Conditional Learning of Rules and Plans by Knowledge Exchange in Logical Agents

Stefania Costantini
University of L'Aquila, Italy

Pierangelo Dell'Acqua
Linköping University, Sweden

Luis Moniz Pereira
New University of Lisbon, Portugal

RuleML, Barcelona, Spain - 19-21 July, 2011

A Criticism to Rule-Based AI Systems

(and then to Logical Agents)

- "Brittleness" problem: automated systems tend to "break"
  when confronted with even slight deviations
  from the situations specifically anticipated by their designers.

- Brittleness and inflexibility are often attributed to rule-based systems
  due to their supposed over-commitment
  to particular courses of action.

- especially in case of self-modifications: SOAR Team: "in rule-based systems every item which is added to memory via a rule must be maintained by other rules . . ."

A-ILTL (Agent Interval LTL-like) rules with Repairs

Towards Flexible Rule-based Logical Agents

- Each A-ILTL rule is attempted at run-time with a certain frequency
- If the current state of affairs does not satisfy any A-ILTL rule, some kind of repair action has to be undertaken wrt. the violated rule

$$NEVER_{m,n}\,(not\,achieved(G), dropped(G)\,) ::$$
$$(goal(G), deadline(G, T), NOW(T1), T1 \leq T) \div \boxed{inc\_comt(T1)}$$

$$incr\_comt(T) \leftarrow \;\ldots$$

Semantics

- Declarative Semantics:

> S. Costantini, A. Tocchio.
> *About declarative semantics of logic-based agent languages*
> Declarative Agent Languages and Technologies, DALT 2005

- Operational Semantics: general agent model which does not stick to any specific approach for defining logical agents

> S. Costantini, A. Tocchio, F. Toni, P. Tsintza.
> *A multi-layered general agent model*
> Artificial Intelligence and Human-Oriented Computing (AI*IA 2007)

Proposed Framework: Implementation

- Our approach has been partly implemented and partly simulated in DALI
  - Logic agent-oriented programming language
  - "Prolog for Agents"

DALI References

S. Costantini, A Tocchio
A logic programming language for multi-agent systems, JELIA 2002, LNAI 2424

S. Costantini, A Tocchio
The DALI logic programming agent-oriented language, Jelia 2004. LNAI 3229

S. Costantini & many others
The DALI web site, download of the interpreter (2010)
http://www.di.univaq.it/stefcost/Sito-Web-DALI/WEB-DALI/index.php

- **Definition** Let $\mathcal{M}$ be an agent model. An *agent program* is a tuple of software components

$$\langle \mathcal{B}, \mathcal{DI}, \mathcal{SC}, \mathcal{BM}, \mathcal{CS}, \mathcal{A}, \mathcal{C}, \mathcal{CI}, \mathcal{MC}, \mathcal{MCI} \rangle$$

$\mathcal{B}$, $\mathcal{DI}$ - agent's beliefs, and desires and intentions

$\mathcal{SC}$ - sensing and communication component

$\mathcal{BM}$ - belief management

$\mathcal{CS}$ - set of constraints

$\mathcal{A}$ - actions that the agent has devised to perform

$\mathcal{C}$, $\mathcal{CI}$ - object-level control component and control information

$\mathcal{MC}$, $\mathcal{MCI}$ - meta-control component and meta-control information

- Each component is defined (or omitted) according to $\mathcal{M}$

Operational Semantics (cont'd)

- The *operational behavior* of the agent results from the control and meta-control components $\mathcal{C}$ and $\mathcal{MC}$ given the control and meta-control information

- The agent actual functioning relies on underlying control $\mathcal{U}$ and meta-control mechanisms $\mathcal{H}$ that implement the practical counterpart of the agent model $\mathcal{M}$

- **Definition** Let $A_0 = \mathcal{P}$ be the initial agent program, and $\mathcal{E} = \{E_0, \ldots, E_n\}$ a sequence of sets of events. The *underlying control mechanism* $\mathcal{U}$ is a transformation function that transforms (E_0, A_0) into a sequence $A_1, \ldots, A_n$ of agents:

$$(E_i, A_i) \xrightarrow{\mathcal{U}(\mathcal{C}_i, \mathcal{CI}_i)} A_{i+1}$$

Operational Semantics (cont'd)

- The meta-control acts by means of single steps similarly to $\mathcal{U}$

- **Definition** Let $\mathcal{E} = \{E_0, \ldots, E_n\}$ be a sequence of sets of events. The *underlying meta-control mechanism* $\mathcal{H}$ is a transformation function that transform (E_0, A_0) into a sequence $A_1, \ldots, A_n$ of agents:

$$(E_i, A_i) \xrightarrow{\mathcal{H}(\mathcal{MC}_i, \mathcal{MCI}_i)} A_{i+1}$$

Operational Semantics (cont'd)

- **Definition** Given the sequence $\mathcal{E} = \{E_0, \ldots, E_n\}$, the *operational behavior* of the agent is a sequence of transformation steps interleaving control and meta-control

- We assume to perform some steps of meta-control after a number of steps of control
  - The number of steps can be specified in the control information

Breaking Brittleness

(in Rule-based Logical Agents)

- make them able to expand the set of perceptions they can recognize, elaborate on and react to;
- make them able to expand their range of expertise.

Learning by Being Told

- One form of learning may consist in acquiring rules from other agents - *learning by being told*

- The acquired rules can define a reaction to a previously unknown event, or represent a plan to reach an objective

- Learning from others is a practical and economical way of increasing abilities, widely used by human beings

  - Avoiding the cost of learning is an important benefit of imitation

  - An agent that learns and re-elaborates the learned knowledge may become itself an *information producer*

Paper Contribution

- We consider the problem of learning in agents by rule exchange
  - Agents should not blindly incorporate the acquired knowledge, but evaluate how *useful* the new knowledge is
- In our framework, usefulness will not be evaluated by a simulation
  - time-costly, and possibly worsen the problem of brittleness
  - rather, it will be evaluated at runtime based on practical usage
- We associate the new knowledge a specific objective and (possibly) a set of conditions, including a time limit
  - Afterwords, the agent will evaluate whether (and to what extent) the objective has been achieved

Proposed Framework

- Our approach to learning is developed in the context of our general agent model

- Monitoring and supervising tasks are performed at ML

Proposed Framework

- Assume that an agent needs to acquire knowledge to cope with a situation that it cannot handle

- The agent can learn reactive rules and plans from other agents. Once acquired, the new knowledge is recorded in two forms:

  - as plain knowledge, added to the set of beliefs at OL

  - as meta-knowledge recording at the ML what was acquired, with which expectations, etc.

The agent may use the meta-knowledge to evaluate whether the expectations of a rule have been met

Proposed Framework

- Meta-knowledge associated with a set of acquired rules:

  react(I, event(E), rules(R1,. . . ,Rn), cond(pos(P),neg(N)), time(T))

I - unique identifier

E - event to be coped with

$R1, \ldots, Rn$ - acquired reactive rules

P - conditions that have to be fulfilled after the execution of the rules

N - conditions that must not hold after the execution of the rules

T - time threshold allowed for conditions fulfillment

Proposed Framework

- Meta-knowledge associated with an acquired plan:

  plan(I, obj(O), steps(S1,. . . ,Sn), cond(pos(P), neg(N)), time(T))

I - unique identifier

O - objective to be reached

$S1, \ldots, Sn$ - steps of the plan

P - conditions that have to be fulfilled after the plan execution

N - conditions that must not hold after the plan execution

T - time threshold allowed for reaching the objective

Meta-history

- Supervising and monitoring activities rely upon a *meta-history* generated during the agent's operation:
  - which goals have been set and at which time
  - which goals have been successful/failed/timed-out and at which time
  - which external events were known (and thus have been reacted to)

Supervising Activity

- The supervising activity is based upon a mechanism similar to that of *internal events* of the DALI system
  - Expectations related to each piece of new knowledge are checked from time to time
  - Actions are undertaken on awareness of their violation

Semantics of Learning by Rule Exchange

- Our approach to rule exchange fits in the above-presented semantic framework
  - The history and the meta-history are included into the control $\mathcal{CI}$ and meta-control information $\mathcal{MCI}$
  - Among the actions devised by an agent at each step there may be a request for new rules to other agents
  - An incoming event can be the arrival of such new rules that will be managed by the meta-control $\mathcal{MC}$, and thus made available to the control component $\mathcal{C}$

Semantics of Learning by Rule Exchange

- To cope with adding and deleting the new knowledge, we rely on the approach of EVOLP that allows (sets of) rules to be conditionally added or deleted from a program

- The EVOLP approach can be smoothly merged into our semantics: some of the evolution steps determined by the meta-control will be (a series of) EVOLP steps that imply requiring, adding or dropping some knowledge pieces

J. J. Alferes, A. Brogi, J. A. Leite and L. M. Pereira
Evolving logic programs
Logics in Artificial Intelligence (JELIA 2002)

Proposed Framework: Implementation

- Our approach has been partly implemented and partly simulated in DALI
- For the experiments, we introduced a “yellow-pages” mediator agent to cope with match-making and trust

Case Study: an Artificial Fish

- We consider a virtual marine world inhabited by a variety of fish

- For simplicity, the behavior of a fish is reduced to eating food and escaping, and is determined by the motivation of it being satiated and safe

- Each fish is described by variables with values in the range [0 1] with higher values indicating a stronger desire to eat or to avoid predators

  In the formalization, we let t denote the clock time of the system

Case Study: an Artificial Fish

- The fish behavior as well as its internal state are modeled by means of an agent program

$$\mathcal{M} = \langle \mathcal{B}, \mathcal{C}, \mathcal{CI}, \mathcal{MC}, \mathcal{MCI} \rangle$$

where $\mathcal{B}$ is the fish's beliefs component

Case Study: an Artificial Fish

- Assume that at some state α the meta-control information component $\mathcal{MCI}_\alpha$ contains the rules

$$
\begin{array}{ll}
(r1) & \textit{prop1(E)} \leftarrow \textit{SOMETIMES not know(E)} \\
 & \textit{prop1(E)}{:}{>}\textit{learn(new\_rule\_for(E))} \\
 & \textit{know(hunger)}
\end{array}
$$

stating that any time there exists an unknown event, then a new rule to cope with that event must be learned

Case Study: an Artificial Fish

- Suppose that at state α the fish knows that it has to search for food when it is hungry. This is formalized in $\mathcal{CI}_\alpha$ with the reactive rule

  hunger(X), X $\geq$ *0.5, not food :> search(food)*

  where X is the value of hungriness

- The stimuli of the fish (i.e., its input vector $\tilde{x}(t)$) are represented at the controller level via the notion of event

  The value v of the stimulus *hungry(t)* of the process is represented as *hunger(v)*

- Suppose that the fish perceives the stimulus of fear

  Being this stimulus unknown, $\mathcal{MC}$ requires a new rule to handle the unknown event via the reactive rule (r1)

- Assume that at a later state α_2, the meta-control receives in response to its request the rule

$$
\begin{array}{ll}
(r2) & \textit{react}(\#2, \textit{event}(\textit{fear}), \\
 & \textit{rules}(\ulcorner \textit{fear}(X), X \geq 0.5, \textit{nearby}(\textit{predator}) :> \textit{flee} \urcorner), \\
 & \textit{cond}(\textit{pos}(\textit{true}), \textit{neg}(\textit{nearby}(\textit{predator}))), \textit{time}(10))
\end{array}
$$

  $\ulcorner r \urcorner$ abbreviates the representation of a rule r

- Rule r2 is a meta-rule aimed at producing actual object-level rules to be employed by the fish

Case Study: an Artificial Fish

- Suppose that the supervisor also receives a related *evaluation rule* which declaratively expresses how to evaluate r2

$$
\begin{array}{ll}
(r3) & \textit{eval(\#2, act(pos(nearby(predator)), neg(false)),} \\
& \textit{obj(pos(true), neg(nearby(predator))),} \\
& \textit{time(20), criticality(high), action(drop\_rule))}
\end{array}
$$

Rule r3 is a meta-meta rule stating the activation conditions to start the evaluation of r2

- The agent program $A_{\alpha 2}$ evolves through a meta-control step as follows:

$$(E_{\alpha_2}, A_{\alpha_2}) \xrightarrow{\mathcal{H}(\mathcal{MC}_{\alpha_2}, \mathcal{MCI}_{\alpha_2})} A_{\alpha_2+1}$$

$$E_{\alpha_2} = \{r2, r3\}$$
$$A_{\alpha_2} = \langle \mathcal{B}_{\alpha_2}, \mathcal{C}_{\alpha_2}, \mathcal{CI}_{\alpha_2}, \mathcal{MC}_{\alpha_2}, \mathcal{MCI}_{\alpha_2} \rangle$$
$$A_{\alpha_2+1} = \langle \mathcal{B}_{\alpha_2+1}, \mathcal{C}_{\alpha_2+1}, \mathcal{CI}_{\alpha_2+1}, \mathcal{MC}_{\alpha_2+1}, \mathcal{MCI}_{\alpha_2+1} \rangle$$

- The aim of this meta-control step is to incorporate the new learned rules into the agent program A_{α_2}

- The new rules may be possibly de-activated later if they are considered not useful

Case Study: an Artificial Fish

- The rules that are (automatically) added to make the incoming rules r2 and r3 operative are rules r4-r12

$(r4)$ *active(#2), fear(X), X $\geq$ 0.5, nearby(predator) :> flee*
$(r5)$ *active(#2)*
$(r6)$ *obj(#2, cond(pos(true), neg(nearby(predator)))) $\leftarrow$nearby(predator), active(#2)*
$(r7)$ *obj(#2,X), not obj_set(#2,_), current_time(T) :> assert(obj_set(#2,X):T,T+10)*
$(r8)$ *obj_achieved(#2):T $\leftarrow$*
  obj_set(#2,cond(pos(P),neg(N))):T1,T2,
  P, not N, current_time(T), T $\leq$ T2
$(r9)$ *obj_achieved(#2):T :> record(obj_achieved(#2):T)*
$(r10)$ *prop3 $\leftarrow$ NEVER obj_set(#2,_), timed_out(#2)*
$(r11)$ *not prop3 :> drop(#2)*
$(r12)$ *timed_out(#2) $\leftarrow$*
  not obj_achieved(#2), obj_set(#2,_):T1,T2
  current_time(T), T > T2

- The agent program A_{α_2+1} is therefore defined as follows (where in EVOLP notation $\circ$ denotes rule assertion):

$$\mathcal{CI}_{\alpha_2+1} = \mathcal{CI}_{\alpha_2} \circ \{r4, r5\}$$
$$\mathcal{MCI}_{\alpha_2+1} = \mathcal{MCI}_{\alpha_2} \circ \{r6 - r12\}$$
$$\mathcal{MC}_{\alpha_2+1} = \mathcal{MC}_{\alpha_2}$$
$$\mathcal{B}_{\alpha_2+1} = \mathcal{B}_{\alpha_2}$$
$$\mathcal{C}_{\alpha_2+1} = \mathcal{C}_{\alpha_2}$$

- A_{α_2+1} is obtained by updating (wrt. the EVOLP semantics) the components of A_{α_2} with the specified sets of rules

- In this kind of setting, preferences/priorities among events are particularly important
  - In our example, fear must be given higher priority than hunger

Related Work

- Our approach brings some similarity with the approach:

Ancona, D., Mascardi, V., Hübner, J.F., Bordini, R.H. *Coo-agentspeak: Cooperation in AgentSpeak through plan exchange* Autonomous Agents and Multiagent Systems (AAMAS 2004)

Bozzo, L., Mascardi, V., Ancona, D., Busetta, P. *Coows: Adaptive BDI agents meet service-oriented computing* E. W. on Multi-Agent Systems (EUMAS 2005)

- In their approach,
  - each BDI agent can define its plans as private, public, or sharable with other trusted agents
  - a BDI agent not possessing a plan to manage an event, can ask a trusted agent for such a plan
- Their approach has been implemented and applied to service-oriented computing
- Our approach adds the aspect of meta-reasoning for evaluating the new knowledge
  - This evaluation can affect the level of trust of source agent

Related Work

- The work below aims at filtering new percepts according to their expected relevance to the current agent's desires and intentions

  Lorini, E., Piunti, M.
  Introducing Relevance Awareness In Bdi Agents
  Programming Multi-Agent Systems (Promas 2009)

  Koster, A., abd F. Dignum, F.K., Sonenberg, L.
  Augmenting BDI with relevance: Supporting agent-based, pervasive applications
  Pervasive Mobile Interaction Device (PERMID 2008)

- The latter proposal adopts meta-reasoning techniques

- The methods outlined in their work might be suitably integrated in our approach to evaluate how relevant the acquired knowledge is

- As our approach is modular wrt. this aspect, we can in future work implement such techniques in the communication layer of our agent architecture

Conclusions

- The presented approach is part of a comprehensive framework
- ... where we aim at breaking brittleness
- ... by means of meta-reasoning via temporal-logic-like meta-axioms
  - applied at a certain (customizable) frequency
  - able to perform self-checking and enforce self-modifications
- Future work: fully implement and further enrich the framework