UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

A CLASSIFICATION OF SCHEMA-BASED MATCHING APPROACHES

Pavel Shvaiko

August 2004

Technical Report # DIT-04-093

Also: in Proceedings of the Meaning Coordination and Negotiation workshop at ISWC'04

A Classification of Schema-Based Matching Approaches

Pavel Shvaiko

University of Trento, Povo, Trento, Italy
pavel@dit.unitn.it

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 taxonomy of schema-based matching techniques that builds on the previous work on classifying schema matching approaches. Some innovations are in introducing new criteria which distinguish between matching techniques relying on diverse semantic clues. In particular, we distinguish between heuristic and formal techniques at schema-level; and implicit and explicit 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.

1 Introduction

Match is a critical operator in many well-known application domains, such as, semantic web, schema/ontology integration, data warehouses, e-commerce, XML message mapping, catalog matching, etc. Many solutions to the matching problem include identifying terms in one information source that "match" terms in another information source. The applications can be viewed as graph-like structures containing terms and their inter-relationships. These might be database schemas, taxonomies, or ontologies, for example [14], etc. Match operator takes two graph-like structures as input and produces a mapping between the nodes of the graphs that correspond semantically to each other as output.

Many diverse solutions to the matching problem have been proposed so far, for example [19, 15, 8, 21, 32, 1, 17, 23, 26, 20], etc. 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 between schema and ontology matching (alignment) 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 referring as to the one problem.

With the emergence and proliferation of the semantic web, the semantics captured in schemas/ontologies should be also handled at different levels of details. Therefore, there is a need in distinguishing between schema/ontology matching techniques relying on diverse semantic clues. In this paper we present

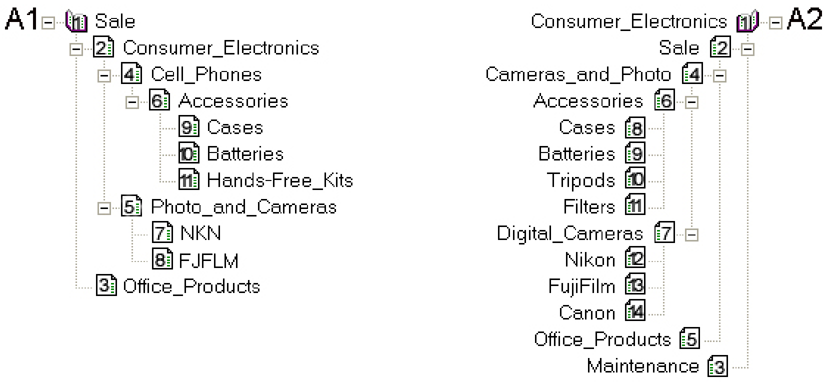


Fig. 1. Two XML schemas

a taxonomy of schema-based matching techniques that builds on the previous work of E. Rahm and P. Bernstein on classifying schema matching approaches [28]. Some innovations are in introducing new criteria which distinguish between schema/ontology matching techniques relying on diverse semantic clues. In particular, we distinguish between heuristic and formal techniques at schema-level; and implicit and explicit techniques at element- and structure-level.

The rest of the paper is organized as follows. Section 2 provides, via an example, the basic motivations to the schema/ontology matching problem. Section 3 introduces the classification of schema-based approaches and discusses in details possible alternatives. Section 4 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 5 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 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 nodes (sometimes in the following we refer to nodes as 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 nodes with labels $Office\_Products$ in A1 and in A2 are the candidates to be merged, while the node with label *Digital_Cameras* in A2 should be subsumed by the node with label *Photo_and_Cameras* in A1.

2.2 Matching: Syntactic vs. Semantic

In this paper we discuss the problem of matching schemas and ontologies from the generic perspective i.e., we analyze information which is exploited by matching systems in order to produce mappings. In this respect, ontology matching differs substantially from schema matching in the following two (among the others, see [25]) 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 [30] allows defining inverse properties, transitive properties; disjoint classes, new classes as unions or intersections of other classes, etc.

However, ontologies can be viewed as schemas for knowledge bases. Having defined classes and slots in the ontology, we populate the knowledge base with instance data [25]. 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 heuristic techniques trying to guess semantics 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.

Apart from the information that matching systems exploit, the other important dimension of schema/ontology matching is a form of the result they produce. Based on these criteria, following the proposal first introduced in [11], schema/ontology matching systems can be viewed as *syntactic* and *semantic* matching systems. Syntactic matching approaches do not analyze term meaning, and thus semantics, directly. In these approaches semantic correspondences are determined using (i) syntactic similarity measures, usually in [0,1] range, for example, with the help of similarity coefficients [19, 10] or confidence measures [32]; and (ii) syntax driven techniques, for instance techniques, which consider labels as strings, etc., see [21, 19, 15]. The first key distinction of the semantic matching approaches is that mappings are calculated between schema/ontology elements by computing *semantic relations* (for example, equivalent ($=$) or subsuming elements ($\sqsubseteq, \sqsupseteq$), etc., see for details [12]). The second key distinction is that semantic relations are determined by analyzing *meaning* (concepts, not labels as in syntactic matching) which is codified in the elements and the structure of schemas/ontologies. These ideas are schematically represented in Figure 2.

Let us define the matching problem in terms of graphs [11]. A *mapping element* is a 4-tuple $< ID_{ij},\ n1_i,\ n2_j,\ R >$, i=1,...,N1; j=1,...,N2; where ID_{ij}

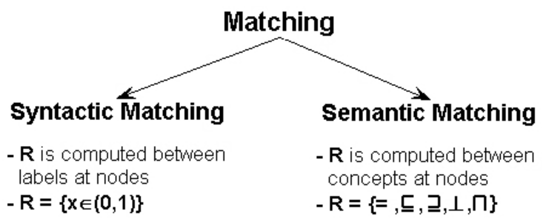

Matching
Syntactic Matching
- R is computed between labels at nodes
- R = {x∈(0,1)}
Semantic Matching
- R is computed between concepts at nodes
- R = {=, ⊑, ⊒, ⊥, ⊓}

Fig. 2. Matching: Syntactic vs. Semantic

is a unique identifier of the given mapping element; $n1_i$ is the i-th node of the first graph, N1 is the number of nodes in the first graph; $n2_j$ is the j-th node of the second graph, N2 is the number of nodes in the second graph; and R specifies a *similarity relation* (a coefficient in [0,1] range or a semantic relation) holding between the nodes $n1_i$ and $n2_j$. For instance, based on linguistic and structure analysis, the similarity coefficient between nodes with labels *Photo_and_Cameras* in A1 and *Cameras_and_Photo* in A2 in Figure 1 could be 0.67. Thus, the corresponding mapping element is $< ID_{54}, n1_5, n2_4, 0.67 >$. The *matching* operation determines a set of mapping elements.

3 Classification of schema-based matching approaches

At present, there exists a line of semi-automated schema/ontology matching systems, see for instance [19, 15, 8, 21, 32, 1, 17, 23, 26, 20], etc. Good surveys are provided in [28, 31, 16]. The classification of [28] distinguishes between *individual* implementations of match and *combinations* of matchers. Individual 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, etc.) can be taken into account. Individual matchers can be used in different ways: directly (*hybrid* matchers), see [19, 1] or combining the results of independently executed matchers (*composite* matchers), see for instance [15, 8, 9].

We focus only on schema-based approaches, and therefore consider only schema/ontology information, not instance data [1]. There are two levels of granularity while performing schema-based matching: element-level and structure-level. Element-level matching techniques compute mapping elements by analyzing individual labels/concepts at nodes; structure-level techniques compute mapping elements by analyzing also subgraphs.

With the emergence and proliferation of the semantic web, the semantics captured in schemas/ontologies should be also handled at different levels of details. Therefore, there is a need in distinguishing between schema/ontology matching techniques relying on diverse semantic clues. We introduce for individual matchers the following classification cretiria:

[1] Prominent solutions of instance-based schema/ontology matching as well as possible extensions of the instance-based part of the classification of [28] can be found in [8] and [17] correspondingly.

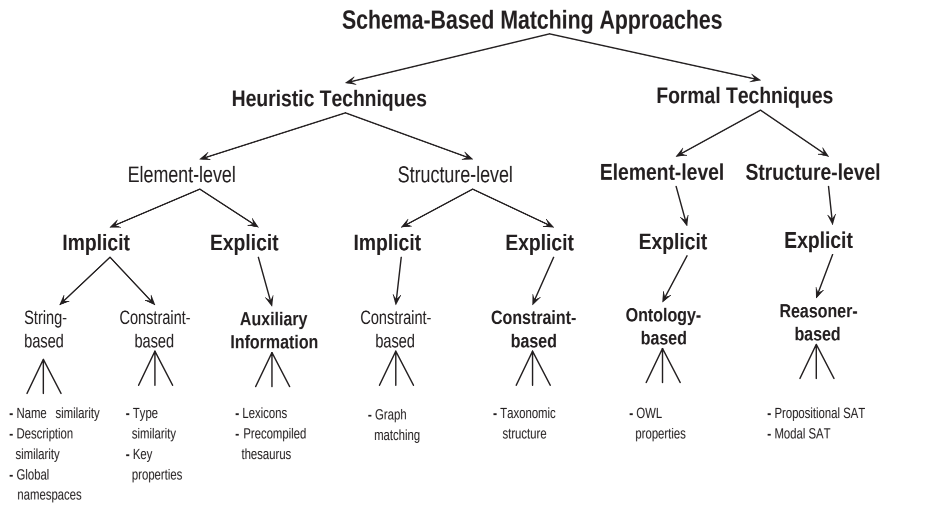


Fig. 3. A revised classification of schema-based matching approaches

- *Heuristic vs formal.* Matching techniques can have either heuristic or formal ground. The key characteristic of the heuristic techniques is that they try to guess relations which may hold between similar labels or graph structures. The key characteristic of the formal techniques is that they have model-theoretic semantics which is used to justify their results.
- *Implicit vs explicit.* These matching techniques rely either on implicitly or explicitly codified semantic information. Implicit techniques are syntax driven techniques: examples are techniques, which consider labels as strings, or analyze data types, or soundex of schema/ontology elements. Explicit techniques exploit the semantics of labels. These techniques are based on the use of tools, which explicitly codify semantic information, e.g., thesauruses, ontologies, etc.

To make the distinctions between the categories proposed more clear, we revised a schema-based part of the classification of matching techniques by E. Rahm and P. Bernstein [28], see Figure 3. All the innovations are marked in bold type. Let us discuss the main alternatives (also indicating in which matching systems they were exploited) according to the above classification criteria in more detail. We omit in our further discussions heuristic element-level implicit techniques as well as heuristic structure-level implicit constrained-based techniques because they appear in a revised classification without changes in respect to the original publication. We also renamed linguistic techniques into string-based techniques, to discard from this category thesaurus look-up methods (they appear in the other category) and methods that perform morphological analysis of strings, which we view only as a preprocessing part, for example, for matching techniques based on lexicons.

3.1 Heuristic techniques

Element-level explicit techniques

- *Precompiled thesaurus.* A precompiled thesaurus usually stores domain knowledge as entries with synonym, hypernym and other relations. For example, in Figure 1 elements *NKN* in A1 and *Nikon* in A2 are treated by a matcher as synonyms from the thesaurus look up: syn key - "NKN:Nikon = syn", see, for instance [19].
- *Lexicons.* The approach is to use lexicons to obtain meaning of terms used in schemas/ontologies. For example, WordNet [24] 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 elements can be computed in terms of bindings between WordNet senses, see, for instance [12, 4]. For example, in Figure 1 a 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 element $Digital\_Cameras$ in A2 should be subsumed by the element $Photo\_and\_Cameras$ in A1.

Structure-level explicit techniques

- *Taxonomic structure.* These matchers analyze and compare positions of terms (labels) within taxonomies. For example, they 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 [26, 9]. The intuition behind taxonomic structure methods 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. 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.

3.2 Formal techniques

Element-level explicit techniques

- *OWL properties.* OWL [30] is ontology web language with clear, model-theoretic semantics, and hence methods exploiting its constructors are formal element-level methods. For instance, *sameClassAs* constructor explicitly states that one class is equivalent to the other, see for a particular implementation [9]. For example, in Figure 1 one of the possible OWL encodings could specify semantics of the element $Digital\_Cameras$ in A2 as follows: $Digital\_Cameras = Camera \sqcap DigitalPhoto\_Producer$. An intuitive reading of the above statement is that digital camera means the same thing as a camera, which encodes and stores images digitally. Then, a matcher would determine that the node 7 in A2 has to be subsumed by the node 5 in A1. Possible extensions to the given category would also exploit other OWL constructors: class properties (e.g., enumeration, disjointness), object properties (inverse-of, symmetric, transitive), etc.

Structure-level explicit techniques

- *Propositional satisfiability (SAT).* As from [11, 4] the approach is to translate the matching problem, namely the two graphs (trees) and mapping queries into a propositional formula and then to check it for its validity. By a mapping query we mean here the pair of nodes and a possible semantic relation between them. Notice that SAT deciders are correct and complete decision procedures for propositional satisfiability, and therefore they can be used for an exhaustive check for all possible mapping elements.
- *Modal SAT.* As from [29] the approach is to delimit propositional SAT which allows handling only unary predicates (e.g., classes) 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.

4 Prototype Matchers

We now look at some recent schema-based state of the art matching systems in light of the classification presented in Figure 3. We also indicate how systems combine individual matchers in their implementations, e.g., in a hybrid or composite manner.

Similarity Flooding (SF). The SF [21] approach as implemented in Rondo [22] utilizes a hybrid matching algorithm based on the ideas of similarity propagation. Schemas are presented as directed labeled graphs; the algorithm manipulates them in an iterative fix-point computation to produce mapping 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 mapping 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 mapping which is further filtered to finalize the matching process.

Artemis. Artemis (Analysis of Requirements: Tool Environment for Multiple Information Systems) [5] 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.

Cupid. Cupid [19] implements a hybrid matching algorithm comprising linguistic and structural schema matching techniques, and computes similarity coefficients with the assistance of a precompiled thesaurus. Input schemas are encoded as graphs. Nodes represent schema elements and are traversed in a combined bottom-up and top-down manner. Matching algorithm consists of three phases and operates only with tree-structures to which no-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 thesaurus look-up. The second phase (structural matching) computes structural similarity coefficients weighted by leaves which measure the similarity between contexts in which individual schema elements occur. The third phase (mapping generation) computes weighted similarity coefficients and generates final mappings by choosing pairs of schema elements with weighted similarity coefficients which are higher than a threshold. Referring to [19], Cupid performs somewhat better overall, then the other hybrid matchers: Dike [27] and Artemis [5].

COMA. COMA (COmbination of MAtching algorithms) [15] 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 [15] it contains 6 individual matchers, 5 hybrid matches, 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 (thesaurus 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 [6], COMA dominates Autoplex[2] and Automatch [3]; LSD [7] and GLUE [8]; SF [21], and SemInt [18] matching tools.

NOM. NOM (Naive Ontology Mapping) [9] adopts the idea of composite matching from COMA [15]. 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.

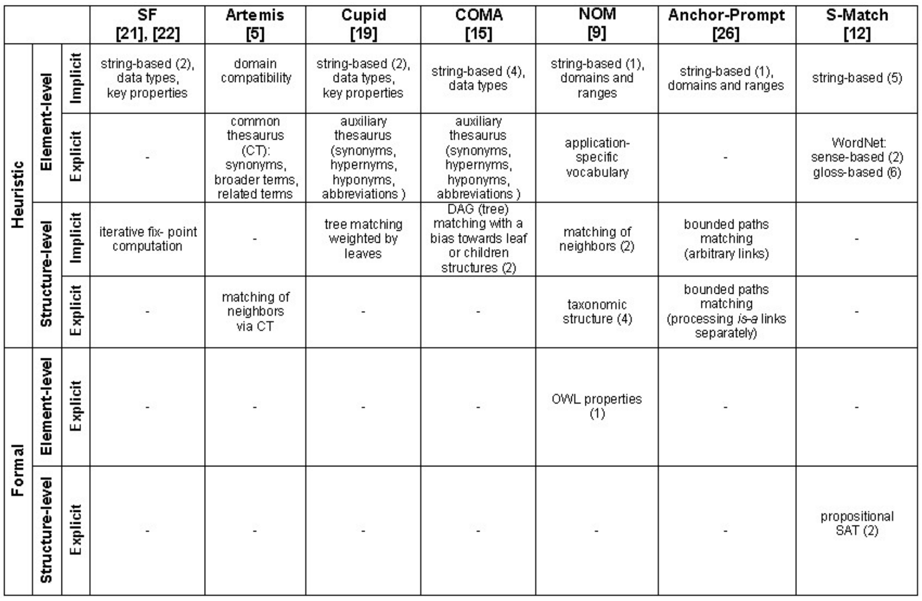
Anchor-PROMPT. Anchor-PROMPT [26] (an extension of PROMPT, also formerly known as SMART) is an ontology merging and alignment tool with a sophisticated prompt mechanism for possible matching terms. The anchor-PROMPT is a hybrid alignment algorithm which takes as input two ontologies, (internally represented 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 [20]. 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.

S-Match. S-Match [11, 12] 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 semantic relations between the nodes of the graphs that correspond semantically to each other as output. Possible semantic relations 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 [4] 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 semantic relations where single components can be plugged, unplugged or suitably customized. It is a hybrid system performing composition of element level techniques. At present, S-Match libraries contain 13 element-level matchers, see [13], and 2 structure-level (JSAT and SAT4J) matchers.

Notice that from the discussed systems, only S-Match returns as output semantic relations, while all the other systems return coefficients rating match quality in [0,1] range. Although, almost all the matching systems analyze term meaning, for example, with the help of a thesaurus, however when they produce a mapping, they encode it in [0,1] range, therefore loosing some information. For example, from a similarity coefficient with value of 0.7 we can not say if the elements it binds are more or less general. We only may conclude that they are similar, and the probability of their equality is 70%. Therefore, only the S-Match system can be considered as a semantic matching system, all other systems, in this sense, are syntactic systems.

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 heuristic element-level implicit matchers (prefix, suffix, edit distance, n-gram, and text corpus, see [13]). Figure 4 also testifies that schema/ontology matching research was mainly focused on heuristic techniques

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



			SF [21], [22]	Artemis [5]	Cupid [19]	COMA [15]	NOM [9]	Anchor-Prompt [26]	S-Match [12]
Heuristic	Element-level	Implicit	string-based (2), data types, key properties	domain compatibility	string-based (2), data types, key properties	string-based (4), data types	string-based (1), domains and ranges	string-based (1), domains and ranges	string-based (5)
		Explicit	-	common thesaurus (CT): synonyms, broader terms, related terms	auxiliary thesaurus (synonyms, hypernyms, hyponyms, abbreviations)	auxiliary thesaurus (synonyms, hypernyms, hyponyms, abbreviations)	application-specific vocabulary	-	WordNet: sense-based (2) gloss-based (6)
	Structure-level	Implicit	iterative fix- point computation	-	tree matching weighted by leaves	DAG (tree) matching with a bias towards leaf or children structures (2)	matching of neighbors (2)	bounded paths matching (arbitrary links)	-
		Explicit	-	matching of neighbors via CT	-	-	taxonomic structure (4)	bounded paths matching (processing *is-a* links separately)	-
Formal	Element-level	Explicit	-	-	-	-	OWL properties (1)	-	-
	Structure-level	Explicit	-	-	-	-	-	-	propositional SAT (2)

so far. Formal element-level and structure-level techniques have been exploited only by two systems: NOM [9] and S-Match [12] correspondingly.

5 Conclusions

This paper presents the taxonomy of schema-based matching approaches, which builds on the previous work by E. Rahm and P. Bernstein on classifying schema matching approaches. We have introduced new criteria which distinguish between schema/ontology matching techniques relying on diverse semantic clues. In particular, we distinguish between heuristic and formal techniques at schema-level; and implicit and explicit techniques at element- and structure-level. We reviewed some of the recent schema/ontology matchers 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 heuristic techniques, and only a few utilize formal techniques. However, the category of formal 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.

Acknowledgements: This work has been partly supported by the European Knowledge Web network of excellence (IST-2004-507482). Thanks to Fausto Giunchiglia, Paolo Bouquet, and Jérôme Euzenat for their insightful comments and suggestions.

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. P. Bouquet, L. Serafini, and S. Zanobini. Semantic coordination: A new approach and an application. In *Proceedings of ISWC*, pages 130–145, 2003.
5. 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.
6. H.H. Do, S. Melnik, and E.Rahm. Comparison of schema matching evaluations. In *Proceedings of workshop on Web and Databases*, 2002.
7. A. Doan, P. Domingos, and A. Halvey. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of SIGMOD*, pages 509–520, 2001.
8. A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halvey. Learning to map ontologies on the semantic web. In *Very Large Databases Journal, Special Issue on the Semantic Web*, 2003. (to appear).
9. M. Ehrig and Y. Sure. Ontology mapping - an integrated approach. In *Proceedings of ESWS*, pages 76–91, 2004.
10. J. Euzenat and P. Valtchev. An integrative proximity measure for ontology alignment. In *Proceedings of Semantic Integration workshop at ISWC*, 2003.
11. F. Giunchiglia and P. Shvaiko. Semantic matching. In *The Knowledge Engineering Review Journal*, number 18(3), pages 265–280, 2003.
12. 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.
13. F. Giunchiglia and M. Yatskevich. Element level semantic matching. In *To appear in Proceedings of Meaning Coordination and Negotiation workshop at ISWC*, 2004.
14. 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.
15. H.H.Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *Proceedings of VLDB*, pages 610–621, 2001.
16. 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.
17. J. Kang and J.F. Naughton. On schema matching with opaque column names and data values. In *Proceedings of SIGMOD*, pages 205–216, 2003.
18. W.S. Li and C. Clifton. Semantic integration in heterogeneous databases using neural networks. In *Proceedings of VLDB*, pages 1–12, 1994.
19. J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proceedings of VLDB*, pages 49–58, 2001.
20. 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.
21. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm. In *Proceedings of ICDE*, pages 117–128, 2002.
22. S. Melnik, E. Rahm, and P. Bernstein. Rondo: A programming platform for generic model management. In *Proceedings of SIGMOD*, pages 193–204, 2003.

23. 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.
24. A.G. Miller. Wordnet: A lexical database for english. In *Communications of the ACM*, number 38(11), pages 39–41, 1995.
25. N. Noy and M. Klein. Ontology evolution: Not the same as schema evolution. In *Knowledge and Information Systems*, in press, 2002.
26. 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.
27. 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.
28. 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.
29. P. Shvaiko. Iterative schema-based semantic matching. Technical Report DIT-04-020, University of Trento, 2004.
30. 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.
31. 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.
32. 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.