

Planning support for Evolving Logic Agents

Peter Klimo

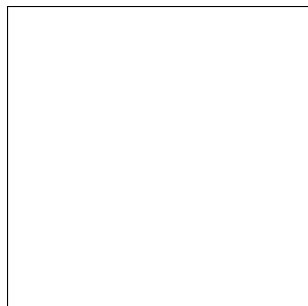
February 26, 2007

Planning support for Evolving Logic Agents - Outline

Planning support for Evolving Logic Agents:

- Initial conditions:
 - Multiagent framework Evolp:
 - based on logic programming
 - nonmonotonic reasoning
 - reasoning about actions
 - BDI aritecture:
 - clearly and formaly well defined
 - 3APL
- My goal:
 - Introduce framework, which combines mentioned approaches
 - Formulate and prove formal properties

Definition of Evolp agent



Agent consist of:

- Knowledge base (\mathcal{KB})
- Capabilities (\mathcal{C})

Deliberation cycle:

- Observe: move input to E
- Think: compute Belief model (\mathcal{BM})
- Act: execute actions from \mathcal{BM}
- Self Update: update \mathcal{KB}

\mathcal{KB} = Evolp program = Logic program + special predicate *assert*
 \mathcal{C} = Set of pairs $\langle Action, Effect \rangle$

Think: $\mathcal{BM} = SelectModel(\mathcal{KB} \oplus E)$

Act: a_i - basic action, $\langle a_i, Effect \rangle \in \mathcal{C}$ then $E = E \cup Effect$

Self Update: $\mathcal{KB} = (\mathcal{KB} \oplus \{r \mid assert(r) \in \mathcal{BM}\})$

Belief Desire Intention Architecture:

- \mathcal{KB} (Knowledge base):
 - input: Events from outside, observations
 - output: Belief Model - model of world (usually set of literals)
- \mathcal{GB} (Goal base): Hierarchical representation of goals
 - input: Belief Model
 - output: Goal Model (maybe inconsistent set of goals)
- \mathcal{INT} (Intention base): Consistent subset of goals, which agent committed to achieve
 - input: Belief Model, Goal Model, Plan Library
 - output: consistent set of pairs $\langle Goal, Plan \rangle$

Languages definition:

- Let \mathcal{A} be a propositional language (world description)
- Let \mathcal{A}_C be a language of basic actions
- Goal language \mathcal{L}_G : if $a_1, a_2, \dots, a_n \in \mathcal{A}$ then $g(a_1, a_2, \dots, a_n) \in \mathcal{L}_G$ representing conjunction of goals.
- Plan languages \mathcal{PLAN} is Evolp program over language $\mathcal{A} \cup \mathcal{A}_C$

Agent definition:

- \mathcal{KB} is Evolp program over language $\mathcal{A} \cup \mathcal{A}_C$
- \mathcal{GB} is Evolp program over language $\mathcal{L}_G \cup \mathcal{A}$, such that $\forall r : Head(r) \in \mathcal{L}_G$
- \mathcal{PL} is finite set of pairs $\langle G, P \rangle, G \in \mathcal{L}_G, P \in \mathcal{PLAN}$
- \mathcal{INT} is subset of $\mathcal{PL}, \langle G, P \rangle$ G-goal agent committed to achieve, P corresponding plan

Evolp Agent with Planning support - Deliberation cycle

The agent evolves from a state i to $i + 1$ using the following deliberation cycle:

- Comp. Belief Model: $KB_i = \{L \mid (\mathcal{KB}_i, E_i) \models_x L\}$
- Comp. Goal Model: $GB_i = \{L \mid (\mathcal{GB}_i, KB_i) \models_y L\}$
- Update intentions: $\mathcal{INT}_{i+1} = UpdateInt(\mathcal{KB}_i, \mathcal{GB}_i, \mathcal{INT}_i)$
- Comp. plan model:
 $PM_i = \{L \mid (\cup_{\langle G,P \rangle \in \mathcal{INT}_{i+1}} P, KB_i \cup GB_i) \models_z L\}$
- Self Update:
 $\mathcal{GB}_{i+1} = (\mathcal{GB}_i, G_{i+1}), G_{i+1} = \{g \mid assert(g) \in GB_i \cup PM_i\}$
 $\mathcal{KB}_{i+1} = (\mathcal{KB}_i, K_{i+1}), K_{i+1} = \{g \mid assert(g) \in KB_i\}$

Inherited properties of agent framework

- Reasoning about action, external / internal update.
- Non-monotonic reasoning in KB, GB and PL
- Multiagent environment: agents can communicate and influent each other (through events).
- Delayed effect of actions, conditional effect of action

Properties of agent framework

- More types of goal are possible: perform, achieve and maintain. Framework allows to represent all of them (Goal base is represented by Evolp program).
- using nondeterministic choice, parallel actions, and comfortable update of Goal base (Plan is represented by Evolp program).
- Different semantics for Goal base and Belief base are possible (sceptically / credulously /select model semantics).

Evolving Agent with planning support:

- Extended Evolp Agent about modules: Goal Base, Intention Base, Plan Library
- Combine logic programming and BDI architecture

Thank you for your attention

Thank you for your attention