

Soundness of Schema Matching Methods

M. Benerecetti¹, P. Bouquet², S. Zanobini²

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. One of the typical solutions for that problem is the definition of a mapping between pairs of schemas, namely a set of point-to-point relations between the elements of different schemas. A lot of (semi-)automatic methods for generating such mappings have been proposed. In this paper we provide a preliminary investigation on the notion of correctness for schema matching methods. In particular we define different notions of soundness, strictly depending on what dimension (syntactic, semantic, pragmatic) of the language the mappings are defined on. Finally, we discuss some preliminary conditions under which a two different notions of soundness (semantic and pragmatic) can be related.

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. The typical solution for that problem is the definition of a mapping between pairs of schemas, namely a set of point-to-point relations between the elements of different schemas. 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 in order to (semi-)automatically compute on fly such mappings¹. 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

¹ A very partial list includes [15, 14, 12, 11, 4, 6, 10, 2, 5, 9, 3]. A detailed description of these methods is out of the scope of this paper.

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 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, recent papers by Giunchiglia and Schvaiko [7, 8] propose 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. First, we propose a simple theoretical model which allows us to classify methods for schema matching in three broad categories (syntactic methods, semantic methods, and pragmatic methods). Then we turn our attention to semantic methods, and propose a characterization of these methods and a notion of (semantic) soundness. Semantic methods have the advantage that are computationally quite good, but we’ll argue that in general they do not guarantee to capture the intuitive notion of a “good” schema matching method. We then introduce a notion of pragmatic methods (and the corresponding notion of soundness), and argue that they more closely corresponds to what is intuitively expected by a schema matching method. However, we also show that in general they can’t be effectively computed. Therefore, in the last part of the paper we try to identify some very general conditions under which a semantically sound method can guarantee pragmatic 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

² Throughout the paper we will use the notation $X/\dots/Y$ 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 node and $X/\dots/Y$ the unique path from X to Y .

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.

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.

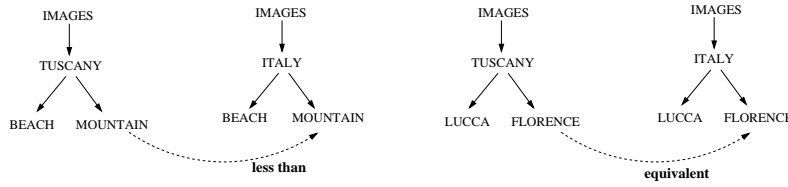


Fig. 1. Two simple topic hierarchies

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:

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

Finally, we can define a hierarchical classification (hereafter HC) simply as a topic hierarchy with an associated classification function τ . Formally:

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} .

Schema matching can be defined as the problem of computing relations between pairs of nodes belonging to different HCs. Let \mathfrak{R} 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 $\mathcal{M}_{A \rightarrow B}$ 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;
- $r \in \mathfrak{R}$ 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 $\mathcal{M}_{A \rightarrow B}$ be a mapping between two HCs \mathcal{H}_A and \mathcal{H}_B . A schema matching method $\alpha : \mathcal{M}_{A \rightarrow B} \rightarrow \{T, F\}$ 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 A three-layer model of schema matching

Before we proceed with our discussion of soundness for schema matching methods, we discuss a simple model which can help us in clarifying what the task of schema matching is from a theoretical point of view.

In a schema matching task, there are three levels that can be taken into account (see Figure 2):

Language level: the language level is the level of expressions (an alphabet and a grammar to build more complex expressions) that can be used to label a schema. Such a language is used to “publish” information about a schema, and possibly to exchange information about the schema with other applications. Therefore, by definition, this level must be publicly accessible. If we do not make any assumption on such a language, then labels should be regarded as mere syntax, with no special meaning. Therefore, if we restrict our analysis to this level, the only kind of mapping that can be found is purely syntactic, and the only information that can be used to compute such a mapping has to do with the syntactic properties of the strings that are used to label nodes, and their arrangement in the schema. However, as we will argue, there are good reasons to assume that labels are meaningful expressions (typically natural language terms), and this has important consequences on how they are treated in schema matching methods.

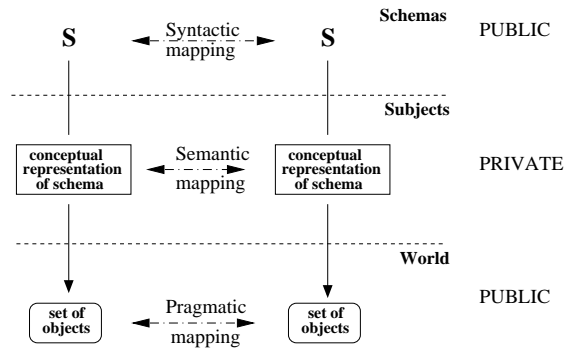


Fig. 2. Three level of schema matching

Concept level: at the concept level, we find a collection of concepts that correspond to the intended meaning of nodes in a schema. Intuitively, this level corresponds to what the creator (or the users) of a schema “had in mind” when the schema was created (or when the schema is used). The main difference between the language level and the concept level is that concepts are not directly accessible, and therefore cannot be used to publish a schema or to convey information about a schema. As a consequence, a schema matching method can compute a mapping between concepts only indirectly, e.g. by making conjectures about the most plausible interpretation of labels. However, this is clearly the level at which most schema matching methods aim, as we are typically interested in the relation between “meanings” and not between syntactic structures.

Object level: at the object level, we find the objects themselves, namely the objects that the schema is supposed to organize. For HCs, the relevant objects are documents, namely the entities that are associated to nodes in a classification schema, and a mapping may be a set-theoretic relation between pairs of sets of documents. Objects are by definition publicly available. However, as we will argue, the fact that the set of documents associated to a node in a schema is, for example, a subset of the set of documents associated to a node in another schema does not tell us much about the concepts associated to the two sets. In particular, it can’t even tell us that, whatever the relevant concepts are, one subsumes the other, as it may well be that the two sets are not sufficiently representative.

Given these three levels, we can imagine three broad classes of methods: (i) *syntactic methods*, namely methods that use only information at the language level to compute mappings across schemata; (ii) *semantic methods*, namely methods that use only information at the conceptual level; and *pragmatic methods*, namely methods that use only information at the object level. In practice, very few methods can be said to belong to a single category, and for good reasons. For example, as we said, most syntactic methods are (often implicitly) based on the assumptions that labels are meaningful (or even natural language expressions), and this justifies for example the use of thesauri that would not be allowed otherwise (if two nodes PICTURES and PHOTOS were mere abstract labels, what would be the justification for exploiting the idea of synonymy to

match them?). However, and even more important, semantic methods – as we will argue in detail – must start from the language level, as concepts cannot be represented directly; and thus moving from the language level to the concept level is a crucial step for any real world semantic method.

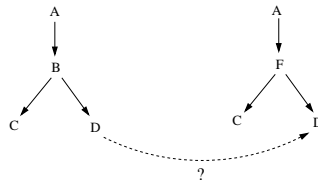
In the rest of the paper, we will disregard purely syntactic methods, as they are of little use in most applications of schema matching³. We will focus on semantic and pragmatic methods. For each of them, we will provide a precise characterization and a notion of soundness.

4 Soundness of semantic methods

Semantic methods are methods which return mappings across concepts associated to nodes in two (or more) schemata. As we argued in the previous section, semantic methods are intrinsically based on two macro steps:

Semantic elicitation: this step takes as input a linguistic description of a node in a schema and returns a representation of its meaning in terms of conceptual representation. As the idea is to design computer-based matching methods, such a meaning must be expressed by some formal object of a logical type, corresponding to the logical type of the node’s intended meaning. Notationally, if n is a node in a HC, then $\mathcal{T}(n)$ denotes the formal representation of its meaning;

³ A simple example will illustrate this claim. Consider the two simple abstract schemata below:



and compare the problem of discovering mappings between nodes of the two abstract schemata with the problem of discovering mappings across schemata with meaningful labels like those in 1. Nodes in abstract schemata do not have an implicit meaning, and therefore, whatever technique we use to map them, we will find that there is some relation between the two nodes labeled D in the two schemata, which depends only on the abstract shape of the two schemata. The situation is completely different for schemata with meaningful labels, as we can make explicit a lot of information that we have about the terms which appear in the graph, and their relations (e.g., that Tuscany is part of Italy, that Florence is in Tuscany, and so on). It is this kind information which allows us to understand why the semantic relation between the two nodes labeled MOUNTAIN and the two nodes labeled FLORENCE is different, despite the fact that the two pairs of schemata are structurally equivalent, and both are structurally isomorphic with the pair of abstract schemata. Indeed, for the first pair of nodes, the set of documents we would classify under the node MOUNTAIN on the left hand side is a subset of the documents we would classify under the node MOUNTAIN on the right; whereas the set of documents which we would classify under the node FLORENCE in the left schema is exactly the same as the set of documents we would classify under the node FLORENCE on the right hand side.

Semantic comparison: given two nodes n and m belonging to different HCs, a semantic method must return a relation which connects the concepts expressing the meanings of the schema elements under comparison. Such a relation must in turn have an interpretation defined with respect to the meaning of the compared elements.

So, according to the first step, a semantic method should explicitly interpret the elements of a HC as concepts, and provide a corresponding formal representation of type concept (e.g., using some Description Logic language [1]). For example, the meaning of the nodes FLORENCE of right and left hand side of schemas of Figure 1 approximately corresponds to the two concepts “Images of Florence in Tuscany” and “Images of 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.

In the second step, a semantic method is supposed to 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. Going back to the example of Figure 1, the relation between the two nodes FLORENCE (interpreted as concepts) is that they are equivalent. We note that, in this case, determining the relation between the two concepts intuitively requires to use further knowledge w.r.t. the one extracted from the two schemata – namely that Tuscany is in Italy. In the following, we will refer to this (possibly external) further 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 provide a formal notion of soundness for semantic methods. Given the two semantic steps 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 ($\mathcal{T}(n)$), (iii) a (possibly empty) ontology \mathcal{O} expressing knowledge about the domain, and (iv) a set of relations \mathfrak{R} to be computed between pairs of nodes. The 4-tuple $\langle \mathcal{L}, \mathcal{O}, \mathcal{T}(), \mathfrak{R} \rangle$ is what we call the *semantic frame* of the method.

We now propose a notion of semantic soundness and completeness of a semantic schema matching method with respect 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 \mathfrak{R} 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 $F = \langle \mathcal{L}, \mathcal{O}, \mathcal{T}(), \mathfrak{R} \rangle$ 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. F if and only if for any mapping element $\langle n_A, n_B, r \rangle$ the following holds:

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

Definition 8 (Semantic Completeness). Let $F = \langle \mathcal{L}, \mathcal{O}, \mathcal{T}(), \mathfrak{R} \rangle$ 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. F if and only if for any two nodes n_A and n_B the following holds:

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

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 about the appropriateness of the meaning elicitation 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 non 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 [13]), 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. From Putnam’s argument, we can derive an even stronger condition: even if subjects shared the function connecting the conceptual level to the linguistic level (and therefore they agreed on the conceptual representation of any statement at the language level), nothing assures us that the function connecting the semantic level to the pragmatic level is also shared. This means that two agents may agree at the conceptual level, but not at the pragmatic level.

Applying this considerations to schema matching methods means that even if we can guarantee that a method is semantically sound and complete, there is nothing that

guarantees that (i) the two elements were correctly interpreted, and (ii) even if they were correctly interpreted, that the relation between the two nodes is the one we expect.

To sum up, assuming the existence of sound reasoners (SAT, DL reasoners, and so on), it seems relatively to come up with a semantically sound method. But this notion of soundness seems to be of little use if we cannot guarantee some form of pragmatic soundness. Let us turn now to this notion.

5 Soundness of pragmatic methods

A pragmatic method is a method which returns mappings across schema elements which depend on some relation between the sets of objects associated to the schema elements themselves. If we restrict our analysis to HCs, a pragmatic method is a method which computes relations between HC nodes through the analysis of set-theoretic relation between the sets of documents actually classified under the two nodes. The intuition underlying pragmatic methods is the following. Suppose we have a collection D of documents, and that a user is required to classify them into two different HCs. Then, if two nodes in two HCs have the “same” meaning, then we may expect that the user will classify the same set of documents under the two nodes; and if the meaning of a node is subsumed by the meaning of another node, then the user will classify under the first node a subset of the documents classified under the second node; and so on. Following this intuition, a pragmatic method may work backward and try to infer the relation between the meaning of two nodes from the relation between the collections of documents associated to the nodes themselves.

In the following, we shall try to make this intuition more precise. Let us first introduce a notation to refer to the set of documents classified under (all the nodes in) a subtree of a topic hierarchy, instead of a single node. The reason is the following. Consider the the right hand side pair of HCs in Figure 1. If we want to compare the nodes labeled TUSCANY and ITALY in the two HCs, it will not be in general sufficient to consider only the documents specifically classified under those two nodes. We should take into account the whole set of documents classified under all nodes belonging to the subtrees rooted in those two nodes. Indeed, any document classified under node FLORENCE is also implicitly classified under node TUSCANY (resp., node ITALY). What one expects in this case is that the set of documents classified in the subtree rooted in TUSCANY be a subset of the set of documents classified in the subtree rooted in ITALY. To capture this intuition, 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 \mathfrak{R} a set of relations between sets of documents (for example, $\mathfrak{R} = \{=, \subseteq, \supseteq, \perp\}$, 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 $r \in \mathfrak{R}$) the following holds:

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

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 ⁴. Notice, however, that this definition presupposes two very strong assumptions:

1. *the set D must be the set of all possible documents.* Indeed, it may well happen that the set of documents actually classified is not sufficient to discriminate between some set-theoretical relations, such as \subset and $=$. In other words, it may be the case that the set of documents considered is not enough to tell two nodes apart, while they will be if we had had more documents available;
2. *each document can be classified in a unique way.* This is not the case in general, as documents are typically *rich objects*, and can be classified under different categories, depending on what aspects of the document are taken as the relevant ones for a given classification task. 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.

To overcome the second assumption, we provide 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. To capture this intuition, we first introduce the following finer notion of classifier:

Definition 10 (Classifier). A classifier C is a set of classification functions $\{\tau_i\}$.

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

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 classifier would consider this result as “acceptable” according to the possible ways he could classify a set

⁴ With an abuse of notation, we use the symbol r to refer both to the (semantic) relation computed by a semantic method and the relation which holds between sets of documents. 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).

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 11 (Pragmatic Soundness). *Let C be a 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 τ_2 of C there is a classification τ_1 of C such that, for any $r \in \mathfrak{R}$, $\mu_{\tau_1}(n_A \downarrow) r \mu_{\tau_2}(n_B \downarrow)$.*

Notice that while this definition allows us to relax the second assumption, the first assumption is still needed to allow for a sensible notion of soundness, and seems to be much harder to relax.

Summarizing, differently to semantic methods, pragmatic methods seem to provide meaningful answers, at least in principle. On the other hand, due to the need to satisfy assumption 1, sound pragmatic methods do not seem to be possible in practice.

6 Can semantic methods be pragmatically sound?

The point we reached can be described as follows. On the one hand, it seems relatively easy to design and implement semantically sound methods, but the answer they provide can be of little use, as we can’t guarantee its pragmatic adequacy. On the other hand, pragmatic methods can provide a provably adequate answer, but they can’t be computed in real cases, due to the strong requirements they presuppose. The solution seems to be either trivial or impossible.

A possible way out of this situation would be to take the advantages of the two methods, while avoiding their drawbacks. This essentially amounts at (i) defining a schema matching method which is semantically sound (as it is ‘easy’ to design), and (ii) to create the conditions under what such method is, at the same time, pragmatically sound (as it provides ‘meaningful’ results). In this section we provide some *preliminary conditions* under which such a result would be possible.

In definition 10 we define a classifier as a set of classification functions. In real cases, we 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. As an example, we expect that a classifier τ classifies a document d (say, a photo of Florence) under a node PHOTO/FLORENCE, or under a node PHOTO/TUSCANY (if the classifier knows that Florence is in Tuscany), and not, say, under a node BOOK/ANIMAL. In other words, a classifier classifies a document with respect to the meaning it associates to the documents. Furthermore, we expect that in presence of both the nodes PHOTO/FLORENCE and PHOTO/TUSCANY, the classifier classifies the document d under the node PHOTO/FLORENCE and not under the node PHOTO/TUSCANY. Essentially, we expect the classifier classifies the documents in the more specific node. When a classifier respects such constraints, is said to be *pragmatically competent*. Such competence intuitively represents a sort of bridge between the semantic and pragmatic spheres: it

says that a document is classified w.r.t. the intended meaning the *classifier associates* to documents itself.

Formally, let $D = \{d_1, d_2, \dots\}$ be the set of all the documents. Furthermore, let $M = \{\phi_1, \phi_2, \dots\}$ be the set of all the intended meanings that can be associated to documents in D . As an example, we could say that the document $d_k \in D$ is the present paper, and that d_k can be associated with the intended meaning ‘semantic web’ (e.g., the topic of the paper), or with the intended meaning ‘paper of 2004’ (e.g., the year when the paper has been written), or the intended meaning ‘submitted paper’ and so on. We assume all these meanings are contained in M . Notationally, we write $\phi^{\tau_j, d_k} \in M$ for indicating the intended meaning in M used by the classification function τ_j for classifying the document d_k .

Definition 12 (Pragmatic competence). *Let $C = \{\tau_i\}$ be a classifier, and $F^C = \langle \mathcal{L}^C, \mathcal{O}^C, \mathcal{T}^C(), \mathfrak{R}^C \rangle$ be a semantic frame. C is pragmatically competent w.r.t F^C if, for any structure \mathcal{H} , for any document $d_k \in D$, for any node $n \in \mathcal{H}$ and for any classification function $\tau_j \in C$, the following holds:*

if $d \in \mu_j(n)$ then:

- a) $\mathcal{O}^C \models_{\mathcal{L}^C} \mathcal{T}^C(n) \sqsupseteq \phi^{\tau_j, d_k}$
- b) for no other node $m \in \mathcal{H}$, $\mathcal{O}^C \models_{\mathcal{L}^C} \mathcal{T}^C(m) \sqsupseteq \phi^{\tau_j, d_k}$ and $\mathcal{T}^C(n) \sqsupseteq \mathcal{T}^C(m)$

The definition above simply requires that a competent classifier classifies documents w.r.t. the intended meaning (requirement a) and under the most specific nodes (requirement b). In particular, a document is classified under a node n if the meaning associated to the document entails the meaning of that node, and if no more specific node m exists into the structure. As an example, imagine our document d_k is a photo of some church in a city of Tuscany (Italy), say Lucca. Imagine that the classifier τ_j associates the intended meaning ‘photo of Lucca’ to the document d_k , say ϕ^{τ_j, d_k} . Furthermore, imagine that the meaning of the node TUSCANY of right hand schema of Figure 1 ($\mathcal{T}^C(\text{TUSCANY})$) is ‘images of Tuscany’. Then, the classification function τ_j is competent if it classifies the document d_k in the node TUSCANY, as a ‘photo of Lucca’ is also an ‘image of Tuscany’ (requirement a) and it doesn’t exist any other more specific node where to classify the document⁵ (requirement b).

A important consequence of Definition 12 is the following:

Proposition 1. *Let $C = \{\tau_i\}$ be a pragmatic competent classifier w.r.t. a semantic frame $F^C = \langle \mathcal{L}^C, \mathcal{O}^C, \mathcal{T}^C(), \mathfrak{R}^C \rangle$. Then, given any two nodes n_A in \mathcal{H}_A and n_B in \mathcal{H}_B , for any $r \in \mathfrak{R}^C$ the following holds: if $\mathcal{O}^C \models \mathcal{T}^C(n_A) r \mathcal{T}^C(n_B)$ then for any classification function $\tau_1 \in C$, there is another $\tau_2 \in C$ such that $\mu_{\tau_1}(n_A \downarrow) r \mu_{\tau_2}(n_B \downarrow)$.*

Proposition 1 simply states 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 w.r.t. the semantic frame F^C , then C must be prepared, possibly by employing some other acceptable classification function of his (namely, a compatible classification function), to classify

⁵ The other node where is possible to classify the node, according to the requirement a, is the node IMAGES. But this is less specific than the node TUSCANY.

under the n_A a set of documents holding the same relation r with the set of documents attached to n_B . Notice that Proposition 1 is an immediate consequence of Definition 12.

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 condition can guarantee that a sound semantic method α is also pragmatically sound⁶. We can state the following proposition:

Proposition 2. *Let $F = \langle \mathcal{L}, \mathcal{O}, \mathcal{T}(\cdot), \mathbb{R} \rangle$ be a semantic frame, α a method semantically sound w.r.t. F , and $C = \{\tau_i\}$ a pragmatically competent classifier with respect to a semantic frame $F^C = \langle \mathcal{L}^C, \mathcal{O}^C, \mathcal{T}^C(\cdot), \mathbb{R} \rangle$. If $\mathcal{O} \sqsubseteq \mathcal{O}^C$ and $\mathcal{T}^C(\cdot) = \mathcal{T}(\cdot)$, then α is pragmatically sound. Moreover, if $|C| = 1$, 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 $\mathcal{T}^C(\cdot)$ and $\mathcal{T}(\cdot)$ is the same (namely, for any node m of any schema \mathcal{S} , $\models \mathcal{T}^C(m) \equiv \mathcal{T}(m)$), 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 Proposition 1 and Definitions 7 and 11. The second part descends from Proposition 1 and Definitions 7, 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) together 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 competent classifier, whatever classification function τ he has used to place documents under node n_A and n_B , by Proposition 1 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 pragmatic 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 already pointed out in Section 4. 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 knowledge 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, as it states that we must determine the ‘right’ meaning (with the respect to the one assigned by the classifier) of each schema element. Notice that weakening the condition on the semantic elicitation functions is problematic. Indeed, a condition as $\models \mathcal{T}(m) \sqsubseteq \mathcal{T}^C(m)$ (which states that the meaning associated to each schema element

⁶ The problem of pragmatic completeness is significantly harder and out of the scope of this paper. We will not discuss it here.

by the matching method is consistent with the meaning associated to the same element by the classifier) preserves the soundness only with respect to the disjointness (\perp). Unfortunately, it doesn't hold with respect to none of the other relations we have been considering in the paper (\sqsubseteq , \sqsupseteq , \equiv).

7 Conclusions

The consequence of Proposition 2 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 ($\mathcal{C} \sqsubseteq \mathcal{O}$) 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 an 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. P. Bouquet, L. Serafini, and S. Zanobini. Semantic coordination: a new approach and an application. In K. Sycara, editor, *Second International Semantic Web Conference (ISWC-03)*, Lecture Notes in Computer Science (LNCS), Sanibel Island (Florida, USA), October 2003.
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 integrative proximity measure for ontology alignment. *Proceedings of the workshop on Semantic Integration*, October 2003.
7. F. Giunchiglia and P. Shvaiko. Semantic matching. *The Knowledge Engineering Review Journal*, 18(3):265–280, 2003.

8. 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.
9. Ryutaro Ichisem, Hiedeaki Takeda, and Shinichi Honiden. Integrating multiple internet directories by instance–base learning. In *AI AND DATA INTEGRATION*, pages 22–28, 2003.
10. Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *The VLDB Journal*, pages 49–58, 2001.
11. 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.
12. Marcello Pelillo, Kaleem Siddiqi, and Steven W. Zucker. Matching hierarchical structures using association graphs. *Lecture Notes in Computer Science*, 1407:3–??, 1998.
13. H. Putnam. *Reason, Truth, and History*. CUP, 1981.
14. 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.
15. 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.