Jason smiles: Incremental BDI MAS Learning

Alejandro Guerra-Hernández, Gustavo Ortiz-Hernández
Wulfrano Arturo Luna-Ramírez

Departamento de Inteligencia Artificial
Universidad Veracruzana
Facultad de Física e Inteligencia Artificial
Sebastián Camacho No. 5, Xalapa, Ver., México, 91000
aguerra@uv.mx, gusorh@gmail.com, wulfranoarturo@gmail.com

Abstract

This work deals with the problem of intentional learning in a Multi-Agent System (MAS). Smile (Sound Multi-agent Incremental LEarning), a collaborative learning protocol which shows interesting results in the distributed learning of well known complex boolean formulae, is adopted here by a MAS of BDI agents to update their practical reasons while keeping MAS-consistency. An incremental algorithm for First-Order Induction of Logical Decision Trees enables the BDI agents to adopt Smile, reducing the amount of communicated learning examples when compared to our previous non-incremental approaches to intentional learning. The protocol is formalized extending the operational semantics of AgentSpeak(L), and implemented in Jason, its well known java-based extended interpreter.

1: Introduction

The problem of learning in BDI Multi-Agent Systems [5, 6, 7, 8] has been studied both individually and socially. A protocol for social learning based on cooperative goal adoption [4] has been proposed in these works, where the agents decide intentionally to cooperate in order to update the context of the plans which execution has failed. Each agent in the MAS is able to remember, to some extent, the beliefs supporting the adoption of the plans as intentions, as well as the result of its execution, e.g., success or failure. In this way, each agent disposes of potential training examples to learn about the context for success and fail while executing plans. An agent should believe practical reasons which ensure the successful execution of his intentions. Then, the contexts may be seen as common knowledge that the agents must keep updated and consistent. While consistency is achieved by the collaborative goal adoption, updating is achieved by Inductive Learning of Logical Decision Trees. Such learning mechanism can be seen as the feedback feature characterizing systems under the Computational Sapience paradigm [9].

In our previous approaches to this problem, every time an agent faces a plan failure, he intends to learn a new plan's context individually. If this intention fails, e.g., he can not learn a new context, he asks for help to other agents sharing the failed plan. These agents also share the intention to learn a new context for the failed plan, so they sent back to the learner their relevant trainning examples. Once a new consistent context is learnt, the result is shared by all the agents participating in this social learning process.

In this work we approach social learning adopting the Smile (Sound Multi-agent Incremental Learning) protocol [3]. By incremental we mean that the agents can recover the local consistency of their hypotheses after each example they get. The MAS-consistency is achieved communicating the local consistent hypothesis of the learner agent to be validated by other agents sharing the same failed plan; or confronted to inconsistent trainning examples. When all the agents participating in the learning process validate the hypothesis, the plan context has been updated while achieving MAS-consistency. The effect of such protocol is that fewer examples are communicated while learning, but an incremental First-Order inductive algorithm is required [17, 19]. The proposed protocol is formalized extending the operational semantics of $AgentSpeak(L)$ [16] and it is implemented in the well known java-based extended interpreter Jason [1].

The organization of the paper is as follows: section 2 introduces briefly the Sound Multi-agent Incremental LEarning protocol; section 3 describes briefly Jason and its operational semantics; section 4 defines an extention of the operational semantics to support the Smile protocol, and its implementation is discussed in section 5. ILDT, the Incremental Inductive First-Order algorithm proposed enabling Smile is detailed in section 5 too; section 6 analyzes incremental learning in some of the MAS examples distributed with Jason. Finally, section 7 offers conclusions and discusses future work.

2: Smile

Smile ;-) [3], an acronym for Sound Multi-agent Incremental LEarning, is a protocol related with social learning in MAS. Each agent in the MAS is assumed to be able to learn from the information it perceives. An agent i is defined as $a_i = \langle B_i, K_i \rangle$, where B_i is the beliefs or knowledge state of agent i; and K_i is the information it perceives. It is assumed that all agents in the MAS shared some common knowledge, denoted by B_C so that $\forall i\ B_C \subseteq B_i$. Common knowledge introduces dependences among the agents: If an agent a_i updates its state B_i obtaining B'_i, but also modifying B_C into B'_C, then every agent a_j must update its state B_j to obtain B'_j so that $B'_C \subseteq B'_j$. An agent a_i updates it state B_i learning from new percepctions K_i to keep his beliefs updated. The agents are supposed to be able of verifying consistence between knowledge states and information, e.g., logical consequence. An agent a_i is *a-consistent* if and only if $Cons(B_i, K_i)$ is true, i.e., the information it perceives is a logical consequence of its beliefs ($B_i \models K_i$). An agent a_i is *mas-consistent* if and only if $Cons(S_i, K_i \cup K)$ is true, where $\forall j \neq i\ K = \bigcup K_j$. A MAS is *consistent* if every agent in the system is *mas-consistent*.

The Smile protocol depends on a consistency revision mechanism M with the following properties: it is *locally efficient* in the sense that an agent i can always update B_i; it is *additive* in the sense that $Cons(B_i, K_1) \wedge Cons(B_i, K_2) \implies Cons(B_i, K_1 \cup K_2)$; it is *coherent* in the sense that $\forall i, j, k\ Cons(B_i, K_i) \wedge Cons(B_j, K_j) \implies (Cons(B_i, K_i \cup k) \iff Cons(B_j, K_j \cup k))$. M is said *a-consistent* and *mas-consistent*, if it preserves such properties. M is said *strong mas-consistent* for agent a_i, if it preserves the *MAS-consistency* $\forall j \neq i\ a_j$.

MAS-consistency [3] can be ensured by the iterative application of a mechanism M, triggered by an agent a_i receiving inconsistent information k, i.e., $Cons(B_i, k) = False$. An interaction between a_i and a critic a_j, denoted by $I(a_i, a_j)$, is performed as follows: a_i sends a_j the updated common beliefs B'_C produced by the local application of M, i.e., a_i is *a-consistent*; a_j If B'_C preserves the *a-consistency* of B'_j, induced by the new common knowledge B'_C, a_j adopts it, sending a_i its acceptance; otherwise he sends his refusal with some information k' s.t. $Cons(B'_j, k') = False$. The protocol finishes when no k' is produced by any agent. If the response of the agents is minimal

and serial (k' is one inconsistent example and a_i sends B'_C sequentially to other agents), then the protocol minimizes the amount of information transmitted by the agents while learning.

Our previous approach to social intentional learning used Tilde [2] as a non-incremental update mechanism which lacked of local efficiency. As a result, communication among agents increased because the learner agent needed all the available examples in the MAS to learn. Here, we adopt an incremental learning algorithm which targets Logical Decision Trees (ILDT), as the update mechanism M supporting Smile (See section 5).

3: Jason

Jason [1] is an extended implementation of AgentSpeak(L) [16]. An agent in Jason is specified by a set of beliefs and plans. Beliefs are First-Order grounded formulae; and the plans have the form $triggerEvent : context \leftarrow body$. *Trigger events* include the addition or deletion of beliefs and goals. $Contexts$ are expressed in conjunctive form. The $body$ can include a sequence of actions, goals, and beliefs updates. Goals include testing beliefs ($?\theta$) and achieving ($!\theta$) literals.

A full operational semantics [10] of Jason, extended for Speech-Acts based communication [11], provides its proof theory in terms of a transition system. An agent circumstance is a tuple $C = \langle I, E, A, M, R, Ap, \iota, \rho, \epsilon \rangle$, where I is a set of intentions (stacks of instantiated plans); E is the set of events, each one being a pair (te, i) where te is a trigger event and i is the intention associated with the event; A is a set of actions; M is a mailbox; R is the set of relevant plans, i.e., plans that match the current event; and Ap is the set of applicable plans, i.e., the relevant plans which context is true; the components ι, ρ, and ϵ keep record of particular intentions, relevant, and applicable plans, respectively. An agent's beliefs and its circumstance form a configuration of the transition system. The transition relation $C, Beliefs \rightarrow C', Beliefs'$ defines the semantics of Jason.

4: Jason Smiles

The context of a plan expresses practical reasons to act in some way and not in another. Intentional learning intends to keep such reasons updated while keeping *MAS-consistency* in order to warrant coordination. Contexts are, at the intentional stance, the common knowledge defined in Smile (B_C). When an agent detects a failed execution of some intention, it desires updating his practical reasons. In what follows we describe the operational semantics for learning intentionally under the Smile protocol to satisfy such desire. We adopted the same notation used in [11, 12]. Given an $AgentSpeak(L)$ circumstance C, we write C_E to denote the component E of C. Similarly for all other components of C. We use $i, i', \ldots$ to denote intentions, and $i[p]$ denotes an intention that has the plan p on its top. As usual, we assume the existence of selection functions: S_E for events, S_{Ap} for applicable plans, S_I for intentions, and S_M for messages in the mailbox.

4.1 Preliminar definitions

First, we define some auxiliar functions. Syntactically a plan p has the form $label\ te : \phi \leftarrow h$. We use $Head(p)$ to denote $te : \phi$; and $Body(p)$ to denote h. $Label(p) = label$ where $label$ is a literal identifying the plan p. $TE(p) = te$ and $Ctxt(p) = \phi$. $Plan(label) = p$. Given a circumstance C, the set of relevant plans C_R is computed by $RelPlans(ps, te) = \{p\sigma \mid p \in ps \wedge \sigma = mgu(te, TrEv(p))\}$; and the set of applicable plans C_{Ap} is computed by $AppPlans(R, beliefs) =$

$\{p\theta \mid R \in \wedge\theta$ is s.t. $Beliefs \models Ctxt(p)\theta\}$. The function $Test(Beliefs, \phi) = \{\theta \mid Beliefs \models \phi\theta\}$, computes test goals of the form $?\theta$.

An agent believes some (or all) of its plans are supervised, e.g., they can be updated by intentional learning:

$$Supervised(p) = \begin{cases} true, & Beliefs \models supervised(l) \wedge Label(p) = l \\ false, & otherwise \end{cases}$$

An agent is able to remember, to some extent, the beliefs supporting the adoption of a plan as an intention, as well as the result of its execution, i.e., success or failure. In this way, each agent disposes of training examples about execution of their intentions. Given a plan p the trainning examples E are given by:

$$E(p) = \{sample(l, B, C) | Beliefs \models sample(l, B, C) \wedge Label(p) = l\}$$

where $C \in \{success, failure\}$ is the class of the trainning example, and B is the set of beliefs holded by the agent when it selected p to form an intention.

The function $ENeg(H, E)$ gives the subset of the trainning examples of E refuting the hypothesis H:

$$ENeg(H, E) = \{e | e \in E^+ \wedge\ H \not\models e \vee e \in E^- \wedge H \models e\}$$

where E^+ is the set of examples which class is $success$ and E^- are those examples with class $c = failure$.

As we mentioned in section 2, Smile requires a learning mechanism M locally efficient. $M(H, e)$ denotes the execution of such mechanisme, where H is an hypotesis, represented as a logical decision tree, and e is a trainning example, such that $M(H, e) = H'$ where H' updates the consistency with e.

4.2 The semantics of intentional learning in Jason

The rules to collect trainning examples about the supervised plans, are as follows. $\mathbf{RecPlan_1}$ adds, without generating any event, the trainning example of the form $sample(l, B, executing)$, where $executing$ denotes that the $success$ or $failure$ of the intention is not known yet.

$$\mathbf{RecPlan_1} \qquad \frac{Supervised(p)}{C, Beliefs \rightarrow C, Beliefs'} \quad C_\rho = p$$
$$where: \quad Beliefs' \models sample(Label(p), Beliefs, executing)$$

but once the result of the execution is known, the believed examples must be completed. The case for success is as follows:

$$\mathbf{RecPlan_{succ}} \quad \frac{Supervised(p)}{C, Beliefs \rightarrow C, Beliefs'} \quad C_\iota = i[Head(p) \leftarrow]$$
$$where: \quad Beliefs' \not\models sample(Label(p), B, executing)$$
$$Beliefs' \models sample(Label(p), B, success)$$

Failure situations include: sub-goal failure, i.e., a plan $[head \leftarrow !at; h]$ fails if there is not any relevant (1_A) or aplicable (1_B) plan for handling the event $+!at$; a test goal fails (2); and, from a declarative perspective, a plan designed to achieve g fails (3) if the plan execution ends and the agent does not believe g [13]:

$$\textbf{RecPlan}_{\textbf{fail 1}_\textbf{A}} \quad \frac{RelPlans(Plans, te) = \emptyset}{C, Beliefs \rightarrow C', Beliefs'} \quad C_\epsilon \neq \_, C_\iota = i[p], C_R = C_{Ap} = \emptyset$$
$$where: \quad Beliefs' \not\models sample(Label(p), Bels, executing)$$
$$Beliefs' \models sample(Label(p), Bels, failure)$$
$$C'_E = C_E \cup \{\langle +fail(Label(p)), T\rangle\}$$

$$\textbf{RecPlan}_{\textbf{fail 1}_\textbf{B}} \quad \frac{AppPlans(Beliefs, C_R) = \emptyset}{C, Beliefs \rightarrow C', Beliefs'} C_\epsilon \neq \_, C_\iota = i[p], C_R \neq \emptyset, C_{Ap} = \emptyset$$
$$where: \quad Beliefs' \not\models sample(Label(p), Bels, executing)$$
$$Beliefs' \models sample(Label(p), Bels, failure)$$
$$C'_E = C_E \cup \{\langle +fail(Label(p)), T\rangle\}$$

$$\textbf{RecPlan}_{\textbf{fail 2}} \quad \frac{Test(Beliefs, at) = \emptyset}{C, Beliefs \rightarrow C, Beliefs'} \quad C_\iota = i[Head(p) \leftarrow ?at, h]]$$
$$where: \quad Beliefs' \not\models sample(Label(p), B, executing)$$
$$Beliefs' \models sample(Label(p), B, failure)$$
$$C'_E = C_E \cup \{\langle +fail(Label(p)), T\rangle\}$$

$$\textbf{RecPlan}_{\textbf{fail 3}} \quad \frac{Beliefs \not\models g,\ TE(p) = +!g,\ Body(p) = \emptyset}{C, Beliefs \rightarrow C', Beliefs'} \quad C_\iota = i[p]$$
$$where: \quad Beliefs' \not\models sample(Label(p), B, executing)$$
$$Beliefs' \models sample(Label(p), B, failure)$$
$$C'_E = C_E \cup \{\langle +fail(Label(p)), T\rangle\}$$

When a failure event, denoted by $\langle +fail(X), T\rangle$ arises, the *a-consistency* must be restored using the M mechanism. After that, the examples used by the agent are forgotten since they are redoundant with the new learned context. Here a selection function $S_{ag}(MAS)$ is assumed to select an agent from MAS (the set of the agents in the system) for playing the role of critic. S_{Ag} is assumed serial according to the protocol.

$$\textbf{FailEv} \quad \frac{}{C, Beliefs \rightarrow C', Beliefs'} \quad C_\epsilon = \langle +fail(l), T\rangle$$
$$where: \quad C'_\epsilon = \_$$
$$Ctxt(Plan(l))' = M(Ctxt(Plan(l)), sample(Label(l), B, C))$$
$$Beliefs' \not\models sample(Label(l), B, C)$$
$$C'_A = C_A \cup \{send(S_{ag}(MAS), AskIf, \langle Ctxt(Plan(l))'\rangle)\}$$

if a critic agent does not find any evidence to refute the context for plan p, it adopts the update proposed by the learning agent id ($\mathbf{SmileRec_1}$). If this is not the case, he sends back one example refuting the proposed new context to the learning agent ($\mathbf{SmileRec_2}$) warranting the minimal caracter of the interactions in Smile.

$$\mathbf{SmileRec_1} \quad \frac{S_M(C_M) = \langle AskIf, id, H\rangle,\ ENeg(H, E(Plan(l))) = \emptyset}{C, Beliefs \rightarrow C', Beliefs}$$
$$where: \quad Ctxt(Plan(l))' = H$$

$$\mathbf{SmileRec_2} \quad \frac{S_M(C_M) = \langle AskIf, id, H\rangle,\ ENeg(H, E(Plan(l))) \neq \emptyset}{C, Beliefs \rightarrow C', Beliefs'}$$
$$where: \quad C'_A = C_A \cup \{send(id, Tell, e \in ENeg(H, E(Plan(l))))\}$$
$$Beliefs' = Beliefs - e$$

When the learning agent receives a *Tell* message, he adopts the new evidence e as a belief and posts a falilure event to start the updating process again:

$$\mathbf{SmileRec_3} \quad \frac{S_M(C_M) = \langle Tell, id, K\rangle}{C, Beliefs \rightarrow C, Beliefs'}$$
$$where: \quad Beliefs' \models K$$
$$C'_E = C_E \cup \{\langle +fail(Label(First(K))), T\rangle\}$$

In figure 1 three BDI agents learn socially under Smile. Transitions represent the application of the semantic rules defining the protocol. Ag_1 is the learner agent, Ag_2 and Ag_3 are critics.

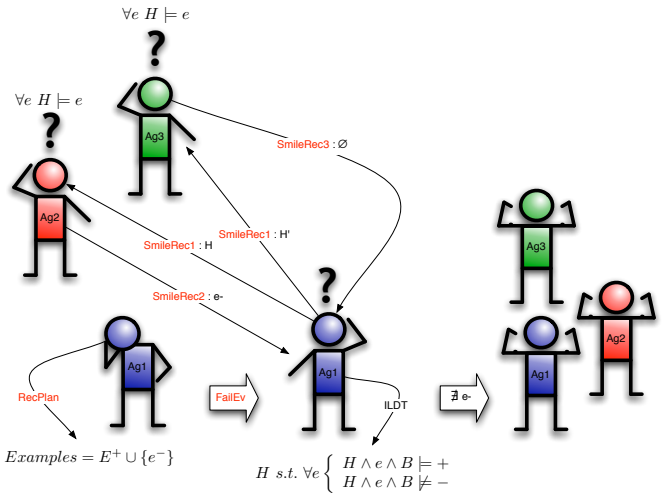

Figure 1. Three BDI agents apply Smile.

5: A locally efficient revision mechanism

Smile relies on a locally efficient revision mechanism M for mantaining the *a-consistency*. With this constrain in mind, we develop ILDT (Algorithm 1), an Incremental inductive algorithm for Logical Decision Trees [17, 19]. A Logical Decision Tree is a binary First-Order decision tree [17] formed by leafs and decision nodes, where each node is a conjunction of First-Order literals. Given a node, its associated query is formed by the conjunction of the literals from the root to the node. Leafs represent the class target and are bi-valuated ($+$ or $-$, true or false).

Algorithm 1 ildt(Example, RModes, Θ, Tree)

1: $Class \leftarrow class(Example)$;
2: **if** $Tree = null$ **then**
3: $\quad Tree \leftarrow inode(leaf(Class), null, null)$;
4: $\quad updateStat(Tree)$;
5: **else**
6: $\quad Class' \leftarrow classify(Example, Tree)$;
7: $\quad updateStat(Tree)$;
8: $\quad G \leftarrow information\_gain(leaf(Class'))$;
9: $\quad$ **if** $(Class \neq Class')\&(partition(G, \Theta) = True)$ **then**
10: $\quad\quad CandidateLits \leftarrow \rho(Example, RModes)$;
11: $\quad\quad BestCandidate \leftarrow best(CandidateLits)$;
12: $\quad\quad C^+ \leftarrow getStats(BestCandidate, +)$;
13: $\quad\quad C^- \leftarrow getStats(BestCandidate, -)$;
14: $\quad\quad C_{left} \leftarrow \max(C^+, C^-)$;
15: $\quad\quad C_{right} \leftarrow \min(C^+, C^-)$;
16: $\quad\quad L \leftarrow getClassCounter(BestCandidate, C_{left})$;
17: $\quad\quad R \leftarrow getClassCounter(BestCandidate, C_{right})$;
18: $\quad\quad NewNode \leftarrow inode(BestCandidate, leaf(L), leaf(R))$;
19: $\quad\quad NewTree \leftarrow poseNode(Tree, NewNode)$;
20: $\quad\quad Tree' \leftarrow swapNodes(NewTree)$;
21: $\quad\quad$ **return** $Tree'$;
22: $\quad$ **end if**
23: **end if**
24: **return** $Tree$;

Logical Decision Trees are isomporphic to propositional decision trees, so that a greedy top-down inductive algorithm, analog to ID3 or C4.5 [15], can be used to induce them, e.g., Tilde [17]. The main difference to consider is the way candidate nodes are computed. As other Inductive Logic Programming (ILP) systems, Tilde computes node candidates using a ρ operator under the generality frame of θ-subsumption [14]. The operator is defined as a set of $rmodes$ defining the language in which the hypothesis are going to be expressed. Details about this setting are reported in [5, 6, 7, 8].

We adopt an incremental strategy for inducing propositional decision trees, in particular ID4 [18], for the induction of Logical Decision Trees. The key idea is to mantain in each node the statistics required to compute the best node candidate using some information metric, e.g., information-gain. These statistics are updated every time a new example is processed. At the beginning the tree is empty, so that $ildt$ assigns a leaf containing the class of the $Example$ as the root of the tree and

```
...
[ag1] saying: PLAN stack FAILED for Goal !stack(b,c)
[ag1] saying: EXECUTE ILDT FOR stack
[report] ag1's CURRENT RELEVANT EXAMPLE FOR stack:
[report] sample(stack,[arm(empty),clear(b),clear(c),onTable(a),onTable(c),on(b,a)],
         failure).
...
[M] THE ILDT FOR stack IS:
[M] <> arm(empty)
[M]    (+) leaf(failure)
[M]    (-) leaf(success)
[ag1] ASKING Ag2 TO REFUTE not arm(empty) AS CONTEXT FOR stack
...
[ag2] REFUSE! EVIDENCE: sample(stack,[on(a,b),clear(a),holding(c),onTable(b)],failure).
...
[M] THE ILDT FOR stack IS:
[M] <> arm(empty)
[M]    (+) failure
[M]    <-> on(Z,Y)
[M]        (+) failure
[M]        (-) success
[ag1] ASKING Ag3 TO REFUTE not arm(empty) and not on(Z,Y) AS CONTEXT FOR stack
[ag3] OK
[ag1] ASKING Ag2 TO REFUTE not arm(empty) and not on(Z,Y) AS CONTEXT FOR stack
[ag2] OK
```

Table 1. Three agents learning in the blocks world.

updates statistics for this new tree (Algorithm 1, lines 2–4). Otherwise the $Example$ is sorted down the $Tree$ updating the statistics of the nodes it traverses. Counters for positive and negative values of class are updated in this way (lines 6–8) and information gain can be computed. If there is a relevant information gain (determined by Θ), the tree grows up. If this is not the case, $ildt$ has updated the statistics for every node in the tree.

A tree grows up due to the the following conditions (line 9): i) an example is misclassified, i.e. the leaf class is not the same of the example class; and ii) according to a splitting criteria (based in the information-gain of the actual leaf after updating its statistics) and a threshold θ. In order to grow up the tree a new node (test) must be computed. The test candidates are generated by ρ (line 10) using the $Rmodes$ language bias and the $Example$. The best candidate, the one that maximizes informationg gain, is selected; and a new node for it is created (line 18-19). Finally, a swapping procedure is used for reviewing the tree, its goal is to keep the best nodes near to the root of the tree, so, when a subtree has an greater information-gain with respect to its father it is swaped in the tree (line 20). Otherwise it keeps its place in the tree.

6: Results

Table 1 shows three agents learning about the failed plan $stack$ in the Blocks World. The ILDT algorithm is used to reestablish the *a-consistency*. First Ag_1 tries to learn the context of the plan $stack$ that has failed. The example includes the beliefs of ag_1 when the plan was selected to form an intention. Once it has learnt a new plan context (H), following Smile, he asks Ag_2 to validate H. Ag_2 refutes H with a new failure example. Ag_1 learns a new H' and asks Ag_3 to validate H'. The learning process finishes when both Ag_3 and Ag_2 do not refute the hypothesis proposed by Ag_1. Provided that the protocol is minimal and serial, the context for $stak$ has been successfuly updated, maintaining the *MAS-consistency*.

The new context is `not armEmpty & not on(Z,Y)`, an equivalent hypothesis to the usual context definition for $stack$: `holding(X) & clear(Y)`. Whenever `holding(X)` is $true$, `arm(empty)` is $false$; similary, if `on(Z,Y)` is $false$, then `clear(Y)` is $true$. It is possible to obtain the usual context defining some equivalences in the background knowledge while learning.

Fewer trainning examples are communicated under Smile, when compared with our previous non-incremental approach to intentional social learning [9] (this work also implements the MAS using Jason, but adopts Tilde as the update mechanism). A consistent hypothesis for $stack$ was induced from two examples only, in contrast with a trainning set of seven examples, required by the non-incremental approach.

7: Conclusions

We have successfully adapted a limited version of the Sound Multi-agent Incremental LEarning for the case of BDI agents. The approach is formalized extending the operational semantics of AgentSpeak(L), following the definitions introduced in Jason. AgentSpeak(L) enables us to review our approach to intentional learning. In particular, we have argue, somewhere else, that intentional learning is related to the problem of intention revision and now we can approach this question formally.

In previous work, we implemented our approach to intentional learning in Lisp. Now, we have implemented the protocol in Jason, a well known AgentSpeak(L) interpreter implemented in Java. The new implementation shows that the intentional learning approach is easy to implement in any BDI architecture.

Testing ILDT in the context of data mining showed that it is extremely sensitive to the order of the examples being processed. This is an undesirable feature inherited from the ID4-like strategy adopted. Future work considers the adoption of a ID5-like strategy for ILDT. The nature of ILDT also makes difficult to adopt the full Smile strategy where learning agents and oracles are adopted dynamically. Here, the first learning agent for a given plan is the only one having access to the statistics used to built the tree, so that if a new failure is detected it has to be sent to the original learning agent! Smile uses a symbolic learning algorithm and each agent keeps its examples to learn again if a new failure is detected. A similar approach can be explored in ILDT if the examples are not eliminated after updating statistics, enabling us to adopt the full Smile strategy.

8: Acknowledgments

The second and third authors are supported by Conacyt Scholarships 197819 and 197840, respectively. Amal El-Fallah Seghrouchni, Henry Soldano, and Gauvin Bourgne have gently shared with us their work and experience on Smile.

References

[1] R. H. Bordini and J. F. Hübner. BDI agent programming in agentspeak using jason. In F. Toni and P. Torroni, editors, *Proceedings of the Sixth International Workshop on Computational Logic in Multi-Agent Systems (CLIMA VI), London, UK, 27-29 June, 2005, Revised Selected and Invited Papers. Lecture Notes in Computer Science.* 3900:143–164, Berlin, 2005. Springer-Verlag.

[2] H. Blockeel et al. Executing query packs in ILP. In J. Cussens and A. Frish, editors, *Inductive Logic Programming, 10th International Conference, ILP2000, London, U.K.*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 60–77, Heidelberg, Germany, July 2000. Springer Verlag.

[3] G. Bourgne, A. El-Fallah-Seghrouchni, and H. Soldano. *SMILE: Sound Multi-agent Incremental LEarning ;-) AAMAS'07, May 14-17 2007, Honolulu, Hawai'i, USA*. ACM, 2007.

[4] C. Castelfranchi. Modelling social action for ai agents. *Artificial Intelligence*, 103(1998):157–182, 1998.

[5] A. Guerra-Hernández, A. El-Fallah-Seghrouchni, and H. Soldano. *Intelligent Agents Technology: Research and Development. Proceedings of the 2nd Asia-Pacific Conference on IAT, Maebashi, Japan, November, 2001*, chapter BDI Multiagent learning based on First-Order induction of Logical Decision Trees, pages 160–169. World Scientific, Singapour, 2001.

[6] A. Guerra-Hernández, A. El-Fallah-Seghrouchni, and H. Soldano. Distributed learning in Intentional BDI Multiagent Systems. In R. Baeza-Yates, M. J.L., and E. Chávez, editors, *Proceedings of the Fifth Mexican International Conference on Computer Science (ENC'04)*, pages 225–232, USA, 2004. IEEE Computer Society.

[7] A. Guerra-Hernández, A. El-Fallah-Seghrouchni, and H. Soldano. Learning in BDI multiagent systems. *Computational Logic in Multi-Agent Systems: 4th International Workshop, CLIMA IV, Fort Lauderdale, FL, USA, January 6–7, 2004, Revised and Selected Papers. Lecture Notes in Artificial Intelligence.* 3259:218–233, 2004.

[8] A. Guerra-Hernández, A. El-Fallah-Seghrouchni, and H. Soldano. On Learning Intentionally. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 9(25):9–18, Septiembre 2005.

[9] A. Guerra-Hernández and G. Ortiz-Hernández. *Towards BDI sapient agents: learning intentionally*. Toward Computational Sapience: Principles and Systems. R.V. Mayorga and L.I. Perlovsky (eds.). Springer-Verlag (In press).

[10] Á. F. Moreira and R. Bordini. An operational semantics for a BDI agent-oriented programming language. In *Proceedings of the Workshop on Logics for Agent-Based Systems (LABS-2002) held with KR2000, April 22–25, Toulouse, France*, pages 45–59, 2002.

[11] Á. F. Moreira and R. Bordini. An operational semantics for a BDI agent-oriented programming language. In A. Omicini, L. Sterling, and P. Torroni, editors, *Declarative Agent Languages and Technologies, First International Workshop, DALT 2003, Melbourne, Australia, July 15, 2003, Revised Selected and Invited Papers.*, volume 2990 of *Lecture Notes in Computer Science*, pages 135–154, Berlin-Heidelberg, Germany, 2003. Springer Verlag.

[12] Alvaro F. Moreira, Renata Vieira, and Rafael H. Bordini. Extending the operational semantics of a BDI agent-oriented programming language for introducing speech-act based communication. In Joao Leite, Andrea Omicini, Leon Sterling, and Paolo Torroni, editors, Declarative Agent Languages and Technologies, Proceedings of the First International Workshop (DALT-03), held with AAMAS-03, 15 July, 2003, Melbourne, Australia (Revised Selected and Invited Papers), number 2990 in Lecture Notes in Articial Intelligence, pages 135-154, Berlin, 2004. Springer-Verlag.

[13] Jomi Fred Hübner, Rafael H. Bordini and Michael Wooldridge. Programming Declarative Goals Using Plan Patterns. In proceedings DALT 2006, pages 123-140.

[14] G. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.

[15] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA., USA, 1993.

[16] A. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In R. van Hoe, editor, *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands, 1996.

[17] Blockeel Hendrik, Ph.D. Thesis *Top-down induction of first order logical decision trees*, Department of Computer Science, K.U.Leuven, December 1998, Leuven Belgium.

[18] Utgoff Paul E., Incremental Induction of Decision, Trees, *Machine Learning*, 4:161-186, 1989.

[19] Driessens Kurt. Relational Reinforcement Learning. PhD Thesis. In *Applied Science*, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 2004.