A Multi-layer Framework for Evolving and Learning Agents

Stefania Costantini
Dipartimento di Informatica
Università degli Studi di L'Aquila
via Vetoio Loc. Coppito
I-67010 L'Aquila, Italy
E-mail: stefcost@di.univaq.it

Pierangelo Dell'Acqua
Dept. of Science and Technology - ITN,
Linköping University
Norrköping, Sweden
E-mail: piede@itn.liu.se

Luís Moniz Pereira
Departamento de Informática
Centro de Inteligência Artificial (CENTRIA)
Universidade Nova de Lisboa
2829-516 Caparica, Portugal
E-mail: lmp@di.fct.unl.pt

Abstract

We illustrate a general agent model which includes a base level BA and a meta-level MA. The MA performs various forms of meta-reasoning including meta-control, which has the role of making meta-level decisions effective on the BA. As, in our view, meta-reasoning and meta-control are often concerned with time, we introduce the possibility of expressing temporal meta-rules. A very important meta-level activity in evolving agents is learning: we propose a general vision for interacting agents, where agents learn their patterns of behavior not only by observing and generalizing their observations, but also by "imitating" other agents, after being told by them. The process of learning by imitation is based on meta-reasoning about various aspects, from self-monitoring to knowledge evaluation. We propose an operational model for knowledge exchange assuming an agent society which is based on concepts of reputation and trust.

Introduction

In this work, we adopt a multi-layered underlying agent model where there is a base (or object) level, that we call BA for "Base Agent", and (at least) two meta-layers. Thus, referring to Figure 1 (which has been taken from the meta-reasoning manifesto (Cox and Raja 2007)), while the BA monitors the "ground level", the MA (Meta-Agent) and the IEA (Information Exchange Agents) compose in our model the "Meta-level". We discuss at some length this agent model and its semantics in the next sections. In defining the agent model, we do not commit to specific agent languages or models: our sole (soft) commitment in fact is to the adoption of computational logic. Many existing agent models and systems can be adopted as "building blocks", and even commitment to computational logic is not really strict.

The MA performs meta-reasoning of various kinds and supervises the BA activities. The MA includes a meta-control component that on the one hand coordinates the BA activities, and on the other hand makes the MA decisions effective by acting upon the BA. The actions that the MA will be able to undertake will include modifications to the BA in terms of adding/removing knowledge (modules) in the attempt at correcting inadequacies and generating a more appropriate behavior. The IEA will be put into action whenever the need and opportunity of social interactions and knowledge exchange has to be evaluated.

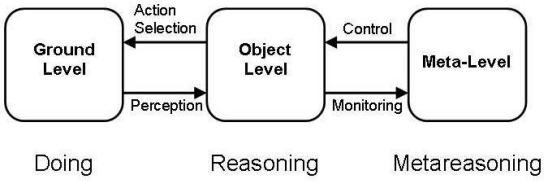

Figure 1: Multi-layered agent model

One may notice that supervising the BA, and thus checking whether the agent performs its tasks in a correct and timely manner, implies several aspects related to time. In this paper, we propose an agent-oriented temporal logic that allows "temporal" meta-rules to be expressed. These meta-rules express constraints whose violation may imply suitable actions to be undertaken. I.e., whenever all rules are complied with the overall agent is supposed to work well. Whenever some rule is violated, the meta-level is supposed to operate so as to restore correct functioning.

In our view, a very important meta-level activity in evolving agents that have to cope with a complex "world" is learning. The process of learning is a form of meta-reasoning that involves various aspects: self-monitoring, for identifying problems and needs; proper learning; evaluation of learned knowledge either according to concepts such as reputation and trust, or according to the usefulness that provisionally learned knowledge demonstrates in practice.

We propose a general vision of interacting agents: in this vision, agents learn their patterns of behavior not only by observing and generalizing their observations, but also by "imitating" other agents, after being told by them. As concerns learning abilities, we assume that "our" agents are able[1] (i) to elicit (e.g., by inductive learning) the behavioral patterns that other agents are adopting, and (ii) to learn rules and plans from other agents by imitation (or "being told"). Indeed, this is a fairly practical and economical way of increasing abilities, widely used by human beings, as widely studied in evolutionary biology (Richardson and Boyd 2005).

[1]We make the simplifying assumption that agents speak the same language, and thus we overlook the problem of ontologies.

We believe in fact that some principles emerging from these studies can equally apply to societies of agents.

The envisaged agents will try to either modify or reinforce the rules/plans/patterns they hold, based on appropriate evaluation performed by the internal meta-level component. The evaluation might also convince the agent to modify its own behavior by means of advanced evolution capabilities.

Avoiding the costs of learning is an important benefit of imitation, but nevertheless learning involves many issues and some potential risks. The involved issues are at least the following: (a) how to ask for what an agent needs; (b) how to evaluate the actual usefulness of the new knowledge. We propose a practical operational model for Information Exchange based on meta-level evaluation and meta-control.

The plan of the paper is the following. First, we comment about our view of meta-reasoning, compared to relevant related work. Then, we introduce our agent model and the new temporal logic. Next, we discuss learning, and argue that our agents will often attempt to learn new knowledge in the form of A-ILTL rules. Finally, we outline an operational model for information exchange and then conclude.

Background and Related Work

In (Raja and Lesser 2007) and (Alexander et al. 2007) and in the references therein (the reader is referred to (Cox 2005) for a review of the main approaches to meta-reasoning), meta-level control is defined as the ability of complex agents operating in open environments to sequence domain and deliberative actions to optimize expected performance. Deliberative control actions may include for instance scheduling domain-level actions and coordinating with other agents to complete tasks requiring joint effort. The meta-control component is triggered based on dynamic occurrence of predefined conditions. Meta-control generates an abstract meta-level problem based on the agent's current problem solving context and available deliberative actions and chooses the appropriate deliberative action which is executed, possibly resulting in further domain-level actions. In their view, meta-level control supports, e.g., decisions on when to accept, delay, or reject a new task; when it is appropriate to negotiate with another agent; whether to renegotiate when a negotiation task fails; how much effort to put into scheduling when reasoning about a new task; and whether to reschedule when actual execution performance deviates from expected performance. Each agent has its own meta-level control component.

In our approach, each agent has its own meta-level component including meta-control. The meta-level is, as illustrated in the next section, a standard component of the agent architecture. However, the meta-level is not just triggered upon specific conditions. Rather, it performs a continuous supervision of object-level activities (in practice, this will result either in real parallelism or in an interleaving of a number of object-level and meta-level steps).

Meta-reasoning can be related to several aspects of the agent operation like, e.g., learning, negotiation, goal evaluation and setting etc. Meta-reasoning involves also checking whether object-level activities are performed timely and correctly (i.e., similarly to previously mentioned work poor performance is considered to be an anomaly). In case a malfunctioning is detected, appropriate deliberations are taken that may include stopping/starting object-level activities and replacing pieces of object-level code. Meta-control coordinates object-level activities by triggering appropriate activities at the appropriate stage. Also, meta-control puts meta-reasoning deliberations into effect, by affecting the base (object) level.

More generally, as widely discussed in (Barklund et al. 2000) and in the references therein, object-level reasoning manipulates elements of the domain of interest in which the agent operates: in logic, this means that constants denote elements of the domain, variables range over elements of the domain, and predicates manipulate these constants and variables. Meta-level reasoning manipulates object-level expressions, including formulas or entire theories: in logic, this means that variables (i.e., meta-variables) range over object-level predicates and formulas, and meta-level predicates and formulas are defined and operate over these representations. A link must exist to correlate the two levels, in one direction for producing meta-level representation and in the other directions to transpose meta-level results to the object-level and put it in operation.

Agent Model

We propose a general agent model that goes in the direction of flexibility and adaptability. This is obtained by providing high flexibility in how an agent is built and may evolve, and by equipping an agent with forms of understanding and awareness that we "situate" at different control layers. Beyond the basic (object) layer we in fact introduce a meta-layer, where non-trivial supervising and tuning tasks (including agent reconfiguration) can be performed.

Agent Model: Knowledge Base

In order to meet the vision outlined in the Introduction, we consider an agent as composed of two layers:

- A *base layer or object layer* BA in charge of interacting with the agent's environment. We assume in this paper that BA is a logic program, but we do not commit to a particular semantics for it (for a survey of semantics for logic programs, see e.g. (Apt and Bol 1994)). We assume however a semantics possibly ascribing multiple models to BA, in order to deal with "uncertainty". One such a semantics might be the stable model semantics (Gelfond and Lifschitz 1988).
- A *meta-layer* MA in charge of performing meta-level activities and meta-control.

Important components of the BA and/or the MA include:

- A *belief*, component, for both the BA and the MA, possibly encompassing several modules for reasoning, planning, reactivity and proactivity, goal identification, etc.
- A set of *desires*, i.e., goals that have been adopted or goals that are under consideration, and *intentions*, i.e., plans in execution and plans under analysis.
- A set of *constraints*, that will in general include temporal constraints; in the MA, these constraints will have the

aim to either induce or verify that the BA performs in the established time, in the correct order and in the appropriate way its activities. Constraints can state what should happen and when or what should not happen.

- In the BA, a set of mechanisms for interacting with the environment.
- A set of mechanisms for *managing beliefs*, including: a learning mechanism and a belief revision mechanism.
- A *control component* for combining, exploiting and supervising the above components, based on *control information* aimed at improving the control mechanism effectiveness.

Agent Model: Operational Behavior

Below we sketch the operational behavior of this agent model, which is further described in (Costantini et al. 2007).

Each agent, once created, will in general pass through a sequence of stages, since it will be affected by the interaction with the environment, that will lead it to respond, to set and pursue goals, to either record or prune items of information, etc. This process, that we can call the agent *life*, is understood here in terms of successive transformations of the initial agent into new agents, that are its descendants where the program has changed by modifying the beliefs, desires, intentions, and by learning and belief revision steps; each transformation is determined by the step that has been done. Formally, an agent starts from a program that defines it, according to the given agent model.

For the sake of generality, we do not go any deeper into the feature of the agent model that we refer to simply as $\mathcal{M}$.

Definition An agent program $\mathcal{P}_{\mathcal{M}}$ is a tuple $\langle BA, MA, \mathcal{C}, \mathcal{CI} \rangle$ of software components where BA and MA are logic programs, $\mathcal{C}$ is the control component and $\mathcal{CI}$ is some related (optional) control information.

The control information $\mathcal{CI}$ is given as input to $\mathcal{C}$ together with BA and MA. While however BA and MA are programs written in a logic agent-oriented language, $\mathcal{CI}$ contains a set of *directives* that can affect in a relevant way the run-time behavior of an agent. Typically, directives will state at least which are the priorities among different events/actions the agent has to cope with and at which frequency they will be taken into consideration. Therefore, by modifying the control information while not touching the code, one can obtain a "lazy" agent rather that an "eager" one or affect the "interest" that the agent will show with respect to different matters.

We can take the agent program as the *initial state* of the agent, where nothing has happened yet.

Definition The *initial agent* A_0 is an agent program $\mathcal{P}_{0\mathcal{M}}$ (or simply $\mathcal{P}_0$ when $\mathcal{M}$ is clear from the context), i.e., $\langle BA_0, MA_0, \mathcal{C}_0, \mathcal{CI}_0 \rangle$.

The operational behavior of the agent will result from the control component and the control information, which rely on an underlying control mechanism that implements the operational counterpart of the agent model.

Definition The *underlying control mechanism* $\mathcal{U}^{\mathcal{M}}$ (or $\mathcal{U}$ in short), able to put in operation the various components of an agent model $\mathcal{M}$, is a transformation function operating in terms of a set of distinguishable *steps*, starting from A_0 and transforming it step by step into $A_1, A_2, \ldots$, given $\mathcal{C}_i$ and $\mathcal{CI}_i$ as defined in $A_0, A_1, A_2, \ldots$ respectively.

Definition Let $\mathcal{P}$ be an agent program. Then, $\forall i \geq 0$, $A_i \rightarrow^{\mathcal{U}(\mathcal{C}_i, \mathcal{CI}_i)} A_{i+1}$.

Operationally, two different solutions are possible. In the first one $\mathcal{U}$ provides different parallel threads for the BA and MA: therefore, MA continuously monitors BA and can possibly make interventions in case of problems. In the second one (where no parallel threads are possible) the control is interleaved between BA and MA where in turn a series of steps is performed at the BA level and a sequence of steps is performed at the MA level. Control will *shift up* from the BA to the MA by an act that is sometimes called *upward reflection* either periodically, in order to perform constraint verification at the meta-level, or upon some conditions that may occur. Control will *shift down* by *downward reflection* from the MA to the BA on completion of the MA activities (a general theory of reflection in logic programming languages has been developed in (Barklund et al. 2000)). How frequently and upon which conditions there will be a shift is control information that can be encoded in $\mathcal{CI}_i$. For a working examples of this kind of behavior, one can consider the *internal events* mechanism of DALI (Costantini and Tocchio 2006).

Notice that the A_js do not deterministically follow from A_0, as there is the unforeseen interaction with the external environment and the agent internal choices are not in general deterministic. A full "evolutionary" declarative semantics that is not specific to a language/approach but is rather designed to encompass a variety of computational-logic based approaches, thus accounting for the agent model proposed here, is described in (Costantini and Tocchio 2006).

A relevant function of the MA is that of evaluating resources used by the BA. Such an evaluation is needed by the MPA layer when it has to make a decision upon which resources to use next for the BA layer. In what follows, we assume to enhance the MA (Meta-Agent) by introducing a meta-meta-level that we call the IEA (Information Exchange Agents), whose role is discussed later.

The MA interventions on the BA may encompass modifications to the BA by replacing some of its rules/components by others. To describe and semantically account for the dynamic changes that the MA performs on BA we rely upon EVOLP, an extension of logic programming (Alferes et al. 2002) that allows to model the dynamics of knowledge bases expressed by programs, as well as specifications that dynamically change.[2] EVOLP augments a given logic programming language by adding the new distinguished atom $assert(R)$, where R is a rule. The intended meaning is that whenever $assert(R)$ is a consequence of the given program, then the rule R is added to the program itself. Symmetrically, a rule can be removed by asserting its negation.

[2] An implementation of EVOLP is available from http://centria.fct.unl.pt/~jja/updates.

Though EVOLP is originally intended for addition/deletion of rules, it may as well be used for addition/deletion of modules (sets of rules). In fact, the result of adding/deleting a whole module consists in the agent program (with its semantics) obtained after having added/deleted all the rules composing the module itself.

Temporal Logic Rules and Meta-rules

The supervision that the MA performs on the BA will in general encompass temporal aspects. As discussed above, MA will check whether some constraints on BA activity are respected. The checks that we consider here are not supposed to be performed in advance in the model-checking style, as done by a number of approaches in the literature that incorporate intervals into temporal logic. Instead, here we consider run-time verification of properties: violations, if present, are treated by means of some kind of *repair*, and by possibly modifying BA.

As it is not possible to foresee in advance all possible states that our agents can reach by interacting with both the user and the environment, we do not adopt a temporal logic based on a branching time, i.e., based on separately considering all paths that the agent may undertake. Rather, we intend to check that some properties are verified anyway, no matter the chosen path. So we adopt and extend LTL, a "Linear Time Logic", that implicitly quantifies universally upon all possible paths. LTL is a propositional logic with additional operators for time, where formulas are constructed from a set of *atomic propositions*. For LTL syntax and semantics the reader may refer to (Ben-Ari, Manna, and Pnueli 1983; Emerson 1990; Lichtenstein, Pnueli, and Zuch 1985)

Extension: A-ILTL

In the context of a resource-bounded agent, it is not possible to verify properties on the entire (infinite) sequence of states. Often this is not even necessary, since one needs properties to hold within a certain time interval. Also, it is useful to be able to state explicitly that a certain undesired property never holds or never holds within a certain time interval. Therefore, we have proposed an extension of LTL that we call A-ILTL, for "Agent Interval LTL", fully described in (Costantini et al. 2008).

We introduce the possibility of accessing the current state (or time): the proposition $\tau(s_i)$ is true if s_i is the current state. Let φ be a proposition. The A-ILTL operators are the LTL operators plus the following ones. X_m, i.e. φ should be true at state s_m. F_m stands for bounded eventually, $F_m\varphi$ means that φ eventually has to hold somewhere on the path from the current state to s_m. $G_{m,n}$ stands for always in a given interval, i.e., $G_{m,n}\varphi$ means that φ should become true at most at state s_m and then hold at least until state s_n. $G_{\langle m,n\rangle}$ means that φ should become true just in s_m and then hold until state s_n, and not in s_{n+1}, where nothing is said for the remaining states. N stands for "never", i.e. $N\varphi$ means that φ should not become true in any future state. $N_{m,n}$ stands for "bounded never", i.e. $N_{m,n}\varphi$ means that φ should not be true in any state between s_m and s_n, included.

A-ILTL Op^k	**OP(m,n;k)**
$\tau(t)$	$NOW(t)$
X^k	$NEXT(1;k)$
${X_j}^k$	$NEXT(j;k)$
F^k	$FINALLY(1;k)$
F_m^k	$FINALLY(m;k)$
G^k	$ALWAYS(1;k)$
${G_{m,n}}^k$	$ALWAYS(m,n;k)$
${G_{\langle m,n\rangle}}^k$	$ALWAYS\_2(m,n;k)$
N^k	$NEVER(1;k)$
${N_{m,n}}^k$	$NEVER(m,n;k)$

Figure 2: Representation in L of A-ILTL operators

In practice, run-time verification of A-ILTL properties may not occur at every state (of the given interval). Rather, properties will be verified with a certain frequency, that can even be different for different properties.

A-ILTL Rules

In our setting, A-ILTL rules are defined upon a logic-programming-like set of formulas where all variables are implicitly universally quantified. In this way, we are able to directly adopt the semantics proposed in (Costantini et al. 2008) for the propositional underlying language. In this setting however, the negation operator not is interpreted (as usual) as negation-as-failure.

Before defining A-ILTL rules, we need to represent A-ILTL operators within the language. Here we follow the naming convention illustrated in Figure 2. When not needed from the context, we omit the arguments of the operator and simply write OP (instead of $OP(m,n;k)$).

Definition Given a set S of literals (i.e., atoms and negated atoms), we write $conj(S)$ to indicate the set of all the conjunctions that can be formed by using literals in S. Let m, n and k be natural numbers. Define the set Q as follows:

- $S \subset Q$
- if $\varphi \in conj(Q)$, then $OP(m,n;k)\,\varphi \in Q$.

An A-ILTL rule is any rule of the form $OP(m,n;k)\,\varphi$, for any $\varphi \in conj(Q)$.

In many monitoring situations, one has to check that what *normally* should happen actually occur. The occurrence of an event is said to be "normal" when it occurs sufficiently often, if not always. We define a new operator called $USUALLY$ in terms of $ALWAYS$ that is checked at a certain frequency, say f, that reinforces "normality".

Definition Given a sentence φ and a natural number f, we let $USUALLY\varphi$ = $ALWAYS(1;f)\varphi$.

Notice that frequencies can possibly be specified separately, e.g., as control information included in $\mathcal{CI}$. For simplicity, both the interval and the frequency, indicated in the definition as $(m,n;k)$, can be omitted if not relevant to understand the context. That is, one can write $ALWAYS\varphi$ if φ has to be checked on all the future states.

To show the potential of A-ILTL rules we define below a check for an agent that, once decided to obtain the goal g, is blindly committed to actually obtain g within a given deadline d. After the deadline, a resource-bounded agent can possibly drop the commitment (or keep it, but only if possible). The fact $goal(g)$ means that g is a goal that has been selected to be executed. $achieved(g)$ means that the plan for reaching the goal g has been successfully completed. In contrast, $dropped(g)$ means that the agent has given up any attempt to achieve g. The following A-ILTL rule checks that an agent respects the blind commitment to its goals.

$$NEVER\,(goal(G), deadline(g, T), NOW(T1), \\ T1 \leq T, not\ achieved(g), dropped(G))$$

In order to fulfill the semantic specification, A-ILTL rules must be ground when they are evaluated, i.e. no variables must occur. For instance in the above example the evaluation will be related to each ground instance obtained by suitably instantiating the variables G, T and $T1$.

A-ILTL Rules with Time-Related Variables The syntax of A-ILTL rules defines time instants as constants. We introduce a further extension where time instants can possibly be variables which are instantiated by what we call an *evaluation context*.

Definition Let $OP(m, n; k)\varphi$ be an A-ILTL rule. The corresponding *contextual A-ILTL rule* has the form $OP(M, N; K)\varphi :: \chi$ where:

- M, N and K can be either variables or constants;
- χ is called the *evaluation context* of the rule, and consists of a quantified-free conjunctions of literals;
- each of the M, N and K which is a variable *must* occur in an atom (non-negated literal) in χ.

A contextual A-ILTL rule will be evaluated whenever ground. The context χ can possibly instantiate not only the time instants, but also other variables occurring in φ. More precisely, all its ground instances are subject to evaluation. In the example below, the contextual A-ILTL rule states that the time-out for completion of a goal is established according to its priority.

$$FINALLY(T; F)\ G :: \\ goal(G), priority(G, P), timeout(P, T), \\ frequency(P, F)$$

A-ILTL Rules with Repairs During the monitoring process, each A-ILTL rule is attempted at a certain frequency and with certain priorities (possibly customizable by means of directives specified in $\mathcal{CI}$). If the current state of affairs satisfies every A-ILTL rule, then no action is required. Otherwise, some kind of repair action has to be undertaken with respect to the violated A-ILTL rule. To this aim, we extend the definition of contextual A-ILTL rules to specify a corresponding repair action.

Definition An A-ILTL rule with a repair is a rule the form: $OP(M, N; K)\varphi :: \chi \div \psi$, where:

- $OP(M, N; K)\varphi :: \chi$ is a contextual A-ILTL rule;
- ψ is called the *repair action* of the rule, and it consists of an atom ψ.

Anytime, the monitoring condition $OP(M, N; K)$ of an A-ILTL rule is violated, the repair action ψ is attempted. The repair action is specified via an atom that is executed as an ordinary goal.

Consider again the previous example which monitors the achievement of goals, but extended to specify that, in case of violation, the present level of commitment of the agent to its objectives has to be increased. This can be specified as:

$$NEVER\,(not\ achieved(G), dropped(G))\ :: \\ (goal(G), deadline(G, T), NOW(T1), T1 \leq T) \div \\ inc\_comt(T1)$$

$$incr\_comt(T) \leftarrow level(commitment, L), \\ increase\_level(L, L1), \\ assert(neg(commitment\_mod(L)), \\ assert(commitment\_mod(L1)), \\ assert(inc\_comt\_at(T))$$

Suppose that at a certain time t the monitoring condition $NEVER\,(not\ achieved(g), dropped(g))$ is violated for some goal g. Upon detection of the violation, the system will attempt the repair action consisting in executing the goal $?{-}inc\_comt(t)$. In turn, its execution will allow the system to perform the specified run-time re-arrangement of the program that attempts to cope with the unwanted situation.

Notice that the above-presented A-ILTL rules performs meta-reasoning rather than object-level adaptation. In fact, it is defined over *any* goal rather than over specific goals as one would expect at the object level, and involves the concept of *level of commitment* which is a meta-level decision. In fact, also in the approach of (Raja and Lesser 2007) and (Alexander et al. 2007) this decision would be subject to meta-control.

Semantically, the execution of the repair action will determine the update of the current agent program $\mathcal{P}_i$, returning a new agent program $\mathcal{P}_{i+1}$.

On Learning and Evolution

We assume that our agent do not act in isolation: rather, they are part of a society of agents. This society in its simplest version can be a set of sibling agents. More generally, it can be a structured society of agents sharing common knowledge and possibly common objectives. We assume that the agents belonging to this society are benevolent and willing to cooperate.

An agent that performs activities of monitoring/training a user must perform at least three kinds of different learning activities:

- **Initialization stage**: in order to start its monitoring/training activities, the agent must receive either by a sibling agent or by the society a set of basic facts and rules that define:
  - the *role* that the agent will impersonate in the society
  - the basic behavior of the agent

  This is clearly a form of *Learning by Being Told*.

- **Subsequent stages, Observation**: the agent will be able to observe the environment along time and in different situations. The agent will collect the observations and will try to *classify* them with the aim of eliciting general rules or at least being able to expect with a reasonable confidence what is likely to happen in the future.
- **Subsequent stages, Interaction**: whenever the agent will have to cope with situations for which it has no sufficient knowledge/expertise, the agent will try to obtain the necessary knowledge either from other agents or from the society. The agent will however in general evaluate the actual usefulness of the acquired knowledge.

We may notice that, in general, at the initialization stage agents will acquire from the society knowledge that they take for granted, while at the interaction stages agents will be told knowledge that may be uncertain, in that it may have in turn learned by the others. The initialization will provide general meta-rules to be included in the MA. The following sample A-ILTL rules for a Personal Assistant agent state that a user should eventually perform necessary actions within the associated time-threshold, and should never perform forbidden actions.

$FINALLY(T)\ A$ ::
$action(A), mandatory(user, A), timeout(A, T)$

$NEVER\ A$ ::
$action(A), forbidden(user, A)$

Vice versa, each agent will give its contribution to the society. For instance, the rule above might be communicated to the society and might (after suitable evaluation by the society itself) be integrated into the society's common knowledge and then communicated to other agents. An agent may contribute to the society's "common belief set" under several respects:

- Provide the others with its own knowledge when required.
- In case of a structured society, insert into a repository whatever it has been able to learn.
- Provide feedback about the usefulness/effectiveness in its own context of the knowledge it has been told by others.
- Participate in possible "collective evaluations" of learned knowledge.

In Observation and Interaction stages, a basic premise that is quite usual in any form of assumption-based reasoning is that the agent distinguishes between *assumable* knowledge that it can try to learn, and basic knowledge that is taken for granted. Based on a record of past observations the agent should be able to produce:

- classifications, such as decision trees aimed at predicting what the other agents will do in some situation described by a set of parameters;
- A-ILTL rules, describing what $ALWAYS$, $USUALLY$ or $NEVER$ should either occur or be done in a certain situation.

The set of A-ILTL rules that an agent is able to learn from can be very important for the society, in that they can form knowledge that they will acquire "by being told". The agent will be later on verify the adequacy of learned rules and will be prompt to revise/retract them in front of new evidence. In fact, since in the real world the most common situation is one where there is incomplete and changeable information, any system making a serious attempt at dealing with real situations must cope with such complexities. The principle to be used here is the *Unknown World Assumption* (UWA) where everything is unknown or undefined until we have some solid evidence of its truthfulness or falseness. This principle differs from the more usual *Closed World Assumption* (CWA) where everything is assumed false until there is solid evidence of its truthfulness. The UWA stance is more skeptical, cautious, and even more realistic than the CWA.

Hopefully, after some iterations along this building/refinement cycle the knowledge built is "good enough" in the sense that the predictions it makes are accurate "enough" concerning the environment observations resulting from experiences. At this point, the theory can be used both to provide explanations to observations as well as to produce new predictions.

The Role of the Society Finding possible alternative pieces of information is one problem; finding which one(s) is(are) the "best" is another issue. In the next section we assume "best" means a minimal well-rated set of hypotheses and we describe the method we use to find such best. Another interpretation of "best" could be "most probable" and in this case the theory inside the agents must contain the adequate probabilistic information.

Ex contradictione quodlibet. This well-known Latin saying means "Anything follows from contradiction". But contradictory, oppositional ideas and arguments can be combined together in different ways to produce new ideas. Since "anything follows from contradiction" one of the things that might follow from it is a solution to a problem to which several alternative positions contribute.

One well known method for solving complex problems widely used by creative teams is that of 'brainstorming'. In a nutshell, every agent participating in the 'brainstorm' contributes by adding one of his/hers idea to the common idea-pool shared by all the agents. All the ideas, sometimes clashing and oppositional among each other, are then mixed, crossed and mutated. The solution to the problem arises from the pool after a few iterations of this evolutionary process.

The evolution of alternative ideas and arguments in order to find a collaborative solution to a group problem is the underlying inspiration of this work.

In fact, we have introduced a meta-meta-level IEA which is present in every agent which participates in the society. This higher level is responsible for information exchange. As discussed in the next section, the IEA can operate by exploiting strategies and techniques involving social evaluation and consensus, credibility measures and overall preferences.

Modeling Imitation Learning

In designing learning agents, particular attention should be dedicated to strategies involving reputation and trust for the evaluation of learned knowledge. The social acceptance of rules can be partly based on existing techniques and algorithms. However, we believe that an extension is necessary because, where learning is concerned, techniques that just measure reputation/trust on the basis of agents' feedback are not sufficient: some kind of economical and efficient evaluation of both the degree of compatibility of the new knowledge with an agent's previous knowledge base and of the performance of the acquired rules with respect to the expected objectives is also required.

In order to obtain an enhanced framework which allows for advanced imitation learning, we have introduced a *meta-meta-layer* IEA (for "Information Exchange Assistant") in charge of exchanging information with other agents. In our framework, the agent's information exchange is controlled by the IEA. The IEA may to this aim also exploit the resource evaluation performed by the MA.

The IEA is activated in at least three situations: (1) when an agent A asks another agent B information about how to solve a certain problem, (2) when A asks B information on how to improve its performance, and (3) when A proactively informs/recommends B a piece of information that A considers relevant for B.

To exchange information, we assume that agents are located in an agent society allowing for social interactions based on the notions of reputation and trust. In such a framework, agents interact with one another through their IEA. Our social structure is inspired and motivated by the findings in trust and distributed reputation systems, see for example (Gupta, Judge, and Ammar 2003; Damiani et al. 2002; Obreiter, Fähnrich, and Nimis 2005). Indeed, trust and reputation systems are receiving considerable attention both in the academic community and the industry as practical methods for assessing the quality of resources and the reliability of entities in online environments. The basic idea of reputation systems is to let entities rate each other and use aggregated ratings about a certain entity to derive its reputation value. Reputation systems have therefore a typical collaborative flavor, reputation being a quantity derived from the underlying social structure which is usually visible to all the members of the network. In contrast, the basic idea of trust systems is to analyze and combine trust relationships in order to derive the measure of trustworthiness of specific nodes[3].

In non-centralized information societies, an agent neighborhood's pooled and polled reputation consulting mechanism will allow for improved trust evaluation, and summation of positive/negative credibility over time. Indeed, in distributed environments, each agent is responsible for collecting and combining ratings from other agents. Since it is often impractical and costly to collect the ratings derived from all the agents interactions within the society, the reputation rating can be based on a subset of the agent interactions, typically relying on neighborhood agents.

[3]See (Jøsang 2007) for a detailed description and overview of trust and reputation systems.

Information Exchange: Operational Semantics

In this section we outline a possible operational semantics of information exchange. We make the following assumptions.

- We let α_i and β_i be the trust and reputation rating/value of an agent towards an agent A_i.
- Sometimes we use i to denote agent A_i.
- We assume that the trust rating α_i towards an agent A_i builds upon its reputation ratings β_i. Thus, we do not explicit require the value of β_i and evaluate the information received by A_i with respect to α_i.
- Every agent will normally have the possibility of interacting with a limited part of the agent society, i.e., with its "neighbor" agents. The notion of neighbor agents depends in general upon the context at hand and may be defined for example in terms of the trust relationship.
- We assume that IEA maintains an *experience repository* containing the trust and reputation ratings of other agents, as well as an history log of previously exchanged information.

The proposed information exchange protocol consists of 4 phases[4].

Phase 1: Resource Searching. Initially, an agent that is searching for new knowledge broadcasts to all its neighbors a query indicating which information it is looking for. We assume that agents reply by sending back an answer equipped with its evaluation. Thus, a reply of an agent A_i takes the form $x_i = (\overline{x}, E_i)$, where $\overline{x}$ is the answer of A_i to the original query and E_i the evaluation (of $\overline{x}$) performed by A_i. In such a framework, $\overline{x}$ can be any piece of information, and E_i meta-knowledge upon $\overline{x}$. In case $\overline{x}$ is any procedural information about how to solve some particular task, then E_i may be a list of facts evaluating $\overline{x}$ itself, like:

- the prerequisites and objectives of $\overline{x}$;
- the conditions for which $\overline{x}$ is not suitable;
- the risk, gain, and probability of success of $\overline{x}$;
- the evaluation of $\overline{x}$ w.r.t. its objectives;
- its performance results.

Phase 2: Resource Selection and Vote Polling. Upon reception of the answers to its query, the agent selects the one that seems to best satisfy its request. To do so, the agent evaluates every received answer x_i:

$$w_x^i = f(x_i, \alpha_i)$$

by taking into consideration also the trust rating of the agent A_i proposing it. Note that f can be any function, and may differ from agent to agent. Before accepting it, the agent inquires its neighbor agents about their opinion concerning

[4]Note that we do not consider here security problems arising during communication, as they are out of the focus of the paper. The interested reader can refer to (Damiani et al. 2002) for details on security problems in a P2P reputation system protocol.

both the information and the proposing agent. Vote polling can be performed via broadcasting. Each agent upon receiving the poll message checks its experience repository and can therefore respond by communicating its vote on the resource as well as on the proposing agent.

Phase 3: Vote Evaluation. During this phase, the agent collects the set of votes on the resource and its offerer. To evaluate the votes received, the agent needs to base its decision on the trust level of the respondents. The weight of the vote is greater if given by a more trusted respondent. Thus, the final evaluation will be:

$$w_x^S = \underset{i \in S}{H}(w_x^i)$$

where S is the set of respondent agents to the voting and H may be for example an average function.

Phase 4: Resource Acceptance. In case the agent considers the resource quality not sufficient, then it can repeat the voting process on another available resource. Otherwise, it will accept the information received by registering it into the experience repository log, and by sending the resource to the MA that will incorporate it in the agent's knowledge base and will possibly set an "a posteriori" evaluation.

Concluding Remarks

There are several future directions for the ideas that we have discussed and sketched in this initial work. A full system corresponding to the architecture outlined in the paper has not been implemented yet. However, some of its building blocks have been implemented, and we mean to build a full implementation in the near future.

For future directions, a perspective that we believe to be important concerns the fact that in many applications intelligent agents will soon require some kind of "moral" or "ethical" reasoning abilities, which can be modeled as a form of meta-reasoning. Some of the authors of this paper have initiated this research, as reported in (Pereira and Saptawijaya 2007).

References

Alexander, G.; Raja, A.; Durfee, E.; and Musliner, D. 2007. Design paradigms for meta-control in multi-agent systems. In *Proceedings of AAMAS 2007 Workshop on Metareasoning in Agent-based Systems*, 92–103.

Alferes, J. J.; Brogi, A.; Leite, J. A.; and Pereira, L. M. 2002. Evolving logic programs. In *Logics in Artificial Intelligence, Proc. of the 8th Europ. Conf., JELIA 2002*, LNAI 2424, 50–61. Springer-Verlag, Berlin.

Apt, K. R., and Bol, R. 1994. Logic programming and negation: A survey. *The Journal of Logic Programming* 19-20:9–71.

Barklund, J.; Dell'Acqua, P.; Costantini, S.; and Lanzarone, G. A. 2000. Reflection principles in computational logic. *J. of Logic and Computation* 10(6):743–786.

Ben-Ari, M.; Manna, Z.; and Pnueli, A. 1983. The temporal logic of branching time. *Acta Informatica* 20:207–226.

Costantini, S., and Tocchio, A. 2006. About declarative semantics of logic-based agent languages. In Baldoni, M., and Torroni, P., eds., *Declarative Agent Languages and Technologies*, LNAI 3229. Springer-Verlag, Berlin.

Costantini, S.; Tocchio, A.; Toni, F.; and Tsintza, P. 2007. A multi-layered general agent model. In *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing, 10th Congress of the Italian Association for Artificial Intelligence*, LNCS 4733. Springer-Verlag, Berlin.

Costantini, S.; Dell'Acqua, P.; Pereira, L. M.; and Toni, F. 2008. Towards a model of evolving agents for ambient intelligence. Submitted.

Cox, M. T., and Raja, A. 2007. Metareasoning: a manifesto. Technical Report BBN TM-2028, BBN Technologies. URL www.mcox.org/Metareasoning/Manifesto.

Cox, M. T. 2005. Metacognition in computation: A selected research review. *Artificial Intelligence* 169(2):104–141.

Damiani, E.; di Vimercati, S.; Paraboschi, S.; Samarati, P.; and Violante, F. 2002. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In 9th ACM Conf. on Computer and Communications Security.

Emerson, E. A. 1990. Temporal and modal logic. In van Leeuwen, J., ed., *Handbook of Theoretical Computer Science, vol. B*. MIT Press.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Logic Programming, Proc. of the Fifth Joint Int. Conf. and Symposium*, 1070–1080. MIT Press.

Gupta, M.; Judge, P.; and Ammar, M. 2003. A reputation system for peer-to-peer networks. In *Proc. 13th Inter. W. on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03)*, 144–152. NY, USA: ACM.

Jøsang, A. 2007. Trust and reputation systems. In Aldini, A., and Gorrieri, R., eds., *Foundations of Security Analysis and Design IV, FOSAD 2006/2007 Tutorial Lectures*, LNAI 3601, 209–245. Springer-Verlag.

Lichtenstein, O.; Pnueli, A.; and Zuch, L. 1985. The glory of the past. In *Proc. Conf. on Logics of Programs*, LNCS 193. Springer Verlag.

Obreiter, P.; Fähnrich, S.; and Nimis, J. 2005. How social structure improves distributed reputation systems - three hypotheses. In *Agents and Peer-to-Peer Computing*, LNAI 3601, 229–236. Springer-Verlag.

Pereira, L. M., and Saptawijaya, A. 2007. Modelling morality with prospective logic. In Neves, J. M.; Santos, M. F.; and Machado, J. M., eds., *Progress in Artificial Intelligence, Procs. 13th Portuguese Intl.Conf. on Artificial Intelligence (EPIA'07)*, LNAI 4874, 99–111. Extended version forthcoming in IJRIS.

Raja, A., and Lesser, V. 2007. A framework for meta-level control in multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 15(2):147–196.

Richardson, P. J., and Boyd, R. 2005. *Not by Genes Alone - How Culture Transformed Human Evolution*. The University of Chicago Press.