UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

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

ITERATIVE SCHEMA-BASED SEMANTIC MATCHING

Pavel Shvaiko

March 2004

Technical Report # DIT-04-020

Iterative Schema-based Semantic Matching

Pavel Shvaiko
PhD thesis proposal (Extended version)

Dept. of Information and Communication Technology
University of Trento,
38050 Povo, Trento, Italy
pavel@dit.unitn.it

Abstract. Schema/ontology matching is a critical problem in many application domains, such as, schema/ontology integration, data warehouses, e-commerce, web services coordination, Semantic Web, semantic query processing, catalog matching, etc. I view *Match* as an operator that takes two graph-like structures (e.g., database schemas or ontologies) and produces a mapping between the nodes of the two graphs that correspond semantically to each other. In this PhD proposal, I first present a classification of state of the art matching techniques. I then introduce a new schema-based approach to matching called *semantic matching* as implemented within the *S-Match* system. At present, the approach allows handling only tree-like structures (e.g., taxonomies or concept hierarchies). Finally, I discuss the main research goals to be achieved within the PhD thesis: (i) extend the semantic matching approach to allow handling graphs; (ii) extend the semantic matching algorithm for computing mappings between graphs; (iii) develop a theory of iterative semantic matching in order to improve quality and efficiency of matching via iterations; (iv) implement the above stated goals within the *S-Match* system and conduct the experimental evaluation in different initial settings.

Keywords: Schema/ontology matching, Semantic Web, Semantic heterogeneity.

1 Introduction

The progress of information and communication technologies, and in particular of the Web, has made a huge amount of disparate information available. The number of different information resources is growing significantly, and therefore the problem of managing semantic heterogeneity is increasing. Many solutions to this problem include identifying terms in one information source that "match" terms in another information source. The applications can be viewed to refer to graph-like structures containing terms and their inter-relationships. These might be database schemas, taxonomies, or ontologies. The *Match* operator then takes two graph-like structures and produces a mapping between the nodes of the graphs that correspond semantically to each other.

Match is a critical operator in many well-known application domains, such as Semantic Web, schema/ontology integration, data warehouses, e-commerce, semantic query processing, and XML message mapping. More recently, new application domains have emerged, such as catalog matching, where the match operator is used to map entries of catalogs among business partners; or web services coordination, where Match is used to identify dependencies among data sources.

In the PhD thesis proposal I focus on *semantic matching*, as introduced in [16] and implemented within the *S-Match* system [15][1]. The key intuition behind semantic matching is that mappings should be calculated by computing the semantic relations holding between the concepts (and not labels!) assigned to nodes. I classify previous approaches to matching under the heading of *syntactic matching* since they do not analyze meaning of labels, and thus semantics, directly. In these approaches semantic correspondences are determined using (i) syntactic similarity measures and (ii) syntax driven techniques, see, for instance [30], [20], [11], [12], [24], [32], [14], etc. The first key distinction of the semantic matching approach is that the mappings are calculated between schema/ontology elements (e.g., nodes of graphs) by computing *semantic*

[1] The current version of *S-Match* is a rationalized re-implementation of the CTXmatch system [8] with a few added functionalities.

relations (for example, equivalent or subsuming elements), instead of computing coefficients rating match quality in the [0,1] range. The second key distinction is that semantic relations are determined by analyzing meaning (concepts and their extensions, not labels as in syntactic matching) which is codified in the elements and the structure of schemas/ontologies. At present, the semantic matching approach as implemented within the *S-Match* system is limited to the case of tree-like structures e.g., taxonomies or concept hierarchies consisting of only classes and single inheritance links.

The main research goals within the PhD thesis proposed are: (i) extend the semantic matching approach to allow handling graphs. Therefore, taking into account, in the case of relational databases: attributes and referential integrity constraints (including cycles), or in the case of ontologies: classes connected by multiple inheritance links, slots, facets and axioms; (ii) extend the semantic matching algorithm for computing mappings between graphs; (iii) develop a theory of iterative semantic matching in order to improve quality and efficiency of matching via iterations; (iv) implement the above stated goals within the *S-Match* system and conduct the experimental evaluation in different initial settings.

2 The Matching Problem

2.1 A Motivating Example

To motivate the matching problem, and the semantic matching approach I am going to elaborate in the PhD thesis, 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.

Fig.1. Two XML schemas

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. 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

I assume that all the data and conceptual models (e.g., database schemas, taxonomies and ontologies) can be represented as graphs, see for a detailed discussion [16]. Therefore, the matching problem 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 *generic matching problem*, very much along the lines of what done in Cupid [20], and COMA [11].

Let us define the notion of matching graphs precisely. *Mapping element* is a 4-tuple $< ID_{ij}, n1_i, n2_j, R>$, $i=1,...,N1; j=1,...,N2$; where ID_{ij} 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* 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.

Matching approaches can be classified into *syntactic* and *semantic* depending on how mapping elements are computed and on the kind of similarity relation *R* used (see [16] for a in depth discussion):

- 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 $< ID_{ij}, l1_i, l2_j, R >$, where $l1_i$ is the *label* at the *i-th* node of the first graph; $l2_j$ 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] range, for instance, *similarity* coefficients [20], [14] or *confidence measures* [32]. Similarity coefficients usually measure the closeness between the two elements linguistically and structurally. 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.

- As its name indicates, 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 $< ID_{ij}, C1_i, C2_j, R >$, where $C1_i$ is the *concept* at the *i-th* node of the first graph; $C2_j$ is the *concept* at the *j-th* node of the second graph; and *R* specifies a similarity relation in the form of a *semantic relation* which holds between the (extensions of) concepts at the given nodes. Possible semantic relations are: *equivalence* (=), *more general* ($\sqsupseteq$), *less general* ($\sqsubseteq$), *mismatch* ($\perp$) and *overlapping* ($\sqcap$). Thus, for instance, the concepts of two nodes are equivalent if they have the same extension, they mismatch if their extensions are disjoint, and so on for the other relations. The relations are ordered according to their binding strength, as they have been listed, from the strongest to the weakest, with *less general* and *more general* having the same binding power. Thus, *equivalence* is the strongest binding relation since the mapping tells us that the concept of the second node has exactly the same extension as the first, *more general* and *less general* give us a containment information with respect to the extension of the concept of the first node, *mismatch* provides a containment information with respect to the extension of the complement of the concept of the first node, while, finally, *overlapping* does not provide any useful information, since we have containment with respect to the extension of both the concept of the first node and its negation. For instance, the semantic relation holding between nodes with labels *Consumer_Electronics* in A1 and *Sale* in A2 in Figure 1 is "equivalence" (see Section 4 for details).

These ideas are schematically represented in Figure 2.

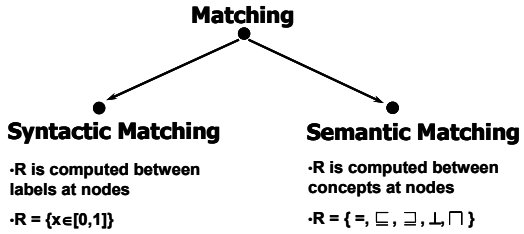

Fig.2. Matching: Syntactic vs. Semantic

It is important to notice that all past approaches to matching I are aware of, with the exception of [8], perform syntactic matching.

3 State of the Art

3.1 Classification of Matching Approaches

At present, there exists a line of semi-automated schema/ontology matching systems, see for instance [20], [11], [12], [24], [32], [2], [18], [25], [27], [21], etc. A good survey on schema matching approaches, up to 2001, is provided by Erhard Rahm and Phil Bernstein in [30] and presented in Figure 3[2].

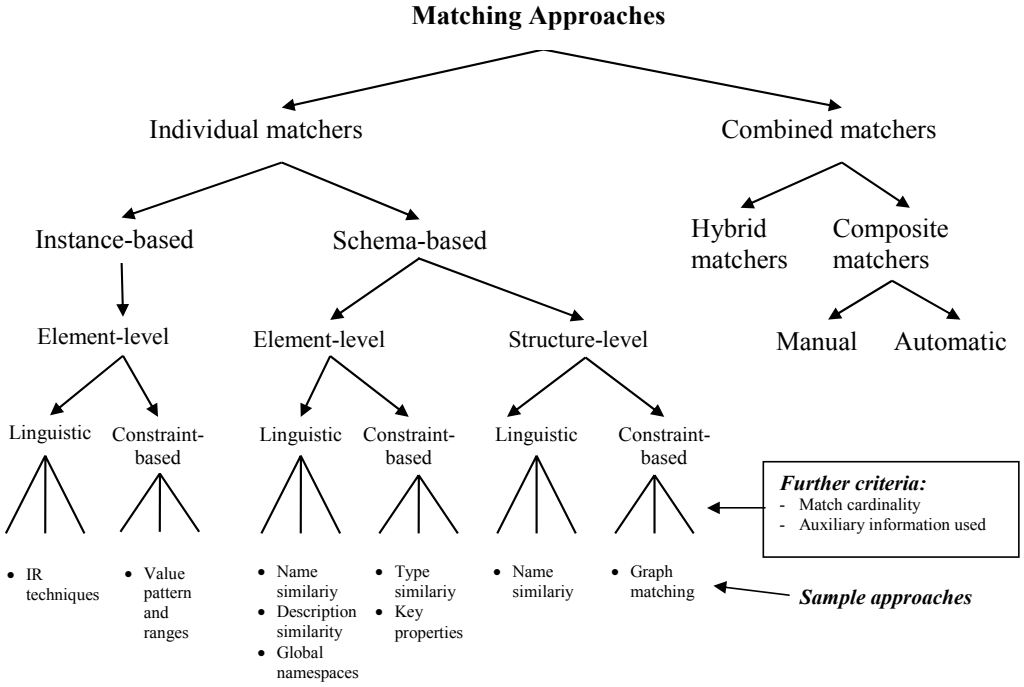

Fig.3. Classification of matching approaches

The classification 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 [20], [2] or combining the results of independently executed matchers (*composite* matchers), see for instance [11], [12].

In this PhD thesis I focus only on schema-based approaches, and therefore consider only schema/ontology information, not instance data[3]. Let us discuss them in more detail. There are two levels of granularity while performing schema-based semantic (and also syntactic matching) 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 [6], 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. One possible solution is to distinguish between techniques which have either *heuristic* or *formal* ground. The key

[2] Without loss of generality the classification of [30] can be also applied to systems performing matching between ontologies. For this reason the title of Figure 3 is stated more generally rather then it appears in the original publication.

[3] Prominent solutions of instance-based schema/ontology matching as well as possible extensions of the instance-based part of the classification presented in Figure 3 can be found in [12] and [18].

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 inferences. The other possible augmentation is also to distinguish between techniques which 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 [20], WordNet [26], etc. To make the distinctions between the categories proposed more clear, I revised a schema-based part of the classification of matching techniques by Erhard Rahm and Phil Bernstein, see Figure 4. All the innovations are marked in bold type. Let us consider them in more detail. I 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 [30]. I also renamed in schema-based part linguistic techniques into string-based techniques, to discard from this category methods which perform morphological analysis of strings, which I view only as a preprocessing part, for example, for matching techniques based on lexicons or string-based techniques.

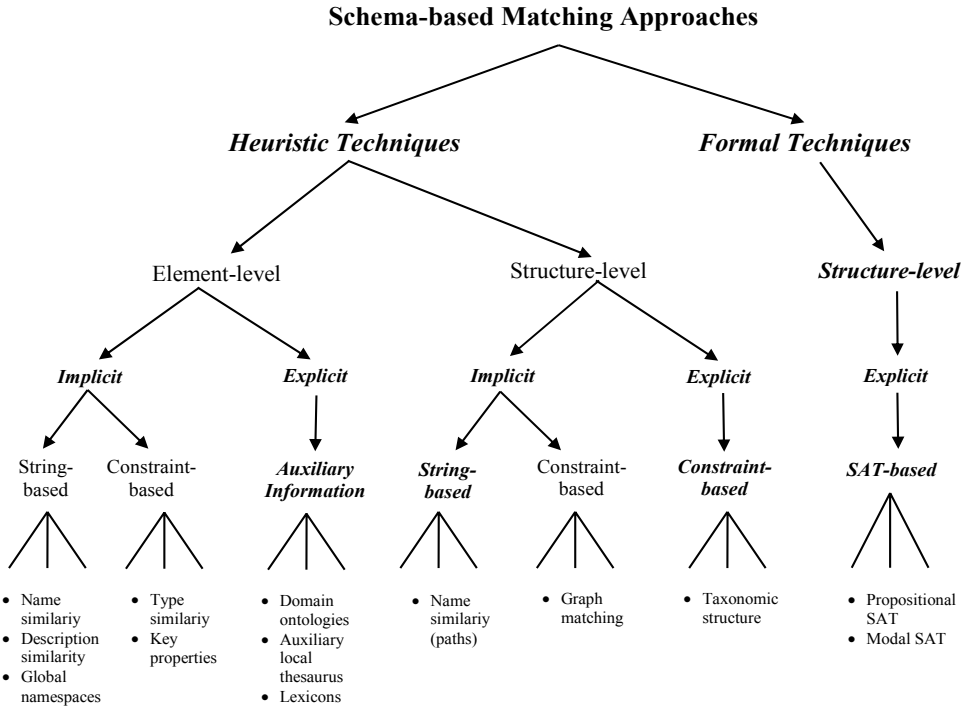

Fig.4. A revised classification of schema-based matching approaches

Heuristic element-level explicit techniques

• *Precompiled thesaurus and domain ontologies.* A precompiled thesaurus usually stores domain knowledge as entries with synonym, hypernym and other 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, see for example [20]. In some cases domain ontologies are used as precompiled thesauruses [25].

• *Lexicons.* The approach is to use lexicons to obtain meaning of terms used in schemas/ontologies. For example, 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. Relations between schema/ontology elements can be computed in terms of bindings between WordNet senses, see for instance [15], [8].

Heuristic structure-level implicit techniques

• *String-based.* These matchers build long labels by concatenating all labels at nodes in a path to a single string. Then string matching techniques look for common prefixes, suffixes and calculate the distance between the two strings. For example, in Figure 1, the node with label *Cases* in A1 can be codified as *Sale.Consumer_Electronics.Cell_Phones.Accessories.Cases*, while the node with label *Cases* in A2 can be codified as *Consumer_Electronics.Sale.Cameras_and_Photos.Accessories.Cases*. Then affix, n-gram, edit distance, etc. techniques can be used by a matcher to compute similarity between the two long strings, see for instance [11].

Heuristic structure-level explicit techniques

• *Taxonomic structure.* These matchers analyze and compare positions of terms (labels) within taxonomies. For examples, they take two paths with links between classes defined by the hierarchical relations or by slots and their domains and ranges, compare terms along these paths and identify similar terms [27].

Formal structure-level explicit techniques

• *Propositional satisfiability (SAT).* As from [16] 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 mapping query I 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 will exhaustively check for all possible mappings.

• *Modal SAT.* The approach is to delimit propositional SAT from the case of trees and DAGs which allows handling only unary predicates (e.g., classes) by admitting binary predicates (e.g., slots, etc.). The key idea is to enhance propositional logics with modal logic (or a kind of description logics) operators. Therefore, the matching problem is translated into a modal logic formula which is further checked for its validity. This is one of the main directions of my PhD thesis.

3.2 Prototype Matchers

We now look at some schema-based state of the art matching systems in light of the classification presented in Figure 4. Notice that non of the below discussed systems exploit formal structure-level techniques.

Similarity Flooding (SF). The SF [24] approach as implemented in Rondo [23] utilizes a hybrid matching algorithm based on the ideas of similarity propagation. Schemas are presented as directed labeled graphs grounding on the OIM specification [28]; 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 of the vertices' names 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 matching process. Referring to the classification in Figure 4, the SF system exploits heuristic techniques: implicit at the element-level and only implicit constraint-based at the structure-level.

Artemis. Artemis (Analysis of Requirements: Tool Environment for Multiple Information Systems) [9] was designed as a module of MOMIS mediator system [2], [3] for creating global views. Artemis does not cover all the issues of matching due to the origin function of schema integration. It performs affinity-based analysis and hierarchical clustering of source schemas elements. Affinity-based analysis is carried out through calculation of the name, structural and global affinity coefficients by exploiting a common thesaurus. The common thesaurus presents a set of terminological and extensional relationships which depict intra- and inter-schema knowledge about classes and attributes of the input schemas, which is built with the help of WordNet [26] and ODB-Tools [1]. A hierarchical clustering technique exploiting global affinity coefficients 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 schemas' attributes is determined through a mapping table. Referring to the classification in Figure 4, the Artemis tool exploits in a hybrid manner heuristic explicit techniques at the element- and structure-level.

Cupid. The Cupid system [20] implements a generic match algorithm comprising linguistic and structural schema matching techniques, and computes normalized 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 and a thesaurus look-up. The second phase (structural matching) computes structural similarity coefficients which measure the similarity between contexts in which individual schema elements occur in the schemas under consideration. The main idea behind the structural matching algorithm is to rely more on leaf level matches instead of the immediate descendents or intermediate substructures when computing similarity between non-leaf elements. 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 then a threshold. In comparison with the other hybrid matchers e.g., Dike [29] and Artemis [9], referring to [20], Cupid performs better in the sense of mappings' quality. Referring to the classification in Figure 4, the Cupid system exploits in a hybrid manner heuristic techniques: implicit and explicit at the element-level and only implicit constraint-based at the structure-level.

COMA. The COMA system [11] is a generic schema matching tool, which implements composite generic matchers. COMA 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 [11] it contains 6 individual matchers, 5 hybrid matches, and one reuse-oriented matcher. Most of them implement string-based techniques as a background idea; others share techniques with Cupid; 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 rooted directed acyclic graphs, where elements are the paths. This fact aims at capturing contexts in which the elements occur. One of the distinct features of the COMA tool is the possibility of performing iterations in matching process. It presumes interaction with a user which approves obtained matches and mismatches to gradually refine and improve the accuracy of match. Based on the comparative evaluations conducted in [10], COMA dominates Autoplex & Automatch [4], [5]; LSD [13] & GLUE [12]; SF [24]; and SemInt [19] matching tools. Referring to the classification in Figure 4, the COMA system exploits in a composite manner heuristic techniques: implicit and explicit at the element-level and implicit at the structure-level.

Chimaera is an environment for merging and testing (diagnosing) large ontologies [21]. Matching in the system is performed as one of the major subtasks of a merge operator. Chimaera search for merging candidates as pairs of *matching terms*, involving term names, term definitions, possible acronym and expanded forms, names that appear as suffixes of other names. It also has techniques to identify terms that should be related by subsumption, disjointness, etc. Thus, referring to the classification in Figure 4, the Chimaera system exploits heuristic techniques: implicit and explicit at the element-level and explicit at the structure-level.

OBSERVER. The OBSERVER approach [25] extends *CLASSIC* description logic model [7] by adding terminological relationships (synonyms, hyponyms, etc.) defined in WordNet [26] and using them for interoperation across ontologies. Matching is performed as a part of query processing. User queries are rewritten in a semantic preserving manner by replacing them with synonym terms from different ontologies. If a synonym is not found for a given term, it is substituted by its definition and then the translation (a kind of matching) algorithm is executed on the definition. Thus, the system exploits only heuristic element-level implicit and explicit techniques.

Anchor-PROMPT. The Anchor-PROMPT [27] (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 alignment algorithm takes as input two ontologies and a set of anchors-pairs of related terms, which are identified with the help of string-based techniques, or defined by a user, or another matcher computing linguistic similarity between frame names (labels at nodes), for example [21]. Then it refines them basing on the ontologies structure and users feedback. Referring to the classification in Figure 4, the Anchor-PROMPT system exploits heuristic techniques: implicit at the element-level, and implicit and explicit constraint-based at the structure-level.

4 Schema-based Semantic Matching

4.1 Semantic Matching via an Example

Let us discuss the semantic matching approach by analyzing how it works on the two XML schemas of Figure 1. Notice that all the links of schemas in Figure 1 have the *containment* semantics. The approach is schema-based, and, as such, it does not exploit the information encoded in data instances.

The key idea of the semantic matching approach is to calculate semantic relations by mapping meaning which is codified in the elements and the structure of the given schemas in two steps:

1. by computing the meaning of labels at nodes;
2. by computing the meaning of the positions that the labels at nodes have in a graph.

Step 1. Labels at nodes can be viewed as concise descriptions of documents that are stored under the nodes. The meaning of a label at a node is computed by taking as input a label, analyzing its real world semantics, and returning as output a *concept denoted by the label*, C_L. For example, when I write $C_{Consumer\_Electronics}$ I mean the concept describing ″all the documents which are (about) consumer electronics″. Notice that by writing $C_{Consumer\_Electronics}$ I move from the meaningless label *Consumer_Electronics* to the concept, label with semantics $C_{Consumer\_Electronics}$, which the given label denotes. Meanings of labels are defined via their senses in WordNet system [26][4].

Step 2. At this step the meaning of the positions that the labels at nodes have in a graph is analyzed. By doing this concepts denoted by labels computed in step 1 are extended to *concepts at nodes*, $\boldsymbol{C_N}$. An extension of concepts denoted by labels is needed to capture the knowledge residing in a structure of a graph in order to define a context in which the given concept denoted by a label occurs. For example, when I write $\boldsymbol{C_{Consumer\_Electronics}}$ I mean the concept describing ″all the documents which are (about) consumer electronics and are also on sale″. Speaking formally, it is defined as an intersection of concepts denoted by labels located above the given node, including the node itself:

$$\boldsymbol{C_{Consumer\_Electronics}} = C_{Sale} \sqcap C_{Consumer\_Electronics}$$

Now the problem of semantic matching can be defined precisely: given two graphs *G1, G2* compute the $N1 \times N2$ mapping elements $<ID_{ij}, n1_i, n2_j, R'>$, with $n1_i \in G1$, $i=1,...,N1$, $n2_j \in G2$, $j=1,...,N2$ and R' the strongest semantic relation holding between the concepts at nodes $n1_i$, $n2_j$. The strongest semantic relation always exists since, when holding together, more general and less general are equivalent to equivalence.

Let us consider the mapping element existing between the node with label *Consumer_Electronics* in A1 and the node with label *Sale* in A2. In A1, the node with label *Consumer_Electronics* is below the node with label *Sale* and, therefore, its concept stands for the set of all documents which are (about) consumer electronics and which are on sale. Building a similar argument for the node with label *Sale* in A2, and assuming that *Sale* and *Consumer_Electronics* in A1 and *Sale* and *Consumer_Electronics* in A2 refer to the same concepts respectively, we can therefore conclude that the concepts at the two nodes have the same extension, namely

$$<ID_{22}, \boldsymbol{C_{Consumer\_Electronics}}, \boldsymbol{C_{Sale,}} =>$$

Considering also the mapping of the node with label *Consumer_Electronics* in A1 to the nodes with labels *Consumer_Electronics* and *Cameras_and_Photo* in A2 we have the following mapping elements:

$$<ID_{21}, \boldsymbol{C_{Consumer\_Electronics}}, \boldsymbol{C_{Consumer\_Electronics}}, \sqsubseteq>$$
$$<ID_{24}, \boldsymbol{C_{Consumer\_Electronics}}, \boldsymbol{C_{Cameras\_and\ Photo,}}, \sqsupseteq>$$

Semantic relations in the above mapping elements are computed in terms of bindings between WordNet senses using the semantic matching algorithm [15].

[4] Extensions to the work would also take Description Logics (DL) representations of the classes as input such as full OWL ontologies [22]. This topic is beyond the scope of my PhD thesis.

4.2 Semantic Matching as a Validity Problem

The key idea behind the semantic matching algorithm is to translate the matching problem, into a logical (propositional in the above case) formula and then to check it for its validity.

A translation encodes concepts at labels/nodes using a logical propositional language where atomic formulas are atomic concepts, written as single words, and complex formulas are obtained by combining atomic concepts using the connectives of set theory. The semantics of this language are the obvious set-theoretic semantics. The semantic relations are also translated into propositional connectives, namely: equivalence into equivalence, more general and less general into implication, and mismatch into negation of the conjunction.

It is necessary to prove that the following formula:

$$Context \rightarrow rel(\boldsymbol{C1_i}, \boldsymbol{C2_j})$$

is valid; where $\boldsymbol{C1_i}$ is the concept of node *i* in graph 1, $\boldsymbol{C2_j}$ is the concept of node *j* in graph 2, *rel* is the semantic relation (suitably translated into a propositional connective) possibly holding between $\boldsymbol{C1_i}$ and, $\boldsymbol{C2_j}$, assuming, as background theory (*context* [17]), all that is possible to infer about the relations holding among the concepts of the labels of the two graphs. Context is the conjunction of all the relations (suitably translated) between concepts of labels mentioned in $\boldsymbol{C1_i}$ and $\boldsymbol{C2_j}$.

For example, let us consider a propositional formula being built for the problem of matching the node with label *Consumer_Electronics* in A1 and the node with label *Sale* in A2. As from above, the propositional formula to test, for instance, if $\boldsymbol{C1_{Consumer\_Electronics}}$ is equivalent to $\boldsymbol{C2_{Sale}}$ is as follows:

$$(C1_{Sale} \leftrightarrow C2_{Sale})) \wedge ((C1_{Consumer\_Electronics} \leftrightarrow C2_{Consumer\_Electronics})) \rightarrow$$
$$\rightarrow ((C1_{Sale} \wedge C1_{Consumer\_Electronics}) \leftrightarrow (C2_{Consumer\_Electronics} \wedge C2_{Sale}))$$

To prove that $(C1_{Sale} \leftrightarrow C2_{Sale})) \wedge ((C1_{Consumer\_Electronics} \leftrightarrow C2_{Consumer\_Electronics})) \rightarrow ((C1_{Sale} \wedge C1_{Consumer\_Electronics}) \leftrightarrow (C2_{Consumer\_Electronics} \wedge C2_{Sale}))$ is valid (a formula is valid iff its negation is unsatisfiable), it is necessary to prove that its negation is unsatisfiabile. In order to do this it is used a propositional satisfiability engine. Thus, SAT run on the following formula $(C1_{Sale} \leftrightarrow C2_{Sale})) \wedge ((C1_{Consumer\_Electronics} \leftrightarrow C2_{Consumer\_Electronics})) \wedge \neg ((C1_{Sale} \wedge C1_{Consumer\_Electronics}) \leftrightarrow (C2_{Consumer\_Electronics} \wedge C2_{Sale}))$ should fail. A quick analysis shows that SAT will return FALSE[5]. Therefore, '=' relation holds between the nodes under consideration. The relation found is the strongest and therefore we don't need to test $\boldsymbol{C1_{Consumer\_Electronics}}$ and $\boldsymbol{C2_{Sale}}$ for any other semantic relations. The above described approach has been implemented within the *S-Match* system [15][6].

5 Goals of the Thesis Work

In the context of the research I am aiming, I will tackle the following issues that need to be addressed in order to make the *S-match* system a *generic* and *general-purpose* matcher:

- *Extend the semantic matching approach to allow handling graphs*. At present, the semantic matching approach and the *S-Match* system is limited to the cases of tree-like structures. Thus, it only computes mappings among simple XML schemas without ID/IDREF pairs in DTDs or key-keyref pairs in XSD; taxonomies or concept hierarchies without shared elements, where all the links have only *containment* semantics. In order to eliminate the above stated limitations, and also to take into account, in the case of relational databases: attributes and referential integrity constraints (including cycles), or in the case of ontologies: classes connected by multiple inheritance links, slots, facets and axioms its necessary to extend the semantic matching approach to allow handling graphs, encoding those data and conceptual models.

[5] See a detailed discussion on explanations of why a particular semantic relation is found or is not found in [31].

[6] The *S-Match* system presents 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. *S-Match* was tested against three state of the art matching systems: COMA [11], Cupid [20], and SF [24] as implemented in Rondo [23]. The results, though preliminary, look promising, in particular for what concerns precision and recall, see [15]. Source files and description of the schemas tested can be found at the project web-site, experiments section: http://www.dit.unitn.it/~accord/.

The Semantic Web proposes the markup of content on the web using ontologies [6]. At present, OWL [22] is believed to be the most promising ontology language. Therefore, a particular attention will be paid to matching ontologies encoded in OWL format.

- *Extend the semantic matching algorithm for computing mappings between graphs*. From a computational point of view, it is necessary to devise new mechanisms calculating concepts denoted by labels and concepts at nodes when dealing with graphs. The semantic matching algorithm calculates mappings between the nodes of the graphs, not links, then when we take in to account, for example, binary relations, like slots in the case of ontologies or referential integrity constraints in the case of databases, we, therefore, need to represent them as nodes. One solution is to interpret nodes standing for relations as *join views*, and hence this could be a possible way of computing concepts at such nodes.

  The other objective is to choose an appropriate formal structure-level technique(s), e.g., modal SAT engine(s) (or DL reasoners), checking satisfiability of the logical formulas generated by graph matching problems.

- *Develop a theory of iterative semantic matching in order to improve quality and efficiency of matching via iterations*. Iterations can be preformed re-running SAT. We may 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. 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 5.

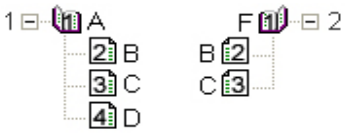

Fig. 5. Not good enough answer

Suppose that we have found out that $C1_2 \sqcap C2_2 \neq \varnothing$, and that we want to improve this result. Suppose that an oracle tells us that $C1_A = C2_F \sqcup C2_G$. In this case the graph on the left in Figure 5 can be transformed into the two graphs, see Figure 6.

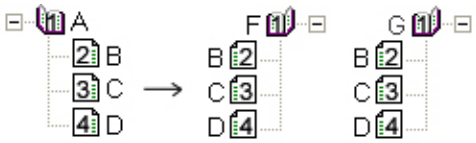

Fig. 6. Extraction of additional semantic information

After this additional analysis we can infer that $C1_2 = C2_2$. As a particular interesting case, consider the following situation, see Figure 7.

Fig. 7. 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. 8. Extraction of additional semantic information. The Result.

Another motivation for multiple iterations is to use the result of a previous match in order to speed up the search of new matches, see [16] for an example.

- *Implement the above stated goals within the S-Match system and conduct the experimental evaluation in different initial settings*. From a technical perspective, development of a generic matcher aims at handling different data and conceptual models and being general-purpose, e.g., serve for many application domains. The models to be matched can be relational schemas, XML schemas, OWL ontologies, or any other. Adding new formal structure-level techniques will increase *flexibility* of the system. Implementing iterations will make the system more *efficient* and *interactive,* providing users with prompts and suggestions when they are asked for a feedback.

  Experimental evaluation should be conducted against the other schema-based state of the art systems and estimate qualitative and quantitative characteristics of the matching results, e.g., precision, recall, overall and time needed to produce mappings. Test schemas/ontologies will be of a different complexity, in order to see how the proposed matching solution may *scale*, and from different application domains. Experimental evaluation should give empirical evidence that the system is generic, general-purpose and produces high-quality results in terms of precision and recall indicators.

Acknowledgments

I am grateful to Fausto Giunchiglia, my supervisor, for many lessons on how to do research and write articles, for being very supportive in my work, for guiding my entrance into AI domain and life in general.

References

1. Beneventano D., S. Bergamaschi, S. Lodi, and C. Sartori: Consistency checking in complex object database schemata with integrity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 10(4):576-598, 1998.
2. Bergamaschi S., Castano S., Vincini M.: Semantic Integration of Semistructured and Structured Data Sources. *SIGMOD Record*, 28(1): 54-59, 1999.
3. Bergamaschi S., Beneventano D., Castano S., Vincini M.: MOMIS: An Intelligent System for the Integration of Semistructured and Structured Data. Technical report, T3-R07, University of Modena, 1998.
4. Berlin, J., Motro A.: Autoplex: Automated Discovery of Content for Virtual Databases. Proceedings of *CoopIS* (2001) 108–122.
5. Berlin, J., Motro A.: Database Schema Matching Using Machine Learning with Feature Selection. Proceedings of *CAiSE* (2002) 452-466.
6. Berners-Lee T., Hendler J., Lassila O.: The Semantic Web. *Scientific American*, May 2001.
7. Borgida A., Brachman, R., McGuinness D., Resnick L.: CLASSIC: A structural data model for objects. Proceedings of *ACM SIGMOD*, (1989) 58-67.
8. Bouquet P., Serafini L., Zanobini S.: Semantic Coordination: A new approach and an application. Proceedings of *ISWC*, (2003) 130-145.
9. Castano S., De Antonellis V., De Capitani di Vimercati S.: Global Viewing of Heterogeneous Data Sources. *IEEE Transactions on Knowledge and Data Engineering*, 13(2): 277-297, 2001.
10. Do H.H., Melnik S., Rahm E.: Comparison of schema matching evaluations. Proceedings of workshop on *Web and Databases*, (2002).
11. Do H.H., Rahm E.: COMA – A System for Flexible Combination of Schema Matching Approaches. Proceedings of *VLDB*, (2002) 610-621.
12. Doan A., Madhavan J., Dhamankar R., Domingos P., Halvey A.: Learning to map ontologies on the semantic web. *VLDB Journal, Special Issue on the Semantic Web*, 2003 (to appear).
13. Doan, A.H., Domingos P., Halevy A.: Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. Proceedings of *SIGMOD*, (2001) 509-520.
14. Euzenat J., Valtchev P.: An integrative proximity measure for ontology alignment. Proceedings of *Semantic Integration* workshop at *ISWC*, (2003).
15. Giunchiglia F., Shvaiko P., Yatskevich M.: S-Match: an algorithm and an implementation of semantic matching. Proceedings of *ESWS*, (2004) 61-75.
16. Giunchiglia F., Shvaiko P.: Semantic matching. *The Knowledge Engineering Review Journal*, 18(3):265-280, 2004. Short versions: Proceedings of *Ontologies and Distributed Systems* workshop at *IJCAI* and *Semantic Integration* workshop at *ISWC*, 2003.
17. Giunchiglia F.: Contextual reasoning. *Epistemologia, special issue on "I Linguaggi e le Macchine"*, vol. XVI: 345-364, 1993.
18. Kang J., Naughton J.F.: On schema matching with opaque column names and data values. Proceedings of *SIGMOD*, (2003) 205-216.
19. Li, W.S., Clifton C.: Semantic Integration in Heterogeneous Databases Using Neural Networks. Proceedings of *VLDB,* (1994) 1-12.
20. Madhavan J., Bernstein P., Rahm E.: Generic schema matching with Cupid. Proceedings of *VLDB*, (2001) 49-58.

21. McGuinness D.L., Fikes R., Rice J., Wilder S.: An environment for merging and testing large ontologies. Proceedings of *KR*, (2000) 483-493.
22. McGuinness D.L., van Harmelen F. 2003: OWL Web Ontology Language Overview. Technical report, World Wide Web Consortium (W3C), December 9. Proposed Recommendation.
23. Melnik S., Rahm E., Bernstein P.: Rondo: A programming platform for generic model management. Proceedings of SIGMOD, (2003) 193-204.
24. Melnik, S.,Garcia-Molina H., Rahm E.: Similarity Flooding: A Versatile Graph Matching Algorithm. Proceedings of *ICDE*, (2002) 117-128.
25. Mena E., Kashyap V.,Sheth A., Illarramendi A.: Observer: An approach for query processing in global information systems based on interoperability between pre-existing ontologies. Proceedings of *CoopIS*, (1996) 14-25.
26. Miller, A.G.: Wordnet: A lexical database for English. *Communications of the ACM*, 38(11): 39-41, 1995.
27. Noy N. and Musen M. A.: Anchor-PROMPT: Using Non-Local Context for Semantic Matching. Proceedings of *IJCAI* workshop *on Ontologies and Information Sharing*, (2001) 63-70.
28. Open Information Model, Version 1.0, Meta Data Coalition. http://mdcinfo/oim/oim10.html, August 1999.
29. Palopoli L., Terracina G., Ursino D.: The System DIKE: Towards the Semi-Automatic Synthesis of Cooperative Information Systems and Data Warehouses. *ADBIS-DASFAA*, Matfyzpress, (2000) 108-117.
30. Rahm E., Bernstein P.: A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4): 334-350, 2001.
31. Shvaiko P., Giunchiglia F., Pinheiro da Silva P., McGuinness D. L.: Web explanations for semantic heterogeneity discovery. Technical Report, DIT-04-012, University of Trento. Also, Technical Report, KSL-04-02, Stanford University, 2004.
32. Xu L., Embley D.W.: Using domain ontologies to discover direct and indirect matches for schema elements. Proceedings of *Semantic Integration* workshop at *ISWC*, (2003).