# Web Explanations for Semantic Heterogeneity Discovery

Pavel Shvaiko[1], Fausto Giunchiglia[1],
Paulo Pinheiro da Silva[2], and Deborah L. McGuinness[2]

[1] University of Trento, Povo, Trento, Italy
{pavel, fausto}@dit.unitn.it
[2] Stanford University, Stanford, USA
{dlm, pp}@ksl.stanford.edu

**Abstract.** Managing semantic heterogeneity is a complex task. One solution involves matching like terms to each other. We view *Match* as an operator that takes two graph-like structures (e.g., concept hierarchies or ontologies) and returns a mapping between the nodes of the graphs that correspond semantically to each other. While some state of the art matching systems may produce effective mappings, these mappings may not be intuitively obvious to human users. In order for users to *trust* the mappings, and thus, use them, they need information about them (e.g., they need access to the sources that were used to determine semantic correspondences between terms). In this paper we describe how a matching system can explain its answers using the Inference Web (IW) infrastructure thus making the matching process *transparent*. The proposed solution is based on the assumption that mappings are computed by logical reasoning. There, *S-Match*, a *semantic matching* system, produces proofs and explanations for mappings it has discovered.

## 1 Introduction

The progress of information and communication technologies, and in particular of the Web, has made available a huge amount of disparate information. The number of different information resources is growing significantly, and therefore, the problem of managing semantic heterogeneity is increasing, see, for instance, [34]. 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 interrelationships. These might be database schemas, concept hierarchies, ontologies.

We view *Match* as one of the key operators for enabling semantic applications since it takes two graph-like structures and produces a mapping between the nodes of the graphs that correspond semantically to each other. Matching, however, requires explanations because mappings between terms are not always intuitively obvious to human users. In fact, if Semantic Web users are going to trust the fact that two terms may have the same meaning, then they need to understand the reasons leading a matching system to produce such a result. Expla-

A. Gómez-Pérez and J. Euzenat (Eds.): ESWC 2005, LNCS 3532, pp. 303–317, 2005.

nations are also useful, when matching applications with hundreds or thousands of nodes, since in these cases automatic matching solutions will find a number of plausible mappings, thus some human effort (e.g., database/knowledge base administrators who need to perform some rationalization of the mapping suggestions) is inevitable.

We focus on *semantic matching* as proposed in [11], and implemented within the *S-Match* [12] system. This matching solution is based on the assumption that mappings are computed by logical reasoning. We have extended *S-Match* to use the Inference Web infrastructure [22] and its Proof Markup Language (PML)[5]. Thus, with the help of IW tools and exploiting PML properties, meaningful fragments of the *S-Match* proofs can be loaded on demand. Users can browse an entire proof or they can restrict their view and refer only to specific, relevant parts of proofs. They can ask for provenance information related to proof elements (e.g., the origin of the terms in the proofs, the authors of the ontologies). Therefore, they can make informed decisions about the mappings.

The rest of this paper proceeds as follows. In Section 2, via an example, we discuss semantic matching approach as implemented within the *S-Match* system. In Section 3 we describe the Inference Web infrastructure. Using the example introduced in Section 2, in Section 4, we present how the Inference Web explanations increase user understanding of the *S-Match* answers. Section 5 presents an experimental study that addresses the scaling of the explanation techniques. Section 6 discusses the related work and Section 7 summarizes the contributions of the paper.

## 2 Semantic Matching

The semantic matching approach is based on two ideas. The first idea is that we calculate mappings between schema/ontology elements by computing *semantic relations* (e.g., equivalence, more general), instead of computing coefficient rating match quality in the [0,1] range, as is the case in other approaches, see, for example, [9, 17, 19]. The second idea is that we determine semantic relations by analyzing the *meaning* (concepts, not labels) which is codified in the elements and the structures of schemas/ontologies. In particular, labels at nodes, written in natural language, are translated into propositional formulas which explicitly codify the label's intended meaning. This allows us to translate the matching problem into a propositional unsatisfiability problem.

We call the *concept of a label* the propositional formula which stands for the set of documents that one would classify under a label it encodes. We call the *concept at a node* the propositional formula which represents the set of documents which one would classify under a node, given that it has a certain label and that it is in a certain position in a tree.

Possible semantic relations that *S-Match* can discover between the concepts of nodes of the two schemas/ontologies are: *equivalence* ($=$); *more general* ($\sqsupseteq$); *less general* ($\sqsubseteq$); *disjointness* ($\perp$). When none of the relations holds, the special *idk* (I dont know) relation is returned. The relations are ordered according to

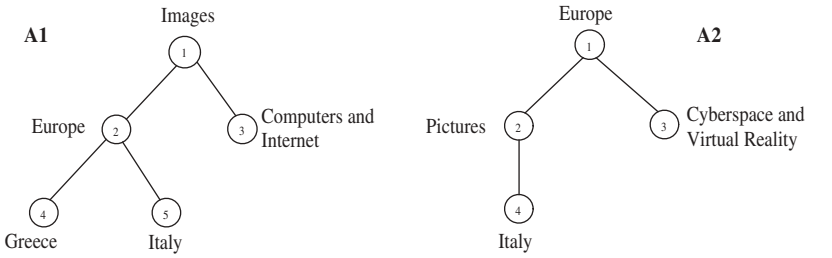


**Fig. 1.** Simple catalog matching problem

decreasing binding strength, i.e., from the strongest (=) to the weakest (*idk*), with more general and less general relations having equal binding power. The semantics of the above relations are the obvious set-theoretic semantics.

A *mapping element* is a 4-tuple $\langle ID_{ij},\ n1_i,\ n2_j,\ R\rangle$, $i$=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 semantic relation which may hold between the concepts of nodes $n1_i$ and $n2_j$. *Semantic matching* can then be defined as the following problem: given two graphs G1, G2 compute the N1 × N2 mapping elements $\langle ID_{ij},\ n1_i,\ n2_j,\ R'\rangle$, 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 of nodes $n1_i, n2_j$. The strongest semantic relation always exists since, when holding together, more general and less general are equivalent to equivalence. *S-Match* is *schema-based*, and therefore, it considers only intentional information, not instance data. In the current version it is limited to the tree-like structures, e.g., taxonomies, or simple XML schemas with attributes.

We concentrate on class matching and motivate the problem by the simple catalog matching example shown in Figure 1. Suppose an agent wants to exchange/search for documents with another agent. The documents of both agents are stored in catalogs according to class hierarchies A1 and A2 respectively. *S-Match* takes as input these hierarchies and computes as output a set of mapping elements in four macro steps. The first two steps represent the pre-processing phase, while the third and the fourth steps correspond to *element level* and *structure level* matching respectively.

**Step 1. For all labels *L* in the two trees, compute *concepts of labels***. We think of labels at nodes as concise descriptions of the data that is stored under the nodes. We compute the meaning of a label at a node by taking as input a label, by analyzing its (real world) semantics, and by returning as output a concept of the label, $C_L$. For example, when we write $C_{Pictures}$ we mean the concept describing all the documents which are (about) pictures. Notice, that by writing $C_{Pictures}$ we move from the natural language ambiguous label *Pictures* to the concept $C_{Pictures}$, which the given label denotes.

Technically, concepts at labels are encoded as propositional logical formulas, where atomic formulas are WordNet [25] *senses* (possible meanings) of single words, and complex formulas are obtained by combining atomic concepts using the connectives of set theory and set-theoretic semantics. For example, $C_{Pictures} = \langle picture, senses_{WN\#11} \rangle$, where $senses_{WN\#11}$ is taken to be disjunction of the eleven senses that WordNet attaches to *pictures*. The process of extraction of logical formulas from natural language labels is described in detail in [20].

**Step 2. For all nodes $N$ in the two trees, compute *concepts of nodes*.** In this step we analyze the meaning of the positions that the labels at nodes have in a tree. By doing this we *extend* concepts of labels to *concepts of nodes*, $\boldsymbol{C_N}$. This is required to capture the knowledge residing in the structure of a graph, namely the context in which the given concept at label occurs. For example, in A2, when we write $\boldsymbol{C_2}$ we mean the concept describing all the documents which are (about) pictures and which are also about Europe. Thus, in Figure 1, following the classification link semantics, which is an *access criterion* [16], the logical formula for a concept at node is defined as a conjunction of concepts of labels located above the given node, including the node itself, for example, in A2, $\boldsymbol{C_2} = C_{Europe} \sqcap C_{Pictures}$.

**Step 3. For all pairs of labels in the two trees, compute *relations* among *concepts of labels*.** Relations between concepts of labels are computed with the help of a library of element level semantic matchers [13]. These matchers take as input two concepts of labels and produce as output a semantic relation between them. For example, in Figure 1, $C_{Images}$ can be found equivalent to $C_{Pictures}$. In fact, according to WordNet, *images* and *pictures* are synonyms. Notice that in WordNet *pictures* has 11 senses and *images* has 8 senses. We use some sense filtering techniques to discard the irrelevant senses for the given context, see [20] for details.

**Step 4. For all pairs of nodes in the two trees, compute *relations* among *concepts of nodes*.** *S-Match* decomposes the tree matching problem into the set of node matching problems. Then, each node matching problem, namely pairs of nodes with possible relations between them, is translated into a propositional formula. The semantic relations are translated into propositional connectives as follows: equivalence into equivalence, more general and less general into implication, and disjointness into negation of the conjunction. As from [11], we have to prove that the following formula:

$$Axioms \longrightarrow rel(\mathbf{C1_i}, \mathbf{C2_j}) \tag{1}$$

is valid, namely that it is *true* for all the truth assignments of all the propositional variables occurring in it. In (1), *Axioms* is the conjunction of all the relations between concepts of labels computed in step 3. $\boldsymbol{C1_i}$ is the propositional formula encoding concept at node $i$ in tree 1, $\boldsymbol{C2_j}$ is the propositional formula encoding concept at node $j$ in tree 2, *rel* is the semantic relation that we want to prove holding between $\boldsymbol{C1_i}$ and $\boldsymbol{C2_j}$.

From the example in Figure 1, trying to prove that *Europe* in A1 is equivalent to *Pictures* in A2, requires constructing formula (2).

$$\underbrace{((C1_{Images} \leftrightarrow C2_{Pictures}) \wedge (C1_{Europe} \leftrightarrow C2_{Europe}))}_{Axioms} \rightarrow (\underbrace{(C1_{Images} \wedge C1_{Europe})}_{\mathbf{C1_2}} \leftrightarrow \underbrace{(C2_{Europe} \wedge C2_{Pictures})}_{\mathbf{C2_2}}) \quad (2)$$

The algorithm checks for the validity of formula (2) by proving that its negation is unsatisfiable. For this purpose, our implementation uses a propositional satisfiability (SAT) engine, in particular JSAT [2]. In this example, the negated formula is unsatisfiable, thus the equivalence relation holds between the nodes under consideration. Since this is the strongest relation, no additional checks need to be made and the *S-Match* algorithm terminates and concludes that documents stored under *Pictures* in A2 are an appropriate match for documents stored under *Europe* in A1, i.e., $\langle ID_{22}, \boldsymbol{C1_2}, \boldsymbol{C2_2}, = \rangle$.

## 3 Inference Web

Inference Web enables applications to generate portable and distributed explanations for any of their answers. In order to explain mappings produced by *S-Match* and thereby increase the trust level of its users, we need to provide information about background theories (for instance, WordNet), and the JSAT manipulations of propositional formulas.

Figure 2 presents an abstract and partial view of the IW framework as used by the *S-Match* system. In order to use IW to provide explanations, question answering systems need to have their reasoners produce proofs of their answers in PML, publish those proofs on the web, and provide IW with a pointer to the last step in the proof. IW also has a registry [23] of meta-information about proof elements, such as sources (e.g., publications, ontologies), inference engines and their rules. In the case of *S-Match*, the IW repository contains meta information about WordNet and JSAT.

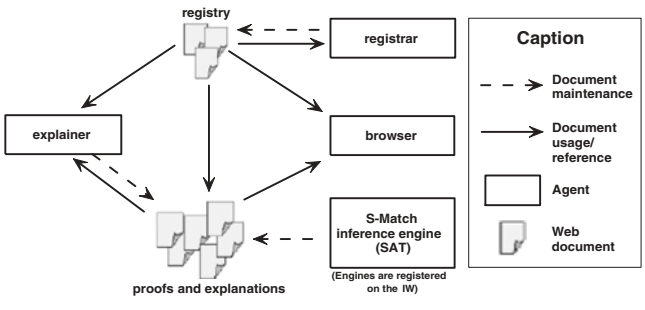


**Fig. 2.** Inference Web framework overview

In the Inference Web, proof and explanation documents are formatted in PML and are composed of PML *node sets*. Each node set represents a step in a proof whose conclusion is justified by any of a set of inference steps associated with a node set. Also, node sets are subclasses of the W3C's OWL Class [33] and they are the building blocks of OWL documents describing proofs and explanations for application answers published on the Web.

The *IW Browser* is used to present proofs and explanations. Exploiting PML properties, meaningful fragments of the *S-Match* proofs can be loaded on demand. Users can browse an entire proof or they can limit their view and refer only to specific, relevant parts of proofs since each node set has its own URI that can be used as an entry point for proofs and proof fragments.

## 4 Producing Explanations

A default explanation of mappings the *S-Match* system produces is a short, natural language, high-level explanation without any technical details. It is designed to be intuitive and understandable by ordinary users.

Let us recall the catalog matching example. Suppose that agent A2 is interested in knowing why *S-Match* suggested a set of documents stored under the node with label *Europe* in A1 as the result to the query - *"find european pictures"*. A default explanation is presented in Figure 3.

From the explanation in Figure 3, users may learn that *Images* in A1 and *Pictures* in A2 are equivalent words, i.e., they can be interchanged, in the context of the query. Also, users may learn that *Europe* in A1 denotes the same concept as *Europe (European)* in A2. Therefore, they can conclude that *Images of Europe* means the same thing as *European Pictures*. Future work includes optional pruning of statements containing information that two concepts are identical.

However, users may not be satisfied with this level of explanation. Let us therefore discuss how they can investigate the details of the matching process by exploiting more verbose explanations. We have implemented two kinds of verbose explanations: background knowledge explanations and logical reasoning explanations. Let us consider them in turn.

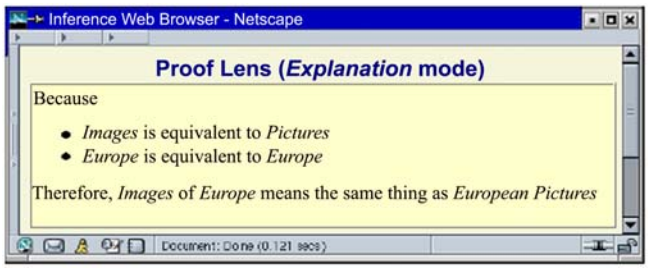


**Fig. 3.** An explanation in English

### 4.1 Explaining Background Knowledge

Suppose that the agent wants to see the sources of background knowledge used in order to determine the mapping. For example, which applications, publications, other sources, have been used to determine that *Images* is equivalent to *Pictures*. Figure 4 presents the source metadata for the default explanation of Figure 3.

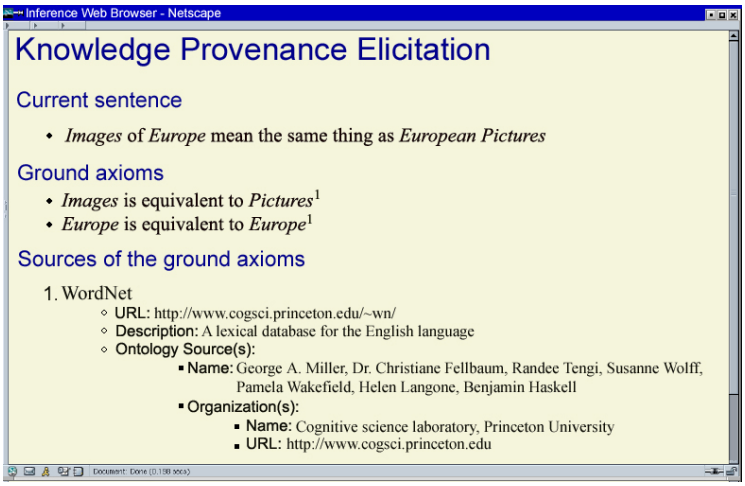


**Fig. 4.** Source metadata information

In this case, both (all) the ground sentences used in the *S-Match* proof came from WordNet. Using WorldNet, *S-Match* learned that the first sense of the word pictures is a synonym to the second sense of the word images. Therefore, *S-Match* can conclude that these two words are equivalent words in the context of the answer. The meta information about WordNet from the IW Registry is also presented in Figure 4 as *sources of the ground axioms*. Further examples of explanations include: providing meta information about the *S-Match* library of element-level matchers [13], i.e., those which are based not only on WordNet, the order in which the matchers are used, and so on.

### 4.2 Explaining Logical Reasoning

A more complex explanation may be required if users are not familiar with or do not trust inference engine(s) embedded in a matching system. As the Web starts to rely more on information manipulations (instead of simply information retrieval), explanations of embedded manipulation/inference engines become more important. In the current version of *S-Match*, a propositional satisfiability engine is used, more precisely, the Davis-Putnam-Longemann-Loveland (DPLL) procedure [6, 7] as implemented in JSAT [2].

The task of a SAT engine is to find an assignment $\mu \in \{\top, \bot\}$ to atoms of a propositional formula $\varphi$ such that $\varphi$ evaluates to *true*. $\varphi$ is *satisfiable* iff $\mu \models \varphi$

for some $\mu$. If $\mu$ does not exist, $\varphi$ is *unsatisfiable*. A *literal* is a propositional atom, or its negation. A *clause* is a disjunction of one or more literals. $\varphi$ is said to be in conjunctive normal form (CNF) iff it is a conjunction of disjunctions of literals. The basic DPLL procedure recursively implements the three rules: *unit resolution*, *pure literal* and *split* [6, 7].

Let $l$ be a literal and $\varphi$ a propositional formula in CNF. A clause is called a *unit clause* iff it has exactly one unassigned literal. *Unit resolution* is an application of *resolution* to a unit clause.

$$unit\ resolution : \frac{\varphi \wedge \{l\}}{\varphi[l \mid \top]}$$

$l$ is called a *pure literal* in $\varphi$ iff it occurs in $\varphi$ only positively or negatively. *Pure literal* removes all clauses in which pure literals occur.

$$pure\ literal : \frac{\varphi}{\varphi[l \mid \top]}$$

*Split rule* performs branching first on truth values of literals then on their false values, iff the above two rules (deterministic choices) cannot be applied.

$$split : \frac{\varphi}{\varphi[l \mid \top] \qquad \varphi[l \mid \bot]}$$

Usually performance of SAT engines is not a concern for producing proofs. Thus, we have modified the JSAT DPLL procedure and enabled it to generate proofs. Next, we discuss the IW proofs and explanations of the *unit resolution* rule in detail. In the current version, the *pure literal* and *split* rules are explained in the same manner as the *unit resolution* rule.

**Unit Resolution Rule.** Let us consider the propositional formula standing for the problem of testing if the concept at node 2 in A1 is less general than the concept at node 2 in A2. In the following, to simplify the presentation we use a label as a placeholder of a concept the given label denotes. The propositional formula encoding the above stated matching problem is as follows:

$$\begin{aligned}&((Images \leftrightarrow Pictures) \wedge (Europe \leftrightarrow Europe)) \wedge \neg \\ &((Europe \wedge Images) \rightarrow (Pictures \wedge Europe))\end{aligned} \tag{3}$$

An intuitive reading of (3) is "is there any situation such that the concept *Images of Europe* is less general than the concept *European Pictures* assuming that *Images* and *Pictures* denote the same concept?". The IW proof of the fact that this is not the case is shown in Figure 5. Notice that, since the DPLL procedure of JSAT handles only CNF formulas, in Figure 5, we show the CNF equivalent of formula (3).

From the explanation in Figure 5, users may learn that the IW proof of the fact that the concept at node 2 in A1 is less general than the concept at node 2 in A2 requires 4 steps and at each proof step (excepting the first one, which is a problem statement) the *unit resolution* rule is applied. Also, users may learn

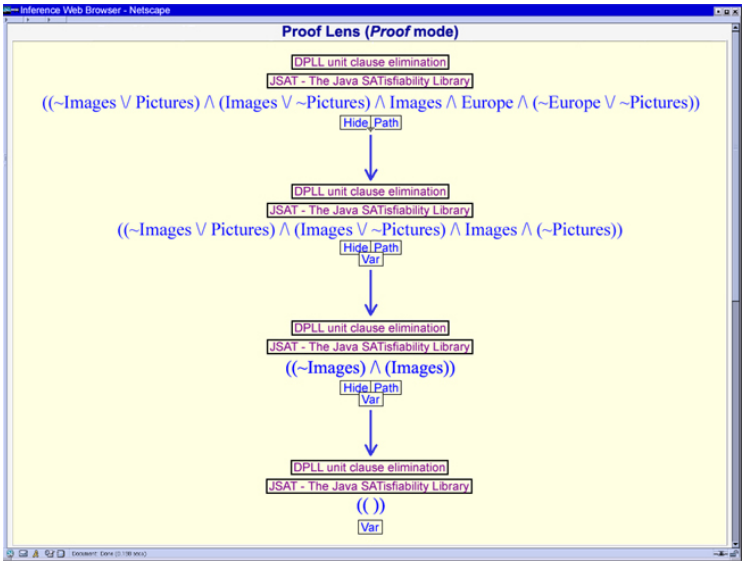


**Fig. 5.** A graphical explanation of the unit clause rule

the assumptions that are made by JSAT. For example, at the second step the DPLL procedure of JSAT assigns the truth value to (all instances of) the atom *Europe*, therefore making an assumption that there is a model, where what an agent says about Europe is always true. According to the *unit resolution* rule, then the atom *Europe* should be stroked out from the input sentence (and, hence it does not appear in the sentence of the step 2).

The explanation of Figure 5 represents some technical details (only the less generality test) of the default explanation in Figure 3. This type of explanations is the most verbose. It assumes (even if the graphical representation of a decision tree is quite intuitive) that the matching system users have some background knowledge in logics and SAT. However, if they do not have it, they have a possibility to learn it by following the publications mentioned in the source metadata information of the DPLL *unit resolution* rule and JSAT (by clicking the *DPLL unit clause elimination* and the *JSAT-The Java SATisfiability Library* buttons respectively).

Two further notes are to be made with respect to the *split* rule. The first is that, it is applied when we need to reason by case distinction, for example, when matching $\boldsymbol{C1_3}$ and $\boldsymbol{C2_3}$. The second note is that, in the case of a satisfiable result, only a path of a decision tree standing for a successful assignment is represented. In the case of an unsatisfiable result a full decision tree is reported.

## 5 Experimental Study

The main goal of the experiments being conducted is to obtain a vision of how the *S-Match* explanations potentially scale to the requirements of the Semantic Web, providing meaningful and adjustable answers in real time.

The semantic (node) matching problem is a CO-NP complete problem, since it is reduced to the validity problem (a formula is valid iff its negation is unsatisfiable) for the propositional calculus. Resolving this class of problems requires exponential time and exponentially long proof logs. However, in all the examples we have done so far proofs are not too long and seem of length polynomial in the length of the input clause. As a matter of fact, [14] shows, that when we have conjunctive concepts at nodes (e.g., $Images \wedge Europe$), these matching tasks can be resolved by the basic DPLL procedure in polynomial time; while when we have full proposition concepts at nodes (e.g., $Images \wedge (Computers \vee Internet)$), the length of the original formula can be exponentially reduced by structure preserving transformations.

In our experiments we have used three test cases: the simple catalog matching problem, presented in the paper, one example from academic and one example from business domains. The business example describes two company profiles: a standard one (mini) and Yahoo Finance (mini). The academic example describes courses taught at Cornell University (mini) and at the University of Washington (mini). Table 1 provides some indicators of the complexity of the test cases[1].

**Table 1.** Some indicators of the complexity of the test cases

| | **Images vs. Europe** | **Yahoo(mini) vs. Standard(mini)** | **Cornell(mini) vs. Washington(mini)** |
|---|---|---|---|
| **#nodes** | 4/5 | 10/16 | 34/39 |
| **max depth** | 2/2 | 2/2 | 3/3 |
| **#leaf nodes** | 2/2 | 7/13 | 28/31 |

We focus on indicators characterizing explanations of mappings. The analysis of the quality of mappings is beyond scope of this paper[2]. In the experimental study we have used the following indicators:

- Number of mapping elements determined by *S-Match* for a pair of schemas/ ontologies. As follows from the definition of semantic matching, this number should be N1 × N2, where N1 is the number of nodes in the first schema/ontology, N2 is the number of nodes in the second schema/ontology.
- Number of steps in a proof of a single mapping element. This indicator represents the number of PML node sets are to be created in the proof.
- Time needed to produce a proof of a single mapping element. This indicator estimates how fast the modified JSAT in producing IW proofs for a particular task.
- Time needed to produce a proof of all mappings determined by *S-Match* for a pair of schemas/ontologies.

[1] Source files and description of the test cases can be found at http://www.dit.unitn.it/~accord/, experiments section.

[2] Analysis of the quality of mappings produced by *S-Match* and a comparative evaluation against state of the art systems, such as COMA [17], Cupid [19], and Rondo [24] can be found in [12].

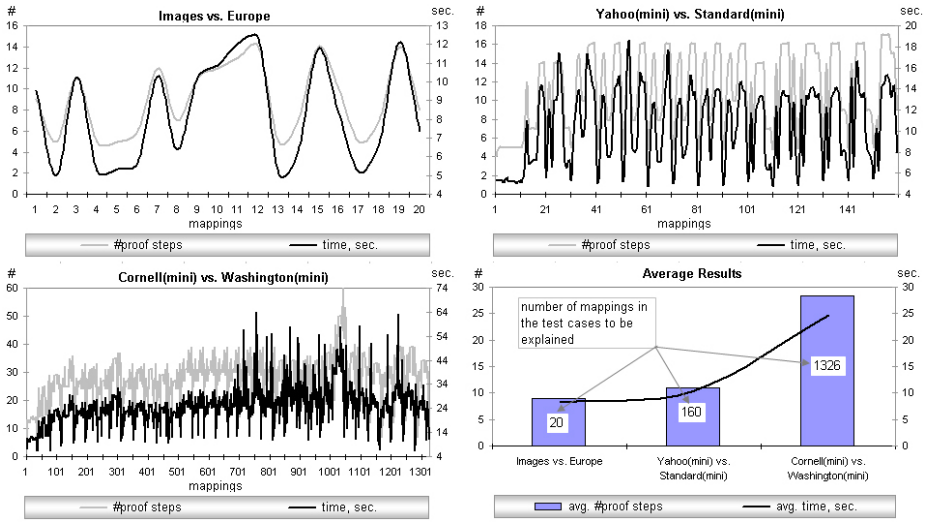


**Fig. 6.** Experimental Results

In order to conduct tests in a real environment, we used the IW web service of KSL at Stanford University (on a P4-2.8GHz, 1.5Gb of RAM, Linux, Tomcat web server) to generate proofs in PML, while the modified JSAT version was run at the University of Trento (on a P4-1.7GHz, 256 MB of RAM, Windows XP). All the tests were performed without any optimizations: for each single task submitted to JSAT, the IW web service was invoked, no compression methods were used while transferring files, etc.

Figure 6 reports on the results of the experimental study. In particular, for each mapping element of the three test cases, it represents the number of proof steps required and the time needed to generate proofs in PML. Notice, that the proof time indicator in Figure 6 takes into account the time needed by the modified version of JSAT to produce proof information, connection time to the IW web service, time for producing and posting PML documents.

An observation of the spikes starting from the mapping #700 in the time line of the Cornell vs. Washington test case is an example of how Internet connection increases the proof time. The average proof length and proof time for a single mapping element in the test cases of Figure 6 constitute 16 steps and 14 seconds. Time needed to produce proofs of all mapping elements in each test case is $2.7min.$ - 20 mappings; $27.7min.$ - 160 mappings; and $546.2min.$ - 1326 mappings respectively. Notice that the modified JSAT version produces proof information on a single mapping element requiring, in the average, less than 1 millisecond, therefore producing proof information for all mappings, for instance, in the case of 1326 mappings, would require less than 1 minute. Moreover, it is hard to imagine that (ordinary) users will be willing to browse explanations of thousands and even hundreds of mappings. However, one dozen seems to be a reasonable number of mappings to be looked through for a short period of time. Also, as

[12] indicates, *S-Match* mappings quality indicators (e.g., precision, recall), on average are above 80%, therefore, may be that users will not need explanations for a large number of mappings.

Results of the experimental study look promising, however there are proof time issues to be addressed. For example, if a user needs explanations aimed at proof generation and manipulation need to be added. Future work also includes further experiments with more complex test cases. However, the experimental study we have conducted gives a preliminary vision that the explanation techniques proposed potentially scale to requirements of the Semantic Web, providing meaningful and adjustable answers in real time.

## 6 Discussion

A line of semi-automated schema/ontology matching systems exists, see for instance [4, 9, 10, 17, 19, 24, 27]. Good surveys are provided in [30–32]. To the best of our knowledge, only the iMap system [8] generates explanations of its matching process. However, it substantially differs from *S-Match*, in the type of the result it returns and in the matching approach. In particular, iMap returns an affinity coefficient in the [0,1] range, it does analyze term meaning, and it does not exploit any inference engines. It is based on a combination of constraint/instance-based matching techniques, called *searchers*. Explanations of mappings in the iMap system are based on the idea of a *dependency graph*, which traces the searchers (memorizing relevant slices of the graph) used to determine a particular mapping. Finally, exploiting the dependency graph, explanations are presented to the user in the English format. Although, the meaning of the affinity coefficient returned remains obscure. Additionally, it becomes more obscure as more operations (e.g., use of particular thresholds or weights) are made on these affinity measures.

The DPLL procedure discussed in the paper constitutes a basic (without heuristics and optimizations) propositional satisfiability search procedure of the state of the art SAT engines, such as Chaff [26], etc. Thus, our approach for producing explanations remains valid also for efficient semantic matching.

Recently there has been some work on verifying SAT solvers, in particular on checking the correctness of unsatisfiability proofs by representing the proof as a chronologically ordered set of conflict clauses [15] and using independent resolution-based checking procedures [36]. The major drawback from the IW perspective is that the above mentioned approaches do not provide proofs as independent (portable) objects, which can be checked by a trusted theorem prover. Another problem is that typical traces of DPLL processing are not logical proofs (e.g., they cannot be translated into natural deduction proofs). One approach describes "equivalent" inferences [3, 21] for use in explaining answers as a correct although potentially alternative deductive path. A direct solution to the above problem is provided in [1].

Also, an emergent and challenging research direction in the SAT community concerns *unsatisfiability cores*, which is a task of extracting a (optionally mini-

mal) subset of clauses of the original formula such that the conjunction of these clauses is still unsatisfiable, see, for example [28, 35]. Typically this subset of clauses is much smaller than the original formula. Although, extracting unsatisfiable cores requires producing another trace file representing a decision tree of an unsatisfiable proof. This direction seems promising with respect to the work on explanations of answers from *S-Match*, since by using unsatisfiability cores, proof logs can be significantly reduced. Moreover, minimal unsatisfiability subformulas should allow for localizing a minimal number of axioms implying a particular semantic relation between the nodes under consideration. This approach focuses a user's attention precisely on a reason why this type of a relation holds.

As the use of matching systems for managing semantic heterogeneity grows, it becomes very important to produce explanations of them in order to make the Semantic Web *transparent* and *trustable*. Some technical details of our solution are:

- We use the Proof Mark-up Language for representing *S-Match* proofs, thus facilitating interoperability;
- We use meaningful terms rather than numbers in the DIMACS format, thus facilitating understandability;
- We use the IW tools, thus facilitating customizable, interactive proof and explanation presentation and abstraction;
- Our solution is potentially scalable to the Semantic Web requirements.

## 7 Conclusions

In this paper, by extending *S-Match* to use the Inference Web infrastructure, we have demonstrated our approach for explaining answers from matching systems exploiting background ontological information and reasoning engines. The explanations can be presented in different styles allowing users to understand the mappings and consequently to make informed decisions about them. The paper also demonstrates that *S-Match* users can leverage the Inference Web tools, for example, for sharing, combining, browsing proofs, and supporting proof meta-information including background knowledge. We also have presented DPLL-based IW explanations of the SAT engine used in the context of *S-Match* tasks. We have tested our approach of explaining *S-Match* answers. The results look promising and demonstrate their potential to scale to the requirements of Semantic Web.

Future work proceeds in at least two directions. Using explanations, a matching system can provide users with meaningful prompts and suggestions on further steps towards the production of a sound and complete result. Having understood the mappings returned by a matching system, users can deliberately edit them manually, therefore providing the feedback to the system. Thus, the first direction includes developing an *environment*, which efficiently exploits the IW proofs and explanations presented in the paper, in order to make the *S-Match* matching process (fully-fledged) *interactive* and *iterative*, involving user in the critical points where his/her input is maximally useful.

The second direction includes (i) improving the *S-Match* proofs and explanations by using *abstraction* techniques more extensively; (ii) conducting a user satisfaction study of the explanations; and (iii) extending explanations to other SAT engines as well as to other non-SAT DPLL-based inference engines, e.g., DLP, FaCT [18], and Pellet [29].

**Acknowledgements.** This work has been partly supported by the European Knowledge Web network of excellence (IST-2004-507482), by the research grant COFIN 2003 Giunchiglia 40100657, and by research grants from DARPA's DAML and PAL programs.

## References

1. C. Barrett and S. Berezin. A proof-producing boolean search engine. In *Proceedings of PDPAR*, 2003.
2. D. Le Berre. JSAT: The java satisfiability library. http://cafe.newcastle.edu.au/daniel/JSAT/, 2001.
3. A. Borgida, E. Franconi, I. Horrocks, D. McGuinness, and P. Patel-Schneider. Explaining ALC subsumption. In *Proceedings of Description Logics workshop*, 1999.
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. P. Pinheiro da Silva, D. L. McGuinness, and R. Fikes. A proof markup language for semantic web services. Technical report, KSL, Stanford University, 2004.
6. M. Davis, G. Longemann, and D. Loveland. A machine program for theorem proving. *Journal of the ACM*, (5(7)), 1962.
7. M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, (7):201–215, 1960.
8. R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: Discovering complex semantic matches between database schemas. In *Proceedings of SIGMOD*, pages 383 – 394, 2004.
9. M. Ehrig and S. Staab. QOM: Quick ontology mapping. In *Proceedings of ISWC*, pages 683–697, 2004.
10. J. Euzenat and P.Valtchev. Similarity-based ontology alignment in OWL-lite. In *Proceedings of ECAI*, pages 333–337, 2004.
11. F. Giunchiglia and P. Shvaiko. Semantic matching. *KER Journal*, (18(3)):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 *Proceedings of Meaning Coordination and Negotiation workshop at ISWC*, 2004.
14. F. Giunchiglia, M. Yatskevich, and E. Giunchiglia. Efficient semantic matching. In *Proceedings of ESWC*, 2005.
15. E. Goldberg and Y. Novikov. Verication of proofs of unsatisability for CNF formulas. In *Proceedings of DATE*, 2003.
16. N. Guarino. The role of ontologies for the Semantic Web (and beyond). Technical report, Laboratory for Applied Ontology, Institute for Cognitive Sciences and Technology (ISTC-CNR), 2004.

17. H.H.Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proceedings of VLDB*, pages 610–621, 2001.
18. I. Horrocks and P. F. Patel-Schneider. FaCT and DLP. In *Automated Reasoning with Analytic Tableaux and Related Methods: Tableaux*, pages 27–30, 1998.
19. J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proceedings of VLDB*, pages 49–58, 2001.
20. B. Magnini, L. Serafini, and M. Speranza. Making explicit the semantics hidden in schema models. In *Proceedings of workshop on Human Language Technology for the Semantic Web and Web Services at ISWC*, 2003.
21. D. L. McGuinness and A. Borgida. Explaining subsumption in description logics. In *Proceedings of IJCAI*, pages 816–821, 1995.
22. D. L. McGuinness and P. Pinheiro da Silva. Infrastructure for web explanations. In *Proceedings of ISWC*, pages 113–129, 2003.
23. D. L. McGuinness and Pinheiro da Silva P. Registry-based support for information integration. In *Proceedings of IJCAI Workshop on Information Integration on the Web*, 2003.
24. S. Melnik, E. Rahm, and P. Bernstein. Rondo: A programming platform for generic model management. In *Proceedings of SIGMOD*, pages 193–204, 2003.
25. A.G. Miller. WordNet: A lexical database for english. *Communications of the ACM*, (38(11)):39–41, 1995.
26. M. Moskewicz, C. Madigan, Y. Zhaod, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of DAC*, 2001.
27. N. Noy and M. A. Musen. Anchor-prompt: Using non-local context for semantic matching. In *Procedings of IJCAI workshop on Ontologies and Information Sharing*, pages 63–70, 2001.
28. Y. Oh, M. N. Mneimneh, Z. S. Andraus, K. A. Sakallah, and I. L. Markov. AMUSE: A minimally-unsatisfiable subformula extractor. In *Proceedings of DAC*, pages 518–523, 2004.
29. B. Parsia, E. Sirin, M. Grove, and R. Alford. Pellet OWL reasoner. http://www.mindswap.org/2003/pellet/index.shtml.
30. E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, (10(4)):334–350, 2001.
31. P. Shvaiko. A classification of schema-based matching approaches. In *Proceedings of Meaning Coordination and Negotiation workshop at ISWC*, 2004.
32. P. Shvaiko and J. Euzenat. A survey of schema-based macthing approaches. Technical report, DIT-04-087, University of Trento, 2004.
33. 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.
34. 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.
35. L. Zhang and S. Malik. Extracting small unsatisfiable cores from unsatisfiable boolean formulas. In *Proceedings of SAT*, 2003.
36. L. Zhang and S. Malik. Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In *Proceedings of DATE*, 2003.