

BDI MULTIAGENT LEARNING BASED ON FIRST-ORDER INDUCTION OF LOGICAL DECISION TREES

ALEJANDRO GUERRA HERNÁNDEZ, AMAL EL-FALLAH SEGHTROUCHNI
AND HENRY SOLDANO

*Université Paris 13 , Laboratoire d'Informatique de Paris Nord, U.P.R.E.S.-A.
CNRS 7030, Institute Galilée, Avenue Jean-Baptiste Clément, Villetaneuse,
93430, France. Email: {agh,elfallah,soldano}@lipn.univ-paris13.fr*

This paper is about learning in the context of Multiagent Systems (MAS) composed by intentional agents, e.g. agents that behave based on their beliefs, desires, and intentions (BDI). We assume that MAS learning differs in subtle ways from the general problem of learning, as defined traditionally in Machine Learning (ML). We explain how BDI agents can deal with these differences and introduce the application of first-order induction of logical decision trees to learn in the BDI framework. We exemplify our approach learning the conditions in which plans can be executed by an agent. Key words: MAS learning, BDI systems, Logical Decision Trees.

1 Introduction

We are interested in learning in the context of Multiagent Systems (MAS) composed by intentional agents, e.g. BDI agents. In this paper, we deal with the issue of adding learning competences to a BDI architecture, which lead us to consider learning methods applied to systems which behavior is explained in terms of beliefs, desires, intentions (BDI propositional attitudes), and partial hierarchical plans, as proposed in practical rationality theories¹, and that can be characterized as autonomous, reactive, pro-active and social¹⁵.

Usually, MAS learning^{10,14} is characterized as the intersection of Machine Learning (ML) and Distributed Artificial Intelligence (DAI). Motivations for this are reciprocal: i) MAS community is interested in learning, because it seems to be central to different properties defining agents; and ii) an extended view of ML dealing with agency and MAS can improve the understanding of general principles underlying learning in natural and artificial systems.

A *learning agent*⁹ can be conceptually divided into four components: i) a learning element responsible for making improvements executing a learning process; ii) a performance element responsible for taking actions, e.g. the agent without learning competences; iii) a critic responsible for providing feedback; and iv) a problem generator responsible for suggesting actions that will lead to informative experiences.

Then, the design of the learning element, and consequently the choice of a

particular learning method, is affected by five major issues: i) which elements of the performance element are to be improved? ii) what representation is used for these components? iii) what feedback is available? iv) what prior information is available? v) is it a centralized or decentralized learning case?

In this paper we expose the way BDI agency can be used to conceive learning agents able to operate in MAS, using induction of logical decision trees. In order to do that, the paper is organized as follows: Section 2 recalls briefly BDI architectures, introducing an example used in the rest of the paper. Section 3 presents our approach to MAS learning, it considers the design of a BDI learning agent, the learning method used (first-order induction of logical decision trees), and examples. Section 5 focuses on discussion, related and future work.

2 BDI Agency

BDI theories of agency are well known. Different aspects of intentionality and practical reasoning have been studied formally using extensions of modal and temporal logics^{5,11,15}. The goal of the section is just to recall the way BDI architectures work to complement the discussion on learning.

Examples in this paper comes from a very simple scenario proposed originally by Charniak and McDermott² (see figure 1). This scenario is composed by a robot with two hands, situated in an environment where there are: i) a board; ii) a sander; iii) a paint sprayer; iv) a vise. Different goals can be proposed to the robot, for example, sand the board or even get self painted! which introduces the case of incompatible goals, since once painted, the robot stops being operational for a while. The robot has different options to achieve its goals, it can use both of its hands to sand the board, for example, or well, use the vise and one hand. Eventually, another robot will be introduced in the environment to deal with examples about different interactions.

In general, a BDI architecture contains four key data structures: beliefs, desires or goals, intentions, and a plan library.

Beliefs represent information about the world. Each belief is represented symbolically as a ground literal of first-order logic. Two activities of the agent update its beliefs: i) the perception of the environment; and ii) The execution of intentions. The scenario shown in Fig. 1 can be represented by the following beliefs of robot r1 as: *somewhere(sander)*, *somewhere(board)*, *somewhere(sprayer)*, *free-hand(left)*, *free-hand(right)*, *operational(r1)*.

Desires, or goals, correspond to the tasks allocated to the agent and are usually considered logically consistent. Two kinds of desires are considered: i) to achieve a desire expressed by a belief formula, i.e. *!sanded(board)*; and

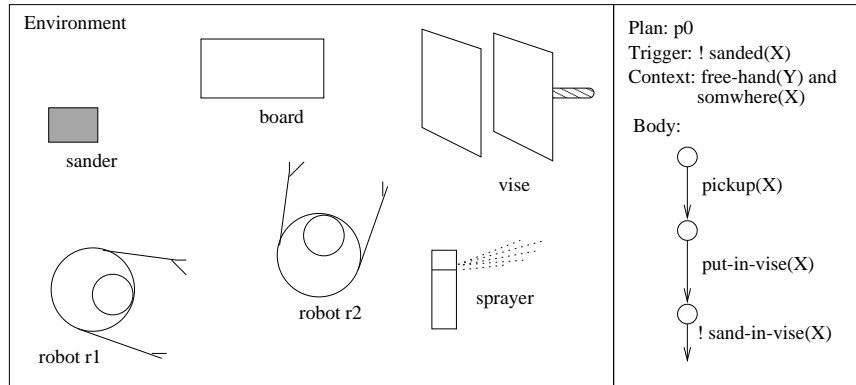


Figure 1. The scenario for examples and an typical plan.

ii) to test a situation expressed as a disjunction and/or conjunction of belief formulae.

Plans have several components. The *invocation* condition specifies, as a trigger event, the circumstances under which the plan should be considered. Four types of *trigger events* are possible: the acquisition of a new belief, the removal of a belief, the reception of a message, and the acquisition of a new (sub)goal. The *context* specifies, as a situation formula, the circumstances under which the execution of the plan may start. The *body* of a plan is represented as a tree where nodes are labeled with states and arcs with actions or subgoals, specifying a course of action. The *maintenance conditions* describe the circumstances that must remain to continue the execution of the plan. Finally, a set of internal actions is specified for the cases of *success* and *failure* of the plan. Figure 1 shows a simplified plan $p0$ to sand an object X . The last branch in the plan is a subgoal, because the robot will need to take the sander to do its work, which involves another plan.

An *intention* is implemented as a stack of plan instances. In response to an event, the agent must find a plan instance to deal with it. Two cases are possible: i) If the event considered is an external one, an empty stack is created and the associated plan is pushed on it, i.e. if the event is $!sanded(board)$, the plan $p0$ is considered, possibly among others, and the substitution $(board/X, left/Y)$ makes it executable. So, this substitution and $p0$ are used to form a new intention stack identified as $ip0$; ii) If the event is an internal one, it means it was produced by some already existing intention. The plan instance generated for the internal event is pushed in the intention stack that gener-

ated the event, i.e. When executing $ip\theta$, the last branch in the plan body is a subgoal, so the event $(!sand-in-vise(X),ip\theta)$ will be posted and will be processed as usual, but the intention formed, will be pushed on the top of $ip\theta$.

A *BDI interpreter*³ manipulates these structures, selecting appropriate plans based on beliefs and desires, structuring them as intentions and executing these ones.

3 BDI Learning Agents

We consider that learning in the MAS context differs in subtle ways from learning in other ML situations. There are two sources for these differences: i) the flexible autonomous behavior defining agency introduces some considerations which are not present in traditional software^{4,8}, i.e. autonomy and pro-activeness; ii) MAS environments are usually complex and dynamic.

This suggests that the same mechanisms controlling the behavior of the agent should be used to control learning processes, e.g. learning processes should be considered as actions of the agent. In particular: i) Agents have to be able to identify situations where learning is necessary (pro-activity); ii) Agents have to evaluate and prioritize their learning processes (action selection); iii) Eventually, agents should be able to cope with simultaneous learning processes, attending different learning goals found by the agent; and iv) The result of the learning processes should be incorporated in the agent architecture.

We have observed that applications and challenges of MAS for ML are indicative of a hierarchy of MAS levels of different complexity, that could be useful to adopt a bottom-up approach in MAS learning research towards a full distributed MAS learning. Levels are as follows: i) In the first level, agents learn from the observation of their environment without direct interaction with other agents (centralized learning); ii) In the second level, an elementary form of direct interaction is introduced: implicit exchange of messages among agents, requests included. Since it is a form of delegation this level introduces social learning in MAS; iii) In the third level, agents are enabled to learn from the observation of the behavior of other agents; and iv) All previous levels are forms of centralized learning. In the fourth level, decentralized learning is considered, i.e. agents with different beliefs participating in the same learning process.

Defining BDI learning agents involves: i) taking into account the above considerations; ii) considering the questions suggested while defining learning agents (section 1) under these considerations; and iii) Choosing a learning method.

3.1 *Defining BDI learning agents*

What components of performance can be improved? Plans are central in our approach: i) the context of each plan determines when they are executable affecting the order in which they are considered, so we want agents to learn the context of their plans that led to successful executions of them; ii) plans will be used as background knowledge; and iii) Success and Failure components of the plan help to build examples using internal actions. BDI learning agents will not learn their beliefs, but use them to build examples to learn. Events can be used in two ways: i) trigger events label the concept to be learn (event satisfied or not); and ii) the set of plans obtained after a given event can be used as background knowledge.

What representation is used for these components? The whole BDI interpreter is built on first-order logic representations. Belief formulae are defined as an atom or the its negation. Beliefs are grounded belief formulae. Situation formulae are a conjunction and/or disjunction of belief formulae. Two goals are considered, achieving a belief formula and testing a situation formula. Actions are seen as procedure calls, possibly including arguments. Plans, as seen, are complex structures. What is relevant here is that the invocation of a plan is represented as a trigger event, the context of a plan is represented as a situation formula, and the body of a plan is a tree which arcs are labelled with either goals or actions. Intentions are built as stacks of plan instances.

What feedback is available? The agent keeps traces of the execution of their intentions. Success in achieving an intention executes a set of internal actions to update the agent structure. These actions can include saving information about the context in which the intention was satisfied. Failures are processed in a similar way, but the event associated originally with the plan is reposted in the queue with the following information i) which plan is producing it, and ii) which plans has failed to satisfy it. This can be complemented with information about the beliefs of the agent when success or failure occurs, to build learning examples.

What prior information is available? Basically, we consider as prior information the bootstrap component of the BDI architecture, i.e. the plan library, and initial beliefs.

3.2 *First-Order Induction of Logical Decision Trees*

After the representations used in BDI architectures, we considered first-order learning methods. Since the context of plans was represented as a disjunction of conjunctions of belief formulae, we decided to use decision trees as target representation.

Decision tree learning is a widely used and very successful method for inductive inference. As introduced in the ID3 algorithm by Quinlan¹³, this method approximates discrete-value target functions. Learned functions are represented as trees and instances as a fixed set of attribute-value pairs. These trees represent, in general, a disjunction of conjunctions of constraints on the attribute values of the instances. Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself is a disjunction of these conjunctions. Decision trees are inferred by growing them the root downward, greedily selecting the next best attribute for each new decision branch added to the tree, in a divide-and-conquer strategy, differing from its rule-based competitors, i.e. CN2 and AQ, which use covering strategies.

Since clausal representation used in inductive logic programming (ILP) exhibits discrepancies with the structure underlying decision trees, Luc de Raedt⁷ introduced the concept of logical decision trees, that are binary decision trees (trees where tests have two possible outputs) constrained by: i) every test is a first-order conjunction of literals; and ii) a variable that is introduced in some node can not occur in its right subtree. This representation that corresponds to a clausal representation known as learning from interpretations paradigm⁶.

The learning from interpretations paradigm can be defined in the following way. Given: i) a set of classes C ; ii) a set of classified examples E ; iii) a background theory B . Find a hypothesis H , such that: $\forall e \in E, H \wedge e \wedge B \models c$ and $\forall e \in E, H \wedge e \wedge B \not\models c'$, where c is the class of the example e and $c' \in C \setminus \{c\}$. The background theory B is used in the following way. Rather than starting from complete interpretations of the target theory, examples are a kind of partial interpretations (sets of facts) that are completed by taking the minimal Herbrand model $M(B \cup I)$ of the background theory B and the partial interpretation I . This paradigm enables the agent to conceive examples as sets of beliefs considered when executing an intention.

Tilde⁷ is a learning from interpretations algorithm, operating on logical decision trees. It uses the same heuristics that C4.5, a predecessor of ID3 (gain ratio, post-pruning heuristics), but the computation of the tests is based on a classical refinement operator under Θ -subsumption.

3.3 Exemplifying the approach

In the scenario proposed in Fig. 1 we can consider the following predicates to specify the actions configuring the behavior of the agent: *pickup*(X), *put-down*(X), *put-in-vise*(X), *sand-in-vise*(X), *sand-in-hand*(X), *paint*(X), *self-paint*(X). To describe the environment where the agent is situated, the fol-

lowing predicates are used: *free-hand(X)* to indicate that the robot has the hand X free; *somewhere(X)* to indicate that the object X is somewhere there; *in-vise(X)* to indicate that the object X is in the vise; *in-hand(X)* to indicate that the object X is in a hand of the robot; *operational(X)* to indicate that the robot X is operational; *sanded(X)* and *painted(X)*.

Then we can consider the simple plan body of *p0* to sand an object X, executing sequentially: *pickup(X)*, *put-in-vise(X)*, and *sand-in-vise(X)*. This plan body is executed if (context of plan): *free-hand(Y)* and *somewhere(board)*. The specification of the plan can be incorporated in the background knowledge, as well as other general knowledge of the agent:

```
board-sanded :- plan(p0,board).
plan(p0,board) :- free-hand(Y), somewhere(board), sanded(board).
sanded(X) :- pickup(X), put-in-vise(X),
sand-in-vise(X).
```

The agent can build examples as models of the cases where the execution of *p0* lead to the board sanded and also for the cases where it does not. For this, the trigger event *!sanded(board)* produces two classes to consider *board-sanded* and *board-not-sanded*. The rest of the models are beliefs the agent had when the intention containing *p0* was executed.

```
begin(model(1)).      begin(model(2)).      begin(model(3)).
board-sanded.        board-sanded.          board-not-sanded.
free-hand(left).     free-hand(right).     free-hand(left).
operational(r1).     operational(r1).      somewhere(board).
somewhere(board).    somewhere(board).     plan(p0).
plan(p0).             plan(p0).              end(model(3)).
end(model(1)).        end(model(2)).

begin(model(4)).     begin(model(5)).      begin(model(6)).
board-not-sanded.   board-sanded.          board-sanded.
free-hand(left).    free-hand(left).       free-hand(left).
somewhere(board).   operational(r1).       operational(r1).
in-vise(sander).    somewhere(board).     somewhere(board).
plan(p0).           plan(p0).              plan(p0).
end(model(4)).      end(model(5)).         end(model(6)).
```

The following pruned tree for this learning setting is obtained by Tilde:

```
operational(A) ?
+--yes: board-sanded [4 / 4] [m1,m2,m5,m6]
+--no:  board-not-sanded [2 / 2] [m3,m4]
```

Fractions in the form $[i / j]$ indicate the number of examples in the class (i) and how many of them were well classified (j). Examples in the class are listed immediately (m1...m6). Induction time, for this example was of 0.03 seconds. The equivalent logic program for this logical decision tree is:

```
n1 :- operational(A).
class(board-not-sanded) :- not n1.
class(board-sanded) :- operational(A).
```

The definite clause $n1 :- operational(A)$ is introduced by the refinement operator of Tilde, because it will be useful to define the branch for the class *board-not-sanded*, which is defined in terms of *not n1*. The decision tree obtained suggests that the agent must add *operational(A)* in the preconditions of the plan $p\theta$.

Observe that examples expressed as models, can include beliefs about other agents, i.e. *operational(r2)* where $r2$ is a different robot, or also beliefs that other agents have sent to robot $r1$, without affecting the learning process. This is very important to scale up the approach to social learning, particularly to the fourth MAS level proposed.

4 Discussion

We have explained and exemplified how BDI agents can learn using First-Order Induction of Logical Decision Trees.

Different triggers have been considered in literature ⁴ to start learning processes associated with specific areas, i.e. expectation violations, and perceived need of improvement. All of them are possible in a BDI agent thanks to the way it uses its plans. We have not considered here expectation violations, but expectations can be represented in the states of plan bodies to verify these conditions. Unsuccessful executions of intentions suggest the need of improvement. The setting used in learning from interpretations are very important here, since using the BDI architecture we can: i) identify a task that is not well accomplished; ii) obtain examples of the execution of intentions (positives and negatives); and iii) obtain background knowledge, defining in this way the area where learning is necessary.

The example introduced suggests, it is possible for the agent to learn with few examples. We think that this is due to the way BDI architectures built windows of rationality¹ enabling the agent to focus on beliefs and plans relevant to particular events. More complicated experiments are necessary to know if Tilde continues to infer useful information with few models, specially in the case of the agent considering interactions with other agents.

We have decided to do our own implementation of a BDI interpreter. The reasons for this decision include i) we knew that different implementations for BDI architectures already existed, e.g. PRS , its re-implementation dMARS³, but we only had access to formal specifications of them, not the source code or low level information that help us modify or extend them accordingly to our needs. We are using Allegro CL 4.3 running on a Linux platform. This lisp interpreter enables us to execute several functions, i.e. agents, sharing the same lisp environment in a multiprocessing way. For the learning algorithm we are using Tilde version 5.5.1.

Some works in the same direction that ours include: Olivia and co-authors¹² present a Case-Based BDI framework applied to intelligent search on the Web, but the interpreter operates in a case-based cycle. Grecu and Brown⁴ have some similar position about the way learning must be incorporated in agent systems, but their agents are not intentional and they use propositional learning. Jacobs et al.⁸ presents the use of ILP systems for the validations of MAS.

Experimental results are promising. Even when the scenario proposed is very simple, extended with a second robot, it seems to be sufficient to experiment different interaction situations among agents. Immediate work to do is completing some details about the interaction of the interpreter and the learning processes, in order to use more realistic scenarios. Experiments done up to now have help us to better understand the interaction of the agents with their learning processes.

5 Acknowledgements

Discussion with David Kinny and Pablo Noriega has been very helpful. The first author is supported by Mexican scholarships from Conacyt, contract 70354; and Promep, contract UVER-53.

References

1. M Bratman, *Intention, Plans, and Practical Reasoning*,(Harvard University Press, Cambridge MA., USA, 1987).

2. E Charniak and D McDermott, *Introduction to Artificial Intelligence*, (Addison-Wesley, USA, 1985).
3. M D'Inverno, D Kinny, M Luck, and M Wooldridge in *Intelligent Agents IV*, Volume 1365 in Lecture Notes in Artificial Intelligence, pages 155-176, (Springer-Verlag, Berlin-Heidelberg, Germany, 1997).
4. D L Grecu and D C Brown in *Proceedings of the Third IFIP Working Group 5.2 Workshop on Knowledge Intensive CAD*, eds. T Tomiyama and M Mantyla, Guiding Agent Learning in Design, pages 237-250, Tokio, Japan, 1998.
5. A Rao and M P Georgeff, Decision Procedures for BDI Logics, *Journal of Logic and Computation* 8(3):293-344, 1998.
6. L De Raedt and Dzeroski, First-order jk-clausal theories are PAC-learnable, *Artificial Intelligence* (70):375-392, 1994.
7. L De Raedt and H Blockeel, Top-Down Induction of Logical Decision Trees, Technical Report, Department of Computer Science, Katholieke Universiteit Leuven, Belgium, 1997.
8. J Nico *et al* in *Inductive Logic Programming*, eds. N Lavrac and S. Dzeroski, Using ILP-Systems for Verification and Validation of Multi-Agent Systems, pages 145-154, (Springer Verlag, Berlin-Heidelberg, Germany, 1997).
9. S J Russell and P Norvig, *Artificial Intelligence, a modern approach*, (Prentice-Hall, New Jersey, USA, 1995).
10. S Sen and G Weiss, *Multiagent Systems, a modern approach to Distributed Artificial Intelligence*, (MIT Press, Cambridge, MA., USA, 1999).
11. M Singh *et al* in *Multiagent Systems, a modern approach to Distributed Artificial Intelligence*, ed. G Weiss, chapter Formal Methods in DAI: Logic-based Representation and Reasoning, (MIT Press, Cambridge MA., USA, 1999).
12. C Olivia *et al* in *AAAI Symposium on Intelligent Agents, Case-Based BDI Agents: an Effective Approach for Intelligent Search on the WWW*, Stanford University, USA, 1999.
13. J R Quinlan, Induction of Decision Trees, *Machine Learning* 1:81-106, 1986.
14. G Weiss and S Sen, *Adaptation and Learning in Multiagent Systems*, Number 1042 in Lecture Notes in Artificial Intelligence (Springer-Verlag, Berlin-Heidelberg, Germany, 1996).
15. M Wooldridge, *Reasoning about Rational Agents*, (MIT Press, Cambridge MA., USA, 2000).