Soundness of Semantic Methods for Schema Matching

M. Benerecetti[1], P. Bouquet[2], S. Zanobini[2]

[1] Department of Physical Science – University of Naples – Federico II
Via Cintia, Complesso Monte S. Angelo, I-80126 Napoli (Italy)
[2]Department of Information and Communication Technology – University of Trento
Via Sommarive, 10 – 38050 Trento (Italy)
bene@na.infn.it, bouquet@dit.unitn.it, zanobini@dit.unitn.it

Abstract. One of the key challenges in the development of open semantic-based systems is enabling the exchange of meaningful information across applications which may use autonomously developed schemata. Semantic coordination is the problem of discovering mappings across schemata and schema matching is one of the proposed approaches. In this paper we provide a preliminary investigation on the notion of correctness of semantic methods for schema matching. We define a first notion of semantic soundness (and completeness), but immediately show that this notion is not appropriate to capture the intuitive notion of correctness for a method. We then introduce the idea of pragmatic soundness, and argue that it corresponds to what we intuitively expect, but that it can't be directly computed. Finally, we discuss some preliminary conditions under which a semantically sound method can guarantee progmatic soundness as well, which is – in our opinion – the best we can get from semantic methods.

1 Introduction

One of the key challenges in the development of open semantic-based systems is enabling the exchange of meaningful information across applications which may use autonomously developed schemata (database schemata, classifications, even directory trees on file systems in peer-to-peer applications) for organizing locally available data. As in open system a beforehand agreement on the meaning of schemata seems impossible in practice, a large number of methods and systems have been proposed to automatically match schemata[1]. The resulting mappings are then used as the basis for a runtime semantic-based coordination of such a network of autonomous applications.

Methods may differ along many dimensions: the type of structures to which they can be applied (e.g., trees, directed acyclic graphs, graphs); the type of result they return (e.g., similarity measures, model-theoretic relations, fuzzy relations); the resources they use to compute such a relation (e.g. external lexical resources, ontologies, string manipulators, graph matching techniques, instance-based techniques). In this paper, for reasons that will be explained in detail, we are mostly concerned with a class of methods that we call *semantic methods*. The general intuition underlying semantic methods is that they aim at discovering relations between (pairs of) entities belonging to different

[1] A very partial list includes [14, 13, 11, 10, 4, 6, 9, 2, 5, 8, 3]. A detailed description of these methods is out of the scope of this paper.

schemata *based on the meaning of the two entities*. However, beyond this point, there is a significant disagreement on what characterizes a semantic method from a non semantic method. For example, a recent paper by Giunchiglia and Schvaiko [7] proposes to include among semantic methods only those methods that directly return a semantic relation (e.g., material implication or logical equivalence), namely a relation with a well-defined model-theoretic interpretation. This analysis is far from being shared in the community, as other people feel that a method is semantic if it uses semantic information to return its results, or if there is a principled way to assign an indirect semantics to its results (e.g., mapping numerical values on semantic relations through the definition of suitable thresholds).

In such a situation, it is not surprising that we still lack a clear definition of the conditions under which a semantic method can be said to work "correctly". Suppose, for example, that we have a method α that takes in input two nodes n_A and n_B from two schemata S_A and S_B respectively and returns `True` if the two nodes represent equivalent concepts, `False` otherwise. Now, imagine that α is fed with the categories `/IMAGES/TUSCANY/FLORENCE` and `/PHOTOS/ITALY/FLORENCE`[2] belonging to two classification schemata, and that it returns `True`. Is the result "correct"? Why? And what if the result were `False`? Under what conditions would we accept this result as "correct"?

This paper aims at answering this kind of questions. We start by providing a precise characterization of schema matching for a special (but interesting) case of schemata, namely hierarchical classifications. Then we propose a characterization of semantic-based methods based on the idea that they must at least provide an explicit and formal interpretation of the entities they compare, and of the resulting relation. Finally, for this class of methods, we define the notions of semantic soundness and completeness, but immediately show that this notion is not appropriate to capture the intuitive notion of correctness for a method. We then introduce the idea of pragmatic soundness, and argue that it corresponds to what we intuitively expect, but that it can't be directly computed. Finally, we discuss some preliminary conditions under which a semantically sound method can guarantee progmatic soundness as well, which is – in our opinion – the best we can get from a semantic method for schema matching.

2 The problem of schema matching

Schema is a broad term, that applies to different kinds of structures. In [3], it was argued that it makes no much sense to speak about schema matching in general, and that the analysis should be done case by case along the dimension of the intended use of a schema. Accordingly, in this paper we restrict our attention to a special kind of schemata, *hierarchical classifications*, whose explicit purpose is to classify objects (e.g., documents). This restriction does not affect the generality of our investigation, as the method of analysis can be applied to study the problem of matching other types of schema, such as database schemata, service descriptions, datatypes.

[2] Throughout the paper we will use the notation `X/.../Y` to refer to a path in schema in analogy with the notation for paths in a file system. If the schema is a tree, then `/` represent the root of the schema and `/X/.../Y` a path from the root to `Y` through `X`.

We start with a few definitions that characterize the kind of schemata we deal with, namely topic hierarchies used as classification schemata.

Definition 1 (Topic hierarchy). *Let Λ be a set of labels (e.g., words in natural language). A* topic hierarchy *$\mathcal{S} = \langle K, E, l \rangle$ is a triple where K is a finite set of nodes, E is a set of arcs on K, such that $\langle K, E \rangle$ is a rooted tree, and l is a function from K to Λ.*

Two simple examples of topic hierarchies are depicted in Figure 1.

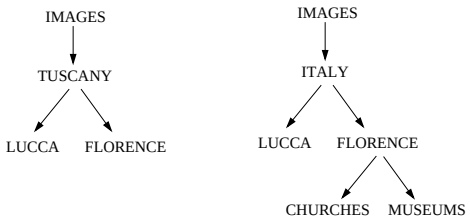

Fig. 1. Two simple topic hierarchies

A possible use for topic hierarchies is to classify documents. To express this formally, we introduce the notion of classification function.

Definition 2 (Classification function). *Let D be a set of documents and $\mathcal{S}$ a topic hierarchy $\langle K, E, l \rangle$. A classification function over $\mathcal{S}$ is a function $\tau : D \rightarrow K$ from documents to nodes of $\mathcal{S}$.*

A classification function places a document under a node in a topic hierarchy. We associate to each classification function a *retrieval function*, which is a function from nodes to the sets of documents attached to them in a topic hierarchy. It essentially plays the inverse rôle of the classification function.

Definition 3 (Retrieval function). *Let D be a set of documents, $\mathcal{S} = \langle K, E, l \rangle$ a topic hierarchy, and τ a classification function over $\mathcal{S}$. The* retrieval function *of τ over $\mathcal{S}$ is a function $\mu_\tau : K \rightarrow 2^D$ satisfying the following condition:*

$$\textit{for every } d \in D,\ d \in \mu_\tau(\tau(d))$$

Finally, we want to formalize the intuition of a classification being a hierarchical classification (hereafter HC). Intuitively, it must satisfy the following requirement: documents classified under a node `Z` along a path `X/.../Y/Z` could be also classified under the ancestor nodes of the same path (though with a lower degree of precision) if `Z` were removed from the topic hierarchy:

Definition 4 (Hierarchical classification). *Given a set of documents D, a* hierarchical classification *$\mathcal{H} = \langle \mathcal{S}, \tau \rangle$ is a pair where $\mathcal{S}$ is a topic hierarchy and τ is a classification function over $\mathcal{S}$ which satisfies the following property: if τ classifies a document d under a node `Z` along a path `X/.../Y/Z`, and we remove `Z` from the path, then τ would assign d to the node `Y` of the path `X/.../Y`.*

Schema matching can be defined as the problem of computing relations between pairs of nodes belonging to different HCs. Let $\Re$ be a set of relations that may hold between two nodes belonging to two distinct schemata $\mathcal{S}_A$ and $\mathcal{S}_B$. Then a mapping is defined as follows:

Definition 5 (Mapping). *A* mapping $\mathcal{M}_{A\to B}$ *between two HCs* $\mathcal{H}_A = \langle \mathcal{S}_A, \tau_A \rangle$ *and* $\mathcal{H}_B = \langle \mathcal{S}_B, \tau_B \rangle$ *is a set of triples* $\langle n_A, n_B, r \rangle$*, where:*

- n_A *and* n_B *are two nodes belonging to* $\mathcal{S}_A$ *and* $\mathcal{S}_B$*, respectively;*
- $r \in \Re$ *is a relation between* n_A *and* n_B*.*

Each triple $\langle n_A, n_B, r \rangle$ belonging to a mapping is called a *mapping element*.

Finally, as our goal is to discuss properties of schema matching methods, we formally define a method as a function which returns true when a given relation holds between two elements of different schemata, false otherwise:

Definition 6 (Schema Matching Method). *Let* $\mathcal{M}_{A\to B}$ *be a mapping between two HCs* $\mathcal{H}_A$ *and* $\mathcal{H}_B$*. A* schema matching method $\alpha : \mathcal{M}_{A\to B} \to \{T, F\}$ *is a function from mapping elements to boolean values.*

Of course, it is more natural to view a method as a function which takes two nodes as input and returns a relation as output. Here we adopt this more abstract (but after all equivalent) characterization as it is more appropriate for our analysis.

3 Semantic methods for schema matching

In the previous section, we deliberately left the definition of schema matching methods quite vague, as we wanted to characterize the problem of schema matching in a very general form. Here we provide a precise characterization of semantic methods in a precise way through two general principles that, in our opinion, distinguish semantic methods from non semantic methods.

A method for schema matchng is a semantic methods if it satisfies the two following principles:

Explicit representation of meaning: a semantic method must match schema elements on the basis of an explicit representation of their meaning, where meaning is a formal object of a logical type which corresponds to the type intended by the schema designer. Notationally, if n is a node of a HC, then $\mathcal{R}(n)$ is the formal representation of its meaning;

Computation of relations based on meaning: given two nodes n and m belonging to different HCs, a semantic method must return a relation which connects the meanings of the schema elements under comparison. Such a relation must in turn have an interpretation defined over the meaning of the compared elements.

So, according to the first principle, a semantic method should explicitly interpret the elements of a HC as concepts, and provide a corresponding formal representation of type concept (e.g., as terms in some Description Logic system [1]). For example,

the meaning of the nodes FLORENCE belonging to the right hand side schema and CHURCHES belonging to the left hand side of schema of Figure 1 approximately corresponds to the two concepts "Images of Florence in Tuscany" and "Images of churches in Florence, in Italy". Notice that a schema describing how a web service works (basically, a finite state automaton) should be interpreted in a completely different way, as nodes would represent states that can be reached through actions associated to arcs.

Then, according to the second principle, when matching HCs, a semantic method should return a relation between concepts (e.g., subsumption, equivalence, and so on). Notice that here we will privilege classical model-theoretic relations, though it is possible to work with fuzzy-theoretic relations between concepts (see e.g. the common framework for ontology. Going back to the example of Figure 1, the relation between the two nodes FLORENCE and CHURCHES (interpreted as we said above) is that the first is more general than the second. We note that, in this case, determining the relation between the two concepts intuitively requires to use knowledge that was not extracted from the two schemata, namely that Tuscany is in Italy. In the following, we will refer to this external knowledge as the *ontology* associated to a method. In analogy to what we said above, a relation between elements of two service description schemata would be completely different.

We are now ready to start our discussion about soundness of semantic methods.

4 Semantic soundness and completeness

Given the two principles discussed above, a semantic method α is defined by: (i) a language $\mathcal{L}$ suitable to explicitly represent the meaning of each schema element, (ii) a procedure for extracting the meaning of each element n ($\mathcal{R}(n)$), (iii) a (possibly empty) ontology $\mathcal{O}$, and (iv) a set of relations $\Re$ that it can compute. The 4-tuple $\langle \mathcal{L}, \mathcal{O}, \mathcal{R}(), \Re \rangle$ is what we call the *semantic frame* of the method.

We now propose a notion of semantic soundness and completeness with rtespect to a semantic frame F. The intuition is the following: a method is *semantically sound* w.r.t. F if, whenever it computes a relation between two elements of distinct schemata, the relation follows from what the method knows about the meaning associated to the two elements; and is *semantically complete* if, whenever one of the relations in $\Re$ between the meaning of two nodes follows from what the method knows, then the method effectively returns that relation. More formally:

Definition 7 (Semantic Soundness). *Let $F = \langle \mathcal{L}, \mathcal{O}, \mathcal{R}(), \Re \rangle$ be the semantic frame of a method α and $\mathcal{H}_A$ and $\mathcal{H}_B$ be two HCs. Then α is* semantically sound *w.r.t. C if and only if for any mapping element $\langle n_A, n_B, r \rangle$ the following holds:*

$$\textit{if } \alpha(\langle n_A, n_B, r \rangle) = T, \textit{ then } \mathcal{O} \models_{\mathcal{L}} \mathcal{R}(n_A)\ r\ \mathcal{R}(n_B)$$

Definition 8 (Semantic Completeness). *Let $F = \langle \mathcal{L}, \mathcal{O}, \mathcal{R}(), \Re \rangle$ be the semantic frame of a method α and $\mathcal{H}_A$ and $\mathcal{H}_B$ be two HCs. Then α is* semantically complete *w.r.t. C if and only if for any two nodes n_A and n_b the following holds:*

$$\textit{if } \mathcal{O} \models_{\mathcal{L}} \mathcal{R}(n_A)\ r\ \mathcal{R}(n_B), \textit{ then } \alpha(\langle n_A, n_B, r \rangle) = T$$

Though these notions of semantic soundness and completeness seem reasonable, it should be quite evident that they do not seem to capture what we have in mind when we say that a method is correct. Indeed, what we would like to say is that a method is sound when it computes the "right" relation between two elements, namely the relation that follows from the "correct" interpretation of the schemata and from the use of the "right" background knowledge. Instead, what the definitions above says is only that, given an ontology and a formal representation of the meaning of two nodes, then a semantic method is sound if and only if it derives only relations that logically follows from the background knowledge provided by its ontology. But this is tantamount as saying that a semantic method is sound if and only if the reasoner used to compute the relation between meanings is sound and complete, which would be a very trivial result. Indeed, imagine a dummy method that associate the same concept k to all the elements of two HCs, and always returns the equivalence relation for any pair of nodes (for all $k \in \mathcal{S}$ and $k' \in \mathcal{S}'$, $\alpha(k, k', \equiv) = T$). Since any concept is always equivalent to itself, then this method is semantically sound. But is this method of any interest?

Intuitively, the problem is that semantic soundness as we defined it (and a similar argument can be done for completeness) does not say anything on the appropriateness of the meaning explicitation performed by the method and on the relation between the meaning of nodes and the available ontology. In short, semantic soundness is a necessary but not sufficient condition to capture the intuitions we have about the correctness of a method. What we need as a sufficient condition is a way for excluding dummy methods like the one described above, namely methods that build arbitrary interpretations and use pertinent knowledge about the meaning of schema elements.

However, this is an extremely tough problem not only in schema matching, but in general for any semantic theory based on formal logic. Indeed, as we know from classical results (see e.g. the model-theoretic argument discussed by the philosopher H. Putnam in [12]), there's nothing we can do to prevent unintended interpretations of a formal language. The form in which Putnam discusses this problem is the following: even if two agents agree on the truth value of all the sentences of a language L (including modal propositions on the necessity of propositions), this is not sufficient to fix the interpretations of the terms they use, which means that they may still be talking about different things. Our version of this argument would be the following: even if we can guarantee that a method is semantically sound and complete, there is nothing that guarantees that the two elements were correctly interpreted, and that relation between the two nodes is the one we expect.

To get around this problem, there are basically two approaches available for semantic methods. The first one, which we will call the *linguistic approach*, is to exploit the fact that almost invariably the labels of schemata are meaningful expressions of natural language, e.g. English. If this is the case, then not every interpretation is acceptable, though ambiguity is still possible. For example, the word "bank" can mean "depository financial institution" or "the slope beside a body of water" (Wordnet 1.7), but cannot mean "animal with four legs", unless we relax the assumptions that labels are taken from English. The second approach, which we call *instance-based approach*, is to exploit the data attached to a schema (e.g., the documents attached to the categories of a HC) to guess the meaning of the category itself. The idea is that we can determine the meaning

of an element in a schema by processing the documents associated to that element, e.g. by analyzing the number of times a word occur in a document, or the co-occurrence of words in the same document. Both approaches, however, have their drawbacks: the linguistic approach suffers from the "ambiguity problem", namely there is still the possibility that we haven't caught the intended interpretation of a schema element (as natural languages are ambiguous in different respects); instance-based methods suffers from the "contingency problem", namely the actual set of documents attached to an element may be insufficient to capture the intended concept (let alone the fact that there may be schemata not populated with documents at all).

To sum up, linguistic and instance-based approaches improve the situation of semantic methods, but do not provide the sufficient conditions we are looking for. Is there another way of defining the correctness of a method which does not refer to the explicitation of meanings? In the next section we propose a possible answer.

5 Pragmatic Soundness

The main reason why people develop schemata is to provide a suitable organization of a body of relevant data, e.g. records in a database, file in a file system, documents in a classification schema. Schema matching methods should allow applications to exchange data in a meaningful way through the exploitation of mappings across schemata. For example, when we match two HCs, the goal is to find mappings that allow us to retrieve documents on a given topic which are classified under (possibly different) categories in different HCs. From this perspective, if a semantically sound method derives that two nodes are equivalent, then one would expect that a user would classify the same documents under those two nodes; if the methods derives that a node is more general than another one, then one would expect that the documents classifiable under the first node are a superset of those classifiable under the second; and so on. If this holds, then the method is correct, as it "does the right thing" for its users. This notion of correctness, based on data rather than on the meaning of schema elements, is pragmatic, as it refers to how people use schemata, and not (directly) to how they interpret it. Let us try to make this intuition more precise.

Preliminary, we introduce the notation to refer to all documents classified in a subtree of a topic hierarchy, and not only in a single node. The reason is the following. Consider the two HCs of Figure 1. Note that the node `FLORENCE` in the right hand side topic hierarchy has two children (`CHURCHES` and `MUSEUMS`), whereas the left one has no children. This means that the documents that in the right hand side topic hierarchy are classified under `CHURCHES`, would be probably classified under the node `FLORENCE` in the left hand side topic hierarchy. Therefore, we introduce the notation $\mu_\tau(n\downarrow)$ to denote the set of documents classified under a subtree rooted at the node n. More formally, let $n\downarrow = \{k \in K \mid k \text{ is a descendant of } n\}$ denote the set of nodes in the subtree rooted at n, then $\mu_\tau(n\downarrow) = \bigcup_{m \in n\downarrow} \mu_\tau(m)$.

Let D be a set of documents and $\Re$ a set of relations between sets of documents (for example, $\Re = \{=, \subseteq, \supseteq, \perp\}$, where $\perp$ means disjoint). Furthermore, imagine that a classifier classifies all documents of D in two different HCs ($\mathcal{H}_A$ and $\mathcal{H}_B$). Then a first tentative definition of pragmatic soundness could be the following:

Definition 9 (Strong Pragmatic Soundness). *Let $\mathcal{H}_A$ and $\mathcal{H}_B$ be two HCs and α a semantic method. Then α is* strongly pragmatically sound *if for any mapping element $\langle n_A, n_B, r\rangle$ (with $r \in \Re$) the following holds:*

$$\textit{if } \alpha(\langle n_A, n_B, r\rangle) = T \quad \textit{then} \quad \mu_\tau(n_A{\downarrow})\ r\ \mu_\tau(n_B{\downarrow})$$

Intuitively, this means that if a semantic method α discovers a relation r between two nodes n_A and n_B, then the corresponding set-theoretic relation r also holds between the sets of documents classified by the function τ in the subtree rooted at the nodes n_A and n_B[3].

However, this definition presupposes two very strong assumptions. First, D must be the set of all possible documents of the universe; otherwise, it may happen that an actual set of documents is not sufficient to discriminate between some set-theoretical relations, such as $\subset$ and $=$ (it may happen that no document belonging to D which would be associated to n_A and n_B, and therefore the two sets would be contingently the same, whereas they would not if we had had more documents available).

But even more important, it presupposes that each document can be classified in a unique way. Of course, this is not the case in general, as documents are typically *rich objects*, and can be classified under different categories, depending on what aspect of the document is taken as dominant. For example, this paper could be classified under different categories (e.g. `SEMANTIC INTEROPERABILITY`, `ONTOLOGY INTEGRATION`, `SCHEMA MATCHING`, `FORMAL MODELS`, ...), and each of these categories would reflect a legitimate point of view on the paper. Therefore, even if two categories in two different HCs – populated by the same classifier – are semantically related, we can't guarantee that the sets of documents classified under those two categories will be in the same relation.

The considerations above suggest that we need a weaker notion of pragmatic soundness, which can take into account the possibility that a classifier (human or automatic) can legitimately classify the same document under different categories. In this situation, the question arises of whether there can be a reasonable notion of correctness. Intuitively, we suggest that a *counterfactual* notion of correctness: a method is correct if a classifier would not disagree with the answers produced by the method; in other words if, no matter what the actual classification is, the classifier could have classified the documents according to the relation discovered by the method.

To capture this intuition, we first introduce the following finer notion of classifier:

Definition 10 (Classifier). *A* classifier *C is a pair $\langle\{\tau_i\}, F\rangle$, where $\{\tau_i\}$ is a set of classification functions, and $F = \left\langle \mathcal{L}^C, \mathcal{O}^C, \mathcal{R}^C(), \Re\right\rangle$ is a semantic frame.*

Associating a set of classification functions to a classifier allows us to capture the fact that he can classify the same set of document in different ways. Therefore, when

[3] To keep the formalism simple, we are abusing our notation by using the symbol r to refer both to the (semantic) relation computed by a semantic method and the relation which holds between sets of documents. In fact, we rely on the intuitive mapping between semantic relations (say, subsumption between concepts) and set-theoretic relations between their interpretation (for subsumption, it would be set inclusion). To be precise, such a mapping should be explicitly defined.

populating a topic hierarchy, we allow classifiers to employ any of their classification functions. Intuitively, the set $\{\tau_i\}$ can be seen as a set of "acceptable" classification functions, in the sense that the classifier will be prepared to accept classifying a document under a given node if there is a classification function belonging to $\{\tau_i\}$ which would classify that document under the same node.

But this is not enough. Indeed, we also expect that there is a rationale behind the classification tasks of any "reasonable" classifier. In other words, we expect that classifiers perform their task based on their knowledge about the documents to be classified and about the available categories. This is where the ontology $\mathcal{O}^C$ and the interpretation function $\mathcal{R}^C()$ associated to a classifier come into play. Intuitively, the ontology and the interpretation function represents the knowledge classifiers use to understand the meaning of a node and, consequently, for classifying a document. Classifiers are then called *pragmatically adequate* if they act consistently with their knowledge. We capture this by imposing the following condition: when a classifier C recognizes a relation holding between (the meanings of) two nodes n_A and n_B of two HCs $\mathcal{H}_A$ and $\mathcal{H}_B$, then – whatever classification function she is actually using – if this function classifies under n_A and n_B the sets X and Y of documents, then there must be a (possibly distinct) acceptable classification function which would classify under the other node a set X' of documents in such a way that the corresponding set-theoretic relation holds between the sets X' and Y. The following definition formalizes this intuition.

Definition 11 (Compatibile classification functions). *Let C be a classifier, τ_1 and τ_2 two classification functions of C, r any relation in $\Re$, and n_A in $\mathcal{H}_A$ and n_B in $\mathcal{H}_B$ two nodes. We say that τ_1 is* compatible with τ_2 w.r.t. n_A and n_B *if the following holds:*

$$\textit{if } \mathcal{O}^C \models \mathcal{R}^C(n_A)\ r\ \mathcal{R}^C(n_B),\ \textit{then } \mu_{\tau_1}(n_A{\downarrow})\ r\ \mu_{\tau_2}(n_B{\downarrow})$$

Intuitively, two classification functions of a classifier are compatible w.r.t. two nodes if their respective way of classifying documents under the two nodes preserve the relation the classifier recognizes between the meanings of the two nodes. We can now formalize the notion of *pragmatically adequate* classifier.

Definition 12 (Pragmatic adequacy). *Let $C = \langle\{\tau_i\}, F\rangle$ be a classifier. Then C is* pragmatically adequate *if, given any two nodes n_A in $\mathcal{H}_A$ and n_B in $\mathcal{H}_B$, and any classification function $\tau_1 \in \{\tau_i\}$, there is another $\tau_2 \in \{\tau_i\}$ which is compatible with τ_1 w.r.t. n_A and n_B.*

This definition simply says that if a classifier C associate a set of documents to some node n_A, and n_A is in a certain relation r with a second node n_B, then C must be prepared, possibly by employing some other acceptable classification function of his (namely, a compatible classification function), to classify under the n_A a set of documents holding the same relation r with the set of documents attached to n_B.

Based on the definitions above, we can now attempt a second definition of *pragmatic soundness* which, we believe, is the best we can expect from a schema matching method. Intuitively, we say that a schema matching method is *pragmatically sound* if whenever it derives a relation r between two nodes n_A and n_B, a pragmatically adequate classifier would consider this result as "acceptable" according to the possible

ways he could classify a set of documents. By "acceptable" here we mean that whatever set of documents C has actually placed under n_A and n_B (using one of his classification functions), C could have placed under n_A, using a possibly different admissible classification function, a set of documents in the same relation r with the set of documents actually placed under n_B. This intuition is captured by the following definition:

Definition 13 (Pragmatic Soundness). *Let $C = \langle\{\tau_i\}, F\rangle$ be a pragmatically adequate classifier, and $\mathcal{H}_A$ and $\mathcal{H}_B$ be two HCs. A method α is* pragmatically sound *w.r.t. C if, for any mapping element $\langle n_A, n_B, r\rangle$, the following holds: if $\alpha(\langle n_A, n_B, r\rangle) = T$, then for any classification $\tau_2 \in \{\tau_i\}$ there is a classification $\tau_1 \in \{\tau_i\}$ such that, for any $r \in \Re$, $\mu_{\tau_1}(n_A\downarrow)\ r\ \mu_{\tau_2}(n_B\downarrow)$.*

6 Can semantic methods be pragmatically sound?

Assume now we have a semantically sound matching method α which can answer whether a semantic relation between two nodes holds or not. In this section we try to answer the question of what can condition can guarantee that a sound semantic method α is also pragmatically sound[4]. We can state the following proposition:

Proposition 1. *Let $F = \langle\mathcal{L}, \mathcal{O}, \mathcal{R}(), \Re\rangle$ be a semantic frame, α a method semantically sound w.r.t. F, and $C = \langle\{\tau_i\}, F^C\rangle$ a pragmatically adequate classifier (where $F^C = \langle\mathcal{L}^C, \mathcal{O}^C, \mathcal{R}^C(), \Re\rangle$). If $\mathcal{O}^C \sqsubseteq \mathcal{O}$ and $\mathcal{R}^C(m) = \mathcal{R}(m)$, then α is pragmatically sound. Moreover, if $|\{\tau_i\}| = 1$, then α is strongly pragmatically sound.*

The proposition states that if (i) α is semantically sound, (ii) the ontology used by α is subsumed by a pragmatically competent classifier knowledge (i.e., it is a sound but not necessarily complete representation of the classifier knowledge), and (iii) the meaning assigned to the nodes by $\mathcal{R}^C(m)$ and $\mathcal{R}(m)$ is the same (namely, $\models \mathcal{R}^C(m) \equiv \mathcal{R}(m)$), then α is also pragmatically sound. If, in addition, (iv) the classifier always uses the same classification function, then clearly α is also strongly pragmatically sound.

The first part of the proposition immediately follows from Definitions 7, 12 and 13. The second part descends from Definitions 7, 12 and 9. A sketch of proof follows. Since any relation between concepts that can be deduced from a less specific ontology ($\mathcal{O}$) can also be deduced by a more specific one ($\mathcal{O}^C$), Condition (i) toghether with Condition (ii) ensure that any relation discovered by the method α would also be inferred by any classifier. Moreover, if C is a pragmatically adequate classifier, whatever classification function τ he has used to place documents under node n_A and n_B, by Definition12 there must be another acceptable classification function τ' of C using which C would have placed under n_A a set of documents holding the relation r with those placed by τ under n_B. Hence pragmatice soundness follows. Adding the additional constraint that the classifier only allows for a single classification function, immediately leads to strong pragmatic soundness.

Let us now briefly comment on the conditions we needed to guarantee pragmatic soundness of a semantic matching method. Condition (i) is quite easy to ensure, as we

[4] The problem of pragmatic completeness is significantly harder and out of the scope of this paper. We will not discuss it here.

already pointed out in Section 3. A logic framework powerful enough to express the desired semantic relations between the concepts of interest, for which decidability is guaranteed will suffice. Condition (ii) seems to be a relatively weak requirement. This is an important observation, since providing a method with complete knowledge with respect to a classifier is likely to be a very hard task, let alone the problem of providing complete knowldge with respect to *any* classifier. Even though the first two conditions seem to be reasonably easy to satisfy, Condition (iii) turns out to be quite strong. Notice that weakening the condition on the semantic explicitation functions is problematic. Soundness with respect to any set of semantic relations can indeed be ensured only if the matching method and the classifier both assing the same interpretation (in terms of concepts) to all the nodes of the HCs. Nevertheless, soundness can still be retained on some specific sets of semantic relations, depending on the cases. For instance, if we consider the set of relations $\{\sqsubseteq, \bot\}$, then any semantically sound method employing a more specific semantic explicitation function than that of the classifier (namely, $\models \mathcal{R}^C(m) \sqsupseteq \mathcal{R}(m)$) will still be paragmatically sound. Unfortunately, soundness with respect to none of the other relations we have been considering in the paper would be preserved in this case. Similarly, any semantically sound method employing a less specific semantic explicitation function than that of the user (namely, $\models \mathcal{R}(m) \sqsupseteq \mathcal{R}^C(m)$) will still be paragmatically sound only with respect to the relation $\sqsupseteq$.

7 Conclusions

The consequence of Proposition 1 is that semantic methods can be guaranteed to obtain pragmatically correct results under conditions (i)–(iii) (also (iv) if we want strong pragmatic soundness). As condition (i) is quite trivial, we can conclude that the roadmap to correct semantic methods is quite clear: (a) we need to build ontology which reflect the classifier's (or the user's) point of view on the world ($\mathcal{O}^{\mathcal{C}} \sqsubseteq \mathcal{O}$) and (b) we need to design tools that interpret a schema element as the user interprets it. These two problems are not trivial, but they can be addressed with well-known methods belonging to disciplines like ontology engineering and knowledge representation. Ontology engineering can help us to design better ontologies, e.g. ontologies that appropriately represent what am individual or a community knows on a given domain; knowledge representation gives us methods for representing the meaning of different types of schemata, beyond classifications.

To conclude, we see our work as a small step towards a much more general goal, namely the construction of a theory which explains how semantically autonomous entities (agents) can communicate without presupposing a beforehand agreement on how things should be represented. In other words, a theory of the role of meaning coordination in a theory of (inter)action. A lot remains to be done, but this goes beyond the scope of this paper.

References

1. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook. Theory, Implementation and Applications*. Cambridge University Press, January 2003.

2. Sonia Bergamaschi, Silvana Castano, and Maurizio Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.
3. Paolo Bouquet, Luciano Serafini, and Stefano Zanobini. Semantic coordination: A new approach and an application. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *The Semantic Web - ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science (LNCS)*, pages 130–145, Sanibel Island (FL, USA), October 2003. Springer Verlag.
4. Jeremy Carroll and Hewlett-Packard. Matching rdf graphs. In *Proc. in the first International Semantic Web Conference - ISWC 2002*, pages 5–15, 2002.
5. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of WWW-2002, 11th International WWW Conference, Hawaii*, 2002.
6. J. Euzenat and P. Valtchev. An integrativive proximity measure for ontology alignment. *Proceedings of the workshop on Semantic Integration*, October 2003.
7. F. Giunchiglia and P. Shvaiko. Semantic matching. *Proceedings of the workshop on Semantic Integration*, October 2003.
8. Ryutaro Ichisem, Hiedeaki Takeda, and Shinichi Honiden. Integrating multiple internet directories by instance–base learning. In *AI AND DATA INTEGRATION*, pages 22–28, 2003.
9. Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *The VLDB Journal*, pages 49–58, 2001.
10. Tova Milo and Sagit Zohar. Using schema matching to simplify heterogeneous data translation. In *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pages 122–133, 24–27 1998.
11. Marcello Pelillo, Kaleem Siddiqi, and Steven W. Zucker. Matching hierarchical structures using association graphs. *Lecture Notes in Computer Science*, 1407:3–??, 1998.
12. H. Putnam. *Reason, Truth, and History*. CUP, 1981.
13. Jason Tsong-Li Wang, Kaizhong Zhang, Karpjoo Jeong, and Dennis Shasha. A system for approximate tree matching. *Knowledge and Data Engineering*, 6(4):559–571, 1994.
14. K. Zhang, J. T. L. Wang, and D. Shasha. On the editing distance between undirected acyclic graphs and related problems. In Z. Galil and E. Ukkonen, editors, *Proceedings of the 6th Annual Symposium on Combinatorial Pattern Matching*, volume 937, pages 395–407, Espoo, Finland, 1995. Springer-Verlag, Berlin.