

# Programming Agents with Emotions

Mehdi Dastani and John-Jules Ch. Meyer<sup>1</sup>

**Abstract.** This paper presents the syntax and semantics of a simplified version of a logic-based agent-oriented programming language to implement agents with emotions. Four types of emotions are distinguished: happiness, sadness, anger and fear. These emotions are defined relative to agent's goals and plans. The emotions result from the agent's deliberation process and influence the deliberation process. The semantics of each emotion type is incorporated in the transition semantics of the presented agent-oriented programming language.

## 1 Introduction

In the area of intelligent agent research many papers have been written on rational attitudes of agents pertaining to beliefs, desires and intentions (BDI) [8]. More recently one sees a growing interest in agents that display behaviour based on mental attitudes that go beyond rationality in the sense that also emotions are considered (e.g. [12]). There may be several reasons to endow artificial agents with emotional attitudes. For instance, one may be interested to imitate human behaviour in a more natural way [11]. This is in the first instance the perspective of a cognitive scientist, although, admittedly, also computer scientists involved in the construction of systems such as games or believable virtual characters may find this interesting. Our interest in emotional agents here is slightly different. We are interested in agents with emotions, since we believe that emotions are important heuristics that can be used to define effective and efficient decision-making process.

As Dennett [4] has pointed out, it may be sensible to describe the behaviour of complex intelligent systems as being generated by a consideration of their beliefs and desires (the so-called intentional stance). Bratman [1] has made a case for describing the behaviour of such systems in terms of their beliefs, desires and intentions. Here intentions play a role of stabilizing agents decisions, i.e., in deliberating about desires make choices and stick to these as long as it is 'rational'. Following this work it has been recognised that to design and create intelligent *artificial* agents this is a fruitful way of proceeding as well, leading to the so-called BDI architecture of intelligent agents [8]. Pursuing this line of developments we believe that it makes sense to go even further and incorporate also emotional attitudes in agent design and implementation. The role of emotions is to specify heuristics that make the decision process more efficient and effective and, moreover, result in more *believable* behavior [11].

If one takes the stance that endowing artificial agents with emotions is a worthwhile idea, the next question is how to construct or implement such agents. We believe that the agent architectures proposed in the literature are very complicated and hard to engineer, and this holds *a fortiori* for architectures that have been proposed for emotional agents, such as Sloman's CogAff architecture [10].

We start by looking at a logical description of agents with emotions as presented in [5]. Inspired by this logical specification, we aim at constructing agents with emotions through dedicated agent-oriented programming language, in which mental (BDI-like) notions have been incorporated in such a way that they have an unambiguous and rigorous semantics drawn from the meaning they have in the logical specification. In agent programming languages one programs the way the mental state of the agent evolves, and the idea goes back to Shoham [9], who proposed AGENT-0. *In particular, we devise a logic-based agent-oriented programming language, which is inspired by 3APL [3], and can be used to implement emotional agents, following the logical specification of [5] as closely as possible.*

The structure of this paper is as follows. In section 2, we briefly discuss four emotion types and explain their roles in agent programming. In section 3, the syntax of a simplified version a logic-based agent-oriented programming language is presented that allows the implementation of emotional agents. Section 4 presents the part of the semantics that is relevant for the involved emotion types. In section 5, the general structure of the deliberation process is presented. Finally, the paper will be concluded in section 6.

## 2 Emotions and Deliberation

Following [6, 7] we will incorporate four basic emotions: happiness, sadness, anger and fear. As in [5] we will concentrate on the functional aspects of these emotions with respect to the agent's *actions* ignoring all other aspects of emotions. We do this since we are (only) interested here in how emotions affect the agent's practical reasoning. Briefly, *happiness* results when in the pursuit of a goal subgoals have been achieved as a sign that the adopted plan for the goal is going well. In this case nothing special has to be done: plans and goals simply should persist. On the other hand, *sadness* occurs when subgoals are not being achieved. In this case we have to deliberate to either drop its goal or try to achieve it by an alternative plan. *Anger* is the result of being frustrated from not being able to perform the current plan, and causes the agent to try harder so that the the plan becomes achievable again. An agent of which a maintenance goal is being threatened becomes *fearful*, which causes it to see to it that the maintenance goal gets restored before proceeding with other activities.

From this description we see that the emotions that we will consider here are relativised with respect to certain parameters such as certain plans and goals. A consequence of this is that agent may at any time have many different emotions regarding different plans and goals. Thus in this paper we do not consider a kind of 'general' emotional state (e.g. happy) which is not further qualified. Another thing that should be clear is that some emotional types (such as happiness and sadness) come about from performing actions while other emotion types (such as anger and fear) come about from monitoring the

---

<sup>1</sup> Utrecht University, The Netherlands, Email: {mehdi,jj}@cs.uu.nl

deliberation process that decides the next action itself. The resulted emotions have an influence on this deliberation process. This is exactly what we are interested in here: how does these emotions arise and affect the agent's course of action.

### 3 Programming Language: Syntax

In this section, we propose a simplified version of a logic-based agent-oriented programming language that provides programming constructs for implementing agents with emotions. This programming language treats an agent's mental attitudes such as beliefs, goals, and plans, as data structures which can be manipulated by meta operations that constitute the so-called deliberation process. In particular, beliefs and goals are represented as formulas in data bases, plans consists of actions composed by operators such as sequence and iterations, and reasoning rules specify which goals can be achieved by which plans and how plans can be modified. The deliberation process, which constitutes the agent's reasoning engine, is a cyclic process that iterates the sense-reason-act operations. The deliberation process can be implemented by a deliberation language [2]. It should be clear that the deliberation language can be considered as a part of the agent-programming language.

In order to focus on different types of emotions and their computational semantics, we ignore many details that may be relevant or even necessary for a practical logic-based agent-oriented programming language. The practical extension of this logic-based programming language for agents without emotions is presented in [3].

Although the implemented agents will be endowed with emotions, the programming language does not provide any programming constructs to implement emotions. In fact, the syntax of the programming language is similar to the syntax of the language presented in [3]. However, the semantics of the language is extended with the emotional state. The idea is that the emotional state of an agent, which is determined by its deliberation process and the effects of the executed actions, influences the deliberation process itself. Thus, while the syntax of the programming language is similar to the one presented in [3], the semantics of this language is extended with an emotional state.

For this logic-based agent-oriented programming language, we assume a propositional language  $L$ , called the *base language*, and the propositional entailment relation  $\models$ . The beliefs and goals of an agent are propositional formulas from the base language  $L$ . In this paper, we distinguish two different goals types: achievement goals and maintenance goals. An achievement goal denotes a state that the agent wants to achieve and will be dropped as soon as the state is (believed to be) achieved. A maintenance goal denotes a state that the agent wants to maintain. In contrast to achievement goals, a maintenance goal can hold even if the state denoted by it is believed to hold.

The plans of an agent are considered as a sequence of basic actions and test actions. Basic actions are specified in terms of pre- and postcondition. They are assumed to be deterministic in the sense that they either fail (if the precondition does not hold) or have unique resulting states. The precondition of a basic action indicates the condition under which the action can be performed and the postcondition of a basic action indicates the expected effect of the action. The test actions check if propositions are believed, i.e., if propositions are derivable from the agent's belief base.

**Definition 1** (*plan*) Let Action with typical element  $\alpha$  be the set of basic actions and  $\phi \in L$ . The set of plans *Plans* with typical element

$\pi$  is then defined as follows:

$$\pi ::= \alpha \mid \phi? \mid \pi_1; \pi_2$$

We use  $\epsilon$  to denote the empty plan.

Planning rules are used for selecting an appropriate plan for a goal under a certain belief condition. A planning rule is of the form  $\beta, \kappa \Rightarrow \pi$  and indicates that it is appropriate to decide plan  $\pi$  to achieve the goal  $\kappa$ , if the agent believes  $\beta$ . In order to check whether an agent has a certain belief or goal, we use propositional formulas from the base language  $L$ .

**Definition 2** (*plan selection rule*) The set of plan selection rules  $\mathcal{R}_{PG}$  is a finite set defined as follows:

$$\mathcal{R}_{PG} = \{\beta, \kappa \Rightarrow \pi \mid \beta, \kappa \in L, \pi \in Plans\}$$

In the following, we use *belief*( $r$ ), *goal*( $r$ ), and *plan*( $r$ ) to indicate, respectively, the belief condition  $\beta$ , the goal condition  $\kappa$ , and the plan  $\pi$  that occur in the planning rule  $r = (\beta, \kappa \Rightarrow \pi)$ .

Given these languages, an agent can be implemented by programming two sets of propositional formulas (representing the agent's beliefs and goals), a set of basic actions specified in terms of their pre- and postcondition, one set of planning rules, and an order on the set of planning rules to indicate the order according to which the planning rules should be applied.

**Definition 3** (*agent program*) An agent program is a tuple  $(\sigma, (\gamma_a, \gamma_m), A, PG, <)$  where  $\sigma, \gamma_a, \gamma_m \subseteq L, A \subseteq Action, PG \subseteq \mathcal{R}_{PG}$ , and  $<$  is a strict order on  $PG$ .

For technical reasons, we demand that if two planning rules in  $PG$  have equivalent goal formulas in their heads, then they will be identical formulas. Moreover, we assume that an agent's plans are generated by applying planning rules and that an agent does not have initial plans. This simplification is due to the focus of this paper and can be relaxed for a practical agent-oriented programming language.

### 4 Programming Language: Semantics

The semantics of the logic-based agent-oriented programming language is defined by means of a transition system. A transition system for a programming language consists of a set of axioms and derivation rules for deriving transitions for this language. A transition is a transformation of one configuration into another and it corresponds to a single computation step. A configuration represents the state of an agent at each point during computation.

For the purpose of this paper, a configuration consists of a belief base  $\sigma$  representing the agent's beliefs, a goal base  $\gamma = (\gamma_a, \gamma_m)$  representing the agent's (achievement and maintenance) goals, an action base  $A$  consisting of the specifications of basic actions, a plan base  $\Pi$  containing the plans, a set of planning rules  $PG$ , an ordering  $<$  on the set of planning rules, and an emotion state consisting of emotions of the agent with respect to certain goals and/or plans.

**Definition 4** (*agent configuration*) Let  $\Sigma = \{\sigma \mid \sigma \subseteq L, \sigma \not\models \perp\}$ . An agent configuration is a tuple  $(\sigma, (\gamma_a, \gamma_m), A, \Pi, PG, <, E)$  where  $\sigma \in \Sigma$  is the belief base,  $\gamma = (\gamma_a, \gamma_m)$  is the goal base,  $A$  is the action base,  $\Pi \subseteq (L \times Plans)$  is the plan base,  $PG \subseteq \mathcal{R}_{PG}$  is a set of planning rules,  $<$  is a strict order on  $PG$ , and  $E = (E_h, E_s, E_a, E_f)$  are emotion bases for happiness, sadness, anger, and fear, where  $E_h \subseteq \{\mathbf{happy}(\pi, \kappa, \kappa') \mid \pi \in Plans \ \& \ \kappa, \kappa' \in L\}$ ,  $E_s \subseteq \{\mathbf{sad}(\pi, \kappa) \mid \pi \in Plans \ \& \ \kappa \in L\}$ ,  $E_a \subseteq \{\mathbf{angry}(\pi) \mid \pi \in Plans\}$ , and  $E_f \subseteq \{\mathbf{fearful}(\kappa) \mid \kappa \in L\}$ .

Note that the elements of the plan base are defined as tuples consisting of a plan and a goal formula, which indicates the original intended state to be reached/maintained by the plan. Note also that the emotions of an agent are with respect to particular goals and plans such that, for example, an agent can be happy with respect to one of its goal/plan while it is sad with respect to another goal/plan. In the sequel, we often omit the set of basic actions, the set of planning rules  $PG$  and the ordering  $<$  in the agent configuration for reasons of presentation, i.e., we use configurations of the form  $\langle \sigma, \gamma, \Pi, E \rangle$  instead of  $\langle \sigma, \gamma, A, \Pi, PG, <, E \rangle$ . This is not problematic, since the set of basic actions, the set of planning rules, and the ordering defined on them does not change during executions of an agent.

Moreover, in order to check if an agent, which is represented by a configuration  $\langle \sigma, (\gamma_a, \gamma_m), \Pi, E \rangle$ , has certain beliefs and (achievement or maintenance) goals, we use  $\models_b$  and  $\models_g$  defined as follows:

$$\begin{aligned} \langle \sigma, (\gamma_a, \gamma_m), \Pi, E \rangle \models_b \phi &\Leftrightarrow \sigma \models \phi, \\ \langle \sigma, (\gamma_a, \gamma_m), \Pi, E \rangle \models_g \text{achieve}(\kappa) &\Leftrightarrow \gamma_a \models \kappa, \\ \langle \sigma, (\gamma_a, \gamma_m), \Pi, E \rangle \models_g \text{maintain}(\kappa) &\Leftrightarrow \gamma_m \models \kappa. \end{aligned}$$

In the following, we do not present the complete operational semantics of the proposed agent programming language, but provide only the transition rules that are related to the emotional states of the agents. Other transition rules for this agent programming language are trivial extensions of the operational semantics presented in [3].

For the transition rules, we assume a belief update operator  $\tau : \Sigma \times \text{Action} \rightarrow \Sigma$  that determines the updates of the belief base by a basic action. Moreover, we assume  $\text{PreCond} : \text{Plans} \rightarrow L$  and  $\text{PostCond} : \text{Action} \rightarrow L$  to be functions that determine the precondition and the expected effect (postcondition) of a basic action, respectively. The precondition of a plan can be determined recursively in terms of the involved basic actions in a STRIPS like fashion. Moreover, we assume that the postcondition of an action is not necessarily derivable from the update of the belief base by the action, i.e., it is not generally the case that  $\tau(\sigma, \alpha) \models \text{PostCond}(\alpha)$ . This means that the expected effect of an action is not necessarily realized by the action execution (due to the unpredictability of the environment).

Before presenting the emotion related transitions, we need to present some preliminary concepts. We will do this in a rather informal way and refer to [5] for a rigorous treatment. First of all, we use dynamic logic expressions to specify the effects of actions:  $[\alpha]\psi$  denotes that after all possible executions of action  $\alpha$  it holds that  $\psi$ .  $\langle \alpha \rangle \psi = \neg[\alpha]\neg\psi$  expresses there is an execution of  $\alpha$  resulting in a state where  $\psi$  holds. Note that  $\langle \alpha \rangle \top$  ( $\alpha$  is executable) is equivalent with  $\text{PreCond}(\alpha)$ , and that  $\langle \alpha \rangle \psi \rightarrow \langle \alpha \rangle \top$  is valid in dynamic logic. Furthermore, we have the validity  $\langle \alpha; \pi \rangle \psi \leftrightarrow \langle \alpha \rangle (\langle \pi \rangle \psi)$ .

As mentioned, a planning rule is meant to indicate which plan can be executed to realize a goal. A planning rule  $r = (\kappa, \beta \Rightarrow \pi)$  is called correct, denoted by  $\text{correct}(r)$ , if the goal  $\kappa$  is achievable by the execution of the plan  $\pi$ , i.e.,  $\text{correct}(\kappa, \beta \Rightarrow \pi) \Leftrightarrow (\beta \rightarrow \langle \pi \rangle \kappa)$  where  $\langle \pi \rangle \kappa$  states that (under condition  $\beta$ ) after the performance of  $\pi$ , the state denoted by  $\kappa$  holds. We assume that the agent programmer is responsible for the correctness of the planning rules.

Moreover, given the agent configuration  $\mathcal{C}$ , an agent *can* perform a plan  $\pi$  to achieve a goal  $\kappa$  if and only if  $\pi$  is executable (i.e., its precondition is derivable from the agent's belief base) and achieves  $\kappa$  after it is executed, i.e.,  $\mathcal{C} \models \text{Can}(\pi, \kappa) \Leftrightarrow \mathcal{C} \models_b \langle \pi \rangle \kappa$ . (Note that  $\langle \pi \rangle \kappa \rightarrow \text{PreCond}(\pi)$ !)<sup>2</sup>. The notion of ‘Can’ can be made more concrete in the present setting involving planning rules as follows:

An agent can ( $\text{Can}'$ ) perform a plan  $\pi$  to achieve a goal  $\kappa$  if and only if there exists a correct planning rule  $r = (\kappa, \beta \Rightarrow \pi'; \pi)$ , and the belief condition  $\beta$  is entailed by the agent's beliefs, i.e.,

$$\begin{aligned} \mathcal{C} \models \text{Can}'(\pi, \kappa) &\Leftrightarrow \exists r \in PG : \text{correct}(r) \ \& \\ &\text{goal}(r) = \kappa \ \& \\ &\text{belief}(r) = \beta \ \& \\ &\exists \pi' \in \text{Plans} : \text{plan}(r) = \pi'; \pi \ \& \\ &\mathcal{C} \models_b \text{belief}(r) \end{aligned}$$

Note that  $\text{Can}'(\pi, \kappa) \Rightarrow \text{Can}(\pi, \kappa)$ . Finally, as in [5] we employ a notion of possible intention to perform a plan  $\pi$  to achieve a goal  $\kappa$  stating that the agent ‘Can’ do  $\pi$  to achieve its goal  $\kappa$ . In the setting here it is operationalised as follows. An agent possibly intends to perform a plan  $\pi$  to achieve a goal  $\kappa$  if and only if the goal  $\kappa$  is a goal of the agent, and the agent can perform  $\pi$  to achieve  $\kappa$ , i.e.,  $\mathcal{C} \models \text{PossIntend}(\pi, \kappa) \Leftrightarrow \mathcal{C} \models \text{Can}'(\pi, \kappa) \ \& \ \mathcal{C} \models_g \text{achieve}(\kappa)$ .

Note that the intended plans are assumed to be generated by the applications of planning rules. Moreover, if the first part  $\pi'$  of an intended plan  $\pi'; \pi$  gets executed, the agent remains intended to perform the rest of the plan (i.e.,  $\pi$ ) to achieve the original goal  $\kappa$ .

## 4.1 Happiness

According to [5], an agent that is happy observes that its subgoals are being achieved. In particular, an agent that has the intention to do  $\pi$  for achieving goal  $\kappa$  (denoted  $I(\pi, \kappa)$  below), and is committed to it, and that believes that by performing the initial part  $\alpha$  the subgoal  $\kappa'$  should be achieved, is *happy* (with respect to the remainder  $\pi \setminus \alpha$  of the plan – to which it is still committed, the goal  $\kappa$  and subgoal  $\kappa'$ ) if after the performance of  $\alpha$  it believes that indeed the subgoal  $\kappa'$  has been achieved. This is formulated as follows:

$$\begin{aligned} I(\pi, \kappa) \wedge \text{Com}(\pi) \wedge \alpha \preceq \pi \wedge \mathbf{B}([\alpha]\kappa') &\rightarrow \\ [\alpha](\mathbf{B}\kappa' \wedge \mathbf{Com}(\pi \setminus \alpha)) &\rightarrow \text{happy}(\pi \setminus \alpha, \kappa, \kappa') \end{aligned}$$

Moreover, it is defined:  $\text{happy}(\pi, \kappa, \kappa') \Leftrightarrow \text{happy}(\pi, \kappa)$  for all subgoals  $\kappa'$  that are deemed important/crucial by the agent. In this work, we assume that all subgoals are crucial to the agent.

We first observe that an agent becomes happy after the execution of actions. Therefore, the transition rule for action execution is an appropriate transition rule through which agents can become happy with respect to a plan and its corresponding goal and subgoal. In order to define the transition rule for action execution, we translate  $I(\pi, \kappa)$  as possible intention  $\text{PossIntend}(\alpha; \pi, \kappa)$ ,  $\text{Com}(\pi)$  as the fact that the agent has the plan  $\pi$  in the plan base, and  $\mathbf{B}([\alpha]\kappa')$  as the fact that the postcondition of the basic action  $\alpha$  is  $\kappa'$ . After the execution of the basic action  $\alpha$ , if the updated belief base entails the postcondition of the basic action, then the agent becomes happy. Note that the agent becomes committed to the rest-plan  $\pi'$  of the plan  $\pi = \alpha; \pi'$  since  $\pi'$  is added to the new plan base  $\Pi'$ . The transition rule for action execution can be defined as follows:

$$\frac{\langle \sigma, \gamma, \Pi, E \rangle \models \text{PossIntend}(\alpha; \pi', \kappa) \ \& \ (\alpha; \pi') \in \Pi \ \& \ \text{PostCond}(\alpha) = \kappa' \ \& \ \tau(\sigma, \alpha) = \sigma'}{\langle \sigma, \gamma, \Pi, E \rangle \rightarrow \langle \sigma', \gamma, \Pi', E' \rangle}$$

where

$$\Pi' = (\Pi \setminus \{(\alpha; \pi', \kappa)\}) \cup \{(\pi', \kappa)\}$$

$$E = (E_h, E_s, E_a, E_f)$$

$$E' = (E_h \cup \{\text{happy}(\pi, \kappa, \kappa')\}, E_s, E_a, E_f) \text{ if } \sigma' \models \kappa' \\ = E \text{ otherwise}$$

According to [5], happiness causes a kind of persistence with respect to possible intention and commitments, i.e.,

$$I(\pi, \kappa) \wedge \text{Com}(\pi) \wedge \text{happy}(\pi, \kappa) \rightarrow [\text{deliberate}]I(\pi, \kappa) \wedge \text{Com}(\pi)$$

<sup>2</sup> In [5] in the ‘Can’ operator also a notion of ability regarding plan  $\pi$  is incorporated, which may be viewed here as the agent's having access to all basic actions involved in the plan. For simplicity we omit this here.

This means that intentions and commitments of an agent with respect to plans and goals persist through the deliberation *operations* (i.e., operations for selecting and applying reasoning rules) when the agent is happy with respect to those goals and plans. In order to ensure the persistence of intentions and commitments through the deliberation operations, we define in section 5 a deliberation *process* that does not drop any intention with respect to which the agent is happy. We assume that the deliberation operation, represented as [*deliberate*], is a part of the deliberation process, represented as [*deliberate*; *executePlans*]. In fact, the deliberation process consists of deliberation operations plus the execution of plans (not a deliberation operation).

## 4.2 Sadness

A sad agent is disappointed about the way its plans are progressing, and will look for ways of revising its plans (or perhaps even adjust the goals to be achieved) and make them more realistic. In particular, an agent, who intends to perform a plan to achieve a goal and believes that the first action of its plan will have a certain effect, will become sad with respect to the rest of that plan after executing the first action if the agent believes that the expected effect is not achieved. This is formally specified as follows.

$$I(\pi, \kappa) \wedge Com(\pi) \wedge \alpha \preceq \pi \wedge \mathbf{B}([\alpha]\kappa') \rightarrow [\alpha]((\mathbf{B}\neg\kappa' \wedge \mathbf{Com}(\pi \setminus \alpha)) \rightarrow \mathbf{sad}(\pi \setminus \alpha, \kappa))$$

Like happiness, we observe that an agent can become sad after executing a basic action, and that the transition rule for action execution is therefore an appropriate transition rule through which agents can become sad with respect to a plan and its corresponding goal. Under similar translation of logical concepts, the transition semantics for the sadness is as follows.

$$\frac{\langle \sigma, \gamma, \Pi, E \rangle \models PossIntend(\alpha; \pi', \kappa) \ \& \ (\alpha; \pi', \kappa) \in \Pi \ \& \ PostCond(\alpha) = \kappa' \ \& \ \tau(\sigma, \alpha) = \sigma'}{\langle \sigma, \gamma, \Pi, E \rangle \rightarrow \langle \sigma', \gamma, \Pi', E' \rangle}$$

where

$$\begin{aligned} \Pi' &= (\Pi \setminus \{(\alpha; \pi', \kappa)\}) \cup \{(\pi', \kappa)\} \\ E &= (E_h, E_s, E_a, E_f) \\ E' &= (E_h, E_s \cup \{\mathbf{sad}(\pi, \kappa)\}, E_a, E_f) \text{ if } \sigma' \not\models \kappa' \\ &= E \text{ otherwise} \end{aligned}$$

Sadness results in a revision of intention/plan or goal. In particular, an agent who is sad with respect to a plan that he intends to perform to achieve a goal, will become committed to either perform the plan if he can perform it, or otherwise generate an alternative plan and performs the new plan. This effect of sadness is specified as follows.

$$I(\pi, \kappa) \wedge Com(\pi) \wedge \mathbf{sad}(\pi, \kappa) \rightarrow [deliberate]\neg I(\pi, \kappa) \vee \neg Com(\pi) \vee Com(\mathbf{if} \ Can(\pi, \kappa) \ \mathbf{then} \ \pi \ \mathbf{else} \ \mathit{replan}(\pi', \kappa); \pi')$$

Observe that after [*deliberate*], it holds:

$$I(\pi, \kappa) \wedge Com(\pi) \rightarrow Com(\mathbf{if} \ Can(\pi, \kappa) \ \mathbf{then} \ \pi \ \mathbf{else} \ \mathit{replan}(\pi', \kappa); \pi')$$

In order to operationalize this, we check after the deliberation operation if the agent still intends to perform (and is committed to) the plans with respect to which the agent is sad. If so, then we should ensure that the agent checks if he 'can' perform the plan before actually executing it; if he 'can not' perform it, then the agent should generate and execute an alternative plan. In order to simplify this, we check after the deliberation operation and after selecting a plan to execute, if the agent is sad with respect to the selected plan (obviously, the agent intend and is committed to the selected plan for execution) and if he still 'can not' perform it (if the plan 'can' be performed, then it will be executed). In such a case, an alternative plan will be generated

and executed. Given  $\pi$  as the selected plan to execute, the following deliberation step should be included in the deliberation process.

If  $\mathbf{sad}(\pi, \kappa) \in E_s \ \& \ \mathcal{C} \not\models Can(\pi, \kappa)$  then  
 $\{replan(\pi', \kappa); execute(\pi')\}$   
else  $execute(\pi)$

We have assumed that the agent can generate an alternative plan by performing the deliberation operation  $replan(\pi', \kappa)$ , which provides an alternative plan  $\pi'$  to achieve the goal  $\kappa$ . If there is no alternative plan possible, this operation provides a plan which has already been generated. Details about this operation can be found in [2].

## 4.3 Anger

An agent gets angry if its plan is frustrated. This is specified as follows:  $Com(\pi) \wedge \neg Can(\pi, \top) \rightarrow \mathbf{angry}(\pi)$

For the transition semantics, an active plan is considered to be frustrated if its precondition does not hold. This implies that the anger of an agent cannot be incorporated in the transition rule for action execution. We introduce an additional transition rule to update the angry state of the agent with respect to the plans that are not executable. This transition rule is specified as follows:

$$\frac{(\kappa, \pi) \in \Pi \ \& \ \langle \sigma, \gamma, \Pi, E \rangle \not\models Can(\pi, \top)}{\langle \sigma, \gamma, \Pi, E \rangle \rightarrow \langle \sigma, \gamma, \Pi, E' \rangle}$$

where

$$\begin{aligned} (\pi, \kappa) &\in \Pi \\ E &= (E_h, E_s, E_a, E_f) \\ E' &= (E_h, E_s, E_a \cup \{\mathbf{angry}(\pi)\}, E_f) \end{aligned}$$

Note that in our present setting we have that  $Can(\pi, \top) \leftrightarrow PreCond(\pi)$ . So,  $Can(\pi, \top)$  is not derivable from an agent's configuration iff the precondition of the plan  $\pi$  does not hold. Note that we should ensure that this transition rule will be applied during each deliberation cycle.

An angry agent will see to it that he *will* be able to achieve its plans and goals. The effect of being angry is specified as follows:

$$\mathit{angry}(\pi) \rightarrow [deliberate]Com(\mathit{st}it(Can(\pi, \top)))$$

The specification of the effect of being angry indicates that the agent should try to realize the precondition of the plan with respect to which the agent is angry. This can be done by adopting the precondition of the plan as a goal. In order to realize this effect, the deliberation process will include the following deliberation step.

For all  $\mathbf{angry}(\pi) \in E_a$  do

if  $\mathcal{C} \not\models_g \mathit{achieve}(PreCond(\pi))$  then  
 $\mathit{adopt\_goal}(PreCond(\pi))$

This deliberation step ensures that the agent will try to realize the conditions that enable the execution of plans with respect to which the agent is angry.

## 4.4 Fear

An agent becomes fearful if one of its maintenance goals is threatened. This is logically specified as:

$$\models \kappa \rightarrow \neg\kappa' \Rightarrow (goal_m(\kappa') \wedge \mathbf{G}(\kappa)) \rightarrow \mathbf{fearful}(\kappa')$$

For the transition semantics, it means that an agent becomes fearful with respect to a goal if a transition takes place through which a goal is adopted that contradicts a maintenance goal. The transition semantics for the fearfulness is as follows.

$$\frac{\langle \sigma, (\gamma_a, \gamma_m), \Pi, E \rangle \rightarrow \langle \sigma', (\gamma'_a, \gamma'_m), \Pi', E' \rangle \ \& \ \gamma'_a \models \kappa \ \& \ \gamma'_m \models \kappa' \ \& \ \models \kappa \rightarrow \neg\kappa'}{\langle \sigma, (\gamma_a, \gamma_m), \Pi, E \rangle \rightarrow \langle \sigma', (\gamma'_a, \gamma'_m), \Pi', E'' \rangle}$$

where

$$E' = (E'_h, E'_s, E'_a, E'_f)$$

$$E'' = (E'_h, E'_s, E'_a, E'_f \cup \{\mathbf{fearful}(\kappa')\})$$

This transition rule states that whenever there is a transition (action execution transition, rule application transition, etc.) possible through which the agent adopts a goal that threaten one of its maintenance goals, then the agent becomes fearful with respect to the maintenance goal.

The effect of the fearfulness is that the deliberation process should ensure that the feared maintenance goal holds before executing any plan. This is logically specified as follows.

$$Goal_m(\kappa') \wedge Com(\pi) \wedge \mathbf{fearful}(\kappa') \rightarrow$$

$$[deliberate]Com(\mathbf{if} \kappa' \mathbf{then} \pi \mathbf{else} stit(\kappa'); \pi)$$

Normally, the deliberation *process* of an agent selects a plan and executes it. In order to ensure the effect of the fearfulness, the deliberation process will adopt the feared maintenance goals (such that the agent generate plans to realize them) before it proceeds with the execution of its selected plan. Below we make sure that the agent proceeds with executing its selected plan after it has adopted its feared goals. Suppose that the deliberation process has the following deliberation operation through which a plan is selected and executed.

$$\pi := SelectPlanToExecute(\Pi)$$

The effect of the fearfulness can be incorporated in the deliberation *process* by replacing this deliberation *operation* with the following sequence of deliberation operations.

$$\pi := SelectPlanToExecute(\Pi);$$

for all  $\mathbf{fearful}(\kappa') \in E_f$  such that  $C \not\models_b \kappa'$  and  $C \not\models_g achieve(\kappa')$  do

$$\{\pi := adopt\_goal(\kappa'); \pi\};$$

Note that this for-loop generates a plan that consists of the selected plan preceded by the operations to adopt all feared maintenance goals. Note also that the effect of the fearfulness can be strengthened by adding the condition that the maintain goal should hold before proceeding with the executions of the selected plan. This can be done by replacing the body of the for-loop with the following statement:

$$\pi := adopt\_goal(\kappa'); \kappa'?; \pi$$

## 5 Deliberation Process

For the purpose of this paper, we present the following general structure of the deliberation *process* for agents with emotional state.

```

While TRUE Do {
  SenseData := perceive(environment);
  BeliefBase := update(BeliefBase, SenseData);
  r := SelectPlanningRule(PG, <);
  ApplyPlanningRule(r);
  For all angry( $\pi_a$ )  $\in E_a$  do
    if  $C \not\models_g achieve(PreCond(\pi_a))$  then
      adopt_goal(PreCond( $\pi_a$ ));
   $\pi := SelectPlanToExecute(\Pi)$ ;
  If sad( $\pi, \kappa$ )  $\in E_s$  &  $C \not\models Can(\pi, \kappa)$  then
    {replan( $\pi', \kappa$ );  $\pi := \pi'$ }
  for all fearful( $\kappa'$ )  $\in E_f$  such that
     $C \not\models_b \kappa'$  and  $C \not\models_g achieve(\kappa')$  do
    { $\pi := adopt\_goal(\kappa')$ ;  $\pi$ };
  execute( $\pi$ );
  Apply transition rules for angry and fearful
}

```

The deliberation process is assumed to be an cyclic process consisting of perception, reason, and act cycles. During each cycle the

agent perceives its environment and updates its belief base accordingly, applies planning rules to generate plans for its goals, reasons about its emotions and realizes the effect of the emotions, selects and executes plans, and finally makes sure that the transition rules related to different types of emotions are applied. This deliberation cycle can be summarized as follows. Note that the transition rules for **angry** and **fearful** are applied during the last deliberation operation. The transition rules for **happy** and **sad** are applied by the statement **Execute** since these transition rules are the transitions for the execution of basic actions. Note also that the deliberation operations (i.e., the deliberation process except its execution part) does not drop any intention and commitment. Therefore, they ensure that all intentions and commitments, with respect to which the agent is happy, persist during the deliberation operation.

## 6 Conclusion

In this paper, we presented a logic-based agent-oriented programming language that allows the implementation of emotional agents. Four emotion types are discussed and their computational semantics incorporated in the transition system of the programming language. For each emotion type, we presented a transition rule that generates that specific emotions. Based on generated emotions, the deliberation process determines the effect of those emotion on the mental state of the agent. We aim to extend the set of emotion types in the future research. Moreover, we are working on an implementation of an interpreter for the presented programming language. The specified semantics of the emotion types can then be evaluated based on implemented agents. In particular we will test our concept of emotional agents in the realization of a companion robot for young children.

## REFERENCES

- [1] M.E. Bratman, *Intentions, Plans, and Practical Reason*, Harvard University Press, Massachusetts, 1987.
- [2] M. Dastani & F. de Boer & F. Dignum and J.-J. Meyer. Programming agent deliberation: An approach illustrated using the 3APL language. In *Proceedings of The Second Conference on Autonomous Agents and Multi-agent Systems (AAMAS'03)*, Melbourne, Australia, 2003.
- [3] M. Dastani & M. B. van Riemsdijk and J.-J. Ch. Meyer. Programming multi-agent systems in 3APL. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.
- [4] D.C. Dennett, *The Intentional Stance*, MIT Press, Cambridge, Mass., 1987.
- [5] J.-J. Ch. Meyer, Reasoning about Emotional Agents, in *Proc. 16th European Conf. on Artif. Intell. (ECAI 2004)* (R. Lopez de Mntaras & L. Saitta, eds.), IOS Press, 2004, pp. 129-133.
- [6] K. Oatley & J.M. Jenkins, *Understanding Emotions*, Blackwell Publishing, Malden/Oxford, 1996.
- [7] A. Ortony, & G. L. Clore and A. Collins, *The Cognitive Structure of Emotions*, Cambridge University Press, Cambridge, UK, 1988.
- [8] A.S. Rao and M.P. Georgeff, Modeling rational agents within a BDI-architecture, in *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)* (J. Allen, R. Fikes & E. Sandewall, eds.), Morgan Kaufmann, 1991, pp. 473-484.
- [9] Y. Shoham, Agent-Oriented Programming, *Artificial Intelligence* 60(1), 1993, pp. 51-92.
- [10] A. Sloman, Motives, Mechanisms, and Emotions, in: *The Philosophy of Artificial Intelligence* (M. Boden, ed.), Oxford University Press, Oxford, 1990, pp. 231-247.
- [11] A. Sloman, What sort of architecture is required for a human-like agent?, Techn. Report CSRP-96-12, School of Computer Science and Cognitive Science Research Centre, Birmingham, 1996. Invited talk for Cognitive Modelling Workshop at AAA-I'96.
- [12] J. Tao & T. Tan and R.W. Picard, *Affective Computing and Intelligent Interaction*, LNCS 3784, Springer, Berlin, 2005.