9

Explaining alignments

Matching systems may produce effective alignments that may not be intuitively obvious to human users. In order for users to trust the alignments, and thus use them, they need information about them, e.g., they need access to the sources that were used to determine semantic correspondences between ontology entities. Explanations are also useful when matching large applications with thousands of entities, e.g., business product classifications, such as UNSPSC and eCl@ss. In such cases, automatic matching solutions will find many plausible correspondences, and hence user input is required for performing cleaning-up of the alignment. Finally, explanations can also be viewed and applied as argumentation schemas for negotiating alignments between agents.

In this chapter we describe how a matching system can explain its answers, thus making the matching result intelligible. The material of this chapter is mainly based on the work in [Shvaiko *et al.*, 2005, McGuinness and Pinheiro da Silva, 2004, Dhamankar *et al.*, 2004, Laera *et al.*, 2006]. We first present the information required for providing explanations of matching and alignments (§9.1). Then, we discuss approaches to explanations of matching by examples of existing systems (§9.2). In turn, details of these approaches are provided in sequel, including default explanations (§9.3), explaining the basic matchers (§9.4), explaining the matching process (§9.5), and negotiating alignments by argumentation (§9.6).

9.1 Justifications

We have presented the matching process as the use of basic matchers combined by strategies. In order to provide explanations to users it is necessary to have information on both matters. In particular, this information involves justifications on the reason why a correspondence should hold or not.

Each correspondence can be assigned one or several justifications that support or infirm the correspondence. We call them *justified correspondences*. For instance, the justified correspondence:

$$\langle e, e', n, \leq, \text{`}I(e) \subseteq I(e')\text{'} \rangle$$

expresses that the correspondence $\langle e, e', n, \leq \rangle$ is thought to hold because '$I(e) \subseteq I(e')$' is verified. Similarly:

$$\langle e, e', n, =, \text{`}DPLL\ entailed\text{'} \rangle$$

expresses that the correspondence $\langle e, e', n, = \rangle$ is thought to hold because it has been proved by the Davis–Putnam–Longemann–Loveland (DPLL) procedure [Davis and Putnam, 1960, Davis *et al.*, 1962].

In fact, justifications can be largely more complex than presented above. For instance, the second justification may involve a full proof of the correspondence and the axioms involved in that proof. This justification information can be found directly within the correspondences or provided on-demand by the matchers to the system requiring explanation.

We explore below what can be found in this justification part.

9.1.1 Information about basic matchers

When matching systems return alignments, users may not know which external sources of background knowledge were used, when these sources were updated, or whether the resulting correspondences was looked up or derived. However, ultimately, human users or agents have to make decisions about the alignments in a principled way. So, even when basic matchers simply rely on some external source of knowledge, users may need to understand where the information comes from, with different levels of detail.

Following [McGuinness and Pinheiro da Silva, 2004], we call information about the origins of asserted facts the provenance information. Some examples of this kind of information include:

- external knowledge source name, e.g., WordNet;
- date and authors of original information;
- authoritativeness of the source, that is whether it is certified as reliable by a third party;
- name of a basic matcher, version, authors, etc. If the basic matcher relies on a logical reasoner, such as a SAT solver, some more meta-information about the reasoner may be made available:
  - the reasoning method, e.g., the Davis–Putnam–Longemann–Loveland procedure;
  - properties, e.g., soundness and completeness characteristics of the result returned by the reasoner;
  - reasoner assumptions, e.g., closed world vs. open world.

Additional types of information may also be provided, such as a degree of belief for an external source of knowledge from a particular community, computed by using some social network analysis techniques.

9.1.2 Process traces

Matching systems typically combine multiple matchers (see Chap. 5). The final alignment is usually a result of synthesis, abstraction, deduction, and some other manipulations of their results. Thus, users may want to see a trace of the performed manipulations. We refer to them as process traces. Some examples of this kind of information include:

- a trace of rules or strategies applied;
- support for alternative paths leading to a single conclusion;
- support for accessing the implicit information that can be made explicit from any particular reasoning path.

Users may also want to understand why a particular correspondence was not discovered, or why a discovered correspondence was ranked in a particular place, thereby being included in or excluded from the final alignment.

9.2 Explanation approaches

The goal of explanations is to take advantage of the above mentioned types of information for rendering the matching process intelligible to the users. A key issue is to represent explanations in a simple and clear way [Léger *et al.*, 2005].

In fact, while knowledge provenance and process traces may be enough for experts when they attempt to understand why a correspondence was returned, usually they are inadequate for ordinary users. Thus, raw justifications have to be transformed into an understandable explanation for each of the correspondences. Techniques are required for transforming raw justifications and rewriting them into abstractions that produce the foundation for what is presented to users. Presentation support also needs to be provided for users to better understand explanations. Human users will need help in asking questions and obtaining answers of a manageable size. Additionally, agents may even need some control over requests, such as the ability to break large process traces into appropriate size portions, etc. Based on [McGuinness and Pinheiro da Silva, 2004], requirements for process presentation may include:

- methods for breaking up process traces into manageable pieces;
- methods for pruning process traces and explanations to help users find relevant information;
- methods for explanation navigation, including the ability to ask follow-up questions;
- methods for obtaining alternative justifications for answers;
- different presentation formats, e.g., natural language, graphs, and associated translation techniques;
- methods for obtaining justifications for conflicting answers;
- abstraction techniques.

There are several approaches to provide explanations of the answers from matching systems. We describe below three such approaches. There are, however, few works on the topic in the literature and even fewer implemented systems. So, this chapter more specifically describes the explanation approaches as implemented in two systems, namely S-Match (§6.1.19) and iMAP (§6.2.6).

9.2.1 The proof presentation approach

Semantic matchers usually produce formal proofs of their inferences as the basis for a correspondence. They can thus benefit from work developed for displaying and explaining proofs.

For instance, S-Match [Shvaiko *et al.*, 2005] has been extended to use the Inference Web infrastructure as well as the Proof Markup Language (PML) [McGuinness and Pinheiro da Silva, 2003, Pinheiro da Silva *et al.*, 2006]. Thus, meaningful fragments of S-Match proofs can be loaded on demand. Users can browse an entire proof or they can restrict their view and refer only to specific, relevant parts of proofs. The proof elements are also connected to information about basic matchers that generated the hypotheses.

9.2.2 The strategic flow approach

Many matchers are composed of other matchers and have to decide in favour of some particular results over others. This composition and decision flow can be recorded in a dependency graph and used for providing explanation to users.

For instance, iMAP [Dhamankar *et al.*, 2004] records dependencies at a very precise level (correspondence per correspondence) and can provide users with justifications for (i) existing correspondences, (ii) absent correspondences, and (iii) correspondence ranking. It provides explanations by extracting in the dependency graph the part that has an influence on the choice of a correspondence and generates an explanation in English from this extracted subgraph.

9.2.3 The argumentation approach

The argumentation approach considers the justifications or arguments in favour or against specific correspondences. Argumentation theories can determine, from a set of arguments, the correspondences which will be considered to hold and those which will not.

Argumentation can be applied to justify the matching results to users on the basis of the arguments and counter-arguments or to negotiate the correspondences that should be in an alignment. So far, this approach has mainly been applied to agents negotiating the alignments [Laera *et al.*, 2006] rather than for explaining them. The argumentative approach is different from the proof presentation approach because it does not follow the formal proof of the correspondences. It is also more suitable when no such a proof exists.

9.3 A default explanation

A default explanation of alignments should be a short, natural language, high-level explanation without any technical details. It is designed to be intuitive and understandable by ordinary users.

9.3.1 The S-Match example

We concentrate on class matching and motivate the problem by the simple catalogue matching example shown in Fig. 9.1. Let us suppose that an agent wants to exchange or to search for documents with another agent. The documents of both agents are stored in catalogues according to class hierarchies o and o', respectively. S-Match takes as input these hierarchies, decomposes the tree matching problem into a set of node matching problems, which are, in turn, translated into a propositional validity problem, which can then be efficiently resolved using sound and complete SAT solver (§4.5.2).

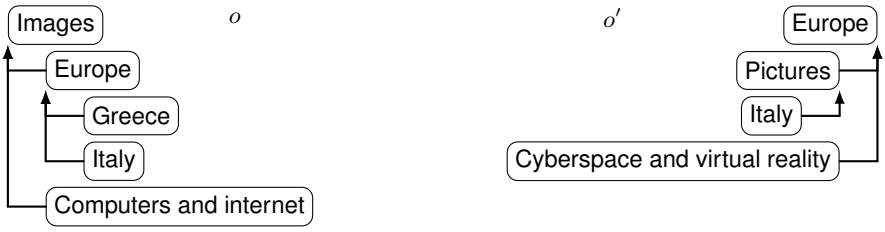

Fig. 9.1. Simple catalogue matching problem.

From the example in Fig. 9.1, trying to prove that the node with label Europe in o (denoted as Europe) is equivalent to the node with label Pictures in o' (denoted as Pictures'), requires constructing the following formula (see Sect. 4.5.2 for details of formula construction):

$$\underbrace{((\text{Images} \equiv \text{Pictures'}) \wedge (\text{Europe} \equiv \text{Europe'}))}_{Axioms} \rightarrow$$
$$((\underbrace{\text{Images} \wedge \text{Europe}}_{Context_c}) \equiv (\underbrace{\text{Europe'} \wedge \text{Pictures'}}_{Context_{c'}}))$$

In this example, the negated formula is unsatisfiable, thus the equivalence relation holds between the nodes under consideration.

Let us suppose that agent o' is interested in knowing why S-Match suggested a set of documents stored under the node with label Europe in o as the result to the query – 'find European pictures'. A default explanation is presented in Fig. 9.2. To simplify

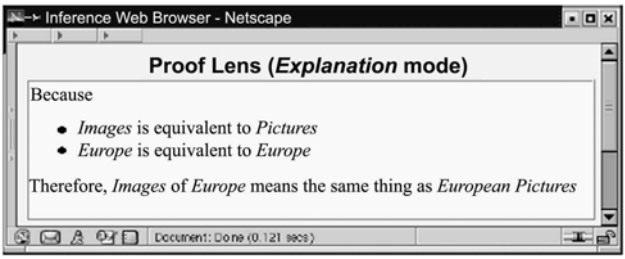

Fig. 9.2. S-Match explanation in English.

the presentation, whenever it is clear from the context to which classification a label under consideration belongs to, we do not tag it with the prime symbol (').

From the explanation in Fig. 9.2, users may learn that Images in o and Pictures in o' can be interchanged, in the context of the query. Users may also learn that Europe in o denotes the same concept as Europe (European) in o'. Therefore, they can conclude that Images of Europe means the same thing as European Pictures.

9.3.2 The iMAP example

iMAP differs substantially from S-Match. It is based on a combination of constraint- and instance-based basic matchers. Once the matchers have produced the candidate correspondences, a *similarity estimator* computes, for each candidate, its similarity score. Finally, by applying the *match selector* the best matches are returned as the final alignment.

Let us consider how iMAP explains why pname = last-name is ranked higher than concat(first-name, last-name). Fig. 9.3 shows the explanation as produced by iMAP [Dhamankar *et al.*, 2004].

```
iMAP:
(1) Searcher Level:
  text searcher generated last-name
  text searcher generated concat(first-name,last-name)
  text searcher ranked concat(first-name,last-name) higher by 0.102
(2) Similarity Estimator Level:
  Name-based evaluator ranked last-name higher by 0.0419
  Naive Bayes evaluator ranked concat(first-name,last-name) higher by 0.01
  Final score for last-name: 0.434
  Final score for concat(first-name,last-name): 0.420
(3) Match Selector Level:
  Match selector did not modify two candidates
(4) Greatest influence on top five candidates for pname: Name-based evaluator
```

Fig. 9.3. iMAP explanation in English.

At the matcher level, concat(first-name, last-name) was ranked higher than the element with label last-name. It also clearly shows that things went wrong at the

similarity estimator level. The naive Bayes evaluator still ranked matches correctly, but the name-based evaluator flipped the ranking, which was the cause of the ranking mistake.

The last line of the explanation also confirmed the above conclusion, since it states that the name-based evaluator has the greatest influence on the top five match candidates for pname. Thus, the main reason for the incorrect ranking for pname appears to be that the name-based evaluator has too much influence. This explanation would allow users to fine tune the system, possibly by reducing the weight of the name-based evaluator in the score combination step.

Users may not be satisfied with this level of explanations. Let us therefore discuss how they can investigate the details of the matching process by exploiting more verbose explanations, which are discussed in the forthcoming sections.

9.4 Explaining basic matchers

Explaining basic matchers requires only to formulate the justification information. It is illustrated through S-Match.

Let us suppose that an agent wants to see the sources of background knowledge used in order to determine the correspondence. For example, which applications, publications, other sources, have been used to determine that Images is equivalent to Pictures. Fig. 9.4 presents the source metadata for the default explanation of Fig. 9.2.

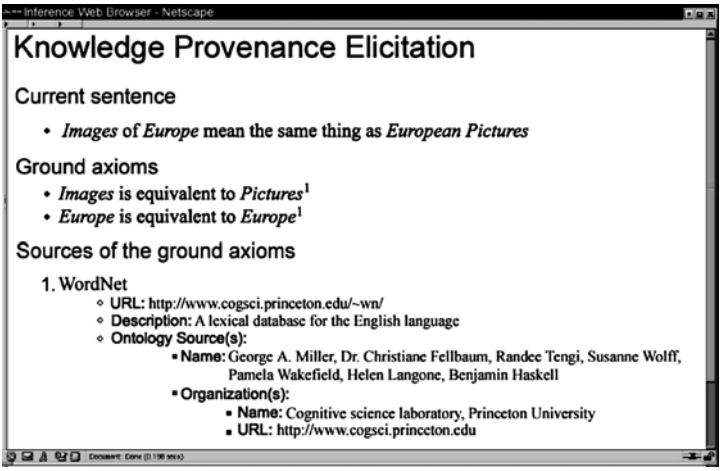

Fig. 9.4. S-Match source metadata information.

In this case, both (all) the ground sentences used in the S-Match proof came from WordNet. Using WorldNet, S-Match learnt that the first sense of the word Pictures

is a synonym to the second sense of the word Images. Therefore, S-Match can conclude that these two words are equivalent words in the context of the answer (§4.2.2). The meta-information about WordNet is also presented in Fig. 9.4 as *sources of the ground axioms*. Further examples of explanations include providing meta information about the S-Match library of element-level matchers, i.e., those which are based not only on WordNet, or the order in which the matchers are used. This use of metadata is not restricted to S-Match and can be applied to any resource used in matching.

9.5 Explaining the matching process

S-Match and iMAP follow different matching strategies. iMAP follows a learning-based solution, while S-Match reduces the matching problem to a propositional validity problem. Let us discuss how they explain the matching process.

9.5.1 Dependency graphs

Explanations of alignments in the iMAP system are based on the idea of a *dependency graph*, which traces the matchers, memorising relevant slices of the graph used to determine a particular correspondence. Finally, exploiting the dependency graph, explanations are presented to users as shown in Fig. 9.3.

The dependency graph is constructed during the matching process. It records the flow of matches, data and assumptions into and out of system components. The nodes of the graph are schema attributes, assumptions made by system components, candidate correspondences, etc. Two nodes in the graph are connected by a directed edge if one of them is the successor of the other in the decision process. The label of the edge is the system component that was responsible for the decision.

Fig. 9.5 shows a dependency graph fragment that records the creation and flow for the correspondence month-posted $=$ monthly-fee-rate. The *preprocessor* finds that both month-posted and monthly-fee-rate have values between 1 and 12 and hence makes the assumptions that they represent months. The date matcher takes these assumptions and generates month-posted $=$ monthly-fee-rate as a candidate correspondence. This candidate is then scored by the name-based evaluator and the naive Bayes evaluator. The scores are merged by a *combining module* to produce a single score. The *match selector* acts upon the several alignment candidates generated to produce the final list of alignments. Here, for the target attribute list-price, the selector reduces the rank of the candidate correspondence price $*$ $(1 +$ monthly-fee-rate$)$ since it discovers that monthly-fee-rate maps to month-posted.

In each case, the system synthesises an explanation in English for the users. To provide explanations, iMAP selects the relevant slices of dependency graph that record the creation and processing of a particular correspondence. For example, the slice for month-posted $=$ monthly-fee-rate is the portion of the graph where the nodes participated in the process of creating that correspondence.

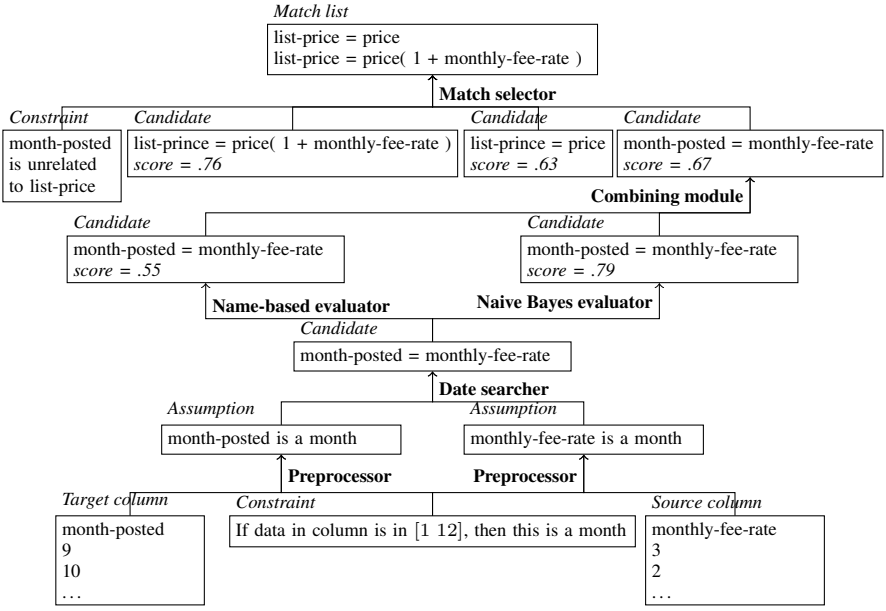

Fig. 9.5. Dependency graph as generated by iMAP [Dhamankar *et al.*, 2004].

9.5.2 Explaining logical reasoning

A complex explanation may be required if users are not familiar with or do not trust the inference engine(s) embedded in a matching system. As the web starts to rely more on information manipulations, instead of simply information retrieval, explanations of embedded manipulation or inference engines become more important. In the current version of S-Match, a propositional satisfiability engine is used (§6.1.19), more precisely, this is the Davis–Putnam–Longemann–Loveland procedure [Davis and Putnam, 1960, Davis *et al.*, 1962] as implemented in JSAT/SAT4J [Le Berre, 2004].

The task of a SAT solver is to find an assignment $\mu \in \{\top, \bot\}$ for atoms of a propositional formula φ such that φ evaluates to *true*. φ is *satisfiable* if and only if $\mu \models \varphi$ for some μ. If μ does not exist, φ is *unsatisfiable*. A *literal* is a propositional atom or its negation. A *clause* is a disjunction of one or more literals. φ is said to be in conjunctive normal form if and only if it is a conjunction of disjunctions of literals. The basic DPLL procedure recursively implements three rules: *unit resolution*, *pure literal* and *split*. We only consider the unit resolution rule to facilitate the presentation.

Let l be a literal and φ a propositional formula in conjunctive normal form. A clause is called a *unit clause* if and only if it has exactly one unassigned literal. *Unit resolution* is an application of *resolution* to a unit clause.

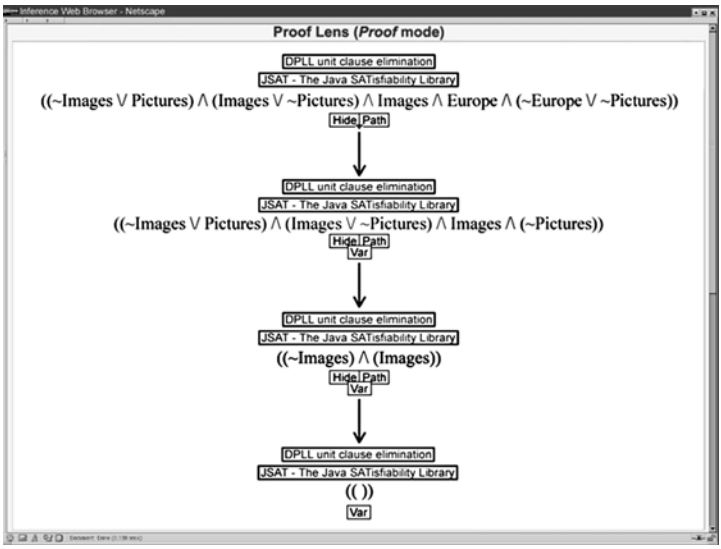

Fig. 9.6. A graphical explanation of the unit clause rule.

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

Let us consider the propositional formula standing for the problem of testing if the concept at node with label Europe in o is less general than the concept at node with label Pictures in o' in Fig. 9.1. The propositional formula encoding the above stated matching problem is as follows:

$$((\mathsf{Images} \equiv \mathsf{Pictures}) \wedge (\mathsf{Europe} \equiv \mathsf{Europe})) \rightarrow ((\mathsf{Images} \wedge \mathsf{Europe}) \rightarrow (\mathsf{Europe} \wedge \mathsf{Pictures}))$$

Its intuitive reading is 'Assuming that Images and Pictures denote the same concept, is there any situation such that the concept Images of Europe is less general than the concept European Pictures?'. The proof of the fact that this is not the case is shown in Fig. 9.6. Since the DPLL procedure of JSAT/SAT4J only handles conjunctive normal form formulas, in Fig. 9.6, we show the conjunctive normal form of the above formula.

From the explanation in Fig. 9.6, users may learn that the proof of the fact that the concept at node with label Europe in o is less general than the concept at node with label Pictures in o' requires 4 steps and at each proof step (excepting the first one, which is a problem statement) the *unit resolution* rule is applied. Moreover, users may learn the assumptions that are made by JSAT/SAT4J. For example, at the second step, the DPLL procedure assigns the truth value to all instances of the atom Europe, therefore making an assumption that there is a model where what an agent says about Europe is always true. According to the *unit resolution* rule, the atom

Europe should then be deleted from the input sentence, and, hence it does not appear in the sentence of the step 2.

The explanation of Fig. 9.6 represents some technical details (only the less generality test) of the default explanation in Fig. 9.2. This type of explanations is the most verbose. It assumes that, even if the graphical representation of a decision tree is quite intuitive, the matching system users have some background knowledge in logics and SAT. However, if they do not, they have a possibility to learn it by following the publications mentioned in the source metadata information of the DPLL *unit resolution* rule and JSAT, by clicking the *DPLL unit clause elimination* and the *JSAT-The Java SATisfiability Library* buttons, respectively.

9.6 Arguing about correspondences

The goal of argumentation is not strictly to explain the alignments, but to give arguments in favour or against the correspondences. It can have two roles:

- negotiating an alignment between two agents, if they accept each others arguments,
- achieving an alignment through matching. In particular, the multiagent negotiation of alignments can be seen as another aggregation technique (§5.2) between two alignments. [Silva *et al.*, 2005] presents such a system based on quantitative negotiation rather than arguments.

Argumentation allows agents to provide counter-arguments and to choose the arguments depending on their preferences. Contrary to the usual explanation work presented above, each agent can generate its own explanation by assembling arguments.

Let us consider two agents C and P using respectively ontology o and o', expressed in description logic as follows:

$$o = \{\mathsf{Micro\text{-}company} = \mathsf{Company} \sqcap \leq_5 \mathsf{employee}\}$$
$$o' = \{\mathsf{SME} = \mathsf{Firm} \sqcap \leq_{10} \mathsf{associate}\}$$

Let us suppose that they have discovered alignment A:

$$A = \{\langle \mathsf{Company}, \mathsf{Firm}, =, .89\rangle, \quad (\gamma_1)$$
$$\langle \mathsf{employee}, \mathsf{associate}, \sqsubseteq, 1.0\rangle, \quad (\gamma_2)$$
$$\langle \mathsf{Micro\text{-}company}, \mathsf{SME}, \sqsubseteq, .97\rangle\} \quad (\gamma_3)$$

The three correspondences are denoted, respectively, as γ_1, γ_2 and γ_3. The set of arguments in favour of γ_1 include:

a_1 all the known Company on the one side are Firm on the other side and vice versa;
a_2 the two names Company and Firm are synonyms in WordNet;

The set of arguments in favour of γ_3 include:

a_3 the alignment (without γ_3) plus the two ontologies entail the correspondence;
a_4 all the known micro-companies on the one side are SME on the other side (and not vice versa);

and the counter-arguments include:

a_5 the two names Micro-company and SME are not alike by any string distance, and they are not synonyms in WordNet;
a_6 the only features they share are associate and employee and they have different domains and cardinalities.

In [Laera *et al.*, 2006], the arguments are expressed following the value-based argumentation framework [Bench-Capon, 2003]. They are made of a flag denoting if they are in favour ($+$) or against ($-$) the correspondence and the type of method that supports this correspondence (basic methods). A simple way to express these arguments is as follows:

$$\begin{aligned}
a_1 &: \langle \text{Company}, \text{Firm}, =, .89, \langle +, extensional \rangle \rangle \\
a_2 &: \langle \text{Company}, \text{Firm}, =, .89, \langle +, terminological \rangle \rangle \\
a_3 &: \langle \text{Micro-company}, \text{SME}, \sqsubseteq, .97, \langle +, semantic \rangle \rangle \\
a_4 &: \langle \text{Micro-company}, \text{SME}, \sqsubseteq, .97, \langle +, extensional \rangle \rangle \\
a_5 &: \langle \text{Micro-company}, \text{SME}, \sqsubseteq, .97, \langle -, terminological \rangle \rangle \\
a_6 &: \langle \text{Micro-company}, \text{SME}, \sqsubseteq, .97, \langle -, structural \rangle \rangle
\end{aligned}$$

Such kind of arguments could be delivered by existing basic matchers. Another, more elaborate way to define arguments is to allow correspondences themselves to be justifications. This permits, for instance, to express that the structural similarity of Micro-company and SME depends on the terminological similarity of employee and associate.

The rationale behind these kinds of arguments is that some agents may prefer, or trust, better some techniques than others. For instance, one can imagine that agent C prefers terminological arguments over extensional arguments, extensional arguments over semantic arguments and semantic arguments over structural arguments. This order induces a partial order on the arguments themselves: $a_5 \succ_C a_2$, $a_1 \succ_C a_2$, $a_5 \succ_C a_4$, $a_1 \succ_C a_4$, $a_2 \succ_C a_3$, $a_4 \succ_C a_3$, $a_3 \succ_C a_6$. Similarly, P could have a different preference ordering favouring structural, semantic, terminological and then extensional arguments.

There are logical theories [Dung, 1995, Amgoud *et al.*, 2000, Bench-Capon, 2003] that, given a set of arguments and the preferences of agents, define what is the consensus alignment between both parties. They usually define an admissible argument a with regard to a set of arguments S as an argument to which every counter-argument is attacked by an argument of S. A set S is said conflict-free if no argument of S attacks another argument of S. A maximal conflict free set of arguments acceptable with regard to S is called admissible. Finally, a preferred extension is an admissible set of arguments where there is no other such set that contains arguments preferred to some in the set that are admissible for their

preferred arguments in the set. For instance, C will have for preferred extension $\{a_5, a_1, a_2, a_6\}$ and P, in turn, will have $\{a_6, a_5, a_2, a_1\}$. However together, the maximal common subset of arguments between C and P is $\{a_1, a_2, a_5, a_6\}$ which selects the preferred alignment made up of γ_1 and γ_2.

A consensus alignment can also be achieved by a dialogue between the agents during which they exchange arguments. Such a dialogue is presented below. The agent C starts the dialogue by asserting alignment A between the two ontologies o and o' (the agent C is committed to support the alignment A and each correspondence it contains). A possible dialogue between C and P is as follows:

//The agent C is committed to support the alignment
$C-$assert(:content A :reply-with 1)$\rightarrow P$
//The agent P asks to justify the correspondence γ_1 (P does not have counter-argument)
$C \leftarrow$ question(:content γ_1 :reply-with 2) - P
// The agent C justifies the correspondence γ_1 with the arguments a_1 and a_2
$C-$support(:content $a_1, a_2 \vdash^+ \gamma_1$:in-reply-to 2)$\rightarrow P$
//The agent P asks to justify the correspondence γ_3 (P is ready to justify the opposite)
$C \leftarrow$ challenge(:content γ_3 :reply-with 3) - P
// The agent C justifies the correspondence γ_3 with the arguments a_3 and a_4
$C-$support(:content $a_3, a_4 \vdash^+ \gamma_3$:in-reply-to 3)$\rightarrow P$
// The agent P contests the correspondence γ_3 with the counter-arguments a_5 and a_6
$C \leftarrow$contest(:content $a_5, a_6 \vdash^- \gamma_3$:in-reply-to 3) - P
// The agent C retracts the correspondence γ_3
$C-$retract(:content γ_3 :in-reply-to 3)$\rightarrow P$

This results in the selection of the alignment $A' = \{\gamma_1, \gamma_2\}$.

These argumentation techniques have not been used in alignment explanation so far. However, they could be used in interactively explaining to users the arguments in favour or against correspondences. In the argumentation dialogue above, one of the agents can be a human user. The system knowing the preferences of users can provide them with a more adapted arguments.

9.7 Summary

Delivering alignments to users for inspection and revision is an important topic not deeply developed so far. Providing the justifications for correspondences can also be used for helping computer systems like agents to better understand alignments and control matching results.

We have presented the type of raw justifications matchers should supply and the manipulations that explanation systems can perform on these justifications in order to provide an intelligible picture of alignments to users. Some of these manipulations are based on proof presentation techniques, some others are based on a kind of dependency graphs. We have also presented techniques used by agents for exchanging justifications of correspondences and reaching a common agreement.

By using explanations, a matching system can provide users with meaningful prompts and suggestions on further steps towards the production of a desired result. Having understood the alignments returned by a matching system, users can deliberately edit them manually, thereby providing the feedback to the system. Beside explanations, matching systems should provide facilities for users to explore the alternative paths not followed by the system. These systems should enable the users to re-launch the matching process with different parameters in an intermediate state.