

Learning Non-Unanimous Ontology Concepts to Communicate with Groups of Agents

Mohsen Afsharchi, Behrouz H.Far
Department of Electrical and Computer Engineering
University of Calgary, Calgary, Canada
{mafsharc, far}@ucalgary.ca

Jörg Denzinger
Department of Computer Science
University of Calgary, Calgary, Canada
denzinge@cpsc.ucalgary.ca

Abstract

We present an extension to the definition of a concept in an ontology that allows an agent to simultaneously communicate with a group of agents that might have different understandings of some concepts. We also provide a way to learn such non-unanimous concepts by using a method for learning concepts from a group of teachers. The general idea of non-unanimous concepts is to use the teachers to identify the core of a concept everyone agrees on and what else at least some of the teachers think belongs into the concept. The learning agent also decides what belongs to the concept for itself and whenever it needs to communicate with a group of other agents and needs to be precise it makes use of these three concept aspects by providing additional example objects for what might be misunderstood.

1. Introduction

The ability to communicate with each other is often a necessity for agents in order to achieve both, group and individual, goals. For many purposes the use of a language is a very economic way to convey information between agents. But in order to allow the use of a language, its semantics must be understood by all agents involved, which in itself requires that agents have a common understanding of the concepts that they are communicating about.

The concepts (and their relations) that an agent knows or understands are commonly known as the ontology of this agent. In order to facilitate communication, a common ontology for all agents involved in a task has often been suggested, but this is only a theoretical solution in many cases. Especially in multi-agent systems with agents developed by and representing different groups, there is often no agreement between these groups on a common ontology and even if agreements can be reached, the implemented on-

ologies might still differ, due to different understandings of the agreed-upon concepts by the different groups. To deal with this problem, in the last few years authors suggested to enable agents to learn new concepts for their ontologies and to learn concepts from other agents to create a better basis for communication (see [8], [5], or [2]). While almost all authors have looked at one agent teaching one other agent, in [1] we presented a general framework that allows an agent to learn a concept from a group of teacher agents.

At first glance, learning from a group of agents instead of a single agent only seems to add potential problems, namely that the teachers might not agree on some aspects of a concept to learn so that it is up to the learning agent to decide on these aspects on its own. And this has the consequence that the concept that the learner has learned is some kind of compromise between the concepts the teachers teach. Would it not be better to learn from just one teacher at a time and to make sure that the learning agent learns exactly the concept of this teacher? But what if this learner has to communicate with several other agents? Naturally, the learner could learn the necessary concepts from each of these agents one by one and then it has an ontology that has concepts like "concept X according to agent Y" and "concept X according to agent Z". As pointed out in [2], it is a rather large effort to create such an ontology. But even more, it does not really solve the problem of how to communicate with all the agents at once (in a kind of broadcast or multicast situation).

Being able to address a group of people is a necessity of human communication. In fact, this paper does exactly this. The way we human beings deal with the fact that our listeners (or readers) might have slightly different interpretations of the concepts we address is to have an idea where everyone agrees and where there is potential for misunderstandings. And then we address the potential misunderstandings by providing in more detail our understanding. In this paper, we present the idea of non-unanimous ontology concepts that allow us to express the range of agreements on a particular concept based on what an agent learns from a group of teacher agents. We modify the approach of [1] to

enable us to learn such non-unanimous concepts that represent a whole spectrum of possible definitions for a concept. The basic idea is to let the learning agent query its teachers regarding all positive and negative examples it receives from them and use the positive examples all agree on (and the other examples as negative examples) to learn the *core* of the new concept. It then uses all positive examples received (and the negative examples not contradicting these examples) to learn the *periphery* of the new concept.

The agent itself then chooses a concept definition that encompasses the core and is itself encompassed by the periphery. When communicating about this particular concept, the agent uses its awareness of the difference between its own definition, the core and the periphery to enhance the usage of the concept name with explicit references to objects in the periphery but not in the core that are relevant to the communication. We provide a case study in ontologies about organizational structure and courses of universities and show on the one hand side the potential for misunderstandings between universities and on the other side how these misunderstandings can be at least reduced if we use non-unanimous concepts for communication.

2 Basic Definitions

In this section, we provide some basic definitions around ontologies and agents on which we will build in the following sections.

2.1 Ontologies and Concepts

A formal definition for ontology has been presented in [6] in which an *ontology* has been defined as a structure $\mathcal{O} := (C, \leq_C, R, \sigma, \leq_R)$. C and R are two disjoint sets with members of C being called *concept identifiers* and members of R are *relation identifiers*. \leq_C is a partial order on C called *concept hierarchy* or *taxonomy* and \leq_R is a partial order on R , named *relation hierarchy*.

$\sigma : R \rightarrow C^+$ is a function providing the argument concepts for a relation such that $|\sigma(r_1)| = |\sigma(r_2)|$ for every $r_1, r_2 \in R$ with $r_1 \leq_R r_2$ and for every projection π_i ($1 \leq i \leq |\sigma(r_1)|$) of the vectors $\sigma(r_1)$ and $\sigma(r_2)$ we have $\pi_i(\sigma(r_1)) \leq_C \pi_i(\sigma(r_2))$. If $c_1 \leq_C c_2$ for $c_1, c_2 \in C$, then c_1 is called a *subconcept* of c_2 and c_2 is a *superconcept* of c_1 . Obviously, the relation \leq_C is supposed to be connected with how concepts are defined. In the literature, taxonomies are often build using the subset relation, i.e. we have

$$c_i \leq_C c_j \text{ iff for all } o \in c_i \text{ we have } o \in c_j.$$

This definition of \leq_C produces a partial order on C as defined above and we will use this definition in the following for the ontologies that our agents use.

Concepts often are seen as collections of objects that share certain *feature* instantiations. In this work, for an on-

tology \mathcal{O} we assume that we have a set of features $\mathcal{F} = \{f_1, \dots, f_n\}$ and for each feature f_i we have its domain $D_i = \{v_{i1}, \dots, v_{im_i}\}$ that defines the possible values the feature can have. Then an object $o = ([f_1 = v_1], \dots, [f_n = v_n])$ is characterized by its values for each of the features (often one feature is the identifying name of an object and then each object has a unique feature combination). By \mathcal{U} we denote the set of all (possible) objects. In machine learning, often every subset of \mathcal{U} is considered as a concept. In this work we want to be able to characterize a concept by using feature values. Therefore, a *symbolic concept* c_k is denoted by $c_k([f_1 = V_1], \dots, [f_n = V_n])$ where $V_i = \{v'_{i1}, \dots, v'_{ij_i}\} \subseteq D_i$ (if $V_i = D_i$ then we often omit the entry for f_i). An object $o = ([f_1 = v_1], \dots, [f_n = v_n])$ is *covered* by a concept c_k , if for all i we have $v_i \in V_i$. In an ontology according to the definition above, we assign a concept identifier to each symbolic concept that we want to represent in our ontology.

2.2 Agents

A general definition that can be instantiated to most of the views of agents in literature sees an agent $\mathcal{A}g$ as a quadruple $\mathcal{A}g = (Sit, Act, Dat, f_{\mathcal{A}g})$. *Sit* is a set of situations the agent can be in, the representation of a situation naturally depending on the agent's sensory capabilities, *Act* is the set of actions that $\mathcal{A}g$ can perform and *Dat* is the set of possible values that $\mathcal{A}g$'s internal data areas can have. In order to determine its next action, $\mathcal{A}g$ uses $f_{\mathcal{A}g} : Sit \times Dat \rightarrow Act$ applied to the current situation and the current values of its internal data areas.

As we want to focus on the knowledge representation used by agents and how this is used for communication, we have to look more closely at *Dat*. We assume that every element of *Dat* of an agent $\mathcal{A}g$ contains an ontology area $\mathcal{O}_{\mathcal{A}g}$ that represents the agent's view and knowledge of concepts. There might be additional data, beyond features, that the agent requires from time to time, about concepts and this data is naturally also represented in *Dat*. Also, there will be additional data areas representing information about the agent itself, knowledge about other agents and the world that the designer of the agent may want to be represented differently than in $\mathcal{O}_{\mathcal{A}g}$.

3 Learning Concepts from Several Teachers

In this section we provide a brief description of the multi-agent concept learning we presented in [1]. As already stated, we have developed a method that demonstrates how an agent can learn new concepts for its ontology with the help of several other agents. This assumes that not all agents have the same ontology. We additionally assume that there are only some base features $\mathcal{F}_{base} \subseteq \mathcal{F}$ that are known

and can be recognized by all agents and that there are only some base symbolic concepts C_{base} that are known to all agents by name, their feature values for the base features and the objects that are covered by them. Outside of this common knowledge, individual agents may come with additional features they can recognize and additional concepts they know. Given this setting, agents will develop problems in working together, since the common grounds for communication are not always there. To come up with a solution for this problem, agents need to acquire the concepts outside of C_{base} that other agents have, at least those concepts that are needed to establish the necessary communication to work together on a given task. The basic idea of [1] is to have an agent *learn* a required concept (or at least a good approximation of it) with the *help* of the other agents acting as teachers.

3.1 The General Interaction Scheme

In the following, Ag_L refers to the agent that wants to learn a new concept and the other agents, Ag_1, \dots, Ag_m , will be its teachers. Ag_L has an ontology $\mathcal{O}_L = (C_L, \leq_C, R_L, \sigma_L, \leq_{R_L})$ and knows a set of features \mathcal{F}_L . Analogously, Ag_i has an ontology $\mathcal{O}_i = (C_i, \leq_C, R_i, \sigma_i, \leq_{R_i})$ and knows a set of features \mathcal{F}_i . For a concept c known to the agent Ag_i , this agent has in its data areas a set $pe x_i^c \subseteq \mathcal{U}$ of positive examples for c that it can use to teach c to another agent. Part of Act_L are actions `QueryConcept`, `AskClassify`, `Learn`, and `Integrate`, while part of the *Act_is* are the actions `FindConcept`, `CreateNegEx`, `ReplyQuery`, `ClassifyEx` and `ReplyClass`. For teaching Ag_L a new concept c_{goal} , we have as general interaction scheme:

After becoming aware that there is a concept that it needs to learn, Ag_L performs the action `QueryConcept`(`identifier`, $\{[f'_1 = V'_1], \dots, [f'_l = V'_l]\}$, O_{goal}). The three parameters of `QueryConcept` allow for three different ways to identify to the teachers what Ag_L is interested in. The parameter `identifier` allows Ag_L to refer to a concept name it observed from other agents, which means that `identifier` is an element of C_i for some agent(s) Ag_i . By $\{[f'_1 = V'_1], \dots, [f'_l = V'_l]\}$, Ag_L can use a selection of features $f'_j \in \mathcal{F}_{base}$ and the values $V'_j \subseteq D_{f'_j}$ that Ag_L thinks are related to the concept c_{goal} . Finally, $O_{goal} \subseteq \mathcal{U}$ is a set of objects that Ag_L thinks are covered by c_{goal} .

Each Ag_i then reacts to Ag_L 's query by performing `FindConcept`(`identifier`, $\{[f'_1 = V'_1], \dots, [f'_l = V'_l]\}$, O_{goal}). Naturally, already each of the parameters can point to different concepts that an agent Ag_i knows of. In fact, if Ag_L provides several objects in O_{goal} , they might be classified by Ag_i into several of its concepts. So, Ag_i first collects all the concepts that fulfill the query into a candidate set C_i^{cand} and then it has to evaluate all these concepts to

determine the concept that is the best fit. So, the output of `FindConcept` is a set of candidate concepts C_i^{cand} . To select the "best" candidate c_i out of C_i^{cand} , there are many different ways how an evaluation of the candidates can be performed. Each of the 3 query parts can contribute to a measure that defines what is "best", but how these contributions are combined can be realized differently. For an example of the selection process, please refer to [1].

The number of examples communicated to Ag_L by each agent is a parameter of our system. So, in the next step, each teacher selects the given number of elements out of the set of positive examples, $pe x_i^{c_i}$, for the best candidate concept c_i and we call this set p_i . Again, there are many possible ways how this selection process can be done, so far we used random sampling of $pe x_i^{c_i}$. By then performing `CreateNegEx`(c_i), the teacher agents produce a given number of (good) negative examples for c_i , which produces the set n_i . Since every concept c_j other than c_i (and its subconcepts) can be categorized as a counter concept, the number of objects associated with these c_j s (which naturally are negative examples) is often very high. In [1] we used both taxonomy information (siblings of c_i) and a relation `is-similar-to` to select the concepts from which we randomly selected examples as elements for n_i .

`ReplyQuery`(c_i, p_i, n_i) is the last action performed by a teacher agent before the initiative goes back to the learner. It sends the result back to the learner. Ag_L collects the answers (c_i, p_i, n_i) from all teachers, $pe x^{c_{goal}} = \bigcup_{i=1}^m p_i$ and $ne x^{c_{goal}} = \bigcup_{i=1}^m n_i$, and then uses a concept learner to learn c_{goal} from the combined examples (action `Learn`($(p_1, n_1), \dots, (p_m, n_m)$)). Naturally, the concept learner only uses features and their values from \mathcal{F}_L . In case of conflicts between the teacher agents, the learner employs one of several methods to resolve these conflicts. Because conflict resolution is strongly related to our new idea of non-unanimous concepts, we will discuss these methods in detail in the next subsection. As the final step of our scheme, Ag_L uses the learned c_{goal} to construct an ontology path C_{path} leading to c_{goal} within its ontology \mathcal{O}_L utilizing action `Integrate`(c_{goal}).

The result of this learning/teaching scheme is the description of c_{goal} in terms of Ag_L 's feature set \mathcal{F}_L and an updated ontology $\mathcal{O}_L^{new} = (C_L^{new}, \leq_C, R_L, \sigma_L, \leq_{R_L})$. Ag_L will also create a set $pe x_L^{c_{goal}}$ in case another agent wants Ag_L to teach it c_{goal} .

3.2 Conflict Resolution

Learning from a group of agents is a very conflict prone process compared to just learning from one agent. It can easily happen that the best concepts c_i and c_j that Ag_i and Ag_j identified are not the same. The worst case can be that an example that Ag_i sent as being positive for c_{goal}

Ag_j sent as a negative one. But we can also have more indirect conflicts where a learning algorithm simply cannot come up with a concept description that covers all objects in $prex^{c_{goal}}$ while not including any objects in $nex^{c_{goal}}$. There are several methods how we can solve this problem and these methods represent different degrees of willingness to satisfy the teacher agents (by Ag_L).

For our system of [1], we have chosen the following conflict resolution to produce c_{goal} for Ag_L . After the learning component of Ag_L has performed `Learn` and produced a more precise c_{goal} , Ag_L will test all elements of $prex^{c_{goal}}$ and $nex^{c_{goal}}$ for correct classification by this new c_{goal} . For all the example objects that are not correctly classified, we go back to the teacher agents and ask them to classify these examples according to the c_i they used to produce their examples. We then treat the answers as votes and include all positive examples for which a majority of the teachers voted, while requiring the exclusion of all negative examples for which a majority voted. This produces some kind of compromise concept that might appeal to most of the teachers (although it might not be identical to any of the c_i).

Obviously, there are other conflict resolution methods. If the learner wants to be very strict and sure that the c_{goal} it produces is a subset of each c_i , it will only accept positive and negative examples for which the vote was unanimous. On the other side of the spectrum of possible methods is to accept every object as positive example for which at least one teacher says this is a positive example (and to adjust the negative examples accordingly). This might result in a very huge concept, but there can be situations where this is what a user might want.

While each possible conflict resolution method will come up with some concept for the learner’s ontology according to our definition of a concept from Section 2.1, the fact that each of these methods makes sense points to a general problem of our definition of concepts in ontologies, at least if we think about our general goal, namely allowing an agent to communicate with other agents without having to use a common ontology. Therefore we have to rethink our concept definition to allow for making use of the information the learner has collected with regard to the opinions of the teachers about objects in and outside of c_{goal} .

4 Non-unanimous Concept Ontologies

One of the basic assumptions of our method from Section 3.1 is that different agents will often have at least slightly different definitions for a concept, due to the reasons mentioned before. This reflects well what we observe among human beings. If all communications only involve two agents/persons, this fact would not produce a lot of problems, since an agent could learn the definition of a concept of every other agent it communicates with and use the

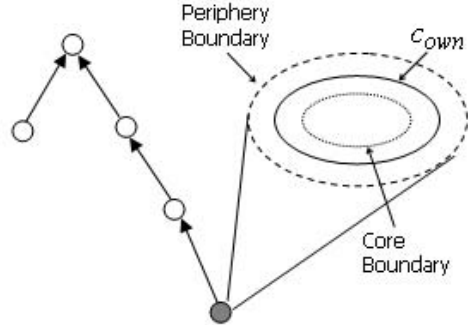


Figure 1. A non-unanimous concept

appropriate definition in each communication. But the moment an agent has to communicate with two or more other agents at the same time, there is a serious problem: what if these agents differ in their definitions of the concepts the communication is about? Obviously, this is rather likely!

The solution used by human beings is to be aware of what (most) agents agree on and to provide additional information to clarify aspects that parts of the audience might misunderstand. Note that this does not mean that an agent is unclear about its own definition of a concept. We believe that for the individual work of an agent having a precise definition of a concept is important. The ability to be aware of potential misunderstandings when communicating with other agents is what we are after. And the method described in Section 3.1 is what suggests to us a way how to obtain this ability for an agent.

4.1 Non-unanimous Concepts

In order to allow to express the range of possible misunderstandings about a concept, instead of representing a concept by one feature set as in Section 2.1, we use 3 such feature sets, which means that we essentially use 3 concepts:

$$c = (c_{core}, c_{own}, c_{periphery}).$$

These three “normal” concepts provide us with two boundaries and the agent’s own definition of the particular concept (represented by c_{own}).

Figure 1 gives a graphical representation of the trio of “old” concepts that we use to represent a *non-unanimous concept*. The inner boundary c_{core} is intended to provide the agent with a concept definition that represents all objects for which there is no doubt among all agents that they belong into the concept, so obviously c_{core} covers the *core* of the concept. The outer boundary $c_{periphery}$ covers all objects that ever might be considered to belong to the concept, which means that all objects not covered by $c_{periphery}$ for sure are not in the concept c . So, essentially $c_{periphery}$ defines the extend of the *periphery* of the concept.

If we want to use non-unanimous concepts within ontologies, then most of what we defined in Section 2.1

does not have to be changed. The only potential problem is \leq_C , since obviously there is always the chance that the peripheries of two concepts might overlap (given that the objects in $c_{periphery} - c_{core}$ are somewhat questionable with regard to really representing the concept and the objects in $c_{periphery} - c_{own}$ are not covered by the concept in the point of view of the agent). If we have to provide the equivalent of \leq_C for non-unanimous concepts, then we will use the relation $\leq_{C_{nu}}$, which we define as

$$(c_{core_1}, c_{own_1}, c_{periphery_1}) \leq_{C_{nu}} (c_{core_2}, c_{own_2}, c_{periphery_2}),$$

iff for all $o \in c_{own_1}$ we have $o \in c_{own_2}$.

This makes sense, since c_{own} represents what an agent thinks is the concept.

Finally, let us take a look at the representation of a non-unanimous concept c_{nu} using features. All the three concepts $c_{core_{nu}}$, $c_{own_{nu}}$ and $c_{periphery_{nu}}$ naturally have a presentation as feature value sets according to Section 2.1. For a feature f_i this means that we have now three value sets, namely $V_{i_{core}}$, $V_{i_{own}}$ and $V_{i_{periphery}}$, with $V_{i_{core}} \subseteq V_{i_{own}} \subseteq V_{i_{periphery}} \subseteq D_i$. So, associated with potential misunderstandings in communication will be certain feature values for some of the features that an agent uses in its ontology.

4.2 Learning Non-unanimous Concepts

Having defined non-unanimous concepts, the next question is obviously how do we create such a non-unanimous concept for an agent. The answer is very straightforward: by learning the non-unanimous concept using our method from Section 3.1. Without non-unanimous concepts, in [1] we had to choose one of the three conflict resolution methods that we described in Section 3.2 and this method was to use the concept that could be learned from the positive and negative examples on which a majority of the teachers agreed on. If we represent concepts by the sets of the objects covered by them, then Figure 2 visualizes the three different intended outcomes of the concept learning process for the three conflict resolution methods for 3 agents Ag_1 , Ag_2 and Ag_3 and the candidates c_1 , c_2 , and c_3 that these agents selected.

If we assume that the set of teachers that an agent uses to learn a concept represents well the different understandings that a group of agents have about a concept, then the three different conflict resolution strategies seem to be a perfect way to learn the 3 concepts $c_{core_{goal}}$, $c_{own_{goal}}$, and $c_{periphery_{goal}}$ that we need to define a non-unanimous concept. We use what the learner produces out of the examples all the teachers agree on (the Intersection area of Figure 2) as $c_{core_{goal}}$ and the result of the concept learning when using everything as positive example that is suggested by at least one teacher as a positive example is $c_{periphery_{goal}}$ (Union area).

In defining $c_{own_{goal}}$ we can use any concept that lies

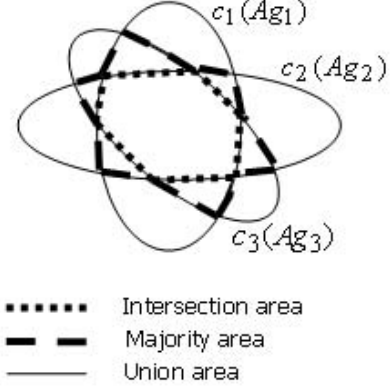


Figure 2. Visualization of conflict resolution

between $c_{core_{goal}}$ and $c_{periphery_{goal}}$ (including these concepts). Since $c_{own_{goal}}$ is intended to express the agent's personal believe in what the concept is, we think that using some kind of compromise concept between the two extremes is a good idea and using the result of what a majority of the teachers agrees on is a good compromise. Given the usage of non-unanimous concepts that we target, it makes sense to create a set of positive examples for each of the 3 concepts $c_{core_{goal}}$, $c_{own_{goal}}$, and $c_{periphery_{goal}}$. We will call these example sets $pe_x^{c_{core_{goal}}}$, $pe_x^{c_{own_{goal}}}$, and $pe_x^{c_{periphery_{goal}}}$.

4.3 Using Non-unanimous Concepts in Group Communications

Let us look at an example that demonstrates the problem we are solving. This example stems from introductory Math classes and essentially represents the question: Is zero a natural number? The concept of natural numbers is introduced in nearly every elementary Math course and while there is absolutely no disagreement between mathematicians with regard to 1, 2, 3 and so on being natural numbers, we often are told that 0 is a natural number in introductory courses of set theory, while introductory courses in number theory are usually seeing 0 not as a natural number. Using non-unanimous concepts, we can model this fact by having c_{core} as the set of number objects 1, 2, 3 and so on and $c_{periphery}$ as the set of number objects 0, 1, 2, 3 and so on.

An agent Ag that has to learn about natural numbers will make its own decision about zero. But regardless of this decision, what if this agent has to communicate with a large group of mathematicians about something related to natural numbers? If Ag has decided to favor the view of 0 being a natural number (i.e. $c_{own} = c_{periphery}$), then by using the concept identifier "natural number" and explicitly stating that whatever it says holds true for 0 (which is the sole element of $pe_x^{c_{own}} - pe_x^{c_{core}}$) Ag can be sure that its communication is understood by everyone. On the other side,

if Ag decided to have $c_{own} = c_{core}$ and wants to express that something does not hold for all natural numbers, then it should again use the concept identifier "natural number" in its communication and explicitly state that this something also does not hold for 0 (since now 0 is the sole element of $pe x^{c_{periphery}} - pe x^{c_{own}}$).

More precisely, assuming that after learning a concept learner and teacher establish a common concept identifier, using non-unanimous concepts to communicate with groups is based on enhancing the usage of the concept identifier with additional objects to convey to the listeners the agent's understanding of the concept. If we communicate positively about a concept, these additional objects are taken from $pe x^{c_{own}} - pe x^{c_{core}}$ (and from $pe x^{c_{periphery}} - pe x^{c_{own}}$ if we want to refer to the universe without the concept).

While it is possible to simply use the whole set $pe x^{c_{own}} - pe x^{c_{core}}$ in the communication, we have to assume that this might be a large set (in fact, there might be infinitely many objects in c_{own} that are not in c_{core} and using the stored positive examples is already just an approximation). Communicating all these examples might be too costly for the agent and therefore we assume that the first step of this agent is to determine a maximal number $exmax$ of objects that it is willing to use in the communication. $exmax$ might be a constant parameter or it could be adjusted based on the success of earlier communications with a group of agents.

In order to select the $exmax$ examples, we naturally want these examples to cover the difference between c_{own} and c_{core} as best as possible. Again, there are different methods how this can be achieved. A simple, but in our opinion good enough, method is the following: Let $FC = \cup_{i=1}^n \{(f_i, v_i) | v_i \in V_{i_{own}} - V_{i_{core}}\}$. For each element o in $pe x^{c_{own}} - pe x^{c_{core}}$ compute $cover(o) = |\{(f, v) \in FC | v \text{ is a value of feature } f \text{ for } o\}|$ and select one of the elements, let's say o' , with the highest $cover$ -value. Then remove all the (feature, feature value) pairs of o' from FC and repeat this selection $exmax - 1$ times.

5 Case Study: University Ontologies

To look at the usefulness of non-unanimous concepts, we use the same application scenario as in [1], namely the university units and courses domain.

5.1 The University Units and Courses Domain

The university units and courses domain consists of files describing the courses offered by Cornell University, the University of Washington and the University of Michigan, together with ontologies for each of the universities describing their organisational structure (see [3] and [7]). In our examples, each of the universities is represented by an agent (Ag_C , Ag_W , Ag_M) and these agents are acting on the one

Unit	c_{core}	c_{own}	$c_{periphery}$
Mathematics	264	392	501
Computer Science	188	381	505
Linguistics	203	283	346

Table 1. Some positive example statistics

hand side as the teachers to an agent Ag_L and on the other side as the agents Ag_L communicates with afterwards.

The objects of this domain are the course files that consist of a course identifier, a plain text course description and the prerequisites of a course. All in all, there are 19061 courses among the three universities and each university's ontology has at least 166 concepts on top of their courses. To create features and their values fulfilling the definitions from Section 2.1 we used or-combinations of key words as single features (with differing key word sets for the different agents to create a situation where the features used by the agents are different) and a feature was fulfilled, if one of the words from the or-combination occurred in the course description. This view of features corresponds nicely with the learning method we chose for Ag_L , namely the method of [4]. For our evaluations regarding non-unanimous concepts, we let Ag_L indeed learn all the concepts we are reporting on using the method described in Section 3.1. To learn the concepts very accurately, we allowed each teacher agent to communicate 110 positive and negative examples to Ag_L .

Table 1 presents some statistics on three concepts/units, namely the number of objects (courses) that are in the three concepts within a non-unanimous concept. While many people might have expected that the difference between the core and the periphery for Computer Science is rather large, it surprised us that this is also true for Mathematics. And even for Linguistics there is still quite a number of courses for which the three universities differ in their categorization. Naturally, we do not have enough space to provide all the details that are represented by Table 1, but we can provide a few details about what happened regarding Mathematics in this subsection and there will be more details about Computer Science in the next subsection.

All universities agree that the courses with titles Calculus I and Mathematical Logic are in c_{core} . But, for example, a course with the name Combinatorial Theory was not classified into Mathematics by Ag_C , while Model Theory did not find the approval by Ag_W . This shows that being able to deal with differences in opinion between agents is something that agents need to be able to do.

5.2 Group Communication: An Example

As already stated, we were not so surprised to see a lot of objects in $c_{periphery}$ that are not in c_{core} for the concept Computer Science. This is due to the fact that already the

taxonomies of the three agents were rather different with regard to where they put the Computer Science concept. For example, Ag_M runs the Computer Science program as a program within the Engineering faculty as does Ag_W . In contrast to this, Ag_C has the Department of Computer Science in its Science faculty. This also means that Ag_L favors for its c_{own} (for Computer Science) the more engineering view, but, in contrast with what we see from time to time with real human agents, it is aware of the potential for misunderstandings due to using non-unanimous concepts.

Consider the following communication problem: Ag_L as representant of a new university that learned their structure from Ag_M , Ag_W , and Ag_C is giving a talk on its university in front of agents from many North-American universities. One of the points of the talk is to tell the other agents that at Ag_L 's university, all courses in the Computer Science program are taught by real professors (no instructors).

Due to learning from the three teachers (that we assume to represent the extremes with regard to opinions on what is part of a Computer Science program, simply because they are the places that are providing the data), Ag_L has Computer Science as a non-unanimous concept. For example, in its c_{core} for Computer Science it has courses with names like Computer Programming I, Design and Analysis of Algorithms II, Computer Networks, or Computer Architecture that have course descriptions that all teacher agents classify into Computer Science. Some of the courses that only one of the teachers voted for as belonging to Computer Science have the names Reliable Computing Systems, Computational Molecular Biology, Computers and Society, or Computational Tools for Finance.

Among the 193 courses that are not in c_{core} but in c_{own} for Computer Science of Ag_L are, for example, courses named Theory of Computing, Introduction to Formal Models, Applied Logic, Parallel Computing, or Computer Game Design and Development. If we would set the parameter $exmax$ of Section 4.3 to 1, then Ag_L would enhance its communication about no instructors in Computer Science by using something like: *In Computer Science including Theory of Computing, we are not using any instructors.* This is because we have the following values for the $cover$ function: $cover(\text{Theory of Computing}) = 5$, $cover(\text{Introduction to Formal Models}) = 4$, $cover(\text{Applied Logic}) = 3$, $cover(\text{Parallel Computing}) = 3$ and $cover(\text{Computer Game Design and Development}) = 2$.

Naturally, $exmax = 1$ still leaves room for misunderstandings (in fact, without stating all 193 courses there is always room for misunderstandings), but for $exmax = 3$ we first add Parallel Computing to the commu-

nication about Computer Science and then we also add Computer Game Design and Development. The $cover$ -values for Introduction to Formal Models and Applied Logic are greatly reduced in the next rounds, since they are using the same features as Theory of Computing. We then get the following communication: *In Computer Science including Theory of Computing, Parallel Computing, and Computer Game Design and Development, we are not using any instructors.*

6 Conclusion

We presented ontologies with non-unanimous concepts as a way to represent potential misunderstandings about concepts that might make communicating simultaneously with a group of agents difficult. By using non-unanimous concepts to enhance communication with references to objects that clarify the possible misunderstandings, simultaneous communication to a group of agents can be substantially improved. To create non-unanimous concepts, they have to be learned from a group of teachers that represent the different understandings of the concept. By using our method for learning of concepts from several teachers of [1] and using it with several conflict resolution strategies we can easily create non-unanimous concepts.

Future work will look into alternative definitions for c_{own} and how to define relations on top of non-unanimous concepts to improve communicating with several agents at once even more.

References

- [1] M. Afsharchi, B.H. Far, J. Denzinger: Ontology-Guided Learning to Improve Communication between Groups of Agents, Proc. AAMAS 2006, Hakodate, 2006, pp. 923–930.
- [2] J. van Diggelen, R.J. Beun, F. Dignum, R.M. van Eijk, J.-J.Ch. Meyer: ANEMONE: An Effective Minimal Ontology Negotiation Environment, Proc. AAMAS 2006, Hakodate, 2006, pp. 899–906.
- [3] Illinois Semantic Integration Archive. <http://anhai.cs.uiuc.edu/archive/>, as seen on Jan 30, 2005.
- [4] D. Koller, M. Sahami: Hierarchically Classifying Documents Using Very Few Words, Proc. ICML-97, 1997, pp. 170–178.
- [5] S. Sen, P.P. Kar: Sharing a concept, AAI Tech Report SS-02-02, Stanford, 2002.
- [6] G. Stumme: Using Ontologies and Formal Concept Analysis for Organizing Business Knowledge, in Becker, Knackstedt (eds.): Wissensmanagement mit Referenzmodellen – Konzepte für die Anwendungssystem- und Organisationsgestaltung, Physica, 2002, pp. 163–174.
- [7] University of Michigan academic units. <http://www.umich.edu/units.html>, as seen on Jan 30, 2005.
- [8] A.B. Williams: Learning to Share Meaning in a Multi Agent System, JAAMAS 8(2), 2004, pp. 165–193.