3

Classifications of ontology matching techniques

Having defined what the matching problem is, we attempt at classifying the techniques that can be used for solving this problem. The major contributions of the previous decades are presented in [Larson *et al.*, 1989, Batini *et al.*, 1986, Kashyap and Sheth, 1996, Parent and Spaccapietra, 1998], while the topic through the recent years have been surveyed in [Rahm and Bernstein, 2001, Wache *et al.*, 2001, Kalfoglou and Schorlemmer, 2003b]. These three works address the matching problem from different perspectives (artificial intelligence, information systems, databases) and analyse disjoint sets of systems. [Shvaiko and Euzenat, 2005] have attempted to consider the above mentioned works together, focusing on schema-based matching methods, and aiming to provide a common conceptual basis for their analysis. Here, we follow and extend this work on classifying matching approaches and use it in the following chapters for organising the presentation.

In this chapter we first consider various dimensions along which a classification can be elaborated (§3.1). We then present our classification based on several of these dimensions (§3.2). Finally, we discuss some alternative classifications of matching approaches that have been proposed so far (§3.3).

3.1 Matching dimensions

There are many independent dimensions along which algorithms can be classified. Following the definition of the matching process in Fig. 2.8, we may primarily classify algorithms according to (i) the input of the algorithms, (ii) the characteristics of the matching process, and (iii) the output of the algorithms. The other characteristics, such as parameters, resources, and input alignments, are considered less important. Let us discuss these three main aspects in turn.

3.1.1 Input dimensions

These dimensions concern the kind of input on which algorithms operate. As a first dimension, algorithms can be classified depending on the data or conceptual models in which ontologies are expressed. For example, the Artemis system (§6.1.6) supports the relational, object-oriented, and entity-relationship models; Cupid (§6.1.11) supports XML and relational models; QOM (§6.3.4) supports RDF and OWL models. A second possible dimension depends on the kind of data that the algorithms exploit: different approaches exploit different information in the input ontologies. Some of them rely only on schema-level information, e.g., Cupid (§6.1.11), COMA (§6.1.12); others rely only on instance data, e.g., GLUE (§6.2.5); and others exploit both schema- and instance-level information, e.g., QOM (§6.3.4). Even with the same data models, matching systems do not always use all available constructs, e.g., S-Match (§6.1.19) when dealing with attributes discards information about datatypes, e.g., string or integer, and uses only the attributes names. In general, some algorithms focus on the labels assigned to the entities, some consider their internal structure and the types of their attributes, and others consider their relations with other entities (see next section for details).

3.1.2 Process dimensions

A classification of the matching process could be based on its general properties, as soon as we restrict ourselves to formal algorithms. In particular, it depends on the *approximate* or *exact* nature of its computation. Exact algorithms compute the precise solution to a problem; approximate algorithms sacrifice exactness for performance [Ehrig and Sure, 2004]. All of the techniques discussed in the remainder of the book can be either approximate or exact. Another dimension for analysing the matching algorithms is based on the way they interpret the input data. We identify three large classes based on the intrinsic input, external resources, or some semantic theory of the considered entities. We call these three classes *syntactic*, *external*, and *semantic* respectively, and discuss them in detail in the next section.

3.1.3 Output dimensions

Apart from the information that matching systems exploit and how they manipulate it, the other important class of dimensions concerns the form of the result these systems produce. The form of the alignment might be of importance: is it a one-to-one alignment between the ontology entities? Has it to be a final correspondence? Is any relation suitable?

Some other significant distinctions in the output results have been indicated in [Giunchiglia and Shvaiko, 2003a]. One dimension concerns whether systems deliver a graded answer, e.g., that the correspondence holds with 98% confidence or 4/5 probability; or an all-or-nothing answer, e.g., that the correspondence definitely holds or not. In some approaches, correspondences between ontology entities are determined using distance measures. This is used for provid-

ing an alignment expressing equivalence between these entities. Another dimension concerns the kind of relations between entities a system can provide. Most of the systems focus on equivalence ($=$), while a few others are able to provide a more expressive result, e.g., equivalence, subsumption ($\sqsubseteq$), and incompatibility ($\perp$) [Giunchiglia *et al.*, 2004, Bouquet *et al.*, 2003c].

In the next section we present a classification of elementary techniques that draws simultaneously on these criteria.

3.2 Classification of matching approaches

To ground and ensure a comprehensive coverage for our classification we have analysed state of the art approaches used for ontology matching. Chap. 6 reports a partial list of systems and approaches which have been scrutinised pointing to (some of) the most important contributions. We have used the following guidelines for building our classification:

Exhaustivity: The extension of categories dividing a particular category must cover its extension, i.e., their aggregation should give the complete extension of the category.

Disjointness: In order to have a proper tree, the categories dividing one category should be pairwise disjoint by construction.

Homogeneity: In addition, the criteria used for further dividing one category should be of the same nature, i.e., should come from the same dimension introduced in Sect. 3.1. This usually helps guarantee disjointness.

Saturation: Classes of concrete matching techniques should be as specific and discriminative as possible in order to provide a fine-grained distinction between possible alternatives. These classes have been identified following a *saturation* principle: they have been added and modified until the saturation was reached, i.e., taking into account new techniques did not require introducing new classes or modifying them.

Disjointness and exhaustivity of the categories ensure stability of the classification, namely that new techniques will not occur in between two categories. Classes of matching techniques represent the state of the art. Obviously, with appearance of new techniques, they might be extended and further detailed.

The exact vs. approximate opposition has not been used because each of the methods described below can be implemented as exact or approximate algorithms, depending on the goals of the matching system.

We build on the previous work on classifying automated schema matching approaches of [Rahm and Bernstein, 2001] which distinguishes between *elementary* (individual) matchers and *composition* of matchers. Elementary matchers comprise *instance-* and *schema-based*, *element-* and *structure-level*, *linguistic* and *constraint-based* matching techniques. Also *cardinality* and *auxiliary information*, e.g., thesauri, global schemas, can be taken into account.

For classifying elementary matching techniques, we have introduced two synthetic classifications in [Shvaiko and Euzenat, 2005], based on what we have found to be the most salient properties of the matching dimensions (see Fig. 3.1). These two classifications are presented as two trees sharing their leaves. The leaves represent classes of elementary matching techniques and their concrete examples. Two synthetic classifications are:

– *Granularity/Input Interpretation* classification is based (i) on the matcher granularity, i.e., element- or structure-level, and then (ii) on how the techniques generally interpret the input information.
– *Kind of Input* classification is based on the kind of input which is used by elementary matching techniques.

The overall classification of Fig. 3.1 can be read both in descending (focusing on how the techniques interpret the input information) and ascending (focusing on the kinds of manipulated objects) manner in order to reach the *Basic Techniques* layer.

Elementary matchers are distinguished by the *Granularity/Input interpretation* layer according to the following classification criteria:

– *Element-level vs. structure-level:* Element-level matching techniques compute correspondences by analysing entities or instances of those entities in isolation, ignoring their relations with other entities or their instances. Structure-level techniques compute correspondences by analysing how entities or their instances appear together in a structure. This criterion for schema-based approaches is the same as first introduced in [Rahm and Bernstein, 2001], while element-level vs. structure-level separation for instance-based approaches follows the work in [Kang and Naughton, 2003].
– *Syntactic vs. external vs. semantic:* The key characteristic of the syntactic techniques is that they interpret the input with regard to its sole structure following some clearly stated algorithm. External are the techniques exploiting auxiliary (external) resources of a domain and common knowledge in order to interpret the input. These resources may be human input or some thesaurus expressing the relationships between terms. Semantic techniques use some formal semantics, e.g., model-theoretic semantics, to interpret the input and justify their results. In case of a semantic-based matching system, exact algorithms are complete with regard to the semantics, i.e., they guarantee a discovery of all the possible alignments, while approximate algorithms tend to be incomplete.

To emphasise the differences with the initial classification of [Rahm and Bernstein, 2001], the new categories or classes are marked in bold face. In particular, in the Granularity/Input Interpretation layer we detail further the element- and structure-level matching by introducing the syntactic vs. semantic vs. external criteria. The reasons for having these three categories are as follows. Our initial criterion was to distinguish between internal and external techniques. By *internal* we mean techniques exploiting information which comes only with the input ontologies. *External* techniques are as defined above. Internal techniques can be further detailed by distinguishing between syntactic and semantic interpretation

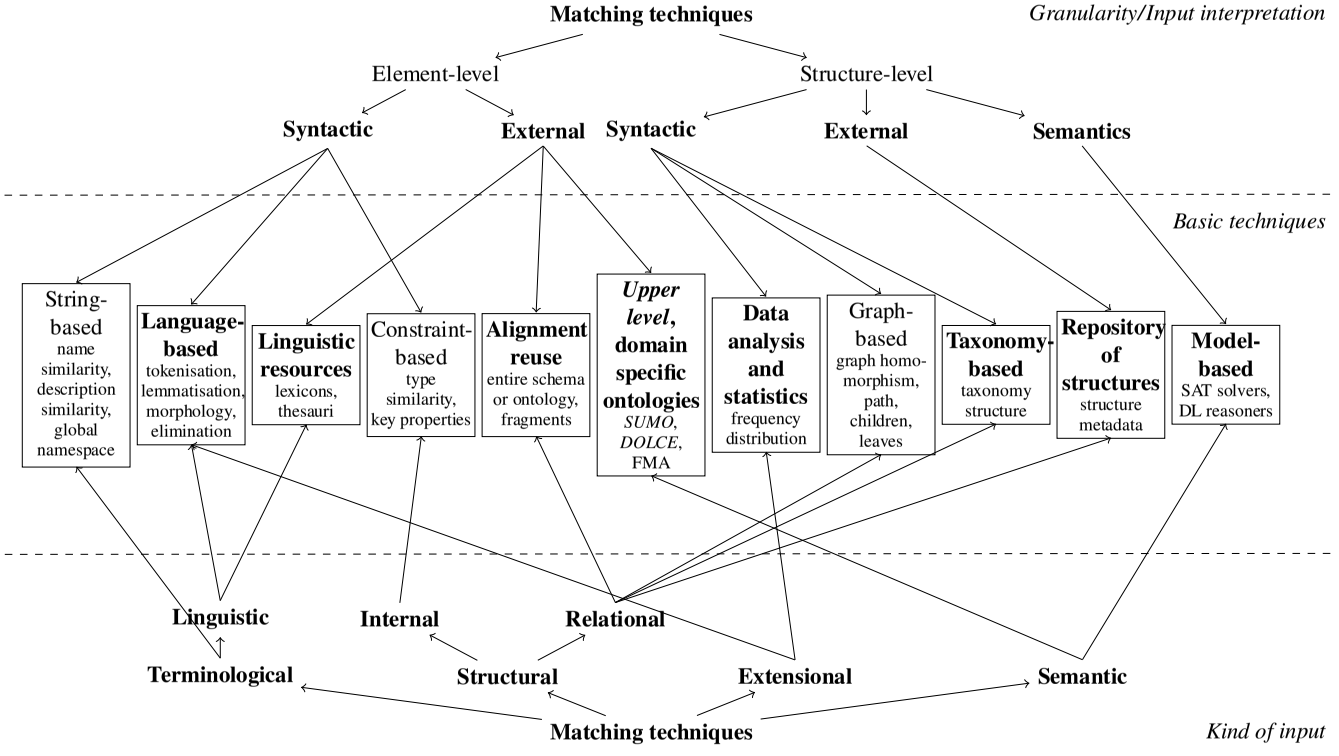

Fig. 3.1. The retained classifications of elementary matching approaches. The upper classification is based on granularity and input interpretation; the lower classification is based on the kind of input. The middle layer features classes of basic techniques. The novelty of this classification in comparison with our previous work in [Shvaiko and Euzenat, 2005] includes *extensional* category of techniques as well as *data analysis and statistics* class of methods.

of input, also as defined above. The same distinction can be introduced, to some extent, for the external techniques. In fact, we can qualify some oracles, e.g., Cyc [Lenat and Guha, 1990], WordNet [Miller, 1995], SUMO [Niles and Pease, 2001], DOLCE [Gangemi *et al.*, 2003], as syntactic or semantic, but not user input. Thus, we do not detail *external* techniques any further and we omit in Fig. 3.1 the theoretical category of *internal* techniques, as opposed to *external*. We also omit in further discussions element-level semantic techniques, since semantics is usually given in a structure, and, hence, there are no element-level semantic techniques.

Distinctions between classes of elementary matching techniques in the *Basic Techniques* layer of our classification are motivated by the way a technique interprets the input information in each concrete case. In particular, a label can be interpreted as a string (a sequence of letters from an alphabet) or as a word or a phrase in some natural language, a hierarchy can be considered as a graph (a set of nodes related by edges) or a taxonomy (a set of concepts having a set-theoretic interpretation organised by a relation which preserves inclusion). Thus, we introduce the following classes of elementary ontology matching techniques at the element-level: string-based, language-based, based on linguistic resources, constraint-based, alignment reuse, and based on upper level and domain specific formal ontologies. At the structure-level we distinguish between graph-based, taxonomy-based, based on repositories of structures, model-based, and data analysis and statistics techniques.

The *Kind of Input* layer classification is concerned with the type of input considered by a particular technique:

- The first level is categorised depending on which kind of data the algorithms work on: strings (*terminological*), structure (*structural*), models (*semantics*) or data instances (*extensional*). The two first ones are found in the ontology descriptions. The third one requires some semantic interpretation of the ontology and usually uses some semantically compliant reasoner to deduce the correspondences. The last one constitutes the actual population of an ontology.
- The second level of this classification decomposes further these categories if necessary: terminological methods can be string-based (considering the terms as sequences of characters) or based on the interpretation of these terms as linguistic objects (*linguistic*). The structural methods category is split into two types of methods: those which consider the internal structure of entities, e.g., attributes and their types (*internal*), and those which consider the relation of entities with other entities (*relational*).

Following the above mentioned guidelines for building a classification, the terminological category should be divided into linguistic and non linguistic techniques. However, since non linguistic techniques are all string-based, this category has been discarded. This presentation is the one followed in the presentation of basic techniques (Chap. 4).

We discuss below the main classes of the *Basic Techniques* layer according to the above classification in more detail. The order follows that of the *Granularity/Input Interpretation* classification and these techniques are divided in two sections concerning element-level techniques (§3.2.1) and structure-level techniques (§3.2.2). Fi-

nally, techniques which are marked in italic in Fig. 3.1 (techniques based on upper level ontologies) have not been implemented in any matching system yet. However, we argue that their appearance seems reasonable in the near future.

3.2.1 Element-level techniques

Element-level techniques consider ontology entities or their instances in isolation from their relations with other entities or their instances.

String-based techniques

String-based techniques are often used in order to match names and name descriptions of ontology entities. These techniques consider strings as sequences of letters in an alphabet. They are typically based on the following intuition: the more similar the strings, the more likely they are to denote the same concepts. Usually, distance functions map a pair of strings to a real number, where a smaller value of the real number indicates a greater similarity between the strings. Some examples of string-based techniques which are extensively used in matching systems are prefix, suffix, edit, and n-gram distances. Various such string comparison techniques are presented in Sect. 4.2.1.

Language-based techniques

Language-based techniques consider names as words in some natural language, e.g., English. They are based on natural language processing techniques exploiting morphological properties of the input words. Several of these techniques are presented in Sect. 4.2.2 (intrinsic techniques).

Usually, they are applied to names of entities before running string-based or lexicon-based techniques in order to improve their results. However, we consider these language-based techniques as a separate class of matching techniques, since they can be naturally extended, for example, in a distance computation (by comparing the resulting strings or sets of strings).

Constraint-based techniques

Constraint-based techniques are algorithms which deal with the internal constraints being applied to the definitions of entities, such as types, cardinality (or multiplicity) of attributes, and keys. These techniques are presented in Sect. 4.3.1.

Linguistic resources

Linguistic resources such as lexicons or domain specific thesauri are used in order to match words (in this case names of ontology entities are considered as words of a natural language) based on linguistic relations between them, e.g., synonyms, hyponyms. Several such methods are presented in Sect. 4.2.2 (extrinsic techniques).

Alignment reuse

Alignment reuse techniques represent an alternative way of exploiting external resources, which record alignments of previously matched ontologies. For instance, when we need to match ontology o' and o'', given the alignments between o and o', and between o and o'' available from the external resource. Alignment reuse is motivated by the intuition that many ontologies to be matched are similar to already matched ontologies, especially if they are describing the same application domain. These techniques are particularly promising when dealing with large ontologies consisting of hundreds and thousands of entities. In these cases, first, large match problems are decomposed into smaller subproblems, thus generating a set of ontology fragments matching problems. Then, reuse of previous match results can be more effectively applied at the level of ontology fragments rather than at the level of entire ontologies. The approach was first introduced in [Rahm and Bernstein, 2001] and was later implemented as two matchers, i.e., (i) reuse alignments of entire ontologies, or (ii) their fragments [Do and Rahm, 2002, Aumüller *et al.*, 2005, Rahm *et al.*, 2004].

Upper level and domain specific formal ontologies

Upper level ontologies can also be used as external sources of common knowledge. Examples are the upper Cyc ontology [Lenat and Guha, 1990], the Suggested Upper Merged Ontology (SUMO) [Niles and Pease, 2001] and Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [Gangemi *et al.*, 2003]. The key characteristic of these ontologies is that they are logic-based systems, and therefore, matching techniques exploiting them are based on semantics. For the moment, we are not aware of any matching system which uses this kind of techniques. However, it is quite reasonable to assume that this will happen in the near future. In fact, for example, the DOLCE ontology aims at providing a formal specification (axiomatic theory) for the top level part of WordNet. Therefore, systems exploiting WordNet in their matching process may also consider using DOLCE as a potential semantic extension.

Domain specific formal ontologies can also be used as external sources of background knowledge. Such ontologies are focusing on a particular domain and use terms in a sense that is relevant only to this domain and which is not related to similar concepts in other domains. For example, in the anatomy domain, an ontology such as The Foundational Model of Anatomy (FMA) can be used as the context for the other medical ontologies to be matched (as long as it is known that the reference ontology covers the ontologies to be matched). This can be used for providing the missing structure when matching poorly structured resources [Aleksovski *et al.*, 2006]. These methods are discussed in Sect. 4.5.1.

3.2.2 Structure-level techniques

Contrary to element-level techniques, structure-level techniques consider the ontology entities or their instances to compare their relations with other entities or their instances.

Graph-based techniques

Graph-based techniques are graph algorithms which consider the input ontologies as labelled graphs. The ontologies (including database schemas, and taxonomies) are viewed as labelled graph structures. Usually, the similarity comparison between a pair of nodes from the two ontologies is based on the analysis of their positions within the graphs. The intuition behind this is that, if two nodes from two ontologies are similar, their neighbours must also be somehow similar. Different graph-based techniques are described in Sect. 4.3.2.

Along with purely graph-based techniques, there are other more specific structure-based techniques, for instance, involving trees.

Taxonomy-based techniques

Taxonomy-based techniques are also graph algorithms which consider only the specialisation relation. The intuition behind taxonomic techniques is that *is-a* links connect terms that are already similar (being interpreted as a subset or superset of each other), therefore their neighbours may be also somehow similar. This intuition can be exploited in several different ways presented in Sect. 4.3.2.

Repository of structures

Repositories of structures store ontologies and their fragments together with pairwise similarity measures, e.g., coefficients in the $[0\ 1]$ range between them. Unlike alignment reuse, repositories of structures store only similarities between ontologies, not alignments. In the following, to simplify the presentation, we call ontologies, or their fragments, as structures. When new structures are to be matched, they are first checked for similarity against the structures which are already available in the repository. The goal is to identify structures which are sufficiently similar to be worth matching in more detail, or reusing already existing alignments, thus, avoiding the match operation over the dissimilar structures. Obviously, the determination of similarity between structures should be computationally cheaper than matching them in full detail. The approach of [Rahm *et al.*, 2004] to matching two structures proposes to use some metadata describing these structures, such as structure name, root name, number of nodes, maximal path length, etc. These indicators are then analysed and aggregated into a single coefficient, which estimates similarity between them. For example, two structures may be found as an appropriate match if they both have the same number of nodes.

Model-based techniques

Model-based (or semantically grounded) algorithms handle the input based on its semantic interpretation, e.g., model-theoretic semantics. The intuition is that if two entities are the same, then they share the same interpretations. Thus, they are well grounded deductive methods. Examples are propositional satisfiability and description logics reasoning techniques. They are further reviewed in Sect. 4.5.

Data analysis and statistics techniques

Data analysis and statistical techniques are those which take advantage of a (hopefully large) representative sample of a population in order to find regularities and discrepancies. This helps in grouping together items or computing distances between them. Among data analysis techniques we discuss distance-based classification, formal concepts analysis (§4.4.1) and correspondence analysis; among statistical analysis methods we consider frequency distributions.

We exclude from this category learning techniques which require a sample of the result. These techniques are considered specifically in Sect. 5.4 as strategies.

3.3 Other classifications

Let us now consider some other available classifications of matching techniques.

[Ehrig, 2007] introduced a classification based on two orthogonal dimensions. These can be viewed as horizontal and vertical dimensions. The horizontal dimension includes three layers that are built one on top of another:

Data layer: This is the first layer. Matching between entities is performed here by comparing only data values of simple or complex datatypes.

Ontology layer: This is the second layer which, in turn, is further divided into four levels, following the 'layer cake' of [Berners-Lee *et al.*, 2001]. These are semantic nets, description logics, restrictions and rules. For example, at the level of semantic nets, ontologies are viewed as graphs with concepts and relations, and, therefore, matching is performed by comparing only these. The description logics level brings a formal semantics account to ontologies. Matching at this level includes, for example, determining taxonomic similarity based on the number of subsumption relations separating two concepts. This level also takes into account instances of entities, therefore, for example, assessing concepts to be the same, if their instances are similar. Matching at the levels of restrictions and rules is typically based on the idea that if similar rules between entities exist, these entities can be regarded as similar. This typically requires processing higher order relations.

Context layer: Finally, this layer is concerned with the practical usage of entities in the context of an application. Matching is performed here by comparing the usages of entities in ontology-based applications. One of the intuitions behind such matching methods is that similar entities are often used in similar contexts.

The vertical dimension represents specific *domain knowledge* which can be situated at any layer of the horizontal dimension. Here, the advantage of external resources of domain specific knowledge, e.g., Dublin Core for the bibliographic domain, is considered for assessing the similarity between entities of ontologies.

[Doan and Halevy, 2005] classifies matching techniques into (i) rule-based and (ii) learning-based. Typically, rule-based techniques work with schema-level information, such as entity names, datatypes and structures. Some examples of rules are that two entities match if their names are similar or if they have the same number of neighbour entities. Learning-based approaches often work with instance-level information, thereby performing matching, for example, by comparing value formats and distributions of data instances underlying the entities under consideration. However, learning can also be done at the schema-level and from the previous matches, e.g., as proposed in the LSD approach (§6.2.4).

[Zanobini, 2006] classifies matching methods into three categories following the cognitive theory of meaning and communication between agents:

Syntactic: This category represents methods that use purely syntactic matching methods. Some examples of such methods include string-based techniques, e.g., edit distance between strings (§4.2.1) and graph matching techniques, e.g., tree edit distance (§4.3.2).

Pragmatic: This category represents methods that rely on comparison of data instances underlying the entities under consideration in order to compute alignments. Some examples of such methods include automatic classifiers, e.g., Bayesian classifier (§4.4), and formal concepts analysis (§4.4.1).

Conceptual: This category represents methods that work with concepts and compare their meanings in order to compute alignments. Some examples of such methods include techniques exploiting external thesauri, such as WordNet (§4.2.2), in order to compare senses among the concepts under consideration.

There were also some classifications mixing the process dimension of matching together with either input dimension or output dimension. For example, [Do, 2005] extends the work of [Rahm and Bernstein, 2001] by adding a *reuse-oriented* category of techniques on top of schema-based vs. instance-based separation, meaning that reuse-oriented techniques can be applied at schema and instance level. However, these techniques can also include some input information, such as user input or alignments obtained from previous match operations.

[Giunchiglia and Shvaiko, 2003a] classified matching approaches into *syntactic* and *semantic*. At the matching process dimension these correspond to syntactic and conceptual categories of [Zanobini, 2006], respectively. However, these have been also constrained by a second condition dealing with the output dimension: syntactic techniques return coefficients in the $[0\ 1]$ range, while semantic techniques return logical relations, such as equivalence, subsumption. Combining methods that work with concepts as well as return logical relations has been defined as a semantic matching problem in [Giunchiglia and Shvaiko, 2003a].

Finally, we notice that the more the ontology matching field progresses, the wider the variety of techniques that come into use at different levels of granularity. For ex-

ample, machine learning methods, which where often applied only to the instance level information, also started being applied more widely to schema level information. We believe that such a cross-fertilisation will gain more evidence in future. Therefore, ultimately, it could be the case that any mathematical method will find appropriate uses for ontology matching.

3.4 Summary

Following the complexity of ontology definition, there is a variety of techniques that can be used. The classifications discussed in this chapter provide a common conceptual basis for organising them, and, hence, can be used for comparing (analytically) different existing ontology matching systems as well as for designing new ones, taking advantages of state of the art solutions. The classifications of matching methods also provide some guidelines which help in identifying families of matching approaches.

This chapter has shown the difficulty of having a clear cut classification of algorithms. In Sect. 3.2 we provided two such classifications based on granularity and input interpretation on the one side and the kind of input on the other side. They will be used for organising the presentation of basic techniques in the next chapter.