Facilitating Agent Communication through Detecting, Diagnosing and Refining Ontological Mismatch

Facilitating Agent Communication Through Detecting, Diagnosing and Refining Ontological Mismatch

Fiona McNeill, Alan Bundy, Chris Walton

Abstract

The development of the semantic web makes the facilitation of agent communication an issue of increasing importance. It is often assumed that agents are using the same ontology and hence can understand one another, but the dynamic and distributed nature of the semantic web can mean that this is not always a valid assumption. We describe a system that can dynamically discover some ontological mismatches between agents during communication and then refine them, to enable communication to proceed successfully between these agents.

1 Introduction

We are exploring the situation in which an agent (which we refer to as the main agent, though the system is distributed) is attempting to achieve goals by forming and executing plans in a given domain. The main agent will form plans to achieve the goals which it is given based on its understanding of the world. Each plan step will be executed through interaction with other agents: for example, a plan step `(Buy Ticket)` may be executed through locating and communicating with a *ticket-selling agent*. The main agent initially makes the assumption that these other agents have the same ontology as it, and thus the conditions under which the main agent believes an action is executable will match the conditions under which the other agents will perform it. Additionally, this assumptions entails that the effects of a given action being performed will match the main agent's expectations of the effects. It is very useful to make such assumptions, as this allows one to form plans based on one's understanding of the world.

However, because ontologies are dynamic and are regularly updated, leading to agents having different versions of the same ontology, and also because users can take an off-the-shelf ontology and the adapt it for their particular needs, we often find that other agents may have ontologies that are similar but are slightly out of sync. This can lead to plan execution failure, since the other agents have expectations of the world that do not match the main agent's expectations.

We use the word *ontology* to mean the whole of the agent's understanding of the world. This includes both the general information about the kinds of things that exist, the *signature*, and the specific details for each case, the *theory*. The signature contains information such as: names of predicates, arity of predicates, the type hierarchy, and so on. The theory contains rules, facts, types of individuals and so on. In our system, we use agents who have ontologies represented in the Knowledge Interchange Format (KIF) [3]. KIF was chosen because it is a recognised representation format for agent ontologies, and because it is extremely expressive, supporting full first-order logic. The expressiveness of KIF allows us to investigate the problem of refinement in as broad and complete a way as possible.

A key concept of the semantic web is the automated processing of ontological knowledge by agents and services. [1] outlines how these are central to the role of the semantic web, and how they require ontological matching to perform appropriately. However, the nature of the semantic web means that strict controls on these ontologies are impractical: it must allow for, amongst other things, partial information, erroneous information and the evolution of ontologies [6]. All of these requirements cause difficulties for agent communication. Our approach is a way of dealing with these ontological mismatches without enforcing tight ontological controls on agents. It is intended to automatically solve a subset of the many problems surrounding ontology mapping, merging and alignment, which are all essential for communication in multi-agent systems. It allows agents to learn about the area in which they are currently active, and thus be more capable of acting successfully within it, without receiving any extraneous information. This is particularly useful in complex and dynamic domains, where a complete and up-to-date representation of the entire domain may be intractable or undesirable.

The ontology mismatch is highlighted by a particular error in the theory, which will usually indicate an error in the underlying signature. Once this signature error has been refined, the parts of the theory which included instantiations of this signature error must also be refined. This will include the original theory error, but also, in most cases, other theory errors. For example, the main agent may have `Money` represented in its signature as a unary predicate `(Money ?Amount)`, whereas the agent with which it is interacting may have this represented as a binary predicate `(Money ?Amount ?Currency)`. This error will come to light through an instantiation of the signature in the theory: perhaps the main agent uses `(Money 10)` where `(Money 10 Dollars)` is expected. In this particular example, diagnosing the appropriate signature refinement is not particularly difficult; if another agents put a question to the main of the form: `(Money 10 Dollars)`, the main agent can examine what questions it expected to be asked (i.e. what preconditions need to be fulfilled for the action to be performed), and explore the relationship between them and the question actually asked. In this case, it will discover that `(Money 10 Dollars)` is related to `(Money 10)`. It can discover the type of the extra argument by searching its ontology for type information about dollars or, if it has none, by asking the other agent for this type information. However, in this case the theory refinement is rather more problematical. It is clear that each instantiation of this ontological object must have an extra argument, and it is clear what the type of this argument must be, but it is not clear what the specific instantiation should be. If we know that the there are three instances of type Currency: Dollars, Sterling and Euros, we will, in most cases, be able to say no more than:

```
(Money Agent-1 10) →
        (Money Agent-1 10 ?Currency),
?Currency = {Dollars,Sterling,Euros}
```

That is, we may know what the possible instantiations are but not which one is valid in this particular case. Even this we cannot be certain about, as there may be other possible instantiations of Currency that we do not currently know about. If there is sufficient information to perform theory refinement, this may then lead to more complex plans. For example, if the main agent has `money 50 Euros` and it knows that `money 10 Dollars` will be required, then some kind of conversion action will need to be part of the new plan.

The system we describe is not intended to be a complete solution to the problem of ontology mismatch. Rather, it is designed to provide a set of tools for refining particular kinds of ontological mismatches which can be expected to be frequently encountered.

2 Related Work

This work draws on many fields of research, including work in planning, abstraction techniques and machine learning. Of particular interest are the fields of belief revision and ontology mapping. Considerable work has been done in the related field of Ontology Mapping, surveyed by Kalfoglou and Schorlemmer [5]. This subject is explored in relation to Semantic Web Services in [2] and [11]. The former has strong correlations to the theory refinement part of the task, but differs from the main focus of our project, which centres on refining ontologies syntactically (signature refinement) rather than purely semantically (theory refinement). The main difference with the latter is that ontology mapping usually assumes that one has complete access to the ontologies one is attempting to map, whereas we are attempting this with the partial information that can be extracted from plan failure and from putting specific questions to other agents. We believe that in the kind of scenario we are envisaging, where disparate agents are interacting in a loosely controlled environment, it is not reasonable to believe that agents will be either willing to or capable of revealing their entire ontologies for others to inspect. Rather, this information must be gathered by putting precise questions to agents. Our work is therefore similar to this body of work as far as performing refinements are concerned, but the diagnosis aspects, which are the central part of this project, are very different. As mentioned above, we do not claim to have produced a general solution to the problem with which we are dealing, but rather a system that can solve this problem in certain situations. Likewise, there is no full solution for automated general ontology mapping, and the techniques used in this field are not guaranteed to be sufficient or complete.

3 Refinement System

Figure 1 illustrates the architecture of our system [7, 8]. There are four main aspects of this system: planning, agent communication, diagnosis and refinement implementation. Of these, the planning and agent communication systems are kept as simple as possible and are both built, to some extent, on work by others. The planning system uses the FF planner, and the agent communication system is built on top of Linda, a Prolog agent platform. The diagnosis and refinement implementation systems are the focus of our work.

The flow of execution in our system is as follows:

1. The main agent sends its ontology, together with the goal, to a planner.
2. The planner produces a plan for achieving the goal which consists of a list of actions. This is sent, together with the original representation of the ontology, to a *plan-deconstructor* [9]. The purpose of this component is to provide a justification for each plan step, in the form of the underlying rule for that step, together with a justification for each of its preconditions. This is necessary so that

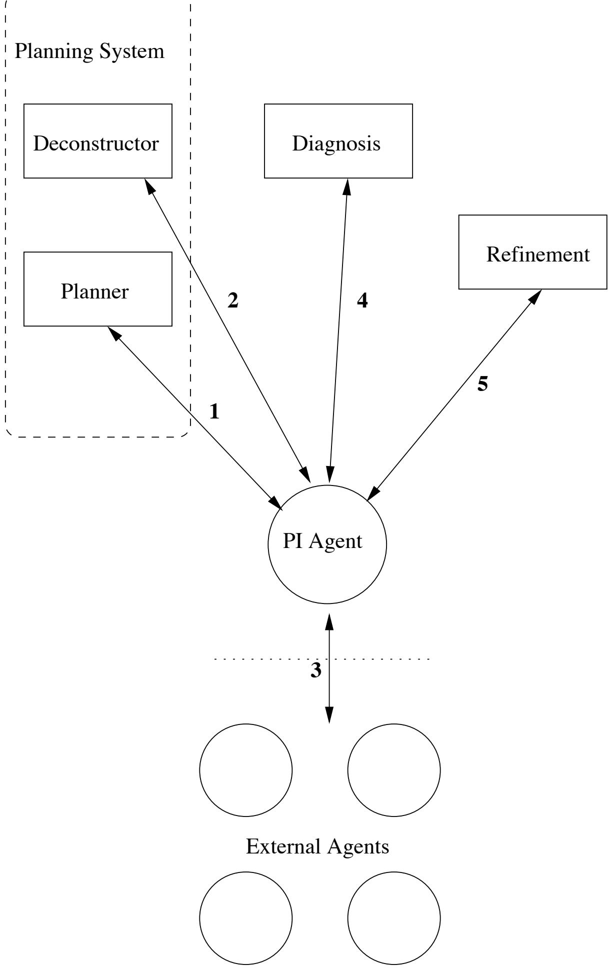

Figure 1: Architecture and interaction of the dynamic ontology refinement system.

when plan failure occurs, this can be linked back to the relevant area of the ontology.

3. The plan, annotated with a justification for each step, is returned to the main agent and execution begins. This execution occurs in an agent communication system, where the main agent can locate the agents with which it needs to interact. If failure occurs, the main agent can refer to the justification of that plan step and find a general area within the ontology where the error occurs. Further agent communication is then used to diagnose the precise ontology mismatch.
4. The diagnosis and information about how it should be refined is passed to the refinement system. This system corrects the error and passes the corrected part of the ontology back to the main agent, which updates its ontology accordingly.
5. The process now begins again, with the main agent forming plans on the basis of an ontology which is likely to be more reliable.

The cycle will continue until the ontology is sufficiently accurate for the goal to be achieved.

3.1 Diagnosing the Problem

When the plan fails, the justification for the plan step is referred to. This justification for an action contains the rule that explains the action. Additionally, there will be information as to why that rule should have been applicable in the particular situation; that is, an explanation as to why each of the preconditions was believed to be true. The rule itself may be incorrect, or any of the preconditions may be erroneously believed to be true. If a precondition is not true, this may be a representational (signature) problem: perhaps it has the wrong name, the wrong arity, and so on, or it may be a theory problem. If the rule is incorrect, this will mean there is a problem either with the preconditions of the rule or with its postconditions.

Much information can be obtained by observing the point of failure, as illustrated in Figure 2. For example, did failure occur immediately after a request was made to another agent? Or did the other agent put some queries to the main? If there were queries, were any of them surprising? The main would expect to be asked about some of its preconditions, so that the other agent could verify that they were correct before performing the action. However, the main would not expect to be asked about anything that did not exactly match one of its preconditions, thus such a query would be deemed *surprising*. Analysing queries, particularly surprising ones, can provide much information about potential failure. If there were no queries, then it is clearly not a response by the main that led to failure. Perhaps the other agent had unexpected preconditions that were not fulfilled, or preconditions that the main shared that were not fulfilled; not all of these will be verified through queries. If there were queries that were expected, then the main's answer to one of these was probably inadequate. If there were surprising queries then perhaps there is a missing precondition for the rule. Alternatively, this query may be related in some way to a precondition, but differ from it either from a signature point of view (incorrect arity, naming, argument order, etc.), or from a theory point of view (incorrectly instantiated).

3.2 Refinement Techniques

In order to develop rules for the kinds of refinements that might be necessary, we are considering the kinds of changes that may have been made to the ontology. We are assuming that changes to an ontology are methodical and sensible and not random. So, for example, detail may be added to a predicate by adding an extra argument, or removed by amalgamating several predicates into one supertype predicate. We are assuming that names will be changed to a sub- or supertype, or possibly a sibling type, but not randomly to an unrelated name. We make these assumptions

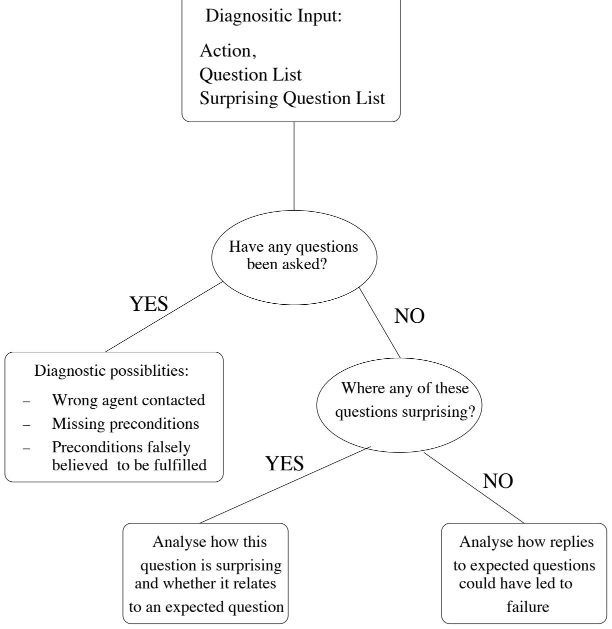
Diagnositic Input:
Action,
Question List
Surprising Question List
Have any questions been asked?
YES
NO
Diagnostic possiblities:
– Wrong agent contacted
– Missing preconditions
– Preconditions falsely believed to be fulfilled
Where any of these questions surprising?
YES
NO
Analyse how this question is surprising and whether it relates to an expected question
Analyse how replies to expected questions could have led to failure

Figure 2: Diagnosis

because we believe that this is the approach that most people will take to altering ontologies, and also because this gives us a basis from which to develop our system. The nature of the system, the fact that there is an unlimited number of ways in which ontologies may be altered, means that we cannot be sure that we will be able to find an appropriate refinement. In some situations, refinement will be impossible. It is, nevertheless, still worthwhile pursuing such an approach, because it will enable refinement in a large number of situations where appropriate interaction would otherwise prove impossible.

We have developed rules for generalising and specialising an ontology, based on *abstraction* and *anti-abstraction* techniques. These abstractions and anti-abstractions are developed from the main types of abstractions identified by Giunchiglia and Walsh in their survey of abstractions [4]. In order to give a taste of the kinds of techniques we are using, we outline some of these below:

1. **Predicate anti-abstractions**
   A single predicate is divided into some number of sub-predicates:
   e.g. `(Money ?X)` maps to `(Dollars ?X)`, `(Euros ?X)`.
2. **Domain anti-abstractions**
   Constants and function symbols are divided up into different cases:
   e.g. `(Money ?X European)` maps to `(Money ?X Euros)`, `(Money ?X Sterling)`.
3. **Propositional anti-abstractions**
   Extra arguments are added to predicates:
   e.g. `(Money ?X)` maps to `(Money ?X Dollars)`, `(Money ?X Sterling)`.
4. **Precondition anti-abstractions**
   Preconditions can be added to rules:
   e.g. `(At Station)` $\rightarrow$ `(Can-buy Ticket)` maps to `(Money ?X)` $\wedge$ `(At Station)` $\rightarrow$ `(Can-buy Ticket)`

Other kinds of errors we have identified include adding new types and predicates, altering the type hierarchy by adding new subtypes, supertypes or siblings, and misordered arguments. In some cases there may be no signature error: a fact may have the correct predicate name, the correct type and order of arguments, and so on, but simply be incorrect. In this case we use the *Shapiro Algorithm*, based on Shapiro's work [10] on detecting errors, to identify the cause of this. This will involve chaining back through the justification to see when this fact came to be believed. If it came to be believed because it was a postcondition of a past rule, we must investigate whether this rule is incorrect.

Of the four types or refinement described above, predicate and precondition anti-abstraction produce no problems in theory refinement. However, domain and propositional anti-abstractions can be difficult to implement in the theory, due to the uncertainty over how to instantiate the extra or incorrect arguments. Abstraction type refinements produce far fewer problems with theory refinement, since removing detail from a theory is much less problematical than adding detail to a theory.

4 Conclusions and Future Work

We have described a system that enables an agent to patch its own ontology in order to successfully interact with other agents who may have ontologies and ontology representations that differ somewhat from its own. We describe some of the types of refinement that we are able to diagnose and correct, and justify why we expect these refinements to cover a large number of ontological mismatches.

A working system that can perform the tasks described above is almost fully implemented. Most of what has yet to be done is focused on communication between the subsystems, which are themselves already in place. This involves automated translation between the different ontological representations required for the various subsystems.

In order to demonstrate the applicability of this system, we intend to identify several off-the-shelf ontologies that have not been developed by us, run agents using different versions of this ontology (where updating of the ontology has gone on between versions), and show that they can use these techniques to facilitate communication.

There is a great deal of future work that we would like to explore with this system. An obvious improvement would be to increase the range of ontological errors that could be identified and patched, and to explore the possibility of using this or a similar system with different ontological representations. With any representation that is as expressive as KIF (i.e. full first-order), we anticipate that this will be a fairly simple process, which would probably just involve some translation. However, many popular representations for agent ontologies, such as OWL, are considerably less expressive. In such cases, we anticipate that the theory behind how to identify and correct errors would remain the same, but the kinds of refinements we would expect to encounter would be somewhat different.

Another interesting avenue for future work would be to introduce some way of remembering past refinements and using this information in making deductions about future refinements. This would be very useful for problems such as identifying anomalous agents, who differ more from other agents than they do from each other: we may not want to refine our ontology to match theirs. Also, we have made the assumption in this system that the main agent is always submissive and willing to change its ontology to fit in with others. It would be interesting to explore a more sophisticated situation, where the main agent was willing to adapt in some situations and not in others; how it could be decided between agents which one was the authority, and so on.

Finally, we have to some extent motivated the project by discussing its relevance to the semantic web. Although we believe that the theory behind the system is indeed highly relevant to any situation where disparate agents from different sources are attempting to interact successfully, the system as it stands is probably not ready to be used directly on the semantic web, mostly due to the ontological restrictions we place on the agents with whom communication can take place (for example, that they have KIF ontologies). We would very much like to make the system more robust and able to deal with a wider range of circumstances, to create a system that can be used immediately in situations such as those encountered on the semantic web.

References

[1] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web, May 2001. Scientific American.

[2] Mark H. Burstein. Ontology mapping for dynamic service invocation on the semantic web. In *Proceedings of Semantic Web Services Symposium*, 2004 AAAI Spring Symposium Series, March 2004.

[3] M. R. Genesereth and R. E. Fikes. Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic-92-1, Stanford, CA, USA, 1992.

[4] F. Giunchiglia and T. Walsh. A theory of abstraction. *Artificial Intelligence*, 56, 1992.

[5] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18:1:1–31, 2003.

[6] Marja-Riitta Koivunen and Eric Miller. W3c semantic web activity. In *Proceedings of the Semantic Web Kick-Off Seminar in Finland*, November 2001.

[7] F. McNeill, A. Bundy, and M. Schorlemmer. Dynamic ontology refinement. In *Proceedings of ICAPS'03 Workshop on Plan Execution*, Trento, Italy, June 2003.

[8] F. McNeill, A. Bundy, and C. Walton. Diagnosing and repairing ontological mismatches. In *Proceedings of the second starting AI Researchers' symposium*, Valencia, Spain, August 2004.

[9] Fiona McNeill, Alan Bundy, Chris Walton, and Marco Schorlemmer. Plan execution failure analysis using plan deconstruction, December 2003. http://planning.cis.strath.ac.uk/plansig/index.php ?page=past22.

[10] Ehud Y. Shapiro. *Algorithmic Program Debugging*. The MIT Press, 1982.

[11] Nuno Silva and Joao Rocha. Ontology mapping for interoperability in semantic web. 2003.