The Use of Collaborative Meta-Agents in Multi-Agent Systems

Charles Dierbach

Towson University
Department of Computer and Information Sciences, Towson, MD 21252, USA
cdierbach@towson.edu

Abstract

In this paper, we discuss the potential application of meta-agents in multi-agent systems. Specifically, we discuss the use of a "collaborative agent," a meta-agent with the goal of facilitating collaboration among agents with similar goals. In this way, collaboration as a goal is moved from agents to the meta-agent. The advantages of such a collaborative agent are given, and a particular formal model of similarity matching is presented. Such a model of similarity matching is a prerequisite part of any such proposed collaborative meta-agent.

Introduction

Software agents are autonomous entities with specific goals. In a multi-agent system (MAS), agents may collaborate, or behave strictly independently. That is, their interactions can be either "cooperative" or "selfish." Therefore, collaboration itself may or may not be a desire of a given agent. Just because a given agent does not pursue collaboration, however, does not mean that collaboration by the agent would not be beneficial. One of the key questions of multi-agent systems is how the inherent interdependence between agents be managed effectively? [Hayes 1999]

In this paper, we investigate the potential use of meta-agents with the goal of facilitating collaboration among agents of a given multi-agent system. Such agents may be referred to as a "collaboration agent." As a result, collaboration as a goal is moved from agents to meta-agent. Since goals among agents of a given domain may be expressed in various ways, there is a fundamental problem of how given sets of goals are found to be similar.

We describe a formal model of similarity matching based on first-order logic to address this problem. Given the work on standardized representations of logic (for example, the first-order logic based representation KIF - Knowledge Interchange Format), we assume the utilization of a standard knowledge representation for the representation of goals among agents in a given multi-agent system.

Multi-Agent Systems and "Collaborative Agents"

Multi-agent systems are viewed as systems of agents capable of solving problems that individual agents cannot. The particular characteristics of such system are ([Sycara 1998]):

- Each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint
- There is no system global control
- Data are decentralized
- Computation is asynchronous.

It is in relation to the first point that collaborative agents may be employed. Such agents can enhance the viewpoint of a given agent by proactively facilitating

cooperation among agents. This contrasts to agents themselves seeking cooperation. This has a number of advantages. First, the facilitation of cooperation is an ongoing process when a collaborative agent is utilized, given that it is the single goal of the agent. Second, more effective collaboration can result given the overview that the collaborative agent has of all collaboration in the system. Third, general strategies of collaboration in multi-agent systems can be more easily controlled, given that such strategies are localized in one agent (the collaboration agent).

Standardized First-Order Logic Based Knowledge Representation

KIF (Knowledge Interchange Format) is a knowledge representation for the exchange of information between computer systems. KIF is a concrete syntax and an instance of a general Common Logic framework for standardized logic representations (Common Logic Standard, [CLSW 2005]). It has the equivalent expressiveness and semantics of first-order logic. In addition, KIF provides for quantification over relations and functions, thus providing capabilities of higher-order logic.

We assume the availability of a standard knowledge representation with the expressiveness of first-order logic (such as KIF) for the representation of agents' goals in a given multi-agent system. Given the various ontologies (e.g., predicates) that may be used by individual agents to express their goals, there remains the issue of how to identify semantic similarity goals constructed over different ontologies.

The differences in representation of semantically similar (or identical) statements may occur in a number of ways. First, two statements may be structurally and semantically similar, but containing different predicates,

Case 1

DOCUMENT(a) ^ SENSITIVE_INFO(a)

TEXT_FILE(b) ^ SENSITIVE_INFO(b)

In this case, the statements are found semantically equivalent (on corresponding entities a and b), if predicates DOCUMENT and TEXT_FILE can be determined to be logically equivalent. Such equivalence can be determined by provided domain knowledge, e.g.,

$$\forall x[\text{DOCUMENT}(x) \Leftrightarrow \text{TEXT\_FILE}(x)]$$

Another way that semantically similar statements may differ is in their syntactic structure,

Case 2

DOCUMENT(a) ^ SENSITIVE_INFO(a)

FILE(b) ^ CONTAINS_TEXT(b) ^ SENSITIVE_INFO(b)

Here, one statement is of the form P ^ Q, and the other P ^ Q ^ R.

Yet another way that similar statements may differ syntactically is in the denotation of entities,

Case 3

$\exists x$ [DOCUMENT(x) ^ SENSITIVE_INFO(x)]

TEXT_FILE(b) ^ SENSITIVE_INFO(b)

In this case, the two statements are structurally similar. However, in one statement, existential quantification is used to denote the entity, and in the other, a constant is used.

These examples demonstrate issues in the identification of similar goals among autonomous agents, when specified in first-order logic, as assumed here. We next

describe a formal model of similarity matching that can address this problem.

The Abstractional Concept Mapping Theory

The Mapping Process

The Abstractional Concept Mapping Theory [Dierbach and Chester, 1997] is a computational model of analogical reasoning allowing for the identification of arbitrary comparisons of similarity (and analogy) between two descriptions given in first-order logic. Such comparisons are themselves represented by (abstracted) first-order logic descriptions and associated mappings from the abstraction to each of the two component descriptions; which aspects put into correspondence is dependent only on the notion of logical truth, and not on the particular vocabulary (ontology) used in each description or the syntactic form of individual statements. Such a mapping, as shown in Figure 1, is seen as a true semantic mapping.

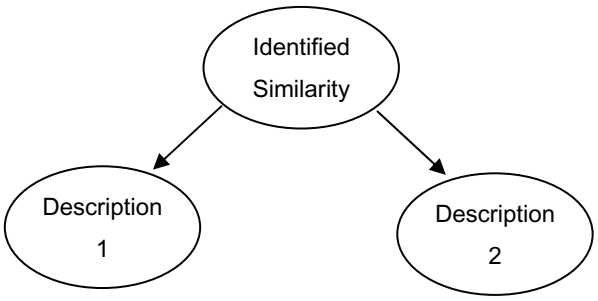

Figure 1 Similarity Mapping

The Abstractional Concept Mapping Theory provides a robust computational model of similarity matching. Thus, given two descriptions, comparisons between the two can be discovered and represented, based solely on the semantics of each description, and not on the ontologies used.

Example Mappings

We consider again the cases of semantically similar statements given above. First, we consider case 1. In order for these two statements to be put into correspondence by abstractional concept mapping, abstract predicates are indirectly mapped to each of the statements, and terms (indirectly) mapped to terms, as given below in figure 2:

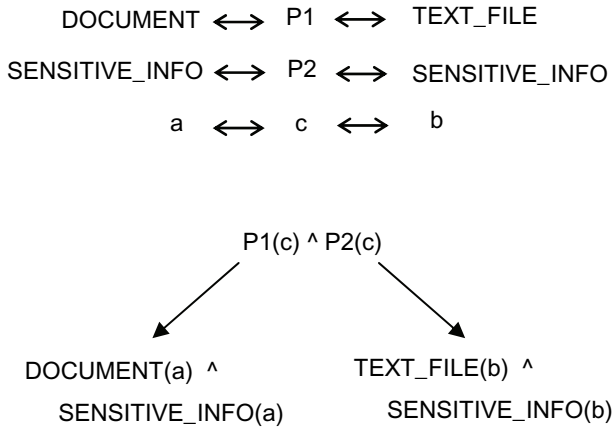

Figure 2 Mapping of Similar Statements by Use of Abstract Predicates

In this mapping, predicates P1 and P2 are referred to as "abstract predicates" given that they have no a priori interpretation. Put another way, they are reusable symbols whose only meaning is constrained by the (concrete) predicates that they are mapped to. By a "concrete predicate" is meant a predicate with a priori interpretation, as given by the domain knowledge. Such domain knowledge, therefore, is what gives meaning to concrete predicates such as predicate DOCUMENT above. For example, the domain knowledge may contain statements such as $\forall x[DOCUMENT(x) \Leftrightarrow TEXT\_FILE(x)]$, as mentioned above.

Thus, application of specific domain knowledge would allow the determination that the two predicates put into correspondence by abstract predicate P1 are in fact semantically equivalent. And given that abstract predicate P2 puts into correspondence predicates that are syntactically (and thus semantically) identical, the statements above are found semantically identical statements (on corresponding entities a and b). Finally, the terms themselves are mapped by a term mapping

putting abstract constant c into correspondence with entity a in the left statement, and entity b in the right statement.

We next consider case 2 above. In order for these two statements to be put into correspondence, the use of abstract predicates alone is not sufficient, given that the two statements do not have the same syntactic form, i.e., P ^ Q versus P ^ Q ^ R. Thus, a more flexible mapping is needed.

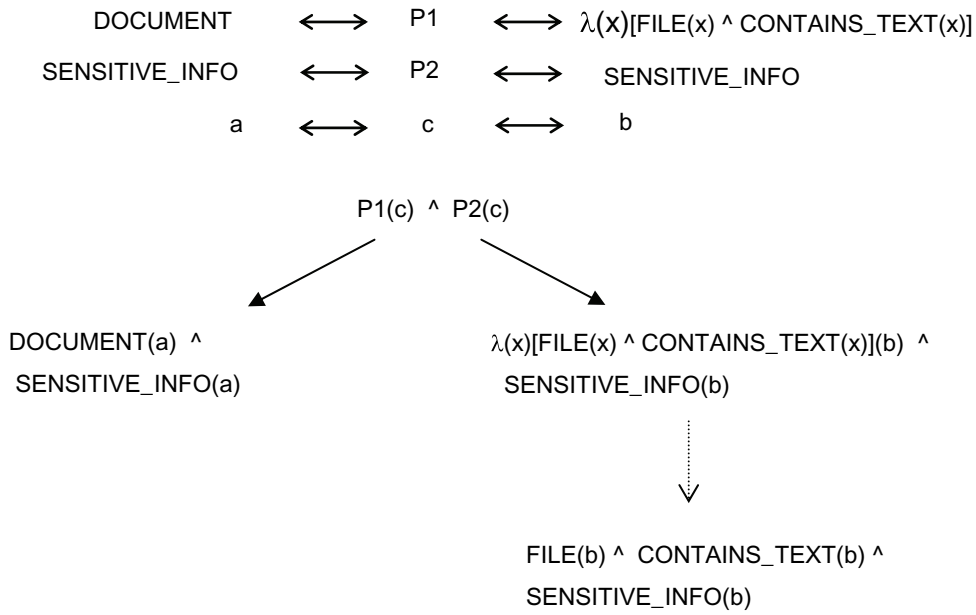

Figure 3 Mapping of Similar Statements of Different Structure

This is provided for in the Abstractional Concept Mapping Theory by the utilization of lambda expressions as "definitions," as shown in figure 3. In this mapping, abstract predicate P1 is mapped to predicate DOCUMENT in the left statement, but to lambda expression λ(x)[FILE(x) ^ CONTAINS_TEXT(x)] in the right mapping. This is in contrast to the previous example, in which abstract predicate P1 was mapped to predicates DOCUMENT and TEXT_FILE. However, in the example here, no single predicate of the right statement has corresponding meaning to predicate DOCUMENT in the left statement. Therefore, the meaning of abstract predicate P1 is mapped to a "definition" described by the lambda expression λ(x)[FILE(x) ^ CONTAINS_TEXT(x)], which when instantiated with the entity b, results in the statement FILE(b) ^ CONTAINS_TEXT(b). I

In order for statements DOCUMENT(a) ^ SENSITIVE_INFO(a) and FILE(b) ^ CONTAINS_TEXT(b) ^ SENSITIVE_INFO(b) to be determined equivalent, the corresponding statements ultimately put into correspondence by abstract predicate P1 and abstract predicate P2 need to be found equivalent. Given that P2 maps to SENSITIVE_INFO in both statements, the two subexpressions SENSITIVE_INFO(a) and SENSITIVE_INFO(b) are clearly equivalent (on corresponding entities a and b). Given that predicate P1, however, maps to DOCUMENT in one statement, and lambda expression λ(x)[FILE(x) ^ CONTAINS_TEXT(x)] in the other, there must be a way to find these two equivalent. To determine this, ∀x[DOCUMENT(x) ⇔ FILE(x) ^ CONTAINS_TEXT(x)] must be derivable from the associated domain knowledge. If derivable, then the two

statements are known to be equivalent. If not derivable, then the particular mapping of these statements is known not to identify an instance of similarity, and thus other potential mappings would be explored (i.e., searched).

Finally, we further show the flexibility of the Abstractional Concept Mapping Theory in the mapping of the statements from case 3 above. Given that the left statement in this example involves existential quantification,

$\exists x$ [DOCUMENT(x) ^ SENSITIVE_INFO(x)]

there is the problem of how to map the entities of the two statements. In order to accommodate such situations, Skolem functions are incorporated into the mapping. Skolem functions are a means of replacing a statement involving existential quantification, with a logically equivalent one involving only definite terms, as in the following example,

$\forall x$[PERSON(x) → $\exists y$ [PERSON(y) ^ FRIEND_OF(x,y)] (unskolemized)

$\forall x$[PERSON(x) → [PERSON(f(x)) ^ FRIEND_OF(x,f(x))] (skolemized)

The meaning these two statements is that "Every person has at least one friend." Here, the Skolem function term f(x) represents a particular person (friend) for any given person x. Thus, for different persons x, a different friend is represented by f(x). While these two statements differ in extensional meaning (since "there exists a y" is meant there exists at least one such y, and f(x) represents one and exactly one entity), they have equivalent intensional meaning. That is, one statement is true if and only if the other is also true. Thus, Skolem functions are appropriate to utilize here, giving the needed definite term required for any mapping.

We give, therefore, the mapping for our third case below.

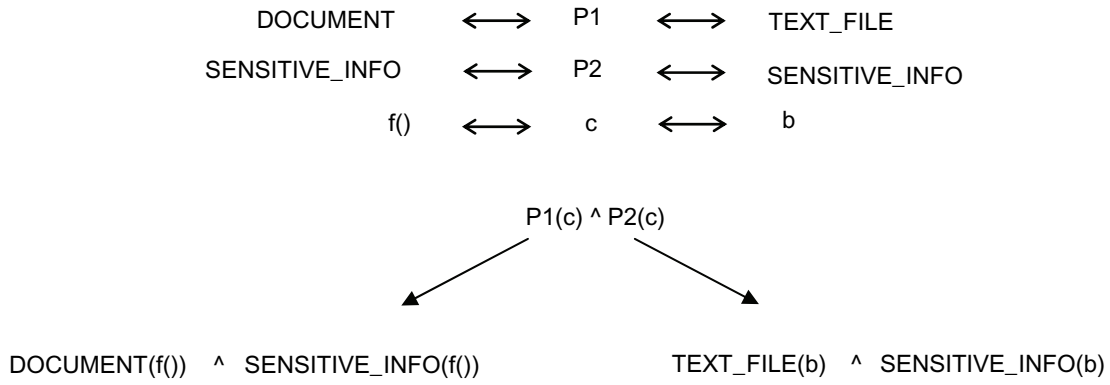

Figure 4 Mapping of Similar Statements with Existential Quantification

Here, Skolem function term f() is utilized. This Skolem function, in fact, is a function with no arguments, given that the existentially quantified variable x in this case is not within the scope of a universally quantified variable (as in the "Every person has a friend" example above). However, the use of Skolem functions is the same. It provides a definite term for abstract constant c in this example to map to, as shown above. Without the use of Skolem functions, therefore, statements involving the use of existential quantification could not be mapped, and therefore not able to be found equivalent.

The Meshing of Viewpoints

The identification of similar goals among autonomous agents by a collaborative agent, as discussed here, can result in overall increased capabilities of a given multi-agent system. For example, two agents, each operating in a "selfish" manner, have no need to understand each others goals. Thus, there would be no need for the two agents to share a common ontology, for example. However, if the two agents were to be coerced into collaboration by a third agent (i.e., a collaboration agent), new capabilities of the system may emerge.

Such a third agent would need to be able to perform robust similarity matching of the goals of agents, regardless of a given agent's ontology. This is where the flexible mapping provided by the Abstractional Concept Mapping may be utilized. In particular, the use of lambda expressions and Skolem functions in the mapping of statements (goals) provides the flexibility that can allow for the identification of similar goals, regardless of the ontology of each agents used.

Summary

We have presented a robust theory of similarity matching, and demonstrated its particular utilization in multi-agent systems. In particular, we propose the use of a collaborative agent in the form of a meta-agent, whose goal is the identification of similar goals among agents. Such collaborative agents can effect "coerced collaboration" among agents, as opposed to proactive collaboration emanating from agents themselves. Thus, whether a given agent's nature is collaborative or "selfish," a collaborative agent can potentially bring all agents of a given system into cooperation.

Three advantages of such a collaborative agent have been identified. First, the facilitation of cooperation is an ongoing process when a collaborative agent is utilized, given that it is the single goal of the agent. Second, more effective collaboration can result given the overview that the collaborative agent has of all collaboration in the system. Third, general strategies of collaboration in multi-agent systems can be more easily controlled, given that such strategies are localized in one agent (the collaboration agent)

References

[CLSW 2005] - Common Logic Standard Working Groups Documents. http://cl.tamu.edu/. 2005)

[Dierbach, C. and Chester, D. 1997] Charles Dierbach and Daniel Chester, "Abstractional Concept Mapping: A Foundational Model for Analogical Reasoning," *Computational Intelligence*, Vol. 13, no. 1 (Jan., 1997), pp. 32-86.

[Hayes, Carolyn C. 1999]. Agents in a Nutshell –A Very Brief Introduction. IEEE Transaction on Knowledge and Data Engineering, vol. 11, no. 1 (Jan/Feb 1999).

[Sycara, Katia 1998]. Muti-Agent Systems. AI Magazine, vol. 19, no. 2, pp. 79-92 (Summer 1998).

[Tharngarajah, John, Padgham, Lin, Harland, James 2001] Representation and Reasoning for Goals in BDI Agents.Twenty-Fifth Australiasian Computer Science Conference (ASCS), Melbourne, Australia (2001).