interpret the input data. In particular, we define three large classes based on the intrinsic input, external resources, or some semantic theory of the considered entities. We call these three classes *syntactic*, *external*, and *semantic* respectively; and discuss them in details in the next section.

**Output dimensions.** Apart from the information that matching systems exploit and how they manipulate it, the other important class of dimensions concerns the form of the result they produce. The form of the alignment might be of importance: is it a one-to-one correspondence between the ontology entities? Has it to be a true mapping? Is any relation suitable? Other significant distinctions in the output results have been indicated in [21]. One dimension concerns whether systems deliver a graded answer, i.e., *identity*, or an all-or-nothing answer. In these approaches correspondences between schema/ontology entities are determined using confidence measures provided with respect to the equivalence relation, usually in [0,1] range, for example, with the help of similarity coefficients [29, 20]. Another dimension concerns the kind of relations between aligned entities a system can provide. Most of the systems focus on equivalence (providing a [0,1] identity for it), while a few other are able to provide a more expressive result (e.g., equivalence (=), subsumption ($\sqsubseteq$), see for details [22]).

In the next sections we first present a classification of elementary matchers (notice that each of them can be implemented as exact or heuristic algorithm, depending on the goals of a matching system), and then we discuss how matching systems themselves can also be classified.

## 4 A retained classification of elementary schema-based matching approaches

At present, there exists a line of semi-automated schema/ontology matching systems, see, for instance [29, 25, 14, 32, 53, 1, 27, 34, 37, 31]. As indicated in introduction of the paper we build on the previous work of classifying automated schema matching approaches of [42]. The classification of [42] distinguishes between *elementary* (individual) matchers and *combinations* of matchers. Elementary matchers comprise *instance-based* and *schema-based*, *element-* and *structure-level*, *linguistic-* and *constrained-based* matching techniques. Also *cardinality* and *auxiliary information* (e.g., dictionaries, global schemas) can be taken into account.

In this section we discuss only schema-based elementary matchers, and we address issues of their combination in the next section. Therefore, only schema/ontology information is considered, not instance data[2]. For classifying elementary schema-based matching techniques, we introduce a three-layered classification, see Figure 3:

- the *upper layer* is based on (i) granularity of match, i.e., element- or structure-level, and then (ii) on how the techniques generally interpret the input information;
- the *middle layer* represents classes of elementary matching techniques and their concrete examples;
- the *lower layer* is based on the kind of input which is used by elementary matching techniques.

The classification of Figure 3 can be read both in descending (focusing on how the techniques interpret the input information) and ascending (focusing on the kind of manipulated objects) manner. To make the distinctions between the categories proposed clearer, we mark in bold type the innovations with regard to the initial classification of [42].

Elementary matchers are distinguished by the upper layer according to the following classification criteria:

- *Element-level vs structure-level.* Element-level matching techniques compute mapping elements by analyzing entities in isolation, ignoring their relations with other entities. Structure-level techniques compute mapping elements by analyzing how entities are related together. This criterion is the same as first introduced in [42].
- *Syntactic vs external vs semantic*. The key characteristic of the syntactic techniques is that they interpret the input in function of its sole structure following some clearly stated algorithm. External are the techniques exploiting in the matching process

[2] Prominent solutions of instance-based schema/ontology matching as well as possible extensions of the instance-based part of the classification of [42] can be found in [14] and [27] respectively.

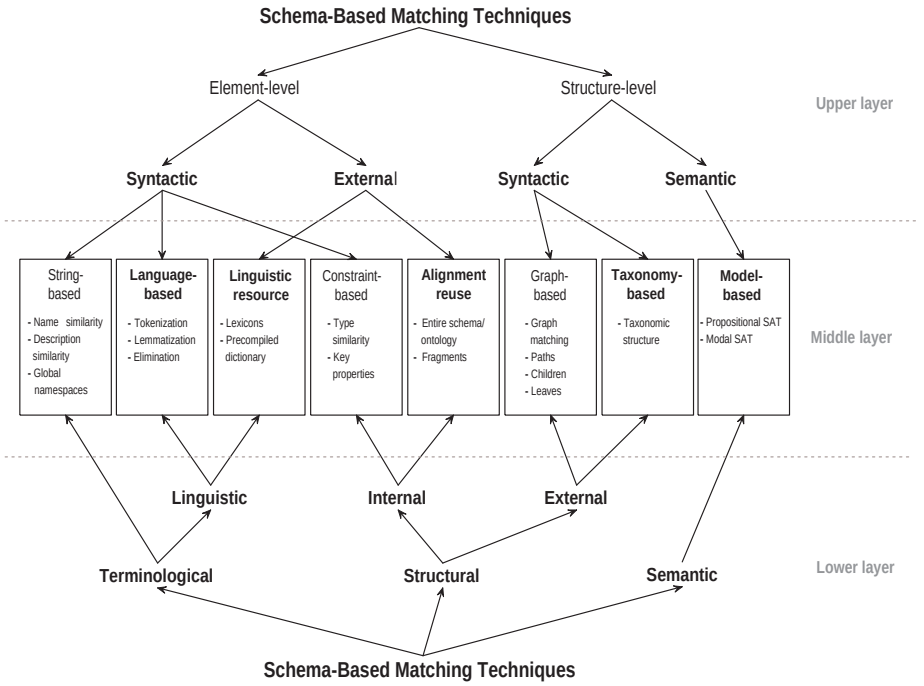


**Fig. 3.** A retained classification of elementary schema-based matching approaches

auxiliary (external) resources of a domain and common knowledge in order to interpret the input. These resources might be human input or some lexicon expressing the relationships between terms or concepts. The key characteristic of the semantic techniques is that they use some formal semantics (e.g., model-theoretic semantics) to interpret the input and justify their results. In case of a semantic based matching system, exact algorithms are complete while heuristic algorithms tend to be incomplete.

Distinctions between classes of elementary matching techniques in the middle layer of our classification are motivated by the way a matching technique interprets the input information in each concrete case. In particular, a label can be interpreted as a string (a sequence of letters from an alphabet) or as a word or a phrase in some natural language, a hierarchy can be considered as a graph (a set of nodes related by edges) or a taxonomy (a set of concepts having a set-theoretic interpretation organized by a relation which preserves inclusion). Thus, we introduce the following classes of elementary schema/ontology matching techniques at the element-level: *string-based*, *language-based*, *based on linguistic resources*, *constraint-based*, and *alignment reuse*. At the structure-level we distinguish between: *graph-based*, *taxonomy-based*, and *model-based techniques*.

The lower layer of the classification is only concerned with the type of input considered by a particular technique. Its first level is categorized on broad classes depending on which kind of data the algorithms work on: strings (*terminological*), structure (*structural*) or models (*semantics*). The second level distinguish between *linguistics* methods, and methods based on *internal* and *external* structures.

We discuss below the main alternatives (also indicating in which matching systems they were exploited) according to the above classification in more detail. Notice, that we omit in further discussions element-level semantic techniques and structure-level external techniques. In the former case, the semantics is usually given in a structure and, hence there are no element-level semantic techniques; regarding the latter case, we are not aware of any matching systems exploiting libraries providing similarities between different structures. We have also split linguistic techniques of [42] into string-based techniques and language-based techniques in order to provide fine-grained distinctions between possible alternatives.

### 4.1 Element-level techniques

**String-based techniques** are often used in order to match names and name descriptions of schema/ontology entities. These techniques consider strings as sequences of letters in an alphabet. A comparison of different string matching techniques, from *distance* like functions to *token-based distance* functions can be found in [9]. Usually, distance functions map a pair of strings to a real number, where a smaller value of the real number indicates a greater similarity between the strings. Some examples of string-based techniques which are extensively used in matching systems are *prefix, suffix, edit distance*, and *n-gram* tests.

- *Prefix*. This test takes as input two strings and checks whether the first string starts with the second one. *Prefix* is efficient in matching cognate strings and similar acronyms (e.g., *int* and *integer*), see, for example [29, 25, 32, 23].
- *Suffix*. This test takes as input two strings and checks whether the first string ends with the second one (e.g., *phone* and *telephone*), see, for example [29, 25, 32, 23].
- *Edit distance*. This test takes as input two strings and calculates the edit distance between the strings. That is, the number of *insertions*, *deletions*, and *substitutions* of characters required to transform one string into another, normalized by $max(length(string1), length(string2))$. For example, the edit distance between *NKN* and *Nikon* is 0.4. Some of matching systems exploiting the given test are [25, 37, 23].
- *N-gram*. This test takes as input two strings and calculates the number of the same n-grams (i. e., sequences of $n$ characters) between them. For example, $trigram(3)$ for the string *nikon* are *nik, iko, kon*. Some of matching systems exploiting the given test are [25, 23].

**Language-based techniques** consider names as words in some natural language (e.g., English). They are based on Natural Language Processing (NLP) techniques exploiting morphological properties of the input words.

- *Tokenization*. Names of entities are parsed into tokens by a tokenizer which recognizes punctuation, cases, blank characters, digits, etc. (e.g., $Hands\_Free\_Kits \rightarrow \langle hands, free, kits \rangle$, see, for example [22].
- *Lemmatization*. Tokens at names are are morphologically analyzed in order to find all their possible basic forms (e.g., $Kits \rightarrow Kit$), see, for example [22].
- *Elimination*. Tokens that are articles, prepositions, conjunctions, and so on, are marked (by some matching algorithms, e.g., [29]) to be discarded.

Usually, the above mentioned techniques are applied to names of entities before running string-based or lexicon-based techniques in order to improve their results. However, we consider these language-based techniques as a separate class of matching techniques, since they are naturally extended in a distance computation (by comparing the resulting strings or sets of strings).

**Constraint-based techniques** are algorithms which deal with the internal constraints being applied to the definitions of entities, such as types, cardinality of attributes, and keys. We omit here a discussion of matching cardinalities and keys as these techniques appear in our classification without changes in respect to the original publication [42]. However, we provide a more detailed overview on matching datatypes.

- *Datatypes comparison* involves comparing the various attributes of a class with regard to the datatypes of their value. Contrary to objects that require interpretations, the datatypes can be considered objectively and it is possible to determine how a datatype is close to another (ideally this can be based on the interpretation of datatypes as sets of values and the set-theoretic comparison of these datatypes, see [49, 50]). For instance, the datatype *day* can be considered closer to the datatype *workingday* than the datatype *integer*. This technique is used in [19].
- *Multiplicity comparison* attribute values can be collected by a particular construction (set, list, multiset) on which cardinality constraints are applied. Again, it is possible to compare the so constructed datatypes by comparing (i) the datatypes on which they are constructed and (ii) the cardinality that are applied to them. For instance, a set of between 2 and 3 children is closer to a set of 3 people than a set of 10-12 flowers (if children are people). This technique is used in [19].

**Linguistic resources** such as lexicons or thesauri are used in order to match words (in this case names of schema/ontology entities are considered as words of a natural language) based on linguistic relations between them (e.g., synonyms, hyponyms).

- *Lexicons and thesauri*. The approach is to use lexicons to obtain meaning of terms used in schemas/ontologies. For example, WordNet [35] is an electronic lexical database for English (and other languages), where various *senses* (possible meanings of a word or expression) of words are put together into sets of synonyms. Relations between schema/ontology entities can be computed in terms of bindings between WordNet senses, see, for instance [22, 6]. For example, in Figure 1, a sense-based matcher may learn from WordNet (with a prior morphological preprocessing of labels performed) that "Camera" in A1 is a hypernym for "Digital Camera" in A2, and, therefore conclude that entity $Digital\_Cameras$ in A2 should

be subsumed by the entity $Photo\_and\_Cameras$ in A1. Another type of matchers exploiting lexicons is based on their structural properties, e.g., WordNet hierarchies. In particular, hierarchy-based matchers measure the distance, for example, by counting the number of arcs traversed, between two concepts in a given hierarchy, see [23]. Several other distance measures for lexicons have been proposed in the literature, e.g., [43, 41].

- *Dictionaries and directories*. A dictionary usually stores some specific domain knowledge (e.g., proper names) as entries with synonym, hypernym and other relations. For example, in Figure 1 entities *NKN* in A1 and *Nikon* in A2 are treated by a matcher as synonyms from the dictionary look up: syn key - "NKN:Nikon = syn", see, for instance [29].

**Alignment reuse** techniques represent an alternative way of exploiting external resources, which contain in this case alignments of previously matched schemas/ontologies. For instance, when we need to match schema/ontology $o'$ and $o''$, given the alignments between $o$ and $o'$, and between $o$ and $o''$ from the external resource, storing previous match operations results. The alignment reuse is motivated by the intuition that many schemas/ontologies to be matched are similar to already matched schemas/ontologies, especially if we they are describing the same application domain. The approach was first introduced in [42], and later was implemented as two matchers, i.e., (i) reuse alignments of entire schemas, or (ii) their fragments, in [25]. To our knowledge, for the moment, this is the unique matching system supporting the alignment reuse.

### 4.2 Structure-level techniques

**Graph-based techniques** are graph algorithms which consider the input as labeled graphs. The applications (e.g., database schemas, taxonomies, or ontologies) are viewed as graph-like structures containing terms and their inter-relationships. Usually, the similarity comparison between a pair of nodes from the two schemas/ontologies is based on the analysis of their positions within the graphs. The intuition behind is that, if two nodes from two schemas/ontologies are similar, their neighbors might also be somehow similar. Below, we present some particular matchers representing this intuition.

- *Graph matching*. There had been a lot of work on graph (tree) matching in graph theory and also with respect to schema/ontology matching applications, see [44, 54]. Typically, the schema/ontology matching problem is encoded as an optimization problem which is further resolved with the help of a graph matching algorithm. Examples of systems exploiting graph matching algorithms based on fix-point computation are [32] and [19]. Some particular matchers handling DAGs and trees are *children* and *leaves*.
- *Children*. The (structural) similarity between inner nodes of the graphs is computed based on similarity of their children nodes, that is, two non-leaf schema elements are structurally similar if their immediate children sets are highly similar. A more complex version of this matcher is implemented in [25].

- *Leaves*. The (structural) similarity between inner nodes of the graphs is computed based on similarity of leaf nodes, that is, two non-leaf schema elements are structurally similar if their leaf sets are highly similar, even if their immediate children are not, see, for example [29, 25].
- *Relations*. The similarity computation between nodes can be also based on their relations. For example, in one of the possible ontology representations of schemas of Figure 1, if class $Photo\_and\_Cameras$ relates to class $NKN$ by relation $hasBrand$ in one ontology, and if class $Digital\_Cameras$ relates to class $Nikon$ by relation $hasMarque$ in the other ontology, then knowing that classes $Photo\_and\_Cameras$ and $Digital\_Cameras$ are similar, and also relations $hasBrand$ and $hasMarque$ are similar, we can infer that $NKN$ and $Nikon$ may be similar too, see [30].

**Taxonomy-based techniques** are also graph algorithms which consider only the specialization relation. The intuition behind taxonomic techniques is that *is-a* links connect terms that are already similar (being a subset or superset of each other), therefore their neighbors may be also somehow similar. This intuition can be exploited in several different ways:

- *Bounded path matching*. These matchers take two paths with links between classes defined by the hierarchical relations, compare terms and their positions along these paths, and identify similar terms, see, for instance [37]. For example, in Figure 1 given that element $Digital\_Cameras$ in A2 should be subsumed by the element $Photo\_and\_Cameras$ in A1, a matcher would suggest *FJFLM* in A1 and *FujiFilm* in A2 as an appropriate match.
- *Super(sub)-concepts rules*. These matchers are based on rules capturing the above sated intuition. For example, if super-concepts are the same, the actual concepts are similar to each other. If sub-concepts are the same, the compared concepts are also similar, see, for example [11, 16].

**Model-based** algorithms handle the input based on its semantic interpretation (e.g., model-theoretic semantics). Thus, they are well grounded deductive methods. Examples are propositional satisfiability (SAT), modal SAT techniques or description logics (DL) reasoning techniques.

- *Propositional satisfiability (SAT)*. As from [21, 6] the approach translates the matching problem, namely the two graphs (trees) and mapping queries into a propositional formula and then checks it for validity. By a mapping query we mean here the pair of nodes and a possible relation between them. Notice that SAT deciders are correct and complete decision procedures for propositional satisfiability, and therefore they can be used in order to exhaustively check for all possible mappings.
- *Modal SAT*. As proposed in [46] the approach is to delimit propositional SAT which allows handling only unary predicates (e.g., classes, XML elements) by admitting binary predicates (e.g., attributes). The key idea is to enhance propositional logics with modal logic (or *ALC* description logics) operators. Therefore, the matching problem is translated into a modal logic formula which is further checked for its validity using sound and complete satisfiability search procedures.

- *DL-based techniques*. Description logics techniques, i.e., subsumption test, can be used to determine the relations between classes in a purely semantic manner. In fact, first merging two ontologies (after renaming) and then testing each pair of concepts and roles for subsumption is enough for aligning terms with the same interpretation (or with a subset of the interpretations of the others). However, we are not aware of existence of any schema/ontology matching systems supporting DL-based techniques for the moment.

There is a line of examples in the literature when DL-based techniques are used in relevant to schema/ontology matching applications. For example, in spatio-temporal database integration scenario, as first motivated in [40] and later developed in [48] the inter-schema mappings are initially proposed by the integrated schema designer and are encoded together with input schemas in $ALCRP(S_2 \oplus T)$ language. Then, the DL reasoning services are used to check the satisfiability of the two source schemas and the set of inter-schema mappings. If some objects are found unsatisfied, then the inter-schema mappings should be reconsidered.

Another example, is when DL-based techniques are used in query processing scenario [34]. The approach assumes that mappings between pre-existing domain ontologies are already specified in a declarative manner (e.g., manually). User queries are rewritten in terms of pre-existing ontologies and are expressed in Classic [4], and further evaluated against real-world repositories, which are also subscribed to the pre-existing ontologies.

Finally, a very similar problem to schema/ontology matching is addressed within the system developed for matchmaking in electronic marketplaces [10]. Demand $D$ and supply $S$ requests are translated from natural language sentences into Classic [4]. The approach assumes the existence of a pre-defined domain ontology $T$, which is also encoded in Classic. Matchmaking between a supply $S$ and a demand $D$ is performed with respect to the pre-defined domain ontology $T$. Reasoning is performed with the help of the NeoClassic reasoner in order to determine the *exact match* ($T \models (D \sqsubseteq S)$) and ($T \models (S \sqsubseteq D)$), *potential match* (if $D \sqcap S$ is satisfiable in $T$), and *nearly miss* (if $D \sqcap S$ is unsatisfiable in $T$). The system also provides a logically based matching results rank operation.

## 5 On classifying matching systems

As the previous section indicates, elementary matchers rely on a particular kind of input information, therefore they have different applicability and value with respect to different schema/ontology matching tasks. State of the art matching systems are made not of a single elementary matcher, usually they combine them in a variety of ways. As noticed in [42] elementary matchers can be used in sequence (*hybrid* matchers), examples are [29, 1], or in parallel (*composite* matchers) combining the results (e.g., taking the average, maximum) of independently executed matchers, see, for instance [25, 14, 16].

Most of the matching algorithms are hybrid, or "compose" hybrid algorithms. The above classification is useful from an architectural perspective, however it does not show how the systems can be distinguished with respect to how they state the matching

problem and what kind of a solution they produce. Thus, it cannot be useful for the customer of alignment.

Finding a better classification is rather difficult. Below, we provide a vision of a classification of matching systems depending on how they consider alignments and the alignment task. This classification captures only a tendency of each algorithm and we think that it is still not exclusive. As a matter of fact, we find that some algorithms may fall between two categories. The proposed classification criteria for matching systems are as follows:

- *Alignments as solutions.* This category covers purely algorithmic techniques that consider that the alignment problem is to be solved and the alignment is a solution to that problem. It could be characterized as a (continuous or discrete) optimization problem. This covers algorithms like [19], or [32] as well as most of the distance-based systems and instance-based systems which learn from the data.
- *Alignments as theorems.* Systems of this category rely on semantics and require the alignment to satisfy it, for example [22]. This category is, strictly speaking, a sub-category of the previous one (the problem is expressed in semantic terms). However, it is sufficiently autonomous for being singled out here. Moreover, it has a strong link with the use of the aligned ontologies.
- *Alignments as likeness clues.* This category refers to the algorithms which do not pretend to find *the* solution to a problem (usually they just measure the distance) but rather to provide reasonable indications to a user for selecting the actual alignment. They can be based on the same techniques as the others but usually do not provide an exact solution to the problem. Examples of matching systems from this category are [29, 25].

## 6 Review of state of the art matching systems

We now look at some recent schema-based state of the art matching systems in light of the classification presented in Figure 3 and criteria highlighted in Section 5.

**Rondo**. The SF [32] approach as implemented in Rondo [33] utilizes a hybrid matching algorithm based on the ideas of similarity propagation. Schemas are presented as directed labeled graphs; grounding on the OIM specification [8] the algorithm manipulates them in an iterative fix-point computation to produce an alignment between the nodes of the input graphs. The technique starts from string-based comparison (common prefixes, suffixes tests) of the vertice's labels to obtain an initial alignment which is refined within the fix-point computation. The basic concept behind the SF algorithm is the similarity spreading from similar nodes to the adjacent neighbors through propagation coefficients. From iteration to iteration the spreading depth and a similarity measure are increasing till the fix-point is reached. The result of this step is a refined alignment which is further filtered to finalize the matching process. SF considers the alignment as a solution to a clearly stated optimization problem.

**Artemis**. Artemis (Analysis of Requirements: Tool Environment for Multiple Information Systems) [7] was designed as a module of MOMIS mediator system [1] for creating global views. It performs affinity-based analysis and hierarchical clustering of source schemas elements. Affinity-based analysis represents the matching step: in a

hybrid manner it calculates the name, structural and global affinity coefficients exploiting a common thesaurus. The common thesaurus is built with the help of ODB-Tools, WordNet or manual input. It represents a set of intensional and extensional relationships which depict intra- and inter-schema knowledge about classes and attributes of the input schemas. Based on global affinity coefficients, a hierarchical clustering technique categorizes classes into groups at different levels of affinity. For each cluster it creates a set of global attributes - global class. Logical correspondence between the attributes of a global class and source schema's attributes is determined through a mapping table. Artemis falls into the alignments as likeness clues category.

**Cupid**. Cupid [29] implements a hybrid matching algorithm comprising linguistic and structural schema matching techniques, and computes similarity coefficients with the assistance of a precompiled dictionary. Input schemas are encoded as graphs. Nodes represent schema elements and are traversed in a combined bottom-up and top-down manner. The matching algorithm consists of three phases and operates only with tree-structures to which non-tree cases are reduced. The first phase (linguistic matching) computes linguistic similarity coefficients between schema element names (labels) based on morphological normalization, categorization, string-based techniques (common prefixes, suffixes tests) and a dictionary look-up. The second phase (structural matching) computes structural similarity coefficients weighted by leaves which measure the similarity between contexts in which elementary schema elements occur. The third phase (mapping generation) computes weighted similarity coefficients and generates final alignment by choosing pairs of schema elements with weighted similarity coefficients which are higher than a threshold. Referring to [29], Cupid performs somewhat better overall, then the other hybrid matchers: Dike [38] and Artemis [7]. Cupid falls into the alignments as likeness clues category.

**COMA**. COMA (COmbination of MAtching algorithms) [25] is a composite schema matching tool. It provides an extensible library of matching algorithms; a framework for combining obtained results, and a platform for the evaluation of the effectiveness of the different matchers. Matching library is extensible, and as from [25] it contains 6 elementary matchers, 5 hybrid matchers, and one reuse-oriented matcher. Most of them implement string-based techniques (affix, n-gram, edit distance, etc.) as a background idea; others share techniques with Cupid (dictionary look-up, etc.); and reuse-oriented is a completely novel matcher, which tries to reuse previously obtained results for entire new schemas or for its fragments. Schemas are internally encoded as DAGs, where elements are the paths. This fact aims at capturing contexts in which the elements occur. Distinct features of the COMA tool in respect to Cupid, are a more flexible architecture and a possibility of performing iterations in the matching process. Based on the comparative evaluations conducted in [12], COMA dominates Autoplex[2] and Automatch [3]; LSD [13] and GLUE [14]; SF [32], and SemInt [28] matching tools. COMA falls into the alignments as likeness clues category.

**NOM**. NOM (Naive Ontology Mapping) [16] adopts the idea of composite matching from COMA [25]. Some other innovations with respect to COMA, are in the set of elementary matchers based on rules, exploiting explicitly codified knowledge in ontologies, such as information about super- and sub-concepts, super- and sub-properties, etc. At present the system supports 17 rules. For example, rule#5 (R5) states that if

super-concepts are the same, the actual concepts are similar to each other, R15 states that two entities are the same if they are binded by *sameClassAs* OWL property. NOM also exploits a set of instance-based techniques, this topic is beyond scope of the paper. The system falls into the alignments as likeness clues category.

**QOM**. QOM (Quick Ontology Mapping) [15] is a successor of the NOM system [16]. The approach is based on the idea that the loss of quality in matching algorithms is marginal (to a standard baseline), however improvement in efficiency can be tremendous. This fact allows QOM producing mappings fast, even for large-size ontologies. QOM is grounded on matching rules of NOM. However, for the purpose of efficiency the use of some rules have been restricted, e.g., R5. QOM avoids the complete pairwise comparison of trees in favor of a (n incomplete) top-down strategy. Experimental study has shown that QOM is on a par with other state of the art algorithms concerning the quality of proposed alignment, while outperforming them with respect to efficiency, and vice versa QOM shows better quality results than approaches within the same complexity class. The system falls into the alignments as likeness clues category.

**OLA**. OLA (OWL Lite Aligner) [19] is designed with the idea of balancing the contribution of each component that compose an ontology (classes, properties, names, constraints, taxonomy, and even instances). As such it takes advantage of all the elementary matching techniques that have been considered in the previous sections, but the semantic ones. OLA is a family of distance based algorithms which converts definitions of distances based on all the input structures into a set of equations. These distances are almost linearly aggregated (they are linearly aggregated modulo local matches of entities). The algorithm then looks for the matching between the ontologies that minimizes the overall distance between them. For that purpose it starts with base distance measures computed from labels and concrete datatypes. Then, it iterates a fix-point algorithm until no improvement is produced. From that solution, an alignment is generated which satisfies some additional criterion (on the alignment obtained and the distance between aligned entities). As a system OLA consider the alignment as a solution to a clearly stated optimization problem.

**Anchor-PROMPT**. Anchor-PROMPT [37] (an extension of PROMPT, also formerly known as SMART) is an ontology merging and alignment tool with a sophisticated prompt mechanism for possible matching terms. The anchor-PROMPT is a hybrid alignment algorithm which takes as input two ontologies, (internally represented as graphs) and a set of anchors-pairs of related terms, which are identified with the help of string-based techniques (edit-distance test), or defined by a user, or another matcher computing linguistic similarity, for example [31]. Then the algorithm refines them by analyzing the paths of the input ontologies limited by the anchors in order to determine terms frequently appearing in similar positions on similar paths. Finally, based on the frequencies and a user feedback, the algorithm determines matching candidates. Anchor-PROMPT falls into the alignments as solutions and alignments as likeness clues categories.

**S-Match**. S-Match [21, 22] is a schema-based schema/ontology matching system implementing semantic matching approach. It takes two graph-like structures (e.g., database schemas or ontologies) as input and returns as output the relations between the nodes of the graphs that correspond semantically to each other. Possible relations

**Fig. 4.** Characteristics of state of the art matching approaches

| | | SF [32,33] | Artemis [7] | Cupid [29] | COMA [25] | NOM [16] | Anchor-Prompt [37] | OLA [19] | S-Match [21,22,23] |
|---|---|---|---|---|---|---|---|---|---|
| Element-level | Syntactic | string-based (2); data types; key properties | domain compatibility; language-based (1) | string-based (2); language-based (2); data types; key properties | string-based (4); language-based (1); data types | string-based (1); domains and ranges | string-based (1); domains and ranges | string-based (3); data types; language-based (1) | string-based (5); language-based (2) |
| | External | - | common thesaurus (CT): synonyms, broader terms, related terms | auxiliary dictionary (synonyms, hypernyms, hyponyms, abbreviations ) | auxiliary dictionary (synonyms, hypernyms, hyponyms, abbreviations ) | application-specific vocabulary | - | WordNet(1) | WordNet: sense-based (2), gloss-based (6) |
| Structure-level | Syntactic | iterative fix-point computation | matching of neighbors via CT | tree matching weighted by leaves | DAG (tree) matching with a bias towards leaf or children structures (2); paths | matching of neighbors (2); taxonomic structure (4) | bounded paths matching (arbitrary links); bounded paths matching (processing *is-a* links separately) | Iterative fix-point computation; matching of neighbors; taxonomic structure | - |
| | Semantic | - | - | - | - | - | - | - | propositional SAT (2) |

are: equivalence (=), more general ($\sqsupseteq$), less general ($\sqsubseteq$), mismatch ($\perp$), and overlapping ($\sqcap$). The current version of S-Match is a rationalized re-implementation of the CTXmatch system [6] with a few added functionalities. S-Match was designed and developed as a platform for semantic matching, namely a highly modular system with the core of computing relations where single components can be plugged, unplugged or suitably customized. It is a hybrid system. At present, S-Match libraries contain 13 element-level matchers, see [23], and 2 structure-level (JSAT and SAT4J) matchers. S-Match falls into the alignments as theorems category.

Figure 4 briefly summarizes how the matching systems cover the solution space in terms of the proposed classification. Numbers in brackets specify how many matchers of a particular type a system supports. For example, S-Match supports 5 string-based element-level syntactic matchers (prefix, suffix, edit distance, n-gram, and text corpus, see [23]), OLA has one element-level external matcher based on WordNet. Figure 4 also testifies that schema/ontology matching research was mainly focused on syntactic and external techniques so far. Semantic techniques have been exploited only by the S-Match system [22].

## 7 Conclusions

This paper presents a new classification of schema-based matching approaches, which improves the previous work on classifying schema matching approaches. We have introduced new criteria which are based on (i) general properties of matching techniques, (ii) interpretation of input information, and (iii) the kind of input information. In particular, we distinguish between heuristic and exact techniques at schema-level; and syntactic, external, and semantic techniques at element- and structure-level. We reviewed some of the recent schema/ontology matching systems in light of the classification proposed pointing which part of the solution space they cover. Analysis of state of the art systems discussed has shown, that most of them exploit only syntactic and external techniques, and only one uses semantic techniques. However, the category of semantic techniques

was identified only recently as a part of the solution space; its methods provide sound and complete results, and, hence it represents a wide area for the future investigations.

The proposed classification provides a common conceptual basis, and hence can be used for comparing (analytically) different existing schema/ontology matching systems as well as for designing a new one, taking advantages of state of the art solutions. As the paper shows, the solution space is quite large and there exists a variety of matching techniques. In some cases it is difficult to draw conclusions from the classification of systems. A complementary approach is to compare matching systems experimentally, with the help of benchmarks. Initial steps have already been done by running I3CON initiative [3] and Ontology Alignment Contest[4].

**Acknowledgements:** This work has been partly supported by the European Knowledge Web network of excellence (IST-2004-507482).

## References

1. S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. In *SIGMOD Record*, 28(1), pages 54–59, 1999.
2. J. Berlin and A. Motro. Autoplex: Automated discovery of content for virtual databases. In *Proceedings of CoopIS*, pages 108–122, 2001.
3. J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proceedings of CAiSE*, pages 452–466, 2002.
4. A. Borgida, R. Brachman, D. McGuinness, and L. Resnick. Classic: A structural data model for objects. In *Proceedings of ACM SIGMOD*, pages 58–67, 1989.
5. P. Bouquet, J. Euzenat, E. Franconi, L. Serafini, G. Stamou, and S. Tessaris. D2.2.1: Specification of a common framework for characterizing alignment. Technical report, NoE Knowledge Web project delivable, 2004. http://www.inrialpes.fr/exmo/cooperation/kweb/heterogeneity/deli/kweb-221.pdf.
6. P. Bouquet, L. Serafini, and S. Zanobini. Semantic coordination: A new approach and an application. In *Proceedings of ISWC*, pages 130–145, 2003.
7. S. Castano, V. De Antonellis, and S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. In *IEEE Transactions on Knowledge and Data Engineering*, number 13(2), pages 277–297, 2001.
8. Meta Data Coalition. Open information model, version 1.0. http://mdcinfo/oim/oim10.html, August 1999.
9. W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string metrics for matching names and records. In *Proceedings of workshop on Data Cleaning and Object Consolidation at KDD*, 2003.
10. T. Di Noia, E. Di Sciascio, F. M. Donini, and M. Mongiello. A system for principled matchmaking in an electronic marketplace. In *Proceedings of WWW*, pages 321–330, 2003.
11. R. Dieng and S. Hug. Comparison of "personal ontologies" represented through conceptual graphs. In *Proceedings of ECAI*, pages 341–345, 1998.
12. H.H. Do, S. Melnik, and E.Rahm. Comparison of schema matching evaluations. In *Proceedings of workshop on Web and Databases*, 2002.

[3] http://www.atl.external.lmco.com/projects/ontology/i3con.html#About
[4] http://www.co4.inrialpes.fr/align/Contest

13. A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of SIGMOD*, pages 509–520, 2001.
14. A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to map ontologies on the semantic web. In *Very Large Databases Journal, Special Issue on the Semantic Web*, 2003. (to appear).
15. M. Ehrig and S. Staab. QOM: Quick ontology mapping. In *Proceedings of ISWC*, 2004.
16. M. Ehrig and Y. Sure. Ontology mapping - an integrated approach. In *Proceedings of ESWS*, pages 76–91, 2004.
17. J. Euzenat. An API for ontology alignment. In *Proceedings of ISWC*, 2004.
18. J. Euzenat, J. Barrasa, P. Bouquet, R. Dieng, M. Ehrig, M. Hauswirth, M. Jarrar, R. Lara, D. Maynard, A. Napoli, G. Stamou, H. Stuckenschmidt, P. Shvaiko, S. Tessaris, S. van Acker, I. Zaihrayeu, and T. L. Bach. D2.2.3: State of the art on ontology alignment. Technical report, NoE Knowledge Web project delivable, 2004.
19. J. Euzenat and P.Valtchev. Similarity-based ontology alignment in OWL-lite. In *Proceedings of ECAI*, 2004.
20. J. Euzenat and P. Valtchev. An integrative proximity measure for ontology alignment. In *Proceedings of Semantic Integration workshop at ISWC*, 2003.
21. F. Giunchiglia and P. Shvaiko. Semantic matching. In *The Knowledge Engineering Review Journal*, number 18(3), pages 265–280, 2003.
22. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-Match: an algorithm and an implementation of semantic matching. In *Proceedings of ESWS*, pages 61–75, 2004.
23. F. Giunchiglia and M. Yatskevich. Element level semantic matching. In *Proceedings of Meaning Coordination and Negotiation workshop at ISWC*, 2004.
24. F. Giunchiglia and I. Zaihrayeu. Making peer databases interact - a vision for an architecture supporting data coordination. In *Proceedings of international workshop on Cooperative Information Agents*, pages 18–35, 2002.
25. H.H.Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proceedings of VLDB*, pages 610–621, 2001.
26. Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. In *The Knowledge Engineering Review Journal*, number 18(1), pages 1–31, 2003.
27. J. Kang and J.F. Naughton. On schema matching with opaque column names and data values. In *Proceedings of SIGMOD*, pages 205–216, 2003.
28. W.S. Li and C. Clifton. Semantic integration in heterogeneous databases using neural networks. In *Proceedings of VLDB*, pages 1–12, 1994.
29. J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proceedings of VLDB*, pages 49–58, 2001.
30. A. Maedche and S. Staab. Measuring similarity between ontologies. In *Proceedings of EKAW*, 2002.
31. D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *Proceedings of KR*, pages 483–493, 2000.
32. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm. In *Proceedings of ICDE*, pages 117–128, 2002.
33. S. Melnik, E. Rahm, and P. Bernstein. Rondo: A programming platform for generic model management. In *Proceedings of SIGMOD*, pages 193–204, 2003.
34. E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. Observer: An approach for query processing in global information systems based on interoperability between pre-existing ontologies. In *Proceedings of CoopIS*, pages 14–25, 1996.
35. A.G. Miller. Wordnet: A lexical database for english. In *Communications of the ACM*, number 38(11), pages 39–41, 1995.
36. N. Noy and M. Klein. Ontology evolution: Not the same as schema evolution. In *Knowledge and Information Systems*, in press, 2002.

37. N. Noy and M. A. Musen. Anchor-prompt: Using non-local context for semantic matching. In *Proceedings of IJCAI workshop on Ontologies and Information Sharing*, pages 63–70, 2001.
38. L. Palopoli, G. Terracina, and D. Ursino. The system dike: Towards the semi-automatic synthesis of cooperative information systems and data warehouses. In *ADBIS-DASFAA, Matfyzpress*, pages 108–117, 2000.
39. M. Paolucci, T. Kawmura, T. Payne, and K. Sycara. Semantic matching of web services capabilities. In *ISWC*, 2002.
40. C. Parent and S. Spaccapietra. Database integration: the key to data interoperability. In M. P. Papazoglou, S. Spaccapietra, and Z. Tari, editors, *Advances in Object-Oriented Data Modeling*. The MIT Press, 2000.
41. R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. In *IEEE Transactions on Systems, Man and Cybernetics*, number 19(1), pages 17–30, 1989.
42. E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. In *Very Large Databases Journal*, number 10(4), pages 334–350, 2001.
43. P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of IJCAI*, pages 448–453, 1995.
44. D. Shasha, J. T. L. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *Proceedings of PODS*, pages 39–52, 2002.
45. P. Shvaiko. A classification of schema-based matching approaches. In *Proceedings of Meaning Coordination and Negotiation workshop at ISWC*, 2004.
46. P. Shvaiko. Iterative schema-based semantic matching. Technical Report DIT-04-020, University of Trento, 2004.
47. M.K. Smith, C. Welty, and D.L. McGuinness. OWL web ontology language guide. Technical report, World Wide Web Consortium (W3C), http://www.w3.org/TR/2004/REC-owl-guide-20040210/, February 10 2004.
48. A. Sotnykova, M-A. Aufaure, N. Bennacer, N. Cullot, and C. Vangenot. Semantic mappings in description logics for database schema integration. Technical report, Swiss Federal Institute of Technology in Lausanne, 2004.
49. P. Valtchev. *Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets*. Thèse d'informatique, Université Grenoble 1, 1999.
50. P. Valtchev and J. Euzenat. Dissimilarity measure for collections of objects and values. *Lecture notes in computer science*, 1280:259–272, 1997.
51. R. van Eijk, F. de Boer, W. van de Hoek, and J. J. Meyer. On dynamically generated ontology translators in agent communication. *International journal of intelligent system*, 16:587–607, December 2001.
52. H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Huebner. Ontology-based integration of information - a survey of existing approaches. In *Proceedings of IJCAI workshop on Ontologies and Information Sharing*, pages 108–117, 2001.
53. L. Xu and D.W. Embley. Using domain ontologies to discover direct and indirect matches for schema elements. In *Proceedings of Semantic Integration workshop at ISWC*, 2003.
54. K. Zhang and D. Shasha. Approximate tree pattern matching. In A. Apostolico and Z. Galil, editors, *Pattern matching in strings, trees, and arrays*, pages 341–371. Oxford University, 1997.