Probabilistic Dialogue Models for Dynamic Ontology Mapping

Paolo Besana and Dave Robertson

Centre for Intelligent Systems and Applications
University of Edinburgh

Abstract Agents need to communicate in order to accomplish tasks that they are unable to perform alone. Communication requires agents to share a common ontology, a strong assumption in open environments where agents from different backgrounds meet briefly, making it impossible to map all the ontologies in advance. An agent, when it receives a message, needs to compare the foreign terms in the message with all the terms in its own local ontology, searching for the most similar one. However, the content of a message may be described using an interaction model: the entities to which the terms refer are correlated with other entities in the interaction, and they may also have prior probabilities determined by earlier, similar interactions. Within the context of an interaction it is possible to predict the set of possible entities a received message may contain, and it is possible to sacrifice recall for efficiency by comparing the foreign terms only with the most probable local ones. This allows a novel form of dynamic ontology matching.

1 Introduction

Agents collaborate and communicate to perform tasks that they cannot accomplish alone. To communicate means to exchange messages, that convey meanings encoded into signs for transmission. To understand a message, a receiver should be able to map the signs in the messages to meanings aligned with those intended by the transmitter.

Therefore agents should agree on the terminology used to describe the domain of the interaction: for example, if an agent wants to buy a particular product from a seller, it must be able to specify the properties of the products unambiguously. Ontologies specify the terminology used to describe a domain [3].

However, shared ontology can be a strong assumption in an open environment, such as a Peer-to-Peer system: agents may come from different backgrounds, and have different ontologies, designed for their specific needs.

In this sort of environment, communication implies translation. The standard approach is to find mappings between the ontologies, creating a sort of bilingual dictionary. Many different techniques have been developed for ontology mapping, but in an open environment it is impossible to know which agents will take part in the interactions; therefore it is impossible to anticipate which ontologies should be mapped.

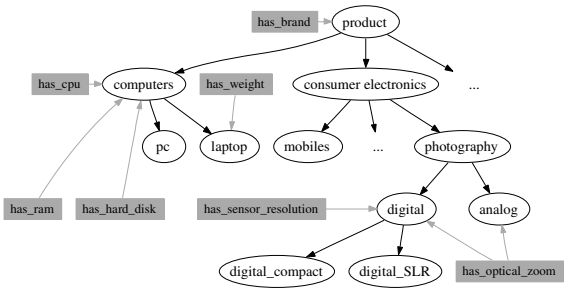

Figure1. Fragment of buyer a_b ontology

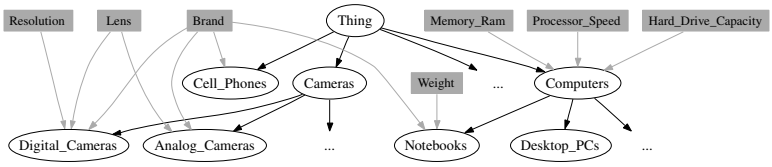

Figure2. Fragment of seller a_s ontology

Agents have to map ontologies dynamically when needed. Mapping full ontologies is a time-consuming task: in the standard process, each term in one ontology is compared with all the terms in the other ontology, and the most similar term is the mapping.

However, agents may meet infrequently, for a single interaction on a specific topic. A full ontology mapping would be a waste of resources: mapping only "foreign" terms that have appeared in the conversation can be more convenient.

Comparing a foreign term in a message with all the terms in the ontology can still be costly. Yet, the entities referred by the signs in the message are not randomly chosen: the dialogue has a meaning because entities are related. For example, if the conversation is about the purchase of a laptop, entities related to cars are unlikely to appear. It is reasonable to compare the signs in the message with entities about laptops, rather than compare with all the entities indiscriminately.

This paper shows how to extract, represent, and use knowledge about the relations and properties of the entities in an interaction to support dynamic ontology mapping.

2 Example scenario

The example scenario is a purchase interaction between the buyer and seller agents a_b and a_s. In the dialogue, the a_b asks a_s about a laptop he needs. The seller a_s inquires about properties of the product in order to make an offer.

The two agents do not share the same ontology: the buyer uses the one in figure 1 and the seller the one in figure 2. In the figures the ovals are classes and

the grey boxes are properties. The classes are structured in taxonomies, and the domains of the properties are shown by grey arrows.

3 Communication

An approach to communication, for which Electronic Institution [8] is an example, focuses on the interaction itself, using norms, laws and conventions to define the expected behaviours of the agents, without specifying their mental state.

As described in [9], norms and conventions form the skeleton for many human coordinated activities, and they work similarly in agents' societies: they provide a template for actions, and simplify the decision-making process, dictating the course of action to be followed in certain situations.

3.1 Lightweight Coordination Calculus

In this paper, interaction are modelled using the *Lightweight Coordination Calculus* [6], that borrows notions from Electronic Institutions.

The *Lightweight Coordination Calculus* (LCC) is an executable specification language adapted to peer-to-peer workflow and has been used in applications such as business process enactment [4] and e-science service integration [1].

LCC is based on process calculus: protocols are declarative scripts written in Prolog and circulated with messages. Agents execute the protocols they receive by applying *rewrite rules* to expand the state and find the next move.

It uses roles for agents and constraints on message sending to enforce the social norms. The basic behaviours are to send ($\Rightarrow$) or to receive ($\Leftarrow$) a message. More complex behaviour are expressed using connectives: `then` creates sequences, `or` creates choices. Common knowledge can be stored in the protocol.

Figure 3 shows and explains the LCC protocol used by the buyer for the interaction in the example scenario. Figure 4 represents the sequence diagram of the exchanged messages and of the constraints satisfied during a run of the protocol for the purchase of a laptop.

3.2 Communication and contexts

The agents execute the protocols inside a separate "box": in theory, it is possible to write a protocol that can be run without requiring any specific knowledge from the agent. It requires that the constraints are satisfied with the information available in the common knowledge.

The "box" in which a protocol is run can be compared to the idea of *context* described by Giunchiglia: in [2] he defines a context c_i as "*partial*" and "*approximate*" theory of the world, represented by the triplet $\langle L_i, A_i, \Delta_i \rangle$. In the tuple, L_i is the language local to the context, A_i is the set of axioms of the context, and Δ_i is the inference engine local to the context. Moreover, a reasoner can connect a deduction in one context with a deduction in another using *bridge rules*.

For the protocol run context $c_r = \langle L_r, A_r, \Delta_r \rangle$, the language L_r is composed by all the terms that can be introduced by the agents involved in the interaction

```
a((buyer(S), B) ::=
ask(Prd) ⇒ a(vendor, S) ← want(Prd)
then
a(neg_buy(Prd, S), B).

a(neg_buy(Prd, S), B) ::=
( ask(Attr) ⇐ a(neg_vend(_), S)
  then
  ( inform(Attr, Val) ⇒ a(neg_vend(_), S) ← required(Prd, Attr, Val) )
  or
  ( dontcare(Attr) ⇒ a(neg_vend(_), S) )
  then
  a(neg_buy(Prd, S), B) )
or
( propose(Prd, Price, Const) ⇐ a(neg_vend(_), S)
  then
  ( accept ⇒ a(neg_vend(_), S) ← afford(Prd, Price) )
  then
  ( ack ⇐ a(neg_vend(_), S) )
  or
  reject ⇒ a(neg_vend(_), S) )
or

sorry ⇐ a(neg_vend(_), S)
```

In the interaction, the agent a_b initially takes the role of buyer: it first sends a request to agent a_s for the products it wants to buy (found satisfying `want(Prd)`) and then becomes a negotiating buyer, waiting for a reply.
The agent a_s receives the request: if it has the product, selects the attributes the buyer need to specify and becomes a negotiating seller; otherwise it says `sorry`. As a negotiating seller, a_s recursively extracts the attributes from the list, and asks about them to a_b, creating a filter with the received information. The buyer agent receives the request, and if it cares and knows about the value of the attribute (if it can satisfy `required(Prd, Attr, Val)`), replies with it, otherwise sends a `dontcare` message.
When the list of attributes is empty, a_s sends an offer using the created filter. The agent a_b accepts the offer if it can afford the price (`afford(Prd, Price)` must be satisfied) or reject it.

Figure3. LCC Dialogue fragment used by the buyer agent

(terms are tagged with their origin: a_b introduces *'Laptop'* satisfying `want(Prd)`, and therefore `Prd` is replaced throughout the protocol with the tagged value `'Laptop'@`a_b); the axioms A_r are the role clauses together with the axioms in the common knowledge and Δ_r is the protocol expansion engine.

Even though protocols can be autonomous from the agent, they become useful only if they can exploit the agents' knowledge, that is if it is possible to bridge the resoning between the interaction context c_r and in the agent's local context c_a. This is accomplished using a bridge rule that connects the constraints in the protocol with the predicates in the agent's local knowledge:

$$\frac{c_r : \kappa_p(W_1, ..., W_n)}{c_a : \kappa_a(Y_1, ..., Y_m)} \tag{1}$$

where κ_p is a formula of a protocol constraint and κ_a is a formula in the agent's local knowledge, that can be satisfied only by using its own language L_a.

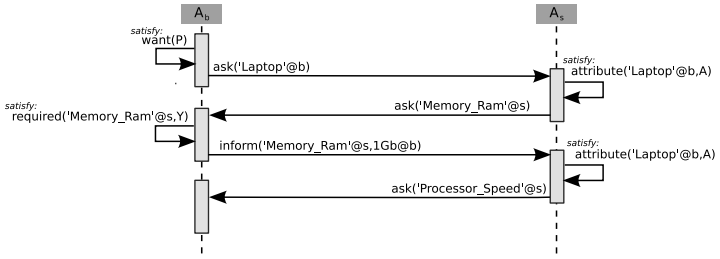

Figure4. A sequence diagram representing a protocol run between a_b and a_s

4 Ontology Mapping

In traditional ontology mapping, the bridges should be valid for any value from L_r and L_a in two contexts c_r and c_a:

$$\forall W_1...W_n \in L_r,\ \exists Y_1...Y_n \in L_a.\ \ c_i : \kappa_p(W_1, ..., W_n) \rightarrow c_j : \kappa_q(Y_1, ..., Y_m) \quad (2)$$

That is, for any value of $W_1, ..., W_n$ in κ_p, it is possible to find the values for $Y_1, ..., Y_n$ so that κ_a is equivalent to κ_p. In the example scenario, the mappings should cover the possible requests from the buyer agent a_b for buying any element in its ontology (see figure 1), such as mobile phones, analog cameras and so on - even if these interactions never take place.

This is a strong requirement: it implies that it is possible to find a corresponding term in L_a for every term in L_r, and this may not always be the case. Static *ontology mapping* tries to achieve this. An ontology mapping function receives two ontologies and returns the relations between their entities:

$map : O \times O \rightarrow \Omega$

where Ω contains all the binary relations r (*equivalence, similarity, generalisation, specialisation, etc*) between entities in O_1 and O_2.

Inconsistencies in the ontologies undermines the possibility of satisfying the definition in expression 2. Mapping systems use various methods to verify the relations between terms: detailed reviews of these approaches can be found in [7,5].

5 Dynamic Ontology Mapping: Motivation

As said in the introduction, it is possible to limit the mappings to those needed to perform the occurring interactions, and there is no need to guarantee complete equivalence between the languages. Therefore an agent needs to map only the terms that appear in κ_p in order to satisfy κ_a :

$$\exists W_1...W \in L_r, Y_1...Y_n \in L_a.\ \ c_r : \kappa_p(W_1, ..., W_n) \wedge c_a : \kappa_a(Y_1, ..., Y_m) \quad (3)$$

that is a much weaker requirement: we need to find the values for $Y_1, ..., Y_n$ so that κ_a is valid for the given instances of $W_1, ..., W_n$. In the example, it means that only the mappings required for buying the laptop are needed.

Not every grounding of the variables is meaningful: some will make κ_a more similar to κ_p than others. The mapping function:

$$singlemap : t_{L_r} \times L_a \rightarrow t_{L_a}$$

is the "oracle" used to search the best possible mapping to make the bridge in expression 1 meaningful. It does this by comparing t_{L_r} with all the terms in L_a.

The values for the variables $W_1, ..., W_n$ in κ_p are introduced by received messages (for example, the first `ask(Attr)` in figure 4 introduces `Memory_Ram@s'`), by satisfying constraints (for example, `want(P)` introduces `'Laptop@b'`) or when a role is invoked with parameters. Only terms introduced by received messages can be defined in other ontologies and require mapping.

Suppose an agent receives a message $m_k(\ldots, w_i, \ldots)$, where $w_i \notin L_a$ is the foreign term. The task of the oracle is to find what entity or concept, represented in the agent's ontology by the term t_m, was encoded in w_i by the transmitter. Not all the comparisons between w_i and terms $t_j \in L_a$ are useful: the aim of this work is to specify a method for choosing the smallest set $\Gamma \subseteq L_a$ of terms to compare with w_i, given a probability of finding the matching term $t_m \in L_a$. We assume that t_m exists and that there is a single best match.

Let $p(t_j)$ be the probability that the entity represented by $t_j \in L_a$ was used in W_i inside m_k. The oracle will find t_m if $t_m \in \Gamma$, event that has a probability:

$$p(t_m \in \Gamma) = \sum_{t_j \in \Gamma} p(t_j)$$

If all terms are equiprobable, then $p(t_m \in \Gamma)$ will be proportional to $|\Gamma|$. For example, if $|L_a| = 1000$, then $p(t_j) = 0.001$. Setting $|\Gamma| = 800$ yields $p(t_m \in \Gamma) = 0.8$, and there is no strategy for choosing the elements to add to Γ.

Instead, if the probability is distributed unevenly, as described in section 6, and we keep the most likely terms discarding the others, we can obtain a higher probability for smaller Γ. For example, suppose that $p(t_j)$ is distributed approximately according to Zipf's law (an empirical law mainly used in language processing that states that the frequency of a word in corpora is inversely proportional to its rank):

$$p(k; s; N) = \frac{1/k^s}{\sum_{n=1}^{N} 1/n^s}$$

where k is the rank of the term, s is a parameter (which we set to 1 to simplify the example), and N is the number of terms in the vocabulary. The probability of finding t_m becomes:

$$p(t_m \in \Gamma) = \frac{\sum_{k=1}^{|\Gamma|} 1/k}{\sum_{n=1}^{|L_a|} 1/n}$$

For $|L_a| = 1000$, then $p(t_m \in \Gamma) = 0.70$ for $|\Gamma| = 110$ and maybe more remarkably $p(t_m \in \Gamma) = 0.5$ for $|\Gamma| = 25$.

Therefore, given a probability distribution for the terms, it is possible to trade off a decrement in the probability of finding the matching term t_m in Γ with an important reduction of comparisons made by the oracle.

The core issue dealt with by this paper is how to create and assign probabilities to the entities that can be used in a message $m_k(\ldots, w_i, \ldots)$. Intuitively,

the type of interaction, the specific topic and the messages already exchanged bind w_i to a set of possible expected entities.

In particular, this paper shows how an interaction model as LCC forms a framework that enforces relations between the entities: the roles provide a first filter for them. For example, messages in a buyer role will likely refer to entities like products, prices, and attributes of the products. Different runs of the same protocol tend to follow the same path, adding predictability to the interaction.

6 Modelling the interactions

6.1 Asserting the possible values

The solution proposed is a model that stores and updates properties of the entities used to instantiate each variable W_i in different runs of the same protocol.

As seen in section 3.1, the variables are replaced by values during protocol execution, and therefore it is not possible to refer to them directly. A variable W_i is a *slot* A (an argument position), in a LCC *node* N (that can be a message, a constraint or a role header) inside a role R, and it is represented as $\langle \mathtt{N}, \mathtt{A} \rangle_{\mathtt{R}}$. For example, the variable `Prd` appears in $\langle \mathtt{want}, \mathtt{1} \rangle_{\mathtt{b}}$, where `b` means `buyer`.

In general, the possible values for the slot $\langle \mathtt{N_i}, \mathtt{A} \rangle_{\mathtt{R}}$ are modelled by M assertions, each assigning a probability to the hypothesis that the matching entity for the slot belongs to a set Ψ:

$$A_j^{\langle \mathtt{Ni},\mathtt{A} \rangle_{\mathtt{R}}} \doteq Pr\left(\langle \mathtt{N_i}, \mathtt{A} \rangle_{\mathtt{R}} \in \Psi | c\right) \tag{4}$$

The probability can be made dependent on the value of another slot. Therefore the assertion is in the form of a posterior probability: the element c can become a constraint on the value of another slot. The probability can also be independent from any other slot: in this case the element c becomes the `true` constant and can be omitted.

How assertions are obtained Assertions are created and updated every time a protocol is executed. Let's suppose that the agents a_s and a_b have already used the protocol in different interactions with other agents. The agent a_b has used it 12 times with different vendors: 6 times searching a laptop, and 6 times seeking a digital camera. In total, it has received 40 times the message `ask(Attr)` that inquired about properties of the requested product. The content of the slot in the received messages has been mapped to the entities from its own ontology (see figure 1) with the frequencies in table 1. The seller agent a_s has used the protocol 100 times with different buyers, receiving every time the message `ask(Prd)`. The content of the slot has been mapped to entities in its own ontology (see figure 2) with the frequencies in table 1. The frequencies of the mappings are used to compute dynamically the probabilities in the assertions.

Assertions about entities Assertions can simply be about the prior probability of entities in a slot, disregarding the values of other slots in the protocol run:

$\langle \mathtt{ask},1\rangle_{\mathtt{nb}}$	$\langle \mathtt{want},1\rangle_{\mathtt{b}} = Laptop$	$\langle \mathtt{want},1\rangle_{\mathtt{b}} = digital\ \ SLR$	**Total**
has_brand	4	5	9
has_cpu	6	0	6
has_ram	6	0	6
has_hard_disk	4	0	4
has_weight	3	1	4
has_optical_zoom	0	5	5
has_sensor_resolution	0	6	6
Total	**23**	**17**	**40**

$\langle \mathtt{ask},1\rangle_{\mathtt{s}}$	Total
Digital_Cameras	40
Cell_Phones	30
Laptops	20
PC_Desktops	10
Total	**100**

Table1. Mappings for $\langle \mathtt{ask},1\rangle_{\mathtt{nb}}$ and $\langle \mathtt{ask},1\rangle_{\mathtt{s}}$

$A_j^{\langle \mathtt{N_i},\mathtt{a}\rangle_{\mathtt{R}}} \doteq Pr\left(\langle \mathtt{N_i},\mathtt{a}\rangle_{\mathtt{R}} \in \{e_q\}\right) = p_j$

In the scenario, assertions about $\langle \mathtt{ask},1\rangle_{\mathtt{nb}}$ are:

$A_1^{\langle \mathtt{ask},1\rangle_{\mathtt{nb}}} \doteq Pr\left(\langle \mathtt{ask},1\rangle_{\mathtt{nb}} \in \{"has\_brand"\}\right) = \frac{9}{40} = 0.225$

...

$A_7^{\langle \mathtt{ask},1\rangle_{\mathtt{nb}}} \doteq Pr\left(\langle \mathtt{ask},1\rangle_{\mathtt{nb}} \in \{"has\_sensor\_resolution"\}\right) = \frac{6}{40} = 0.15$

More precise assertions can be about the posterior probability of the entity given the values of previous slots:

$A_j^{\langle \mathtt{N_i},\mathtt{a}\rangle_{\mathtt{R}}} \doteq Pr\left(\langle \mathtt{N_i},\mathtt{a}\rangle_{\mathtt{R}} \in \{e_q\} \,|\langle \mathtt{N_{i-d}},\mathtt{a}\rangle_{\mathtt{R}} = e_k\right) = p_j$

In the example scenario, we have:

$A_{10}^{\langle ask,1\rangle_{\mathtt{nb}}} \doteq Pr\left(\langle \mathtt{ask},1\rangle_{\mathtt{nb}} \in \{"has\_brand"\} \mid \langle \mathtt{want},1\rangle_{\mathtt{b}} = "Laptop"\right) = \frac{4}{23} = 0.174$

$A_{11}^{\langle ask,1\rangle_{\mathtt{nb}}} \doteq Pr\left(\langle \mathtt{ask},1\rangle_{\mathtt{nb}} \in \{"has\_cpu"\} \mid \langle \mathtt{want},1\rangle_{\mathtt{b}} = "Laptop"\right) = \frac{6}{23} = 0.260$

...

Assertions about properties and relations Assertions can also be about ontological relations between the entities in the slot and other entities. The possible relations depend on the expressivity of the ontology: if it is a simple list of allowed terms, it will not possible to verify any relation; if it is a taxonomy, subsumption can be found; for a richer ontology, more complex relations such as domain or range can be found. The assertions about the probabilities of ontological relations are obtained generating hypotheses about different relations and counting the frequencies of the proved ones.

The hypotheses can be about an ontological relation between the entity in the slot and an entity e_k in the agent's ontology:

$A_j^{\langle \mathtt{N_i},\mathtt{a}\rangle_{\mathtt{R}}} \doteq Pr\left(\langle \mathtt{N_i},\mathtt{a}\rangle_{\mathtt{R}} \in \{X|rel\left(X,e_k\right)\}\right) = p_j$

In the example scenario, the seller can prove some relations between the entities in $\langle \mathtt{ask},1\rangle_{\mathtt{s}}$ and other entities in its ontology (see figure 2):

$A_1^{\langle \mathtt{ask},1\rangle_{\mathtt{s}}} \doteq Pr\left(\langle \mathtt{ask},1\rangle_{\mathtt{s}} \in \{X|subClass(X,"Computers")\}\right) = \frac{30}{100} = 0.3$

The assertions can also regard the relation with another slot in the protocol:

$A_j^{\langle \mathtt{N_i},\mathtt{a}\rangle_{\mathtt{R}}} \doteq Pr\left(\langle \mathtt{N_i},\mathtt{a}\rangle_{\mathtt{R}} \in \{X|rel\left(X,\langle \mathtt{N_{i-d}},\mathtt{a_k}\rangle_{\mathtt{R}}\right)\}\right) = p_j$

In the example scenario the buyer can prove the relation between $\langle ask, 1\rangle_{nb}$ and $\langle \texttt{want}, \texttt{1}\rangle_{\texttt{b}}$ in its ontology (see figure 1):

$A_{20}^{\langle \texttt{ask},1\rangle_{\texttt{nb}}} \doteq Pr\left(\langle \texttt{ask},\texttt{1}\rangle_{\texttt{nb}} \in \left\{X \,|\, hasDomain\left(X, \langle \texttt{want},\texttt{1}\rangle_{\texttt{b}}\right)\right\}\right) = 1.0$

which means that the domain of the entity in the $\langle \texttt{ask},\texttt{1}\rangle_{\texttt{nb}}$ in the negotiator clause is always the content of the first slot in the node `want` in the `buyer` role.

Assertion reliability Assertions that assign probabilities to entities work correctly in well known and stationary situations. But interactions can have different content, such as the purchase of a different product, and the probabilities of entities can change over time (for example, a type of product may go out of fashion). Assertions about ontological relations can work on new content, but sometimes they can over fit the actual relations in interactions.

6.2 Using assertions

When a known protocol about a role R is used and the message $m_k\,(\ldots, w_i, \ldots)$ arrives, the system computes the probability distribution for the terms in $\langle \texttt{m}_\texttt{k}, \texttt{i}\rangle_{\texttt{R}}$: all the assertions relative to the slot are selected and instantiated if needed.

In the example in figure 4, a_b receives the message `ask('Memory_Ram'@s)`, and $\langle \texttt{want},\texttt{1}\rangle_{\texttt{b}}$ contains *'Laptop'@b* . Thus, the assertions about $\langle \texttt{ask},\texttt{1}\rangle_{\texttt{nb}}$ are:

$A_1^{\langle \texttt{ask},1\rangle_{\texttt{nb}}} \doteq Pr\left(\langle \texttt{ask},\texttt{1}\rangle_{\texttt{nb}} \in \{"has\_brand"\}\right) = 0.225$

$\ldots$

$A_6^{\langle \texttt{ask},1\rangle_{\texttt{nb}}} \doteq Pr\left(\langle \texttt{ask},\texttt{1}\rangle_{\texttt{nb}} \in \{"has\_optical\_zoom"\}\right) = 0.125$

$\ldots$

$A_{10}^{\langle ask,1\rangle_{\texttt{nb}}} \doteq Pr\left(\langle \texttt{ask},\texttt{1}\rangle_{\texttt{nb}} \in \{"has\_brand"\}\,|\,true\right) = 0.174$

$\ldots$

$A_{20}^{\langle \texttt{ask},1\rangle_{\texttt{nb}}} \doteq Pr\left(\langle \texttt{ask},\texttt{1}\rangle_{\texttt{nb}} \in \{"has\_brand", "has\_cpu", "has\_ram"\quad "has\_hard\_disk", "has\_weight"\}\right) = 1.0$

The assertions can be generated using different strategies, and assign probabilities to overlapping sets that can be either singletons or larger. The motivation of the work is to select the most likely entities for a slot in order to reach a given probability of finding the mapping, and therefore we need to assign to the terms the probabilities computed with the assertions.

This requires two steps. First, probabilities given to sets are uniformely distributed among the members: according to the *principle of indifference*, the probability of mutually exclusive elements in a set should be evenly distributed. Then, the probability of an entity t_i is computed by summing all its probabilities, and dividing it by the sum of all the probabilities about the slot:

$$p\left(t_i\right) = \frac{\sum A_j^{\langle N,A\rangle_R}\left(\langle N,A\rangle_R \in \{t_i\}\right)}{\sum A_k^{\langle N,A\rangle_R}} \tag{5}$$

In the example above, the entities will have the probabilities:

$P(has\_brand) = \frac{A_1 + A_{10} + A_{20}/5}{A_1 + \ldots + A_{20}} = \frac{0.225 + 0.174 + 0.2}{3} = 0.2$

$\ldots$

$P(has\_sensor\_resolution) = \frac{A_7}{A_1 + \ldots + A_{20}} = \frac{0.15}{3} = 0.05$

The probabilities of terms related to the interactions have higher probabilities than those of unrelated terms. Using the first four terms for the set Γ of terms to compare with the term `'Memory_Ram'@s` in the received message yields a probability of finding the mapping in Γ greater than 0.8.

7 Conclusion

In this paper we showed an approach for dynamic ontology mapping that exploits knowledge about interactions to reduce the waste of resources normally employed to verify unlikely similarities between unrelated terms in the different ontologies.

The traditional approaches aim at finding all the possible mappings between the ontologies, so that any possible interaction that can occur. As shown in section 5, our goal is pragmatic: only the mappings required for the interactions that take place need to be found. For an agent, this means that only the terms in received messages and defined in external ontologies will be mapped.

In the standard approach, an ontology mapper oracle compares these "foreign" terms with all the terms in the agent's ontology, although most of the compared terms are not related. However, the terms that appear in messages are not all equally probable: given the context of the interaction, some will be more likely than others. The use of protocols allows us to collect consistent information about the mappings used during an interaction: in section 6 we show first how to create and update a probabilistic model of the content of the messages and then how to use the model to select what are the most likely entities contained in a message, so that the mapper oracle can focus on them, improving the efficiency.

References

1. A Barker and B Mann. Agent-based scientific workflow composition. In *Astronomical Data Analysis Software and Systems XV*, volume 351, pages 485–488, 2006.
2. F Giunchiglia. Contextual reasoning. Technical report, IRST, Istituto per la Ricerca Scientifica e Tecnologica, 1992.
3. T R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
4. Li Guo, D Robertson, and Y Chen-Burger. A novel approach for enacting the distributed business workflows using bpel4ws on the multi-agent platform. In *IEEE Conference on E-Business Engineering*, pages 657–664, 2005.
5. Y Kalfoglou and M Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.
6. D Robertson. A lightweight coordination calculus for agent systems. In *Declarative Agent Languages and Technologies*, pages 183–197, 2004.
7. P Shvaiko and J Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics*, 4:146–171, 2005.
8. C Sierra, J Rodriguez Aguilar, J Noriega, P; Arcos, and M Esteva. Engineering multi-agent systems as electronic institutions. *European Journal for the Informatics Professional*, 4, 2004.
9. M Wooldridge. *An Introduction to Multiagent Systems.* John Wiley and Sons, 2002.