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 training 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 training 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 extension 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 its state $B_i$ learning from new perceptions $K_i$ 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 $Cons(B_i, K_1) \land Cons(B_i, K_2) \implies Cons(B_i, K_1 \cup K_2)$; it is coherent in the sense that $\forall i, j, k \ Cons(B_i, K_j) \land Cons(B_j, K_j) \implies (Cons(B_i, K_1 \cup k) \iff Cons(B_j, K_j \cup k))$. $M$ 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., $a_j$ 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 (ILD), 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 $\text{triggerEvent} : \text{context} \leftarrow \text{body}$. 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 $\iota$, $\rho$, and $\epsilon$ 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 $C, \text{Beliefs} \rightarrow C', \text{Beliefs}'$ 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 Preliminary definitions**

First, we define some auxiliary functions. Syntactically a plan $p$ has the form $\text{label} te : \phi \leftarrow h$. We use $\text{Head}(p)$ to denote $te : \phi$; and $\text{Body}(p)$ to denote $h$. $\text{Label}(p) = \text{label}$ where label is a literal identifying the plan $p$. $\text{TE}(p) = te$ and $\text{Ctxt}(p) = \phi$. $\text{Plan}(\text{label}) = p$. Given a circumstance $C$, the set of relevant plans $C_R$ is computed by $\text{RelPlans}(ps, te) = \{ p | p \in ps \land \sigma = mgu(te, TrEv(p)) \}$; and the set of applicable plans $C_{Ap}$ is computed by $\text{AppPlans}(R, beliefs) = \{ p | p \in Ap \land \sigma = mgu(te, TrEv(p)) \}$. The choice of $\text{RelPlans}$ is made because it does not consider the order of plans in the stack.
\{p \theta \mid R \land \theta \text{ 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:

\[
\text{Supervised}(p) = \begin{cases} 
    \text{true}, & \text{Beliefs} \models \text{supervised}(l) \land \text{Label}(p) = l \\
    \text{false}, & \text{otherwise} 
\end{cases}
\]

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 training examples \(E\) are given by:

\[
E(p) = \{\text{sample}(l, B, C) \mid \text{Beliefs} \models \text{sample}(l, B, C) \land \text{Label}(p) = l\}
\]

where \(C \in \{\text{success}, \text{failure}\}\) is the class of the training 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 training examples of \(E\) refuting the hypothesis \(H\):

\[
ENeg(H, E) = \{e \mid e \in E^+ \land H \not\models e \lor e \in E^- \land H \models e\}
\]

where \(E^+\) is the set of examples which class is \text{success} and \(E^-\) are those examples with class \(e = \text{failure}\).

As we mentioned in section 2, Smile requires a learning mechanism \(M\) locally efficient. \(M(H, e)\) denotes the execution of such mechanism, where \(H\) is an hypothesis, represented as a logical decision tree, and \(e\) is a training 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 training examples about the supervised plans, are as follows. \(\text{RecPlan}_1\) adds, without generating any event, the training example of the form \(\text{sample}(l, B, \text{executing})\), where \(\text{executing}\) denotes that the \text{success} or \text{failure} of the intention is not known yet.

\[
\begin{array}{c}
\text{RecPlan}_1 \\
\hline
\frac{Supervised(p)}{C, Beliefs \rightarrow C, Beliefs'} \quad C_{\rho} = p \\
\text{where : } \text{Beliefs'} \models \text{sample}(\text{Label}(p), \text{Beliefs}, \text{executing})
\end{array}
\]

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

\[
\begin{array}{c}
\text{RecPlan}_{\text{suc}} \\
\hline
\frac{Supervised(p)}{C, Beliefs \rightarrow C, Beliefs'} \quad C_i = i[\text{Head}(p) \leftarrow] \\
\text{where : } \text{Beliefs'} \not\models \text{sample}(\text{Label}(p), B, \text{executing}) \quad \text{Beliefs'} \models \text{sample}(\text{Label}(p), B, \text{success})
\end{array}
\]
Failure situations include: sub-goal failure, i.e., a plan $\text{[head } \leftarrow \text{!at;} h\text{]}$ fails if there is not any relevant ($1_A$) or applicable ($1_B$) plan for handling the event $+!\text{at}$; 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]:

$$
\text{RecPlan}_{\text{fail1}_A} \quad \text{RelPlans}(\text{Plans,te}) = \emptyset \quad C_c \neq \_C_i = i[p], C_R = C_{Ap} = \emptyset
$$

where:

- $\text{Beliefs} \neq \text{sample(Label(p), Bels, executing)}$
- $\text{Beliefs} \models \text{sample(Label(p), Bels, failure)}$
- $C'_E = C_E \cup \{(+ \text{fail}(\text{Label(p)}), T)\}$

$$
\text{RecPlan}_{\text{fail1}_B} \quad \text{AppPlans(}\text{Beliefs}, C_R = \emptyset \quad C_c \neq \_C_i = i[p], C_R \neq \emptyset, C_{Ap} = \emptyset
$$

where:

- $\text{Beliefs} \neq \text{sample(Label(p), Bels, executing)}$
- $\text{Beliefs} \models \text{sample(Label(p), Bels, failure)}$
- $C'_E = C_E \cup \{(+ \text{fail}(\text{Label(p)}), T)\}$

$$
\text{RecPlan}_{\text{fail2}} \quad \text{Test}(\text{Beliefs, at}) = \emptyset \quad C_c = i[\text{Head(p)} \leftarrow \text{?at;} h]]
$$

where:

- $\text{Beliefs} \neq \text{sample(Label(p), B, executing)}$
- $\text{Beliefs} \models \text{sample(Label(p), B, failure)}$
- $C'_E = C_E \cup \{(+ \text{fail}(\text{Label(p)}), T)\}$

$$
\text{RecPlan}_{\text{fail3}} \quad \text{Beliefs} \neq g, \text{TE}(p) = +!g, \text{Body}(p) = \emptyset \quad C_c = i[p]
$$

where:

- $\text{Beliefs} \neq \text{sample(Label(p), B, executing)}$
- $\text{Beliefs} \models \text{sample(Label(p), B, failure)}$
- $C'_E = C_E \cup \{(+ \text{fail}(\text{Label(p)}), T)\}$

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

$$
\text{FailEv} \quad C, \text{Beliefs} \rightarrow C', \text{Beliefs} \quad C_c = \langle + \text{fail}(l), T \rangle
$$

where:

- $C'_E = \_$
- $\text{Ctxt(Plan(l))}' = M(\text{Ctxt(Plan(l))}, \text{sample(Label(l), B, C)})$
- $\text{Beliefs} \neq \text{sample(Label(l), B, C)}$
- $C'_A = C_A \cup \{\text{send}(S_{ag}(\text{MAS}), \text{AskIf}, \langle \text{Ctxt(Plan(l))}' \rangle)\}$
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 $id$ ($\text{SmileRec}_1$). If this is not the case, he sends back one example refuting the proposed new context to the learning agent ($\text{SmileRec}_2$) warranting the minimal character of the interactions in Smile.

$$\text{SmileRec}_1: \quad \frac{S_M(C_M) = \langle \text{AskIf}, id, H \rangle, \ E\text{Neg}(H, E(\text{Plan}(l))) = \emptyset}{C, \text{Beliefs} \rightarrow C', \text{Beliefs}}$$

where:

$$\text{Ctxt}(\text{Plan}(l))' = H$$

$$\text{SmileRec}_2: \quad \frac{S_M(C_M) = \langle \text{AskIf}, id, H \rangle, \ E\text{Neg}(H, E(\text{Plan}(l))) \neq \emptyset}{C, \text{Beliefs} \rightarrow C', \text{Beliefs}'}$$

where:

$$C'_A = C_A \cup \{\text{send}(id, \text{Tell}, e \in E\text{Neg}(H, E(\text{Plan}(l))))\}$$

$$\text{Beliefs}' = \text{Beliefs} - e$$

When the learning agent receives a $\text{Tell}$ message, he adopts the new evidence $e$ as a belief and posts a failure event to start the updating process again:

$$\text{SmileRec}_3: \quad \frac{S_M(C_M) = \langle \text{Tell}, id, K \rangle}{C, \text{Beliefs} \rightarrow C, \text{Beliefs}'}$$

where:

$$\text{Beliefs}' \models K$$

$$C'_E = C_E \cup \{\langle +\text{fail}(\text{Label(First(K)))}, T \rangle\}$$

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.](image-url)
5: A locally efficient revision mechanism

Smile relies on a locally efficient revision mechanism $M$ for maintaining 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

```
1: Class ← class(Example);
2: if Tree = null then
3:   Tree ← inode(leaf(Class), null, null);
4:   updateStat(Tree);
5: else
6:   Class′ ← classify(Example, Tree);
7:   updateStat(Tree);
8:   G ← information_gain(leaf(Class′));
9:   if (Class ≠ Class′)&(partition(G, Θ) = True) then
10:      CandidateLits ← ρ(Example, RModes);
11:      BestCandidate ← best(CandidateLits);
12:      C+ ← getStats(BestCandidate, +);
13:      C− ← getStats(BestCandidate, −);
14:      Cleft ← max(C+, C−);
15:      Cright ← min(C+, C−);
16:      L ← getClassCounter(BestCandidate, Cleft);
17:      R ← getClassCounter(BestCandidate, Cright);
18:      NewNode ← inode(BestCandidate, leaf(L), leaf(R));
19:      NewTree ← poseNode(Tree, NewNode);
20:      Tree′ ← swapNodes(NewTree);
21: return Tree′;
22: end if
23: end if
24: return Tree;
```

Logical Decision Trees are isomorphic 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 maintain 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 $\Theta$), 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 $\theta$. In order to grow up the tree a new node (test) must be computed. The test candidates are generated by $\rho$ (line 10) using the Rmodes language bias and the Example. The best candidate, the one that maximizes information 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 stack has been successfully updated, maintaining the MAS-consistency.
The new context is not \texttt{armEmpty} \& not \texttt{on}(Z,Y), an equivalent hypothesis to the usual context definition for \texttt{stack} : \texttt{holding}(X) \& clear(Y). Whenever \texttt{holding}(X) is \texttt{true}, \texttt{arm}(\texttt{empty}) is \texttt{false}; similarly, if \texttt{on}(Z,Y) is \texttt{false}, then \texttt{clear}(Y) is \texttt{true}. It is possible to obtain the usual context defining some equivalences in the background knowledge while learning.

Fewer training 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 \texttt{stack} was induced from two examples only, in contrast with a training 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 build 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


