Semantic Matching

Fausto Giunchiglia and Pavel Shvaiko

DIT – Dept. of Information and Communication Technology
University of Trento, 38050 Povo, Trento, Italy
{fausto, pavel}@dit.unitn.it

Abstract

We think of *match* as an operator that takes two graph-like structures (e.g., database schemas or ontologies) and produces a mapping between elements of the two graphs that correspond semantically to each other. The goal of this paper is to propose a new approach to matching, called *semantic matching*. The contributions of this paper are (i) a rational reconstruction of the major matching problems and their articulation in terms of the more generic problem of matching graphs; (ii) the identification of semantic matching as a new approach for performing generic matching; and (iii) a proposal of implementing semantic matching by testing propositional satisfiability.

1 Introduction

Due to the progress of information and communication technologies the number of different information resources is rapidly increasing, and the problem of semantic heterogeneity is becoming more and more severe, see for instance [Washe *et al*., 2001], [Goh, 1997], [Giunchiglia and Zaihrayeu, 2002]. One proposed solution is matching. *Match* is an operator that takes two graph-like structures (e.g., database schemas or ontologies) and produces a mapping between elements of the two graphs that correspond semantically to each other. So far, with the noticeable exception of [Serafini *et al*, 2003], the key intuition underlying *all* the approaches to matching has been to map labels (of nodes) and to look for similarity (between labels) using syntax driven techniques and syntactic similarity measures; see for instance [Do and Rahm, 2002], [Madhavan *et al*., 2001]. We say that all these approaches are different variations of *syntactic matching.* In syntactic matching semantics are not analyzed directly, but semantic correspondences are searched for only on the basis of syntactic features.

In this paper we propose a novel approach, called *semantic matching*, with the following main features:

- We search for semantic correspondences by mapping meanings (concepts), and not labels, as in syntactic matching.
- We use semantic similarity relations between elements (concepts) instead of syntactic similarity relations. In particular, we consider relations, which relate the extensions of the concepts under consideration (for instance, more/less general relations).

The contributions of this paper are (i) a rational reconstruction of the major matching problems and their articulation in terms of the more generic problem of matching graphs; (ii) the identification of semantic matching as a new approach for performing generic matching; and (iii) a proposal of using a decider for propositional satisfiability (SAT) as a possible way of implementing semantic matching. The algorithm proposed works only on Directed Acyclic Graphs (DAG's) and *is-a* links. It is important to notice that SAT deciders are correct and complete decision procedures for propositional logics. Using SAT allows us to find only and all possible mappings between elements. This is another major advantage over syntactic matching approaches, which are based on heuristics. The SAT-based algorithm discussed in this paper is a minor modification/extension of the work described in [Serafini *et al*, 2003].

The rest of the paper is organized as follows. Section 2 defines the notion of matching and discusses the essence of semantic matching. Section 3 provides guidelines to the implementation of semantic matching. Section 4 overviews the related work. Section 5 reports some conclusions.

2 Matching

We assume that all the data and conceptual models (e.g., relational db schemas, OODB and XML schemas, concept hierarchies and ontologies) can be represented as graphs, see for a detailed discussion [Giunchiglia and Shvaiko, 2003]. Therefore, the problem of matching heterogeneous and autonomous information resources can be decomposed in two steps:

1. extract graphs from the data or conceptual models,
2. match the resulting graphs.

Notice that this allows for the statement and solution of a more *generic matching problem*, very much along the lines of what done in Cupid [Madhavan *et al*., 2001], and COMA [Do and Rahm, 2002].

Let us define the notion of matching graphs more precisely. *Mapping element* is a 4-tuple $< m_{ID}, N^i_1, N^j_2, R >$, $i=1...h; j=1..k$; where m_{ID} is a unique identifier of the given mapping element; N^i_1 is the *i-th* node of the first graph, h is the number of nodes in the first graph; N^j_2 is the *j-th* node of the second graph, k is the number of nodes in the second graph; and R specifies a *similarity relation* of the given nodes. A *Mapping* is a set of mapping elements. *Matching* is the process of discovering mappings between two graphs through the application of a matching algorithm. There exist two approaches to graph matching, namely *exact matching* and *inexact* or *approximate matching*. For obvious reasons we are interested in inexact matching.

We classify matching into *syntactic* and *semantic matching* depending on how matching elements are computed and on the kind of similarity relation R used.

- In *syntactic matching* the key intuition is to map labels (of nodes) and to look for the similarity using syntax driven techniques and syntactic similarity measures. Thus, in the case of *syntactic matching*, mapping elements are computed as 4-tuples $< m_{ID}, L^i_1, L^j_2, R >$, where L^i_1 is the *label* at the *i-th* node of the first graph; L^j_2 is the *label* at the *j-th* node of the second graph; and R specifies a similarity relation in the form of a coefficient, which measures the similarity between the *labels* of the given nodes. Typical examples of R are coefficients in [0,1], for instance, *similarity* coefficients [Madhavan *et al.*, 2001]. Similarity coefficients usually measure the closeness between the two elements linguistically and structurally. For instance, based on linguistic analysis, the similarity coefficient between elements "telephone" and "phone" from the two hypothetical schemas could be 0,7.
- As from its name, in *semantic matching* the key intuition is to map meanings (concepts). Thus, in the case of *semantic matching*, mapping elements are computed as 4-tuples $< m_{ID}, C^i_1, C^j_2, R >$, where C^i_1 is the *concept* of the *i-th* node of the first graph; C^j_2 is the *concept* of the *j-th* node of the second graph; and R specifies a similarity relation in the form of a semantic relation between the *extensions of concepts* at the given nodes. Possible R's between nodes are *equality* (=), *overlapping* ($\cap$), *mismatch* ($\perp$), or *more general/specific* ($\subseteq$, $\supseteq$).

These ideas are schematically represented in Figure 1. It is important to notice that all past approaches to matching we are aware of, with the exception of [Serafini *et al*, 2003], are based on syntactic matching.

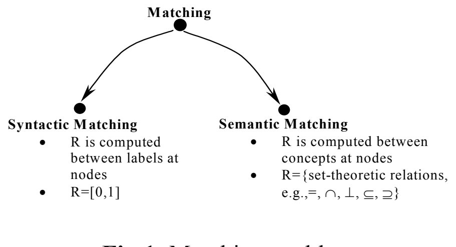
Matching
Syntactic Matching
R is computed between labels at nodes
R=[0,1]
Semantic Matching
R is computed between concepts at nodes
R={set-theoretic relations, e.g.,=, ∩, ⊥, ⊆, ⊇}

Fig.1. Matching problems

Let us consider some examples, which make the consequences of the observation described above clearer. For any example we also report the results produced by the state of the art matcher, Cupid [Madhavan *et al.*, 2001], which exploits very sophisticated syntactic matching techniques. Notationally, A stands for the label at a node; C_A stands for the concept denoted by A; $\boldsymbol{C_i}$ stands for the concept at the node i (in the following we sometimes confuse concepts with their extensions), numbers in circles are the unique identifiers of the nodes under consideration. In order to keep track of the graph we refer to we index nodes, labels, concepts and their extensions with the graph number (which is "1" for the graph on the left and "2" for the graph on the right). Thus we have, for instance, A_1, 5_1, C_{A_1}, $\boldsymbol{C_{5_1}}$.

Analysis of siblings. Let us consider Figure 2. Structurally the graphs shown in Figure 2 differ in the order of siblings. Suppose that we want to match node 5_1 with node 2_2.

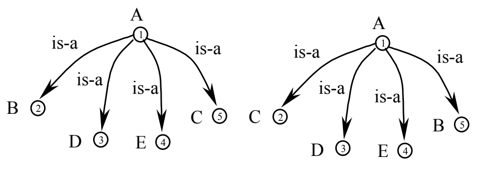

Fig.2. Analysis of siblings. Case 1

Cupid finds the similarity coefficient between labels at the given nodes, which equals to 0,8. This is because $A_1=A_2$, $C_1=C_2$ and we have the same structures on both sides. . A semantic matching approach compares concepts $C_{A_1} \cap C_{C_1}$ with $C_{A_1} \cap C_{C_1}$ and produces $\boldsymbol{C_{5_1}} = \boldsymbol{C_{2_2}}$.

Analysis of ancestors. Let us consider Figure 3. Suppose that we want to match nodes 5_1 and 1_2.

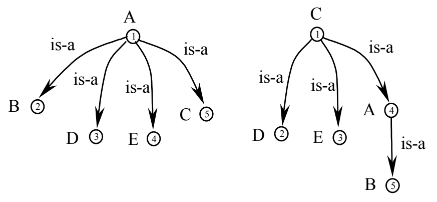

Fig.3. Analysis of ancestors. Case 1

Cupid does not find a similarity coefficient between the nodes under consideration, due to the significant differences in structure of the given graphs. In semantic matching, the concept denoted by the label at node 5_1 is C_{C_1}, while the concept at node 5_1 is $\boldsymbol{C_{5_1}} = C_{A_1} \cap C_{C_1}$. The concept at the node 1_2 is $\boldsymbol{C_{1_2}} = C_{C_2}$. By comparing the concepts denoted by the labels at nodes 5_1 and 1_2 we have that, being identical, they denote the same concept, namely $C_{C_1}=C_{C_2}$. Thus, the concept at node 5_1 is a subset of the concept at node 1_2, namely $\boldsymbol{C_{5_1}} \subseteq \boldsymbol{C_{1_2}}$.

Let us complicate the example shown in Figure 3 by allowing for an arbitrary distance between ancestors, see Figure 4. The asterisk means that an arbitrary number of nodes are allowed between nodes 1_2 and 5_2. Suppose that we want to match nodes 5_1 and 5_2.

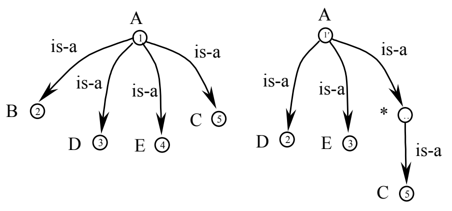

Fig.4. Analysis of ancestors. Case 2

Cupid finds out that the similarity coefficient between labels C_1 and C_2 is 0,86. This is because of the identity of labels (A_1=A_2, C_1=C_2), and due to the fact that nodes 5_1 and 5_2 are leaves. Notice how Cupid treats very differently the two situations represented here and in the example above, even if, from a semantic point of view, they are similar. Following semantic matching, the concept at node 5_1 is $\boldsymbol{C}_{5_1} = C_{A_1} \cap C_{C_1}$; while the concept at node 5_2 is $\boldsymbol{C}_{5_2} = C_{A_2} \cap * \cap C_{C_2}$. Since we have that $C_{A_1} = C_{A_2}$ and $C_{C_1} = C_{C_2}$, then $\boldsymbol{C}_{5_2} \subseteq \boldsymbol{C}_{5_1}$.

Enriched analysis of siblings. Suppose that we want to match nodes 2_1 and 2_2, see Figure 5.

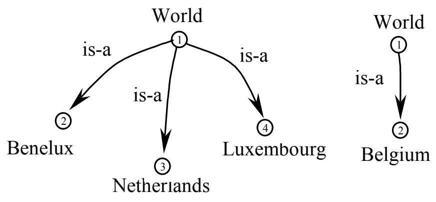

Fig.5. Analysis of siblings. Case 2

Cupid without thesaurus doesn't find a match; with the use of thesaurus it finds out that the similarity coefficient between nodes with labels *Benelux*$_1$ and *Belgium*$_2$ is 0,68. This is mainly because of the entry in the thesaurus specifying *Belgium* as a part of *Benelux,* and due to the fact that the nodes with labels *Benelux*$_1$ and *Belgium*$_2$ are leaves.

Following semantic matching, both concepts $C_{Benelux_1}$ and $C_{Belgium_2}$ are subsets of the concept $C_{World_{1,2}}$. Let us suppose that an oracle, for instance WordNet, states that *Benelux* is a name standing for *Belgium*, *Netherlands* and *Luxembourg*. Therefore, we treat $\boldsymbol{C}_{2_1}$ in Figure 5 as $C_{Benelux_1} \cap C_{Netherlands_1} \cap C_{Luxembourg_1} = C_{Belgium_1}$. Thus, $\boldsymbol{C}_{2_1} = \boldsymbol{C}_{2_2}$.

3 Implementing Semantic Matching

There are two levels of granularity while performing semantic (and also syntactic matching) matching: *element-level* and *structure-level*. Element-level matching techniques compute mapping elements between individual labels/concepts at nodes; structure-level techniques compute mapping elements between subgraphs.

3.1 Element-level Semantic Matching

Element-level semantic techniques analyze individual labels/concepts at nodes. At the element-level we can exploit all the techniques discussed in the literature, see for instance [Do and Rahm, 2002], [Melnik *et al.*, 2002], [Rahm and Bernstein, 2001]. The main difference here is that, instead of a syntactic similarity measure, these techniques must be modified to return a semantic relation R, as defined in Section 2. We distinguish between *weak semantics* and *strong semantics* element-level techniques. *Weak semantics* techniques are syntax driven techniques: examples are techniques, which consider labels as strings, or analyze data types, or soundex of schema elements. Let us consider some examples.

Analysis of strings. String analysis looks for common prefixes or suffixes and calculates the distance between two strings. For example, the fact that the string "phone" is a substring of the string "telephone" can be used to infer that "phone" and "telephone" are synonyms. Before analyzing strings, a matcher could perform some preliminary parsing, e.g., extract tokens, expand abbreviations, delete articles and then match tokens. The analysis of strings discovers only equality between concepts.

Analysis of data types. These techniques analyze the data types of the elements to be compared and are usually performed in combination with string analysis. For example, the elements "phone" and "telephone" are supposed to have the same data type, namely "string" and therefore can be found equal. However, "phone" could also be specified as an "integer" data type. In this case a mismatch is found. As another example the integer "Quantity" is found to be a subset of the real "Qty". This kind of analysis can produce any kind of semantic relation.

Analysis of soundex. These techniques analyze elements' names from how they sound. For example, elements "for you" and "4 U" are different in spelling, but similar in soundex. This analysis can discover only equality between concepts.

Strong semantics techniques exploit, at the element-level, the semantics of labels. These techniques are based on the use of tools, which explicitly codify semantic information, e.g. thesauruses [Madhavan *et al.*, 2001], WordNet or combinations of them [Castano *et al.*, 2000]. Notice that these techniques are also used in syntactic matching. In this latter case, however, the semantic information is lost before moving to structure-level matching and approximately codified in syntactic relations.

Precompiled thesaurus. A precompiled thesaurus usually stores entries with synonym and hypernym relations. For example, the elements "e-mail" and "email" are treated as synonyms from the thesaurus look up: *syn key* - "e-mail:email = syn". Precompiled thesauruses (most of them) identify equivalence and more general/specific relations. In some cases domain ontologies are used as precompiled thesauruses [Mena *et al.*, 1996].

WordNet. WordNet is an electronic lexical database for English (and other languages), where various *senses* (namely, possible meanings of a word or expression) of words are put together into sets of synonyms (synsets). Synsets in turn are organized as hierarchy. Following [Serafini *et al*, 2003] we can define the semantic relations in terms of senses. *Equality*: one concept is equal to another if there is at least one sense of the first concept, which is a synonym of the second. *Overlapping*: one concept is overlapped with the other if there are some senses in common. *Mismatch*: two concepts are mismatched if they have no sense in common. *More general / specific*: One concept is more general than the other iff there exists at least one sense of the first concept that has a sense of the other as a hyponym or as a meronym. One concept is less general than the other iff there exists at least one sense of the first concept that has a sense of the other concept as a hypernym or as a holonym. For example, according to WordNet, the concept "hat" is a holonym for the concept "brim", which means that "brim" is less general than "hat".

3.2 Structure-level Semantic Matching

The approach we propose is to translate the matching problem, namely the two graphs and our *mapping queries* into a propositional formula and then to check it for its validity. By mapping query we mean here the pair of nodes that we think will match and the semantic relation between them. In the following we show how, limited to the case of DAG's and *is-a* hierarchies, we can check validity by using propositional satisfiability (SAT) decider. Notice that SAT deciders are correct and complete decision procedures for propositional satisfiability and therefore will exhaustively check for all possible mappings. Being complete, they automatically implement all the examples described in the previous section, and more. This is another advantage over syntactic matching, whose existing implementations are based only on heuristics.

Our SAT based approach to semantic matching incorporates six steps. We describe below its intended behavior by running these six steps on the example shown in Figure 3 and by matching nodes 5_1 and 1_2 (steps 2-5 are taken from [Serafini *et al*, 2003]).

1. **Extract the two graphs.** Notice that during this step, in the case of DB, XML or OODB schemas, it is necessary to extract useful semantic information, for instance in the form of ontologies. There are various techniques for doing this, see for instance [Davis and Aiken, 2000], [Mena *et al.*, 1996]. The result is the graph in Figure 3.
2. **Compute element-level semantic matching.** For each node, compute semantic relations holding among all the concepts denoted by labels at nodes under consideration. In this case C_{A_1} has no semantic relation with C_{C_2} while we have that $C_{C_1} = C_{C_2}$.
3. **Compute concepts at nodes.** Starting from the root of the graph, attach to each node the concepts of all the nodes above it. Thus, we attach $\boldsymbol{C_{1_1}} = C_{A_1}$ to node 1_1; $\boldsymbol{C_{5_1}} = C_{A_1} \cap C_{C_1}$ to node 5_1; $\boldsymbol{C_{1_2}} = C_{C_2}$ to node 1_2 in the *is-a* hierarchy. As it turns out we have that $\boldsymbol{C_{5_1}} \subseteq \boldsymbol{C_{1_2}}$.
4. **Construct the propositional formula**, representing the matching problem. In this step we translate all the semantic relations computed in step 2 into propositional formulas. This is done according to the following transition rules:

$$C_{A_1} \supseteq C_{A_2} \Rightarrow C_{A_2} \rightarrow C_{A_1}$$
$$C_{A_1} \subseteq C_{A_2} \Rightarrow C_{A_1} \rightarrow C_{A_2}$$
$$C_{A_1} = C_{A_2} \Rightarrow C_{A_1} \equiv C_{A_2}$$
$$C_{A_1} \perp C_{A_2} \Rightarrow \neg(C_{A_1} \wedge C_{A_2})$$

   Subset translates into implication; equality into equivalence; disjointness into the negation of conjunction. In the case of Figure 3 we have that $C_{C_1} \equiv C_{C_2}$ is an axiom. Furthermore, since we want to prove that $\boldsymbol{C_{5_1}} \subseteq \boldsymbol{C_{1_2}}$, our goal is to prove that $((C_{A_1} \wedge C_{C_1}) \rightarrow C_{C_2})$. Thus, our target formula is $((C_{C_1} \equiv C_{C_2}) \rightarrow (C_{A_1} \wedge C_{C_1}) \rightarrow C_{C_2}))$.
5. **Run SAT.** In order to prove that $((C_{C_1} \equiv C_{C_2}) \rightarrow (C_{A_1} \wedge C_{C_1}) \rightarrow C_{C_2}))$ is valid, we prove that its negation is unsatisfiabile, namely that a SAT solver run on the following formula $((C_{C_1} \equiv C_{C_2}) \wedge\neg (C_{A_1} \wedge C_{C_1}) \rightarrow C_{C_2}))$ fails. A quick analysis shows that SAT will return FALSE.
6. **Iterations.** Iterations are performed re-running SAT. We need iterations, for instance, when matching results are not good enough, for instance no matching is found or a form of matching is found, which is too weak, and so on[1]. The idea is to exploit the results obtained during the previous run of SAT to tune the matching and improve the quality of the final outcome. Let us consider Figure 6.

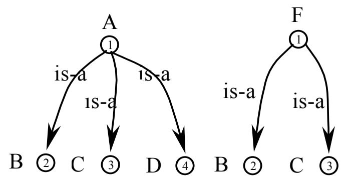

Fig.6. Not good enough answer

Suppose that we have found out that $\boldsymbol{C_{2_1}} \cap \boldsymbol{C_{2_2}} \neq \varnothing$, and that we want to improve this result. Suppose that an oracle tells us that $C_{A_1} = C_{F_2} \cup C_{G_2}$. In this case the graph on the left in Figure 6 can be transformed into the two graphs in Figure 7.

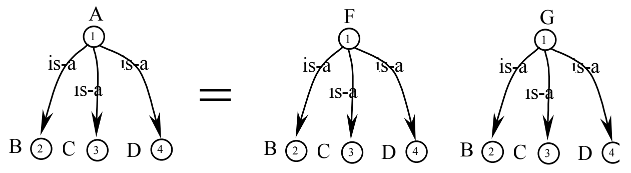

Fig.7. Extraction of additional semantic information

After this additional analysis we can infer that $\boldsymbol{C_{2_1}} = \boldsymbol{C_{2_2}}$. As a particular interesting case, consider the following situation, see Figure 7.1

[1] [Giunchiglia and Zaihrayeu, 2002] provides a long discussion about the importance of dealing with the notion of "good enough answer" in information coordination in peer-to-peer systems.

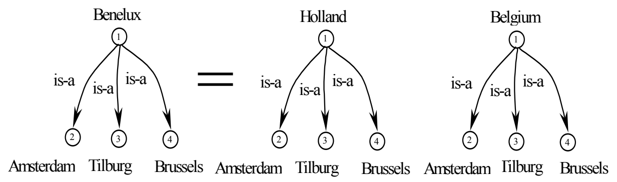

Fig.7.1. Extraction of additional semantic information. Example

In this case the concept *Brussels* in the graph on the left (after the sign "=") becomes inconsistent (empty intersection) and can be omitted; and the same for the concepts at nodes *Amsterdam* and *Tilburg* in the graph on the right. The resulting situation is as follows:

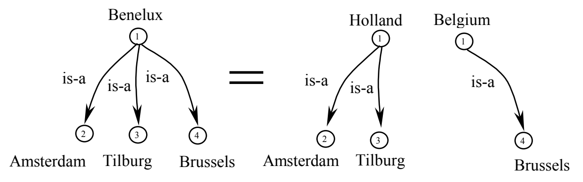

Fig.7.2. Extraction of additional semantic information. Example

Another motivation for multiple iterations is to use the result of a previous match in order to speed up the search of new matches. Consider the following example.

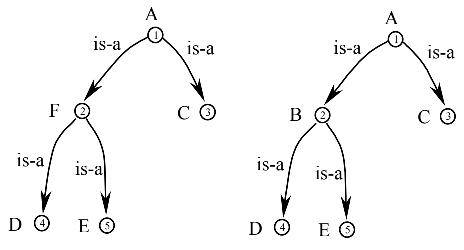

Fig.8. Iterations

Having found that $C_{2_1} \subseteq C_{2_2}$, we can automatically infer that $C_{5_1} \subseteq C_{5_2}$, without rerunning SAT, for obvious reasons, and the same for C_{4_1} and C_{4_2}. As a particular case consider the following situation:

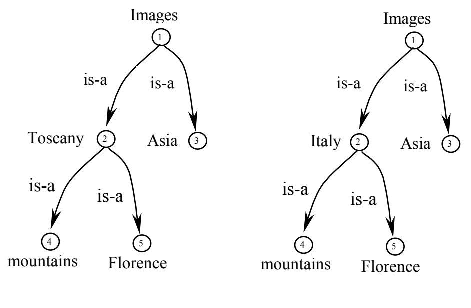

Fig. 8.1. Iterations. Example

Our algorithm allows us to find that $C_{5_1} \subseteq C_{5_2}$, while, being Tuscany in Italy we actually have $C_{5_1} = C_{5_2}$. This is an acceptable result as long as we are not looking for the strongest possible relation holding between two nodes.

4 Related Work

From a technical point of view the matcher we have proposed in this paper is a function *MatchNodesR(G_1,G_2,n_1,n_2,R)* which takes two graphs, two nodes, and a relation and returns a Yes/No answer. Most matchers proposed in the literature are a function *Match(G_1,G_2)* which takes two graphs and returns a set of mappings (n_1, n_2, R). However, it is easy to see how we can build an analogous function. The naive approach being to triple loop on the nodes of the graphs and on the set of proposed relations and, at each loop, call *MatchNodesR*.

At present, there exists a line of semi-automated schema matching and ontology integration systems, see for instance [Madhavan *et al.*, 2001], [Do and Rahm, 2002], [Li and Clifton, 2000], [Castano *et al.*, 2000], [Arens *et al.*, 1996], [Mena *et al.*, 1996], [Doan *et al.*, 2002], etc. Most of them implement syntactic matching. A good survey, up to 2001, is provided in [Rahm and Bernstein, 2001]. The classification given in this survey distinguishes between individual implementations of match and combinations of matchers. Individual matchers comprise instance- and schema-level, element- and structure-level, linguistic- and constrained-based matching techniques. Individual matchers can be used in different ways, e.g. simultaneously (hybrid matchers), see [Li and Clifton, 2000], [Castano *et al.*, 2000], [Madhavan *et al.*, 2001] or in series (composite matchers), see for instance [Doan *et al.*, 2002], [Do and Rahm, 2002].

The idea of generic (syntactic) matching was first proposed by Phil Bernstein and implemented in Cupid system [Madhavan *et al.*, 2001]. Cupid implements a complicated hybrid match algorithm comprising linguistic and structural schema matching techniques, and computes normalized similarity coefficients with the assistance of a precompiled thesaurus. COMA [Do and Rahm, 2002] is a generic schema matching tool, which implements more recent composite generic matchers. With respect to Cupid, the main innovation seems to be a more flexible architecture.

A lot of state of the art syntactic matching techniques exploiting weak semantic element-level matching techniques have been implemented. For instance, in COMA, schemas are internally encoded as DAG's, where the elements are the paths, which are analyzed using string comparison techniques. Similar ideas are exploited in Similarity Flooding (SF) [Melnik *et al.*, 2002]. SF is 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 mappings between the nodes of the input graphs. The technique uses a syntactic string comparison mechanism of the vertices' names to obtain an initial mapping, which is further refined within the fix-point computation.

Some work has also been done in strong semantics element-level matching. For example, [Castano *et al.*, 2000] utilizes a common thesaurus, while [Madhavan *et al.*, 2001] has a precompiled thesaurus. In MOMIS [Castano *et al.*, 2000] element-level matching using a common thesaurus is carried out through a calculation of the name, structural and global affinity coefficients. The thesaurus presents a set of intensional and extensional relations, which depict intra- and inter-schema knowledge about classes, and attributes of the input schemas. All these systems implement syntactic matching and, when moving from element-level to structure-level matching, don't exploit the semantic information residing in the graph structure, and just translate the element-level semantic information into affinity levels.

As far as we know the only example where element-level and a simplified version of structure- level strong semantics matching have been applied is CTXmatch [Serafini *et al*, 2003]. The main problem of CTXmatch is that its rather limited in scope (it applies only to concept hierarchies), and it is hard to see the general lessons behind this work. This paper provides the basics for a better understanding of the work on CTXmatch.

5 Conclusions and Future Work

In this paper we have stated and analyzed the major matching problems e.g., matching database schemas, XML schemas, conceptual hierarchies and ontologies and shown how all these problems can be defined as a more generic problem of matching graphs. We have identified semantic matching as a new approach for performing generic matching, and discussed some of its key properties. Finally, we have identified SAT as a possible way of implementing semantic matching, and proposed an iterative semantic matching approach based on SAT.

This is only very preliminary work, some of the main issues we need to work on are: develop an efficient implementation of the system, do a thorough testing of the system, also against the other state of the art matching systems, study how to take into account attributes and instances, and so on.

Acknowledgments

Thanks to Luciano Serafini, Paolo Bouquet for many discussions on CTXmatch. Stefano Zanobini and Phil Bernstein have proposed very useful feedback on the semi-final version of this paper. Also thanks to Michail Yatskevich for his work on running SAT solvers on our matching problems.

References

[Arens *et al.*, 1996] Yigal Arens, Chun-Nan Hsu, and Craig A. Knoblock. Query processing in the SIMS information mediator. In *Advanced Planning Technology*. AAAI Press, California, USA, 1996.

[Buneman, 1997] Peter Buneman. Semistructured data. In *Proc. of PODS*, pages 117–121, 1997.

[Castano *et al.*, 2000] Castano S., V. De Antonellis, S. De Capitani di Vimercati. Global Viewing of Heterogeneous Data Sources. *IEEE Trans. on Knowledge and Data Engineering*, 2000

[Doan *et al.*, 2002] A. Doan, J.Madhavan, P. Domingos, and A. Halvey. Learning to map between ontologies on the semantic web. In *Proc. Of WWW-02, 11th International WWW Conf.*, Hawaii 2002.

[Do and Rahm, 2002] Hong H. Do, Erhard Rahm. COMA – A System for Flexible Combination of Schema Matching Approach. *VLDB Journal*, pages 610-621, 2002

[Goh, 1997] Cheng Hian Goh. *Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources*. Phd, MIT, 1997.

[Giunchiglia and Shvaiko, 2003] Fausto Giunchiglia and Pavel Shvaiko. Semantic Matching. *Technical Report #DIT-03-013*. http://www.dit.unitn.it/~p2p/

[Giunchiglia and Zaihrayeu, 2002] Fausto Giunchiglia and Ilya Zaihrayeu. Making peer databases interact - a vision for an architecture supporting data coordination. *Proceedings of the Conference on Information Agents*, Madrid, September 2002.

[Li and Clifton, 2000] W. Li, C. Clifton: SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering*, 33(1): 49-84, 2000.

[Davis and Aiken, 2000] Kathi Hogshead Davis, Peter H. Aiken: Data reverse engineering: a historical survey, WCRE'00, 2000.

[Madhavan *et al.*, 2001] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with Cupid. *VLDB Journal*, pages 49-58, 2001

[Melnik *et al.*, 2002] Melnik, S., H. Garcia-Molina, E. Rahm: Similarity Flooding: A Versatile Graph Matching Algorithm. *ICDE*, 2002.

[Mena *et al.*, 1996] 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 1st International Conference on Cooperative Information Systems*. Brussels, 1996.

[Rahm and Bernstein, 2001] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4): 334-350, 2001.

[Serafini *et al*, 2003] Luciano Serafini, Paolo Bouquet, Bernardo Magnini, and Stefano Zanobini. An Algorithm for Matching Contextualized Schemas via SAT. In *Proc. of CONTEX 03,* June 2003.

[Washe *et al.*, 2001] 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 *Proc. of IJCAI*, August 2001.