Automatic ontology mapping for agent communication

Floris Wiesman Nico Roos Paul Vogt

Universiteit Maastricht, Infonomics / IKAT, P.O.Box 616, 6200 MD Maastricht

Abstract

Agent communication languages such as ACL and KQML provide a standard for agent communication. These languages enable an agent to specify the intention and the content of a message as well as the protocol, the language, and the ontology that are used. For the protocol and the language some standards are available and should be known by the communicating agents.

The ontology used in a communication depends on the subject of the communication. Since the number of subjects is almost infinite and since the concepts used for a subject can be described by different ontologies, the development of generally accepted standards will take a long time. This lack of standardization, which hampers communication and collaboration between agents, is known as the *interoperability* problem. To overcome the interoperability problem, agents must be able to establish a mapping between their ontologies.

This paper investigates a new approach to the interoperability problem. The proposed approach uses no background knowledge and requires neither a correspondence between concepts used in the ontologies nor a correspondence between the structure of the ontologies. It only requires that some instances of the subject about which the agents try to communicate are known by both agents.

1 Introduction

The rapid growth of networks such as the Internet offers new possibilities for accessing information. At the same time increasingly more information is generated. To keep up with this growing supply of information, intelligent tools are required. Agent technology is one of the most promising ways of distributing and gathering information. The reason for this is that communication and collaboration are central issues of Multi Agent Systems (MAS).

Agent communication languages such as ACL and KQML provide a standard for agent communication in an open MAS. These languages enable an agent to specify the intention and the content of a message as well as the protocol, the language and the ontology that are used. For the protocol and the language, some standards are available and should be known by the communicating agents (e.g., FIPA protocols, KIF, and SP).

The ontology [2] used in a communication depends on the subject of the communication. Since the number of possible subjects is almost infinite and since the concepts used for a subject can be described by different ontologies, the development of generally accepted standards will take a long time. This lack of standardization, which hampers

communication and collaboration between agents, is known as the *interoperability* problem [7, 12, 13].

The interoperability problem also occurs in the area of heterogeneous databases [1, 3, 5, 6]. The Internet makes it possible to access (legacy) databases that have been developed in isolation, either because they belong to different legal entities or because they are located at different sites between which no communication was possible before the era of the Internet. Asking queries that require access to several of these databases, is impossible unless we know how to relate the information of the databases. One way to relate the information of different database is to use an ontology to describe the underlying semantic structure of a database.

The remainder of the paper is organized as follows. Section 2 discusses the interoperability problem in more detail and Section 3 points out some problems in current approaches. Section 4 outlines our approach and Section 5 reports on the experiments with our approach. Section 6 concludes the paper.

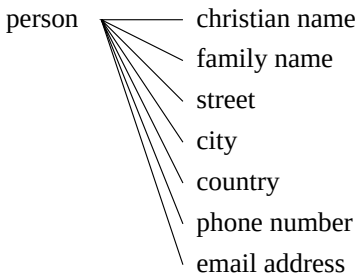
2 Interoperability

In order to reach interoperability, two problems must be dealt with, namely: *structural heterogeneity* and *semantic heterogeneity* [8, 12]. Structural heterogeneity concerns the different representations of information. Information described by the same ontology can be represented in different ways. This is a problem for heterogeneous databases but not for agents. In a multi agent system an ontology is the basis for communication. The actual way information is stored by an agent is shielded from the environment by the agent.

Semantic heterogeneity concerns the intended meaning of described information. Information about, for instance, persons can be described by different ontologies. We distinguish the following difference between ontologies: (1) different semantic structures, *structural conflict* [1], (2) different names for the same type of information or the same name for (slightly) different types of information, *naming conflicts* [1, 12], and (3) different representations of the same data, *data conflicts* [4]. The data conflict can be refined in conflicts because of *different units*, conflicts because of *different precision*, and conflicts because of *different expressions* (e.g., using 'van den Herik' or 'Herik, van den' to describe a person's family name).

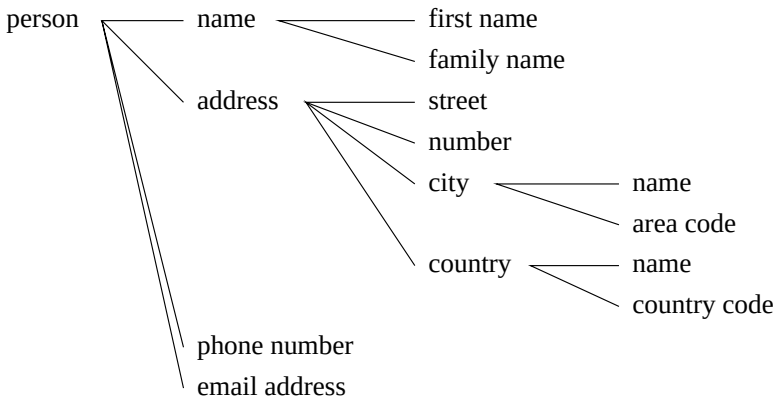
The following two ontologies illustrate some forms of semantic heterogeneity.

Ontology 1



In ontology 1, ‘street’ also describes the house number and ‘phone number’ describes the country code, the area code, and the local number.

Ontology 2



In ontology 2, ‘phone number’ only describes the local number. The ‘area code’ and the ‘country code’ are stored with, respectively, the city and the country.

Each ontology clearly has a different structure. Ontology 1 is flat while ontology 2 has a hierarchical structure. This structural conflict can be solved relatively easy because ontology 2 more or less extends ontology 1. When the two ontologies have completely different hierarchical structures, the structural conflict becomes more serious.

The naming conflicts between the two ontologies form a more severe problem. Different concept names are used for the same type of data (e.g., ‘first name’ and ‘christian name’). Moreover, the same concept name is used for slightly different types of data; e.g., ‘street’. In ontology 1 ‘street’ denotes both the *street name* and the *house number* while in ontology 2 it only denotes the *street name*. Hence, in order to reach interoperability, we must be able to split and merge data fields. For instance, the concept ‘street’ in ontology 1 containing the data ‘Castle Lane 1’ must be split into ‘Castle Lane’ and ‘1’ in order to map ’street’ in ontology 1 to ’street’ and ’number’ in ontology 2. The inverse mapping requires merging ‘Castle Lane’ and ‘1’.

We can conclude that to reach interoperability we have to find a mapping form the concepts of one ontology to the concepts of another ontology while instances of the concepts can be split and merged.

3 Problems in current approaches

To deal with semantic heterogeneity, several solutions have been proposed. Many of the proposed solutions try to derive a common ontology by some (semi) automatic process, see for instance [1, 3, 5, 7, 13]. These approaches heavily rely on assumptions such as:

- concepts are defined using a set of shared primitive concepts,
- different ontologies are the result of differentiations of one initial ontology,

- a human specifies relations between concepts of different ontologies and resolves possible conflicts.

Beside the problem that the above mentioned assumptions often cannot be met, ontology is an indirect way of establishing a mapping between two ontologies.

Some approaches addresses the problem of establishing a mapping directly [11, 6]. Papazoglou et al. [6] assume that the same naming conventions are used in different databases and that for each database an abstract description model describes the types of relations that hold between concepts that are specified. The possible relation types are common knowledge. From this information a mapping between the databases can be derived. The disadvantage of this approach is that it cannot handle naming conflicts.

Van Eijk et al. [11] give a characterization of establishing a mapping using first order logic. They assume a multi agent system in which agents communicate using first order formulas. A mapping consists of a set of translations formulas each expressing an equivalence between expressions. The disadvantage of this approach is that it cannot handle structural conflicts, and naming conflicts that are the result using the same names for different concept. Moreover, they do not provide an method for establishing a mapping.

In a recent survey of existing approaches [12], Wache et al. point out that establishing a mapping between two ontologies is still an important open problem. In this paper, we propose a new approach that solves part of the problem. Our approach handles structural, naming and some data conflicts that are the result of using different expressions, fully automatically. We only require that there are a number of instances of the concepts making up the ontology that are known by the two agents that wish to communicate. In terms of the two ontologies of Section 2, there must be several persons whose data are represented in both ontologies. Moreover, no background knowledge is required.

4 Learning ontology mappings

Suppose that agent 1 wishes to know the phone number and email address of some persons. Agent 1 knows that the information is (probably) available in a database managed by agent 2. Therefore, agent 1 contacts agent 2. In order for agent 1 to put forward its request, the agents first have to establish whether both use the same ontology or whether they use an ontology of which the other agent knows how to map it on its ontology. If the agents use different ontologies and if no mapping is known, the agents should try to establish a mapping.

The way the agents establish a mapping is inspired by language games [9, 10]. In a language game, an agent (robot) tries to interpreted the utterance of another agent by creating and evaluating associations between the received utterances and a categorization of an observed entity, the *joint attention*.

To illustrate the idea behind using language games for ontology mapping, suppose that two agents wish to communicate about a concept such as a ‘person’. Moreover, the agents use different conceptualizations of the concept ‘person’, and some persons are known by both agents.

A concept such as a ‘person’ may consists of a hierarchy of sub-concepts. For the *leaf* concepts in this hierarchy, an instance specifies the actual data values. For example, an instance could be a person called ‘Haddock’, who lives at ‘Castle Lane 1, Marlinspike’,

with phone number ‘421’. By finding an instance of the concept ‘person’ known by both agents, the agents determine *joint attention*. This joint attention will be the basis of the language game.

To establish the joint attention, one agent produces an utterance containing a unique representation of a concept and instance of the concept. The other agent, upon receiving the utterance, investigates whether it has a concept of which an instance matches to a certain degree the communicated instance. For this the agent measures the proportion of words that two instances have in common. The instance with the highest proportion of corresponding words, forms, together with the communicated instance, the joint attention – provided that the correspondence is high enough.

After establishing joint attention, one of the agents tries to establish a mapping between the leaf concepts that make up the concept. For this the agent establishing the mapping needs an utterance from the other agent and itself. Each of these utterances uniquely represents the leaf concepts of the concept followed by corresponding the data of an instance of the concept. The used representations of the leaf concepts can be any symbol, in principle, even Egyptian hieroglyphs. The only thing that is required is that each representation uniquely represents a leaf concept. Hence, the structure of the ontology plays no role. We may use, for instance, the term ‘pnfn’ or a term representing the place of a leaf concept in the ontology ‘person.has.name.has.first name’ to denote a person’s first name in a communication.[1]

If agent 1 uses ontology 1 of Section 2 and agent 2 uses ontology 2, agent 1 might formulate the following utterance:

person.has.christian name:Archibald
person.has.family name:Haddock
person.has.street:Castle Lane 1
person.has.city:Marlinspike
person.has.country:Belgium
person.has.phone number:06229–421
person.has.email address:haddock@herge.be

Agent 2 receiving this utterance might formulate its own utterance about the subject of joint attention:

person.has.name.has.first name:Archibald
person.has.name.has.family name:Haddock
person.has.address.has.street:Castle Lane
person.has.address.has.number:1
person.has.address.has.city.name:Marlinspike
person.has.address.has.city.area code:06229
person.has.address.has.country.name:Belgium
person.has.address.has.country.county code:32

[1]There is an advantage in using a representation that describes the place of a leaf concept in the ontology of an agent. Though it is not necessary for learning a mapping between leaf concepts that can be used for communication, it will enable agents to derive a more accurate mapping between their ontologies. If, for instance, both ontologies have a similar hierarchical structure for a person’s address, the agent might derive from the learned associations that the concept ‘address’ is used in both ontologies to denote a person’s address. The derivation of a more accurate mapping may, however, be limited by structural conflicts.

person.has.phone number:421
person.has.e-mail:haddock@herge.be

Next, agent 2 tries to establish associations between the different leaf concepts. Agent 2 generates associations between the leaf concepts of the two utterances on the basis of the proportion of corresponding words in pairs of leaf concepts, one from each utterance. Possible associations are:

field $x \leftarrow$ field y.
field $x \leftarrow$ field y, split(s), first.
field $x \leftarrow$ field y, split(s), last.
field $x \leftarrow$ field y, field z, merge (t).
field $x \leftarrow$ field y, split(s), last, field y., split(s), first, merge(t).
$\vdots$

Here, the operator field denotes the selection of a leaf concept where x, y and z represent the leaf concepts to be selected. As explained in Section 2, the operators split divides a data field into two sub-fields using the separator s to determine the point of division. We consider the following separators: ‘ ’, ‘,’, ‘;’, and TC (a type change, i.e., a change from letters to digits or vice versa). After splitting a data field the operators first and last can be used to select either the first or the last sub-field. The operator merge takes two data fields and merges them into one data field adding the separator t in between. The following separators can be added: ‘’, ‘ ’, ‘,’ and ‘;’. The following associations illustrates a mapping from agent 1 to agent 2.

field person.has.address.has.number $\leftarrow$ field person.has.street, split(TC), last.

Agent 2 searches through a space of possible associations guided by the proportion of words that instances of concepts have in common. Each new utterance from agent 1 enables agent 2 to update the strength of the associations. After having received a number of utterances, agent 2 may accept certain associations as being correct. Agent 2 has established a complete mapping form agent 1 to itself when it has a unique association for each leaf concept in its ontology. Note that the mapping is asymmetric and that it only enables communication in one direction. For full communication, agent 2 also must establish a mapping in the other direction.

When searching through the space of possible associations, agent 2 may consider for each leaf concept in its ontology, all combinations of the leaf concepts of agent 1. The number of these associations is $n2^m$, where m and n are the number of leaf concepts of agent 1 and agent 2 respectively. The number of possible associations is even higher since we may also split instances of concepts and, even worse, splits can be done in various ways. To reduce this complexity agent 2 only generates associations between leaf concepts if the proportion of words the corresponding data fields have in common is high. Agent 2 uses a matrix of leaf concepts of both ontologies for this purpose. Using this information only a few associations are possible.

5 Experiments

We have conducted experiments for *a proof of concept*. The experiments did not involve communicating agents; we focused on the process of finding the mapping. Further research will be embedded in an agent context.

In our experiments, we tried to establish a mapping between two address databases. For this we used the address database of our department and two small artificial address databases. The structure of the ontology of the address databases used was not very complex. It is even simpler than the example ontologies of Section 2. This does not weaken our results since, as was pointed out in the previous section, we only need unique references to leaf concepts. Structural conflicts therefore do not play a role. The difficulty in establishing a mapping between the ontologies, lays in finding an association between the references of concepts and in combining and splitting instances of these leaf concepts.

The departmental database (DDB) that we used in the experiment had 13 fields and contained 502 records. The first artificial database (ADB1) had 4 fields (identical to the first 4 of ontology 1 in Section 2) and 23 records, and the second artificial database (ADB2) had 5 fields and 5 records.

In ADB1 and ADB2 there were three matching instances, which were all found by the system. For 4 out of 5 fields, the mapping from ADB1 to ADB2 was correct for all records. The algorithm cannot match the abbreviated christian names in ADB1 to the full christian names in ADB2, therefore the christian name field mapping is left empty. The results of the reverse mapping, ADB2 to ADB1, were similar. However, the system failed to match properly names of the form 'van den Herik' to 'Herik, van den'. Merging poses no problem (using split(','), merge(' ')), but *splitting* compound names requires knowledge on Dutch names.

In ADB1 and DBB there were two matching instances, which were found both. The mapping from ADB1 to DBB was correct for all fields for all records. The reverse mapping showed some problems with foreign addresses, which are structured differently from Dutch addresses (e.g., 1 Castle Lane vs. Castle Lane 1). Again, special knowledge would be required to solve this shortcoming. The ADB2–DBB results were equal to the ADB1–DBB results.

6 Conclusions

We introduced an approach to learning mappings between the ontologies of two agents. The agents engage in a dialogue where utterances are exchanged that are formulated using the agents' own ontologies. The approach requires that both ontologies have at least some instances in common.

The approach was tentatively tested with a system that made mappings between two small and one large address databases. The results showed that our concept works. The shortcomings identified were due to missing knowledge about the structure of names and addresses. Since our approach aims to be a generic solution, we do not plan to add this knowledge.

Future work will focus on embedding the system in a real MAS, since that is the intended application of our approach. Moreover, we will generalize the system to deal

with ontologies in which instances are organized differently from databases.

References

[1] S. Bergamaschi, S. Castano, S. Vermercati, S. Montanari, and M. Vincini. An intelligent approach to information integration, 1998.

[2] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43:907–928, 1995.

[3] J. Hammer and D. McLeod. An approach to resolving semantic heterogeneity in a federation of autonomous, heterogeneous database systems. *Journal for Intelligent and Cooperative Information Systems*, 2(1):51–83, 1993.

[4] W. Kim and J. Seo. Classifying schematic and data heterogeneity in multidatabase systems. *IEEE Computer*, 24(12):12–18, 1991.

[5] 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.

[6] M. P. Papazoglou, N. Russell, and D. Edmond. A translation protocol achieving consensus of semantics between cooperating heterogeneous database systems. In *Conference on Cooperative Information Systems*, pages 78–89, 1996.

[7] H. S. Pinto. Some issues on ontology integration. In *IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5)*, 1999.

[8] A. P. Sheth and J. A. Larson. Federated database systems for managing dristributed, heterogeneous, and autonomous databases. *ACM Computing Serveys*, 22:183–236, 1990.

[9] L. Steels. Emergent adaptive lexicons. In P. Maes, editor, *From Animals to Animats 4: Proceedings of the Fourth International Conference On Simulating Adaptive Behavior*, Cambridge Ma., 1996. The MIT Press.

[10] L. Steels and P. Vogt. Grounding adaptive language games in robotic agents. In C. Husbands and I. Harvey, editors, *Proceedings of the Fourth European Conference on Artificial Life*, Cambridge Ma. and London, 1997. MIT Press.

[11] R. M. van Eijk, F. S. de Boer, and W. van der Hoek. On dynamically generated translators in agent communication. *International Journal of Intelligent Systems*, 16:587–607, 2001.

[12] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hbner. Ontology-based integration of information - a survey of existing approaches. In *IJCAI 2001 Workshop on Ontologies and Information Sharing*, 2001.

[13] P. C. Weinstein and W. P. Birmingham. Agent communication with differentiated ontologies: eight new measures of description campatibility. Technical Report CSE-TR-383-99, Artificial Intelligence Laboratory, University of Michigan, 7 1999.