Meta-reasoning for Agents' Private Knowledge Detection

Jan Tožička[1], Jaroslav Bárta[2], and Michal Pěchouček[1]

[1] Gerstner Laboratory, Czech Technical University in Prague,
[2] Center of Applied Cybernetics, Czech Technical University in Prague,
Technicka 2, 166 27, Prague 6, Czech Republic
{tozicka, barta, pechouc}@labe.felk.cvut.cz

Abstract. Agent's meta-reasoning is a computational process that implements agent's capability to reason on a higher level about another agent or a community of agents. There is a potential for meta-reasoning in multi-agent systems. Meta-reasoning can be used for reconstructing agents' private knowledge, their mental states and for prediction of their future courses of action. Meta-agents should have the capability to reason about incomplete or imprecise information. Unlike the ordinary agents, the meta-agent may contemplate about the community of agents as a whole. This contribution presents application of the meta-reasoning process for the agent's private knowledge detection within the multi-agent system for planning of humanitarian relief operations.

1 Introduction

Multi-agent systems are collections of autonomous, heterogeneous agents with specialized functionalities. Distributed problem solving architectures provide important features, e.g. capability to find 'reasonably-good' solutions efficiently, robustness and a very high degree of fault-tolerance, reconfigurability capabilities, 'openness' of the community to integrate new agents or to replace the disappearing, etc.

We have demonstrated [1] that the concept of the multi-agent system is appropriate for planning humanitarian relief operations. The agents representing the humanitarian organizations control their actions with respect to their private restrictions. In this contribution, we focus on detection of these constrains by monitoring the community communication. We use a formal model of meta-agent presented in [2,3] for implementation of the meta-reasoning using different methods of artificial intelligence.

Section 1.1 introduces CPlanT – the target domain for meta-reasoning activities. In section 1.2, we define basic meta-reasoning terms. We describe used meta-reasoning architectures in section 2. In section 3, we describe the methods used in meta-reasoning implementation. In section 4, we summarize and briefly compare these methods.

M. Klusch et al. (Eds.): CIA 2003, LNAI 2782, pp. 316–323, 2003.

1.1 CPlanT Multi-agent System

Planning humanitarian relief operations [1] within a high number of hardly collaborating and vaguely linked non-governmental organizations is a challenging problem. Owing to the very special nature of this specific domain, where the agents may eventually agree to collaborate but are very often reluctant to share their knowledge and resources, we have combined classical negotiation mechanisms, teamwork theory [4] with the acquaintance models and social knowledge techniques[5].

Agents can create a *coalition*, a set of agents, which agreed to cooperate on a single, well-specified task.

We have defined agents' multi-level knowledge organization:

- **private knowledge** - accessible to no other agent,
- **semi-private knowledge** - accessible to other agents in a well-defined subset of the community and
- **public knowledge** - accessible to all agents in the community.

In the CPlanT system, every agent in the community decides in accordance to his private knowledge whether he will agree to participate in a coalition solving given task. The detection of this part of the agent's private information is the studied task of meta-reasoning. Its detection can help meta-agent to reduce necessary communication and to improve the quality of created coalitions. It is not necessary to find out the private knowledge in an exact form (e.g. as stored in the agent's memory), it is sufficient if we will be able to evaluate given task.

1.2 Shortly about Meta-Rrasoning and Meta-agents

We refer to **meta-reasoning** as an agent's capability to reason about the knowledge, mental states and reasoning processes of other members of the multi-agent community. We will refer to **object agents** which are subject of another agent's meta-reasoning process. Meta-reasoning can be carried out either by the object agent or by a specific agent, whose role is only to carry out the meta-reasoning related process. We will refer to **meta-agent** as to any agent with the meta-reasoning ability. Meta-reasoning is an important and inseparable part of reflective behavior of a multi-agent system [6]. A reflective system consists of object-level components and a reflective component.

According to the role/contribution of the object agents to the meta-reasoning process we distinguish between two different types of meta-reasoning. In **collaborative meta-reasoning**, the object agents are aware of being monitored, which is what they agree with and support. The purpose of collaborative meta-reasoning is very often an improvement of the object-agents' collective behavior. While in **non-collaborative meta-reasoning**, the object-agents do not want to be monitored and are not supporting the meta-reasoning process.

Meta-Agents. Many researchers have approached the problem of meta-agency from different points of view. Very often the agents, that reason or maintain knowledge about the community of object agents, are linked to the agents' interaction platform (e.g. *facilitators* [7]). However, meta-reasoning can be implemented also by loosely coupled agents such as *brokers* [8], *matchmakers* [9], or *mediators* [8]. If these agents are tightly connected to the implementation platform, they have been classified as *middle agents* [10].

Unlike the middle agents, the meta-agent is a loosely coupled agent that does not facilitate any functionality that is inevitable for operation of the object-agents. The meta-agent observes the behavior of the community and tries to draw some assumptions about the agents' behavior. This type of meta-agent can help other agents to make optimal decisions and therefore improve the performance of the multi-agent system as a whole. By introducing the concept of the meta-agent, one may get an overall view of the community's functionality.

2 Meta-agent Reasoning Architecture

The central point of the meta-agent's operation is an appropriate model of the community. This model has to be expressed in an appropriate language of adequate granularity. For any reasonable manipulation with the model we need to specify the relevant semantic properties of the object-level community it supposed to model. This is what we will refer to as the **background knowledge** – bk.

According to the model of social knowledge [5], the **agent's social neighborhood** $\mu(A)$ is a collection of agents that are subject of agent's A meta-reasoning processes. While $\mu^+(A)$ is a set of agents, which are monitored by the agent A, $\mu^-(A)$ is a set of all agents that monitor the agent A. Provided that A denotes any agent (including meta-agents), we can say that:

$$\forall A \in \mu^-(B) : B \in \mu^+(A). \tag{1}$$

Object-level community model. Let us define the model of the object-level community $\mathsf{model}(m)$ as a structure, that collects the facts that the meta-agent m knows about the group of agents $\theta \subseteq \mu^+(m)$. As we admit the meta-agent to be mistaken we do not require necessarily truthfulness of these facts, while we want the meta-agent to believe in their truthfulness: bel_m^θ.

Now, let us briefly mention how the model may be constructed and exploited. The meta-reasoning process in multi-agent system is built upon three mutually interconnected computational processes:

1. **monitoring** – process that makes sure that the meta-agent knows the most it can get from monitoring the community of object agents.
2. **reasoning** – this process manipulates the model of the community so that true facts (other than monitored) may be revealed.
3. **community revision** – a mechanism for influencing operation of the object agents' community.

In the following section, we will discuss the reasoning phase in more details. Description of the monitoring phase and the community revision phase can be found in [2].

2.1 Reasoning

The meta-reasoning agent's reasoning operation can be carried out in three different phases of the object agents' community life-cycle:

***init*-time:** initialization time when the meta-agent starts to reason about the system before he receives any event from the community,

***revision*-time:** the instance of the time when an event in the community happens and the community model is automatically revised or

***inspection*-time:** when the user (or other agent possibly) queries the model in order to find out about truthfulness of the goal hypotheses.

Balancing the amount of computational processes in the *revision*-time and the *inspection*-time is really crucial. The proper design depends on the required meta-reasoning functionality. While for visualization and intrusion detection the most of computation is required in the *revision*-time, for explanation, simulation and prediction an important part of computational processes will be carried out in the *inspection*-time.

Community Model Revision. Let us introduce the **community model revision** operator - $\uplus$, that is expected to happen in the *revision*-time exclusively. The community model revision represents the change of the model $\mathsf{model}_t(m)$ in the time t with respect to the new formula $\mathtt{event}_t$ that describes an event in the community:

$$\mathsf{model}_{t+1}(m) = \mathsf{model}_t(m) \uplus \mathtt{event}_t \tag{2}$$

There are different types of events that initiate the community-model-revision process in the *revision*-time. We talk primarily about initiating a contract-net-protocol, offer for collaboration, accepting or rejection of the collaboration proposal, informing about actual resources, etc.

Community Model Inspection. During the *inspection*-time, the computational process of **community model inspection** provides the user (or any other agent) the reply for a queried question – about the truthfulness of the formula `goal`. We introduce an operator $\looparrowright$ for model inspection, which replies as follows:

$$\mathsf{model}(m) \looparrowright \mathtt{goal} = \begin{cases} \mathit{yes} & \text{if } \mathsf{model}(m) \text{ supports the truthfulness of } \mathtt{goal}, \\ \mathit{no} & \text{if } \mathsf{model}(m) \text{ supports the falseness of } \mathtt{goal}, \\ \mathit{unsure} & \text{otherwise.} \end{cases} \tag{3}$$

If the goal formula contains variables, the reply can also contain their possible substitution.

3 Meta-reasoning Implementation

A meta-reasoning method consists of the structure $\mathsf{model}(m)$ describing community model, revision operator $\uplus$ and inspection operator $\looparrowright$. This abstract architecture has been used for agent's private knowledge detection in the CPlanT multi-agent system. We have implemented two meta-reasoning methods which represents two different possibilities of use of the theoretical background described above. First method uses explicit representation of true facts in the community and theorem proving algorithms as operators. The second one creates and manages hypothesis about every agent's decision rule, it uses machine learning algorithm to revise and inspect the model. Both methods uses the same monitoring and preprocessing to get events from the system.

3.1 Monitoring and Preprocessing

The meta-agent communicates with the object agents via TCP/IP connection by messages in ACL language in KIF format. The meta-agent subscribes the object agents for copies of their communication messages[1]. The meta-agent transforms results of the object agents' decision activity to the events, which are used by reasoning methods for further analysis.

Every agent shares with all other agents in his alliance his semi-private information - mainly for maintaining actual resources capacities. If an agent has not enough resources to satisfy task's demands, it is sure, that the agent will refuse the participation on this task. Because this rejection does not give us any information about agents private knowledge, so we can stop in dealing with this event - this functionality is added to the revision operator $\uplus$. On the other hand, during the model inspection ($\looparrowright$ operator) we can directly reply, that the agent will refuse if we know, that he has not enough resources for a given task.

3.2 Theorem Proving and Automated Reasoning

We have experimented with the explicit representation of the true facts in the meta-reasoning model. We have defined the model of the object-level community as a set of true facts about the respective agents. We collect the facts that the meta-agent m knows about the agents θ into a belief-set bel_m^θ.

The model, managed by the meta-agent m, is then a set as follows:

$$\mathsf{model}(m) = \bigcup_{\theta \subseteq \mu^+(m)} \mathsf{bel}_m^\theta \tag{4}$$

It contains all beliefs about all subset of agents in the community. For the implementation of $\uplus$ and $\looparrowright$ operators, we have chosen a theorem proving method.

[1] We are talking about collaborative meta-reasoning.

Theorem Proving. Theorem proving and automated reasoning covers an important part of the traditional symbolic artificial intelligence, where the essence of building artificially intelligent systems is rooted in manipulation with the symbolic representation of the environment - the object level multi-agent system, in our case. The truthful facts about the object agents are represented by means of logical formulae and syntactical manipulation with these formulas is given by the rules of logical deduction.

Probably the most popular calculus used in the implementation of reasoning programs is based on the resolution principle [11]. All the logical formulas, that describe a model $\mathsf{model}_t(m)$, are supposed to be encoded as a conjunction of clauses, where each is a disjunction of literals – CNF (conjunctive normal form). By resolving clauses the inference machine tries to find a contradiction in the model with negation of the goal hypothesis. Had there been a contradiction found, the goal hypothesis is proved.

Community Model Revision. When designing the $\uplus$ operator, one needs to take into account the background knowledge – bk (mentioned in section 2). We may distinguish between two marginal effects of the meta-reasoning operation – $\uplus^{\max}$ and $\uplus^{\min}$ as follows (φ is a formula):

$$\mathsf{model}_t(m) \uplus^{\max} \mathtt{event}_t = \{\varphi : \mathsf{bk} \cup \mathsf{model}_t(m) \cup \{\mathtt{event}_t\} \vdash \varphi\} \tag{5}$$

$$\mathsf{model}_t(m) \uplus^{\min} \mathtt{event}_t = \mathsf{model}_t(m) \cup \{\mathtt{event}_t\} \tag{6}$$

The $\uplus^{\max}$ operator revises the model so that it contains all possible true facts that logically follow from the background knowledge – bk, actual model – $\mathsf{model}_t(m)$ in union with the new observation – $\mathtt{event}_t$. The $\uplus^{\min}$ operator only appends the new formula to the model. In many cases, the $\uplus^{\max}$ operator is hard to achieve as the resulting model may be infinite - we introduce such a model as an abstract marginal concept. When designing the community-model-revision process, we seek such an operation $\uplus$ that

$$\mathsf{model}_t(m) \uplus^{\max} \mathtt{event}_t \supseteq \mathsf{model}_t(m) \uplus \mathtt{event}_t \supseteq \mathsf{model}_t(m) \uplus^{\min} \mathtt{event}_t. \tag{7}$$

The closer our operation gets to $\uplus^{\min}$ the faster is the model revision process and more complex should be the computational process in the *inspection*-time. The closer we are to $\uplus^{\max}$ the easier should be the query process while the revision process is getting really complex. The concept of model revision is closely related to the concept of weak and strong update in the knowledge engineering area [12].

We have experimented with different model revision operators, all based on the suggested novel heuristic strategy – referring to as *minimization strategy*. This strategy generates a new clause only if the length of the new clause is smaller than one of its parents clauses (for experiments see [3]).

Community Model Inspection. The minimal version of the community model inspection process corresponds directly to *instantiation* of the goal formula within the model with no further reasoning:[2]

$$\mathsf{model}(m) \not\rightarrow^{\min} \mathtt{goal} = \begin{cases} yes & \text{if } \mathtt{goal} \in \mathsf{model}(m), \\ no & \text{if } \neg\mathtt{goal} \in \mathsf{model}(m), \\ unsure & \text{otherwise.} \end{cases} \tag{8}$$

In order to use the minimal version of the model inspection, we need to use the maximal (or close to maximal) model revision operator $\uplus^{\max}$.

If the reasoning triggered by the event has not produced the queried formula, the inspection process will be a more complex operation than simply parsing the existing model. The meta-agent is expected to employ reasoning in order to find out whether the requested goal formula logically follows from the model:

$$\mathsf{model}(m) \not\rightarrow \mathtt{goal} = \begin{cases} yes & \text{if } \mathsf{model}(m) \vdash \mathtt{goal}, \\ no & \text{if } \mathsf{model}(m) \vdash \neg\mathtt{goal}, \\ unsure & \text{otherwise.} \end{cases} \tag{9}$$

We have experimented with different theorem proving strategies for implementing the community model inspection operator.

3.3 Machine Learning Algorithm

Let us now talk about the reasoning about singleton object agent only. Watching object agent's reaction for a given tasks, we can learn how he is deciding. For this task it is possible to use a machine learning algorithm. Community model is then a structure used by the learning algorithm and model revision and inspection operators are represented by a learning and evaluation functions of chosen machine learning algorithm.

To allow the meta-agent using such a community model to reason about several agents, it is necessary to create this model for every object agent. But it can be very difficult (or even impossible) to implement more complicated interactions between several object agents. For our experiments, we have chosen version space algorithm [13] and inductive logic programming system. Both methods are able to learn and reply the query in a short time.

4 Conclusion and Comparison

We have presented two possible implementations the meta-reasoning process, i.e. representation of the community model and model revision and inspection operators. The first one is the resolution algorithm based on theorem proving paradigm. It tries to directly create a model containing formulas logical following

[2] The goal formula can contain free variables that get bound during the instantiation.

from the known facts. Second presented approach tries to learn how the object agent is deciding, it can be used with machine learning algorithms.

Next phases of our project will focus on abduction and multiple meta-agents cooperation. In [3], you can find methods and strategies comparison and some results of our experiments.

Acknowledgement. The project work has been co-funded by European Office for Aerospace Research and Development (EORD) Air Force Research Laboratory (AFRL) - contract number: FA8655-02-M-4056, Office of Naval Research (ONR) - award number: N00014-03-1-0292 and by the Grant No. LN00B096 of the Ministry of Education, Youth and Sports of the Czech Republic.

References

1. Pěchouček, M., Mařík, V., Bárta, J.: A knowledge-based approach to coalition formation. IEEE Intelligent Systems **17** (2002) 17–25
2. Pěchouček, M., Štěpánková, M., Mařík, O., Bárta, J.: Abstract architecture for meta-reasoning in multi-agent systems. In Mařík, Muller, P., ed.: Multi-Agent Systems and Applications III. Number 2691 in LNAI. Springer-Verlag, Heidelberg (2003)
3. Bárta, J., Tožička, J., Pěchouček, M., Štěpánková, O.: Meta-reasoning in CPlanT multi-agent system. Technical report, Technical Report of the Gerstner Laboratory ©, Czech Techical University in Prague (2003)
4. Tambe, M.: Towards flexible teamwork. Journal of Artificial Intelligence Research **7** (1997) 83–124
5. Mařík, V., Pěchouček, M., Štěpánková, O.: Social knowledge in multi-agent systems. In Luck, M., et al., eds.: Multi-Agent Systems and Applications. Number 2086 in LNAI. Springer-Verlag, Heidelberg (2001)
6. Pěchouček, M., Norrie, D.: Knowledge structures for reflective multi-agent systems: On reasoning about other agents. Research Report 538, Calgary: University of Calgary (2000)
7. McGuire, J., Kuokka, D., Weber, J., Tenebaum, J., Gruber, T., Olsen, G.: Shade: Technology for knowledge-based collaborative engineering. Concurrent Engineering: Research and Applications **1(3)** (1993)
8. Shen, W., Norrie, D., Barthes, J.: Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing. Taylor and Francis, London (2001)
9. Decker, K., Sycara, K., Williamson, M.: Middle agents for internet. In: Proceedings of International Joint Conference on Artificial Intelligence 97, Japan (1997)
10. Sycara, K., Lu, J., Klusch, M., Widoff, S.: Dynamic service matchmaking among agents in open information environments. ACM SIGMOID Record 28 (1999) 211–246
11. Chang, C., Lee, C.R.: Symbolic Logic and Mechanical Theorem Proving. Academic Press New York and London (1973)
12. Pěchouček, M., Štěpánková, O., Mikšovský, P.: Maintenance of discovery knowledge. In Zitkow, R.J., ed.: Priciples of Data Mining and Knowledge Discovery. Springer-Verlag, Heidelberg (1999) 476–483
13. Mitchell, T.: Generalization as search. Artificial Intelligence (1982) 203–226