Emotions to Control Agent Deliberation*

Bas R. Steunebrink
Department of ICS,
Intelligent Systems Group,
Utrecht University
bass@cs.uu.nl

Mehdi Dastani
Department of ICS,
Intelligent Systems Group,
Utrecht University
mehdi@cs.uu.nl

John-Jules Ch. Meyer
Department of ICS,
Intelligent Systems Group,
Utrecht University
jj@cs.uu.nl

ABSTRACT

The execution of an artificial agent is usually implemented with a sense–reason–act cycle. This cycle includes tasks such as event processing, generating and revising plans, and selecting actions to execute. However, there are typically many choices in the design of such a cycle, which are often hard-coded in the cycle in an ad hoc way. The question of this paper is how one decides, in a principled way, how often and which reasoning rules to apply, how to interleave the execution of plans, or when to start replanning. This paper proposes and formalizes the eliciting conditions of hope, fear, joy, and distress according to a well-known psychological model of human emotion. These conditions are then used to reduce the choices an agent can make in each state. They formalize the idea that emotions focus an agent's attention on what is important in each state.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Miscellaneous

General Terms

Theory

Keywords

Agent programming languages, Emotions

1. INTRODUCTION

Artificial agents are often implemented by means of a sense–reason–act cycle. The sense–reason–act cycle is based on the idea that, in order to decide which action(s) to execute, an agent must reason about its knowledge of its environment and its goals; and in order to keep its knowledge of its environment actual, it must use its sensing capabilities to update its knowledge. Although this general idea may seem quite sensible, actually implementing such a sense–reason–act cycle forces one to make a lot of design choices, many of which are not trivial. For example, at any particular time, an agent may have the option to generate new plans for its goals, to revise existing plans, and to execute a previously generated plan. But based on what principles does one decide when and which plan to generate, revise, or execute?

In the most general sense–reason–act cycle, all such decisions are left unspecified, and so all choices are made nondeterministically. For example, if a plan can be revised according to three different procedures, the procedure to apply is chosen nondeterministically. But there exist many agent programming languages that implement the deliberation of agents by means of some form of sense–reason–act cycle, for example 2APL [5], GOAL [11], Jason [2], Jadex [17], and Jack [20]. These agent implementations do not actually choose nondeterministically, but often make use of ad hoc rules for breaking ties. For example, in 2APL reasoning rules are always tried in the order in which they appear in the agent program, even though its formal semantics leave this to nondeterminism. The main question of this paper is then how one can specify, in a principled way, a sense–reason–act cycle that reduces nondeterminism without resorting to ad hoc decisions.

The approach taken in this paper is to look at psychological models of human emotions for ways in which agent deliberation can be controlled. According to psychological literature, emotions function as a feedback mechanism with respect to one's performance. Particularly in task-oriented situations, positive and negative emotions function as signals for continuing or halting current inclinations, respectively. In this paper we investigate how emotions can be used to specify constraints on an agent's sense–reason–act cycle; these constraints then reduce nondeterminism in a principled way.

Our approach is to take a slightly simplified version 2APL [5] and strip its sense–reason–act cycle down to the point where it is completely general and nondeterministic. We then take four emotion types (i.e., hope, fear, joy, distress) from a well-known psychological model (i.e., the "OCC model") and for each emotion type we formally specify a constraint corresponding to that emotion type. The hope-based constraints turns out to closely match one of the ad hoc choices used in 2APL, whereas the fear-based constraint differs from the corresponding ad hoc choice used in 2APL. The joy-based and distress-based constraints, however, introduce entirely new dynamics. Although in the end there is still nondeterminism, this can be further reduced by introducing more principled, possibly emotion-inspired, constraints.

This paper is structured as follows. In section 2 we present the semantics of a slightly simplified 2APL and illustrate the nondeterminism involved in its sense–reason–act cycle. Section 3 then introduces a formalization of emotion triggers grounded in 2APL. Section 4 then presents the constraints based on the emotion types hope, fear, joy, and distress. Some related work is discussed in section 5. Proofs of propositions can be found in appendix A.

*This work is supported by SenterNovem, Dutch Companion project grant nr: IS053013.

Cite as: Emotions to Control Agent Deliberation, Bas R. Steunebrink, Mehdi Dastani and John-Jules Ch. Meyer, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 973-980

2. A PRACTICAL AGENT PROGRAMMING LANGUAGE

In this section we illustrate the choices that have to be made when designing the sense–reason–act cycle of a BDI agent. To this end, we will define the semantics of a simple agent. These semantics are based on 2APL [5] ("A Practical Agent Programming Language"), but the message of this paper extends to any BDI agent programming language that is based on beliefs, declarative goals, and reasoning rules. Such programming languages include GOAL [11], Jason [2], Jadex [17], and Jack [20]. Although 2APL is a platform for programming multi-agent system, we will only consider the single-agent case in this paper. Also, we will not be concerned with the external environment in which the agent might 'live' and therefore leave this out of our formalization for simplicity.

In 2APL, an agent is programmed in terms of beliefs, declarative goals, plans, and two kinds of reasoning rules: plan generation (PG) rules and plan revision (PR) rules. The analysis of 2APL presented in this paper is purely semantic; the exact syntax of 2APL programs is not of concern here and can be found elsewhere [5]. On the semantic level, then, an agent is represented as a *configuration* containing data structures for storing beliefs, goals, plan, and reasoning rules. This configuration is changed by the execution of actions; the precise effects of actions on a configuration is specified using *transition rules*.

DEFINITION 1. *An agent* configuration *is a tuple* $\langle i, B, G, P, PG, PR\rangle$*, where* i *is the agent* name*,* B *is the* belief base*,* G *is the* goal base*,* P *is the* plan base*,* PG *is the set of* plan generation rules*, and* PR *is the set of* plan revision rules*.*

We will typically use the symbol $\mathcal{C}$ to refer to an agent configuration. It should be noted that during the execution of an agent, only its beliefs, goals, and plans can change. So an agent's name and reasoning rules are static. Beliefs and goals have the following form.

DEFINITION 2. *Let* $\mathcal{L}$ *be a set of well-formed formulas, built as usual by induction from a set of atomic propositions* ATM *and the usual propositional connectives. The set of all possible belief bases is* $\Sigma = \{\, \sigma \subseteq \mathcal{L} \mid \sigma \nvdash_{PC} \bot \,\}$ *and the set of all possible goal bases is* $\Gamma = 2^{\mathcal{K}}$*. Goal formulas are thus drawn form the set* $\mathcal{K}$*, which is the set of conjunctions of atomic propositions, defined as follows:*

$$\mathcal{K} = \{\, \bigwedge X \mid \varnothing \subset X \subseteq \text{ATM} \,\} \tag{1}$$

$$\overline{\mathcal{K}} = \{\, \bigwedge X \mid \varnothing \subset X \subseteq \overline{\text{ATM}} \,\} \tag{2}$$

where $\overline{\text{ATM}} = \{\, \neg p \mid p \in \text{ATM} \,\}$*. The set* $\overline{\mathcal{K}}$ *of conjunctions of negated propositions will be useful for representing unachieved goals. The following typical elements are used:* $\kappa \in \mathcal{K}$*,* $\overline{\kappa} \in \overline{\mathcal{K}}$*, and* $\beta \in \mathcal{L}$*.*

It is thus assumed for each configuration that $B \in \Sigma$ and $G \in \Gamma$. So an agent's belief base is assumed to be consistent, and the goals of an agent are assumed to be specified as conjunctions of atomic propositions. With slight abuse of notation, we use the 'overline' to convert from $\mathcal{K}$ to $\overline{\mathcal{K}}$ and vice versa. So we have that $\overline{\mathcal{K}} = \{\, \overline{\kappa} \mid \kappa \in \mathcal{K} \,\}$ and that $\overline{\overline{\kappa}} = \kappa$. In the following it will be useful to be able to talk about subgoals of goals. This notion is formalized using the following relation.

DEFINITION 3. *The subset relation* $\sqsubseteq$ *on* $\mathcal{K}$ *is defined as:*

$$\sqsubseteq = \{\, (\bigwedge X, \bigwedge Y) \mid \varnothing \subset X \subseteq Y \subseteq \text{ATM} \,\} \tag{3}$$

So $\kappa' \sqsubseteq \kappa$ iff the conjuncts comprising κ' are a non-empty subset of those comprising κ. Normally, we will use this relation when κ is a goal; in that case $\kappa' \sqsubseteq \kappa$ expresses that κ' is a *subgoal* of κ.

Belief bases and goal bases can be queried as follows.

DEFINITION 4. *A belief base is queried using the relation* $\models_b \subseteq \Sigma \times \mathcal{L}$ *and a goal base is queried using the relation* $\models_g \subseteq \Gamma \times \mathcal{K}$*.* $\models_b$ *and* $\models_g$ *are specified as:*

$$B \models_b \beta \;\Leftrightarrow\; B \vdash_{PC} \beta \tag{4}$$

$$G \models_g \kappa \;\Leftrightarrow\; \exists \gamma \in G : \kappa \sqsubseteq \gamma \tag{5}$$

Note that a closed world assumption is used with respect to beliefs; for each formula, either the formula or its negation follows from the belief base. A (sub)goal κ is said to have been *achieved* if it follows from the belief base, i.e., $B \models_b \kappa$. It is assumed that goals are removed as soon as they have been achieved, i.e., $\gamma \in G$ implies $B \models_b \neg\gamma$.

A plan base P consists of plan–(sub)goal–rule triples. For example, $(\pi, \kappa, r) \in P$ means that the agent is committed to performing plan π in order to achieve (sub)goal κ, and that π was obtained by applying rule r. Plans may consist of belief queries, belief updates, external actions (such as sending a message and sensing and manipulating the environment, but we will not go into this any further here), if–then–else and while–do constructs, and sequential compositions of actions. It is assumed that plans are removed as soon as the associated goal has been achieved or has become irrelevant, i.e., $(\pi, \kappa, r) \in P$ implies $B \models_b \neg\kappa$ and $G \models_g \kappa$.

PG-rules are of the form $\kappa \mid \beta \rightarrow \pi$, which specifies that if the agent believes β to be the case, then it can perform plan π to achieve (sub)goal κ. PR-rules are of the form $\pi' \mid \beta \rightarrow \pi$, which specifies that if the agent believes β to be the case, then it can rewrite (replace) π' by plan π. In the following, we will call κ respectively π' the head of the rule, β the guard (belief condition) of the rule, and π the plan of the rule. We will use three accessors for rules: $\mathcal{H}(r)$ denotes the head of rule r (i.e. κ or π'), $\mathcal{G}(r)$ denotes the guard of rule r (i.e. β), and $\mathcal{P}(r)$ denotes the plan of rule r (i.e. π).

DEFINITION 5. *A PG-rule* $r = (\kappa \mid \beta \rightarrow \pi)$ *is* applicable *with respect to a configuration* $\mathcal{C} = \langle i, B, G, P, PG, PR\rangle$ *iff the head of the rule (*κ*) follows from the goal base (*G*) but not from the belief base (*B*) and the guard of the rule (*β*) follows from the belief base. This is abbreviated as follows:*

$$Applicable_{\mathcal{C}}(r) \stackrel{\text{def}}{=} r \in PG \;\&\; G \models_g \kappa \;\&\; B \models_b (\beta \wedge \neg\kappa) \tag{6}$$

Similarly, a PR-rule $r = (\pi' \mid \beta \rightarrow \pi)$ *is* applicable *to a plan* π'' *iff* π'' *is in the plan base, the head of the rule (*π'*) is unifiable with* π''*, and the guard of the rule follows from the belief base. This is abbreviated as follows:*

$$Applicable_{\mathcal{C}}(r, \pi'') \stackrel{\text{def}}{=} r \in PR \;\&\; \exists \kappa', r' : (\pi'', \kappa', r') \in P \;\&\; Unifiable(\pi', \pi'') \;\&\; B \models_b \beta \tag{7}$$

In 2APL, the head π' and plan π of a PR-rule $\pi' \mid \beta \rightarrow \pi$ can contain variables to allow PR-rules to be applied to many different instances of plans. However, we cannot go into the definition of $Unifiable$ without considering the syntax of plans in 2APL. Therefore we simply assume a suitable definition of $Unifiable$ such that it indicates whether or not two plans match. The interested reader can find a proper specification of plan unification in [5].

We will now specify the *effects* of applying a PG-rule or PR-rule. For the application of PG-rule r we specify the meta-action $Gen(r)$ ("generate"), and for the application of PR-rule r' we specify the

meta-action $Rev(r')$ ("revise"). These are meta-actions because they are part of the sense–reason–act cycle of an agent. Gen and Rev cannot appear in the plans of an agent; they are only used in specifying the execution of the agent's deliberation cycle.

If a PG-rule r is *applied*, the triple $(\mathcal{P}(r), \mathcal{H}(r), r)$ is added to the plan base P. If a PR-rule r is applied to $(\pi, \kappa, r') \in P$, the triple (π, κ, r') is replaced by $(\mathcal{P}(r), \kappa, r)$ in the plan base P. But of course a rule can only be applied if it is applicable (in the sense of Definition 5). The meta-actions $Gen(r)$ and $Rev(r)$ are specified using transition semantics, as follows:

$$\frac{Applicable_{\mathcal{C}}(r)}{\mathcal{C} \stackrel{Gen(r)}{\Longrightarrow} \mathcal{C}'} \qquad \frac{\exists \pi : Applicable_{\mathcal{C}}(r, \pi)}{\mathcal{C} \stackrel{Rev(r)}{\Longrightarrow} \mathcal{C}''} \tag{8}$$

The plan base of configuration $\mathcal{C}'$ is $P' = P \cup \{(\mathcal{P}(r), \mathcal{H}(r), r)\}$. The plan base of configuration $\mathcal{C}''$ is $P'' = (P \setminus \{(\pi, \kappa, r')\}) \cup \{(\mathcal{P}(r), \kappa, r)\}$.

We will use a third meta-action which specifies the execution of a plan of an agent. This meta-action is denoted as $Do(\pi)$ and is specified with the following transition rule:

$$\frac{\mathcal{C} \xrightarrow{\alpha} \mathcal{C}'''}{\mathcal{C} \stackrel{Do(\pi)}{\Longrightarrow} \mathcal{C}'''} \tag{9}$$

where α is the first action of plan π, i.e., $\pi = \alpha; \pi'$ for some (possibly empty) π'. So the meta-action Do only executes the first action of the provided plan. This allows plans to be interleaved if the plan base contains more than one plan. $\mathcal{C} \xrightarrow{\alpha} \mathcal{C}'''$ means that configuration $\mathcal{C}$ can make a 'normal' transition to $\mathcal{C}'''$ by performing action α. Here we will not go into the transition rules for actions that can be performed by an agent; for 2APL all such transition rules can be found in [5]. These transition rules typically include a condition to ensure that the plan being executed (i.e., π) is in the plan base and that the goal associated with the plan is still relevant (e.g., $\exists \kappa, r : (\pi, \kappa, r) \in P \ \& \ G \models_g \kappa \ \& \ B \models_b \neg\kappa$). Note that the meta-action for executing a plan only executes one action at the time, i.e., plans are not assumed to be atomic so this meta-action will have to be performed repeatedly to finish a plan.

We now have sufficient ingredients to illustrate that a sense–reason–act cycle typically contains many choice points. In the following, we use notation as is common in Dynamic Logic, i.e., ';' denotes sequential composition, '*' denotes repetition (execute zero or more times), and '+' denotes nondeterministic choice. A complete sense–reason–act cycle is then denoted as $(Sense; Reason\_Act)^*$, where $Sense$ is a meta-action that senses the environment and updates the belief base and goal base accordingly. Although sensing is an essential part of any sense–reason–act cycle, we will focus on the reasoning and acting part in the rest of this paper. The reasoning and acting of an agent can be described using the meta-actions introduced above, as follows.

DEFINITION 6. *The 'reason' and 'act' part of a sense–reason–act cycle is defined in its most general form as follows:*

$$Reason\_Act \stackrel{\text{def}}{=} ApplyPGrule^*; ApplyPRrule^*; ExecutePlan^*$$

$$ApplyPGrule \stackrel{\text{def}}{=} Gen(pg_1) + \cdots + Gen(pg_n)$$

$$ApplyPRrule \stackrel{\text{def}}{=} Rev(pr_1) + \cdots + Rev(pr_m)$$

$$ExecutePlan \stackrel{\text{def}}{=} Do(\pi_1) + \cdots + Do(\pi_k)$$

where $PG = \{pg_1, \ldots, pg_n\}$, $PR = \{pr_1, \ldots, pr_m\}$, *and* $P = \{(\pi_1, \kappa_1, r_1), \ldots, (\pi_k, \kappa_k, r_k)\}$.

So with respect to reasoning and acting, the agent can do three kinds of things: apply PG-rules, apply PR-rules, and execute plans. The above specification is completely general: all these meta-actions can be done in any particular order; all choices are nondeterministic; there is no commitment to any particular order of execution of the presented meta-actions.

Of course choices have to be made when implementing an agent programming language. For example, in 2APL, PG-rules are tried and applied in the order in which they appear in the source code of the agent program; PR-rules are only applied when execution of a plan fails; and plans are interleaved if there is more than one in the plan base. Although seemingly reasonable choices, they remain ad hoc choices. It will be clear that, among the many different ways of designing a sense–reason–act cycle, some will be better than others. For example, constantly checking each PG-rule and each PR-rule for applicability after performing a single atomic action will obviously yield a very inefficient agent. But how does one decide, in a principled way, how to choose between generating, revising, and executing plans? In the next section we propose to use emotions as such design principles.

3. A FORMALIZATION OF EMOTION TRIGGERS

There is little consensus among psychologists as to what exactly constitutes an emotion and how it differs from related affective processes such as moods and impulses. However, this does not mean that making broad classifications is impossible or useless. According to a classification by Gross [10], *emotions* typically have specific objects and give rise to action tendencies relevant to these objects. Moreover, emotions can be both positive and negative. Emotions are often distinguished from *moods*, which are more diffuse and last longer than emotions. Other affective processes include *stress*, which arises in taxing circumstances and produces only negative responses; and *impulses*, which are related to hunger, sex, and pain and give rise to responses with limited flexibility. Of these four types of affective processes, we focus on *emotions* in this paper.

With respect to emotions, usually three phases are distinguished. First, the perceived situation is *appraised* by an individual based on what he or she thinks is relevant and important. For example, Alice, who likes receiving presents, is given a necklace by Bob. Alice then judges receiving the necklace as desirable and Bob's action as praiseworthy. Consequently, the appraisal of this action and its outcome causes gratitude towards Bob to be triggered for Alice. Note that different types of emotions may be triggered simultaneously by the same situation, some of which may even be seen as conflicting. For example, Alice may at the same time be disappointed because it was not the necklace she had hoped to receive. Emotion theories dealing with appraisal are for example [8, 16, 13, 14, 18]. Second, the appraisal of some situation can cause the triggered emotions, if exceeding some threshold, to create a conscious awareness of emotional feelings, leading to the *experience* of having emotions. For example, Alice's gratitude towards Bob will have a certain intensity and will probably decrease over a certain amount of time. All this may depend on, e.g., the degree of desirability of receiving a necklace and Alice's previous attitude towards Bob. Emotion theories dealing with these quantitative aspects of emotions are for example [16, 8, 7]. Third, emotional feelings need to be *regulated*. For example, Alice may want to organize her behavior such that positive emotions are triggered as often as possible and negative emotions are avoided or drowned by positive ones. She could do this by being nice to Bob so that he will give her more presents, or avoiding him altogether so that she will never again be confronted with his bad taste in jewelry. In fact, some emotion theories posit that the main purpose of emotions is to function as a heuristical mechanism

for selecting behaviors [4, 14, 13]. Emotion theories dealing with behavioral consequences of emotions are for example [8, 7, 12, 14].

This section presents a formalization of the eliciting conditions of four emotion types ('hope', 'fear', 'joy', 'distress') as described in the psychological model of Ortony, Clore & Collins [16]. We have chosen the "OCC model" because it provides a clear classification of emotion types, it lists concise descriptions of the conditions that elicit emotions, and for this it uses concepts that are well studied and relatively straightforward to formalize. The presented formalization constitutes a formal model for the appraisal part of the OCC emotion theory. Here we translate the conditions that trigger 'hope', 'fear', 'joy', and 'distress' into concepts that are used in agent programming and in 2APL in particular, thus formalizing the appraisal process corresponding to these emotion types. It should be noted beforehand that we use these emotion types only as convenient labels to describe particular cognitive states of an agent. No representation of emotions will be stored in the agent configuration; consequently, we will not be concerned with issues such as whether or not an agent subjectively experiences an emotion. In this paper emotion models are only used to inform the design of a sense–reason–act cycle in a principled way.

The OCC model specifies eliciting conditions of 'joy' as "pleased about a desirable event" and 'distress' as "displeased about an undesirable event." On the other hand, 'hope' is specified as "pleased about the prospect of a desirable event" and 'fear' as "displeased about the prospect of an undesirable event."[1] It may appear that the specifications of hope and fear subsume those of joy and distress, e.g., that hope is a special kind of joy, namely one concerning a prospect. However, personal communication with Ortony and Clore has revealed that this is not the intended reading [15]. The specifications of joy and distress assume a default disregard of prospects. In the following, we will call an event 'actual' if it is not prospective. Note that 'actual' only applies to an agent's perceptions; as soon as (a consequence of) an event is perceived, it is called 'actual', even though the event may have occurred some time in the past. The specifications of joy and distress may then be read as "(dis)pleased about an actual (un)desirable event."

Both joy/distress and hope/fear actually do subsume another type of emotions, namely 'pleased' and 'displeased'. These two have been chosen by OCC to function as labels for the most general type of valenced reactions to consequences of events, because they are neutral sounding words with respect to intensity of experience, focus of attention, motivational and behavioral effects, etc. Here we use pleased/displeased to compose our formalization of the specifications of hope/fear and joy/distress, as shown in Definition 7. In the following, we will not be concerned with the emotion types 'pleased' and 'displeased', except in using them to formalize 'hope', 'fear', 'joy', and 'distress' in accordance with the OCC model.

Below is a BDI-based formalization of hope, fear, joy, and distress, and their constituent constructs. This formalization concerns the *eliciting conditions* of these four emotion types; this means that they express the moments at which hope, fear, joy, or distress are triggered. For example, $\mathbf{Hope}_i^{\mathbf{T}}(\varphi)$ is read as "hope about consequence φ (of an event) is triggered for agent i." It is important to note that this is *not* the same as "agent i hopes φ will happen." The latter phrase expresses a kind of *experience* of the feeling of hope. In the formalization below, we use a superscript "T" as a constant reminder that all that is expressed is the satisfaction of the conditions that can trigger an emotion, regardless of whether the emotion in question actually is or ever will be experienced.

[1]Events are always appraised with respect to their *consequences*, so each instance of the phrase "(un)desirable event" is actually an abbreviation of "(un)desirable consequence of an event" [15].

DEFINITION 7. *Let φ be a typical formula of a language built from the following constructs:*

$$\varphi ::= p\ (\in \text{ATM}) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{Prev}\,\varphi \mid \mathbf{Fut}\,\varphi \mid \mathbf{B}_i\varphi \mid \mathbf{Des}_i(\varphi) \mid \mathbf{Undes}_i(\varphi) \quad (10)$$

where $\mathbf{Prev}\,\varphi$ is read as "in the previous state, φ was true;" $\mathbf{Fut}\,\varphi$ is read as "in some possible future state (excluding the present), φ is true;" $\mathbf{B}_i\varphi$ is read as "agent i believes φ;" and $\mathbf{Des}_i(\varphi)$ (resp. $\mathbf{Undes}_i(\varphi)$) is read as "agent i appraises consequence φ (of an event) as desirable (resp. undesirable)." The eliciting conditions of 'hope', 'fear', 'joy', and 'distress' are then formalized in accordance with the OCC model as the following macros:

$$\mathbf{Hope}_i^{\mathbf{T}}(\varphi) \stackrel{\text{def}}{=} \mathbf{Pleased}_i^{\mathbf{T}}(\varphi) \wedge \mathbf{Prospect}_i(\varphi) \quad (11)$$

$$\mathbf{Fear}_i^{\mathbf{T}}(\varphi) \stackrel{\text{def}}{=} \mathbf{Displeased}_i^{\mathbf{T}}(\varphi) \wedge \mathbf{Prospect}_i(\varphi) \quad (12)$$

$$\mathbf{Joy}_i^{\mathbf{T}}(\varphi) \stackrel{\text{def}}{=} \mathbf{Pleased}_i^{\mathbf{T}}(\varphi) \wedge \mathbf{Actual}_i(\varphi) \quad (13)$$

$$\mathbf{Distress}_i^{\mathbf{T}}(\varphi) \stackrel{\text{def}}{=} \mathbf{Displeased}_i^{\mathbf{T}}(\varphi) \wedge \mathbf{Actual}_i(\varphi) \quad (14)$$

$$\mathbf{Pleased}_i^{\mathbf{T}}(\varphi) \stackrel{\text{def}}{=} \mathbf{PrcvCnsq}_i(\varphi) \wedge \mathbf{Des}_i(\varphi) \quad (15)$$

$$\mathbf{Displeased}_i^{\mathbf{T}}(\varphi) \stackrel{\text{def}}{=} \mathbf{PrcvCnsq}_i(\varphi) \wedge \mathbf{Undes}_i(\varphi) \quad (16)$$

$$\mathbf{PrcvCnsq}_i(\varphi) \stackrel{\text{def}}{=} \mathbf{Prospect}_i(\varphi) \vee \mathbf{Actual}_i(\varphi) \quad (17)$$

$$\mathbf{Prospect}_i(\varphi) \stackrel{\text{def}}{=} \mathbf{Future}_i(\varphi) \vee \mathbf{Uncertain}_i(\varphi) \quad (18)$$

$$\mathbf{Actual}_i(\varphi) \stackrel{\text{def}}{=} \mathbf{New}\,\mathbf{B}_i\varphi \quad (19)$$

$$\mathbf{Future}_i(\varphi) \stackrel{\text{def}}{=} \mathbf{New}\,(\mathbf{B}_i\neg\varphi \wedge \mathbf{B}_i\mathbf{Fut}\,\varphi) \quad (20)$$

$$\mathbf{Uncertain}_i(\varphi) \stackrel{\text{def}}{=} \mathbf{New}\,(\neg\mathbf{B}_i\varphi \wedge \neg\mathbf{B}_i\neg\varphi) \quad (21)$$

$$\mathbf{New}\,\varphi \stackrel{\text{def}}{=} \varphi \wedge \mathbf{Prev}\,\neg\varphi \quad (22)$$

(11)–(14) read just as the specifications given in the OCC model, as quoted earlier, except that (13) and (14) explicate the default focus on 'actual' events. These four emotion types all depend on either 'pleased' (15) or 'displeased' (16), which are triggered when a consequence (of an event) is perceived that is appraised as desirable or undesirable, respectively. (17) expresses that two types of consequences of events are considered: prospective and actual ones. (18) explicates the (intentionally) ambiguous notion of 'prospect' as used in the OCC model; namely, to expresses both future and uncertain (past or present) events. (19) expresses that the perception of an actual consequence is modeled as a 'belief update'. (20) expresses that the perception of a future consequence is modeled as a 'belief update' expressing that something which is not true now may be true some time in the future. (21) expresses that the perception of an uncertain consequence is modeled as an 'uncertainty update'. Finally, (22) expresses that a formula is 'new' iff it is true now but not previously.

With respect to $\mathbf{Prev}$, we say "*the* previous state" because we assume a linear past (and a branching future). Note that $\mathbf{Actual}$, $\mathbf{Future}$, and $\mathbf{Uncertain}$ are mutually exclusive if it is assumed that beliefs are consistent (i.e., $\neg(\mathbf{B}_i\varphi \wedge \mathbf{B}_i\neg\varphi)$ is valid). With this assumption the triggering conditions of joy and hope are also mutually exclusive, and so are those of distress and fear. Note that this does not mean that an agent cannot experience both fear and distress; it only ensures that fear and distress cannot both be triggered by the same consequence of an event, because a consequence is either prospective or actual.

We are now in a position to define a satisfaction relation $\models$ for agent configurations, such that the eliciting conditions of 'hope',

'fear', 'joy', and 'distress' can be said to hold in a particular configuration. In order to be able to interpret the $\mathbf{Prev}$ construct, it is assumed that agent configurations contain an additional structure $\mathcal{C}_{-1}$, storing the belief base, goal base, and plan base of the previous configuration. So if a transition $\mathcal{C} \longrightarrow \mathcal{C}'$ is made, where $\mathcal{C} = \langle i, B, G, P, PG, PR, \mathcal{C}_{-1} \rangle$ and $\mathcal{C}' = \langle i, B', G', P', PG, PR, \mathcal{C}'_{-1} \rangle$, then $\mathcal{C}'_{-1} = \langle B, G, P \rangle$. So only the previous configuration is stored, not a complete history. Formulas are then interpreted on agent configurations as follows.

DEFINITION 8. *Let $\mathcal{C} = \langle i, B, G, P, PG, PR, \mathcal{C}_{-1} \rangle$; then $\mathcal{C} \models \varphi$ is specified as follows:*

$$\begin{array}{lll} \mathcal{C} \models \neg\varphi & \text{iff} & \text{not } \mathcal{C} \models \varphi \\ \mathcal{C} \models \varphi_1 \wedge \varphi_2 & \text{iff} & \mathcal{C} \models \varphi_1 \,\&\, \mathcal{C} \models \varphi_2 \\ \mathcal{C} \models \mathbf{B}_i\beta & \text{iff} & B \models_b \beta \\ \mathcal{C} \models \mathbf{Des}_i(\kappa) & \text{iff} & G \models_g \kappa \\ \mathcal{C} \models \mathbf{Prev}\,\varphi & \text{iff} & \mathcal{C}_{-1} \models \varphi \\ \mathcal{C} \models \mathbf{B}_i\mathbf{Fut}\,\xi & \text{iff} & \mathit{Fut}_{\mathcal{C}}(\xi) \end{array}$$

where $\mathit{Fut}_{\mathcal{C}}(\xi)$ abbreviates, for $\xi \in \mathcal{K} \cup \overline{\mathcal{K}}$:

$$\mathit{Fut}_{\mathcal{C}}(\kappa) \stackrel{\text{def}}{=} \exists r \in PG : \kappa \sqsubseteq \mathcal{H}(r) \,\&\, B \models_b \mathcal{G}(r) \qquad (23)$$

$$\mathit{Fut}_{\mathcal{C}}(\overline{\kappa}) \stackrel{\text{def}}{=} \exists (\pi, \kappa', r) \in P : \overline{\kappa} \in \mathit{PostCond}_{\mathcal{C}}(r) \qquad (24)$$

Below we will explain Definition 8 in detail.

In the interpretation of $\mathbf{Prev}\,\varphi$, writing $\mathcal{C}_{-1} \models \varphi$ is strictly speaking not allowed, because $\mathcal{C}_{-1}$ is not a proper configuration. Indeed, $\mathcal{C}_{-1} \models \varphi$ is short for $\langle i, B', G', P', PG, PR, \bot \rangle \models \varphi$, where $\mathcal{C}_{-1} = \langle B', G', P' \rangle$. This is possible because the name and reasoning rules are assumed to be static. Although this construction still fails if $\mathbf{Prev}$ is nested (as in $\mathcal{C} \models \mathbf{Prev}\,\mathbf{Prev}\,\mathbf{B}_i p$), this is not problematic because in this paper no situation will occur where nesting of $\mathbf{Prev}$ is required. Note that such nesting also fails for $\mathbf{B}$, $\mathbf{Des}$, and $\mathbf{Fut}$. So the kinds of formulas that can be interpreted on agent configurations is limited with respect to the grammar introduced in Definition 7, but sufficient for the purposes of this paper. Indeed, Definition 8 does not allow arbitrary formulas to be interpreted on agent configurations, but it does allow all eliciting conditions of emotions (as presented in Definition 7) to be interpreted. For example, according to Definition 8 the $\mathbf{Fut}$ operator can only be interpreted inside a $\mathbf{B}$ operator, but as can be seen in Definition 7 $\mathbf{Fut}$ is never used outside $\mathbf{B}$, so this restriction does not pose a problem.

The reasoning behind the $\mathit{Fut}_{\mathcal{C}}$ construct is as follows. Suppose $\kappa \mid \beta \rightarrow \pi$ is a PG-rule of agent i; this means that the agent programmer promises that κ can be achieved by executing π in a state where β holds. Thus $(\kappa \mid \beta \rightarrow \pi) \in PG$ expresses that agent i believes that $\beta \rightarrow \langle\pi\rangle\kappa$ is (always) true.[2] So we would have that $\mathcal{C} \models \mathbf{B}_i(\beta \rightarrow \langle\pi\rangle\kappa)$ iff $(\kappa \mid \beta \rightarrow \pi) \in PG$. Now $\mathbf{Fut}\,\varphi$ expresses that φ will hold in *some* possible future, i.e., it can be seen as an existential quantification over all possible plans (note that $\mathbf{Fut}$ does not incorporate intention). So, replacing $\langle\pi\rangle$ by $\mathbf{Fut}$, we would have that $\mathcal{C} \models \mathbf{B}_i(\beta \rightarrow \mathbf{Fut}\,\kappa)$ iff $\exists r \in PG : \kappa = \mathcal{H}(r) \,\&\, \beta = \mathcal{G}(r)$. But the guard of the rule is satisfied if $B \models_b \mathcal{G}(r)$, and if κ holds after executing π, then so do all its subgoals. The definition of $\mathit{Fut}_{\mathcal{C}}(\kappa)$ above is then the result of putting together all these ideas.

[2]Here we use $\langle\pi\rangle\kappa$ to express that κ is a possible result of performing π, as usual in Dynamic Logic. But note that $\beta \rightarrow \langle\pi\rangle\kappa$ will not appear in our actual object language; it is only used to clarify the interpretation of $\mathbf{B}_i\mathbf{Fut}\,\xi$.

As for the second $\mathit{Fut}_{\mathcal{C}}$ construct, the fact that a plan π is in the plan base means that the agent is committed to performing the plan. Thus the agent should believe each *postcondition* of plan π to be possibly true after the execution of π. So we would have that $\mathcal{C} \models \mathbf{B}_i\langle\pi\rangle\overline{\kappa}$ iff $(\pi, \kappa', r) \in P$ and $\overline{\kappa} \in \mathit{PostCond}_{\mathcal{C}}(r)$ (the function $\mathit{PostCond}_{\mathcal{C}}$ will be explained below). Again, we replace $\langle\pi\rangle$ by $\mathbf{Fut}$, obtaining formula (24).

In order to determine which formulas can possibly hold after the execution of a plan in the plan base, we thus make use of the $\mathit{PostCond}_{\mathcal{C}}$ function. In the following we are actually only interested in a special kind of formula as postcondition, namely inverted goal formulas. The reason for this is that we want to find out whether one plan threatens the goal of another plan. A plan threatens a goal if an inverted subgoal of the goal is among the postconditions of the plan. The set of all (sub)goals that can possibly be threatened is $\mathcal{K}_{\mathcal{C}} = \{\, \kappa \mid r \in PG,\ \kappa \sqsubseteq \mathcal{H}(r) \,\}$. The set of inverted subgoals is then $\overline{\mathcal{K}}_{\mathcal{C}} = \{\, \overline{\kappa} \mid \kappa \in \mathcal{K}_{\mathcal{C}} \,\}$. The function $\mathit{PostCond}_{\mathcal{C}}$ then associates with each PG-rule and PR-rule a subset of $\overline{\mathcal{K}}_{\mathcal{C}}$ indicating which subgoal may be false ("undermined") after performing the plan of the rule. Its mapping is thus $\mathit{PostCond}_{\mathcal{C}} : (PG \cup PR) \rightarrow 2^{\overline{\mathcal{K}}_{\mathcal{C}}}$. It should be noted that the function $\mathit{PostCond}_{\mathcal{C}}$ only depends on the reasoning rules of a configuration, which are static, and that it is not assumed to take into account preconditions at 'run-time'. Therefore we may assume that the postconditions of all rules are determined at 'compile-time', thereby making $\mathit{PostCond}_{\mathcal{C}}$ a cheap lookup function. The postconditions of a rule can be determined by performing some analysis of the rule's plan, or by letting the agent programmer annotate each reasoning rule with the (relevant) postconditions.

Of all constructs used in the formalization of eliciting conditions, only the construct $\mathbf{Undes}$ for expressing undesirability is still undefined. Here we define undesirability simply as an 'inverse' of desirability:

$$\mathbf{Undes}_i(\varphi) \stackrel{\text{def}}{=} \mathbf{Des}_i(\overline{\varphi}) \qquad (25)$$

So the satisfaction of undesirability in a configuration becomes $\mathcal{C} \models \mathbf{Undes}_i(\overline{\kappa})$ iff $G \models_g \kappa$ (recall that $\overline{\overline{\kappa}} = \kappa$). Note that this definition of undesirability is not unreasonable in light of a restriction to achievement goals.[3] In fact, it is noted by OCC that if an event is desirable to some degree, the absence of that event may be undesirable to the same degree [16]. Because we have formalized a desirable event as the satisfaction of a goal formula, the absence of a desirable event can thus be formalized as the satisfaction of an inverted goal formula.

Using formulas (11)–(22) plus (25) as macros, it will now follow that these equivalences hold for all possible agent configurations:

$$\mathbf{Hope}^{\mathbf{T}}_i(\kappa) \leftrightarrow \mathbf{Des}_i(\kappa) \wedge \mathbf{New}\,(\mathbf{B}_i\neg\kappa \wedge \mathbf{B}_i\mathbf{Fut}\,\kappa) \qquad (26)$$

$$\mathbf{Fear}^{\mathbf{T}}_i(\overline{\kappa}) \leftrightarrow \mathbf{Des}_i(\kappa) \wedge \mathbf{New}\,(\mathbf{B}_i\neg\overline{\kappa} \wedge \mathbf{B}_i\mathbf{Fut}\,\overline{\kappa}) \qquad (27)$$

$$\mathbf{Joy}^{\mathbf{T}}_i(\kappa) \leftrightarrow \mathbf{Des}_i(\kappa) \wedge \mathbf{New}\,\mathbf{B}_i\kappa \qquad (28)$$

$$\mathbf{Distress}^{\mathbf{T}}_i(\overline{\kappa}) \leftrightarrow \mathbf{Des}_i(\kappa) \wedge \mathbf{New}\,\mathbf{B}_i\overline{\kappa} \qquad (29)$$

When comparing (26) and (27) to Definition 7, it may appear that they miss the 'uncertainty' aspect (recall that 'prospect' was used to refer to both current uncertainty and future possibility). However, because of our closed world assumption on belief bases, uncertainty in that sense cannot exist; a proposition either follows from the belief base or it does not. Given (25), it is easy to verify that (28) and (29) do correspond directly to Definition 7.

[3]When types of goals other than achievement goals are incorporated, the framework may have to be extended such that desirability and undesirability are defined independently.

The propositions above are translated to the level of agent configurations as follows. Let $\mathcal{C} = \langle i, B, G, P, PG, PR, \mathcal{C}_{-1} \rangle$ where $\mathcal{C}_{-1} = \langle B', G', P' \rangle$. Then the following equivalences hold:

$$\mathcal{C} \models \mathbf{Hope}_i^{\mathbf{T}}(\kappa) \;\Leftrightarrow\; G \models_g \kappa \;\&\; \neg Fut^+_{\mathcal{C}_{-1}}(\kappa) \;\&\; Fut^+_{\mathcal{C}}(\kappa) \quad (30)$$

$$\mathcal{C} \models \mathbf{Fear}_i^{\mathbf{T}}(\overline{\kappa}) \;\Leftrightarrow\; G \models_g \kappa \;\&\; \neg Fut^+_{\mathcal{C}_{-1}}(\overline{\kappa}) \;\&\; Fut^+_{\mathcal{C}}(\overline{\kappa}) \quad (31)$$

$$\mathcal{C} \models \mathbf{Joy}_i^{\mathbf{T}}(\kappa) \;\Leftrightarrow\; G \models_g \kappa \;\&\; B' \models_b \neg\kappa \;\&\; B \models_b \kappa \quad (32)$$

$$\mathcal{C} \models \mathbf{Distress}_i^{\mathbf{T}}(\overline{\kappa}) \;\Leftrightarrow\; G \models_g \kappa \;\&\; B' \models_b \neg\overline{\kappa} \;\&\; B \models_b \overline{\kappa} \quad (33)$$

where $Fut^+_{\mathcal{C}}(\xi)$ is defined for $\xi \in \mathcal{K} \cup \overline{\mathcal{K}}$ as follows:

$$Fut^+_{\mathcal{C}}(\xi) \stackrel{\text{def}}{=} (B \models_b \neg\xi) \;\&\; Fut_{\mathcal{C}}(\xi) \quad (34)$$

So we have that $Fut^+_{\mathcal{C}}(\xi)$ iff $\mathcal{C} \models \mathbf{B}\neg\xi \wedge \mathbf{B}\,\mathbf{Fut}\,\xi$, and thus that $\mathcal{C} \models \mathbf{New}\,(\mathbf{B}\neg\xi \wedge \mathbf{B}\,\mathbf{Fut}\,\xi)$ iff $\neg Fut^+_{\mathcal{C}_{-1}}(\xi) \;\&\; Fut^+_{\mathcal{C}}(\xi)$.

In preparation of the next section, we will show two more interesting propositions with respect to 'hope' and 'fear'. Let r be a PG-rule of configuration $\mathcal{C}$. The following macro then expresses that hope is triggered with respect to r as soon as the PG-rule becomes applicable:

$$Hope_{\mathcal{C}}(r) \stackrel{\text{def}}{=} Applicable_{\mathcal{C}}(r) \;\&\; \neg Fut^+_{\mathcal{C}_{-1}}(\mathcal{H}(r)) \quad (35)$$

The term $\neg Fut^+_{\mathcal{C}_{-1}}(\mathcal{H}(r))$ expresses that rule r cannot have been applicable in the previous configuration. We re-emphasize that including this term ensures that $Hope_{\mathcal{C}}(r)$ accurately expresses that hope is *triggered* (i.e., it does not necessarily express emotional experience), as is shown by the following property:

$$Hope_{\mathcal{C}}(r) \;\Rightarrow\; \mathcal{C} \models \mathbf{Hope}_i^{\mathbf{T}}(\mathcal{H}(r)) \quad (36)$$

It should be noted that, although we put r in $Hope_{\mathcal{C}}(r)$, the object of the triggered hope is actually the (sub)goal that can be achieved by applying the PG-rule (and executing its plan).

Finally, we have that 'fear' is triggered as soon the plan base contains two conflicting plans, in the sense that a postcondition of one plan contradicts the goal of the other plan. To express this, we define the following macro:

$$Fear_{\mathcal{C}}(\pi, \overline{\kappa}) \stackrel{\text{def}}{=} \neg Fut^+_{\mathcal{C}_{-1}}(\overline{\kappa}) \;\&\; \exists (\pi_1, \kappa_1, r_1), (\pi_2, \kappa_2, r_2) \in P : \overline{\kappa} \in PostCond_{\mathcal{C}}(r_2) \;\&\; \kappa \sqsubseteq \kappa_1 \;\&\; \pi = \pi_2 \quad (37)$$

Note again the use of the term $\neg Fut^+_{\mathcal{C}_{-1}}(\overline{\kappa})$ to ensure that what is expressed is a triggering condition. Thus $Fear_{\mathcal{C}}(\pi, \overline{\kappa})$ expresses that fear is *triggered* because plan π threatens (sub)goal κ (by promising $\overline{\kappa}$) of another plan. It should be noted that it is possible that $(\pi_1, \kappa_1, r_1) = (\pi_2, \kappa_2, r_2)$. This does not necessarily mean that π_1 is a 'bad' plan; it might simply be the case that π_1 is not guaranteed to succeed in achieving its goal. The fact that $Fear_{\mathcal{C}}(\pi, \overline{\kappa})$ accurately expresses that fear is triggered is shown by the following property:

$$Fear_{\mathcal{C}}(\pi, \overline{\kappa}) \;\Rightarrow\; \mathcal{C} \models \mathbf{Fear}_i^{\mathbf{T}}(\overline{\kappa}) \quad (38)$$

We now have all ingredients necessary to formally specify principled constraints on the deliberation cycle based on the emotion types 'hope', 'fear', 'joy', and 'distress'.

4. EMOTION-CONTROLLED DELIBERATION

In this section, we will specify emotion-inspired constraints in order to limit the choices in applying reasoning rules and executing plans. In psychological literature, affective feelings (including emotions) are often described as informing an individual about his or her performance. In particular, when one is task-oriented, positive emotions function as a "go" signal for pursuing currently accessible inclinations, whereas negative emotions function as a "stop" signal to allow for seeking alternatives [3]. This view of emotions as "go" and "stop" signals can be used to decide when to generate a new plan ("go ahead and do it"), when to revise a plan ("stop and reconsider"), which plan to choose for execution ("do what is making you feel good"), and when to stop a plan's execution ("stop when you don't feel good about what you're doing"). Indeed, we will now show how 'hope', 'fear', 'joy', and 'distress' can be used as the described signals, respectively, to constrain to deliberation cycle.

4.1 Hope

The generation of new plans is constrained by only allowing a PG-rule to be applied when hope is triggered with respect to the head of the rule. This is done by replacing Gen, as used in Definition 6, by Gen'. The transition rule for Gen' is specified as follows:

$$\frac{Hope_{\mathcal{C}}(r)}{\mathcal{C} \stackrel{Gen'(r)}{\Longrightarrow} \mathcal{C}'} \quad (39)$$

where $\mathcal{C}'$ is updated as in rule (8). Recall from formula (35) that $Hope_{\mathcal{C}}(r)$ includes $Applicable_{\mathcal{C}}(r)$, which was the condition for Gen used in rule (8). What the condition $Hope_{\mathcal{C}}(r)$ adds to the original (ad hoc) condition is that previously applicable PG-rules are not reconsidered for application.

4.2 Fear

The revision of existing plans is restricted by only allowing a PR-rule to be applied when fear is triggered with respect to a possible conflict between two plans, in the sense of formula (37). This is done by replacing Rev, as used in Definition 6, by Rev'. The transition rule for Rev' is specified as follows:

$$\frac{\exists \pi, \overline{\kappa} : Fear_{\mathcal{C}}(\pi, \overline{\kappa}) \;\&\; Applicable_{\mathcal{C}}(r, \pi)}{\mathcal{C} \stackrel{Rev'(r)}{\Longrightarrow} \mathcal{C}''} \quad (40)$$

where $\mathcal{C}''$ is updated as in rule (8). The condition that fear must have been triggered for a PR-rule to be applied is different from the condition used in 2APL [5]. In 2APL, the deliberation cycle only tries to find applicable PR-rules when an action of a plan has failed. In 2APL's precursor 3APL, PR-rules were applied whenever applicable, and applicable PR-rules were sought each time a single action had been executed. The constraint presented above takes a middle road by allowing threatening plans to be revised as soon as a possible conflict is perceived.

It may be interesting to note that it is not guaranteed that a revised plan is any less threatening. So after plan revision, a new fear may immediately be triggered, thus allowing for multiple successive revisions, until a non-threatening plan has been found.

4.3 Joy

The nondeterministic choice between which plan to execute can be constrained by specifying a preference on the plans in the plan base. Specifically, the "go" signal given by joy can make an agent prefer to execute plans that are going well, i.e., for which subgoals are being achieved. This is done by replacing $ExecutePlan$, as used in Definition 6, by $ExecutePlan'$. $ExecutePlan'$ is defined as follows:

$$ExecutePlan' \stackrel{\text{def}}{=} Do(\pi) \quad (41)$$

where $(\pi, \kappa, r) = \min_{\prec_{\mathcal{C}}} P$ (ties broken arbitrarily). The preference relation $\prec_{\mathcal{C}}$ is a strict partial order on plan base P of configu-

ration $\mathcal{C}$, defined as:

$$\prec_{\mathcal{C}} = \{\, (\pi,\kappa,r) \in P \mid Joy_{\mathcal{C}}(\kappa) \,\} \times \{\, (\pi,\kappa,r) \in P \mid \neg Joy_{\mathcal{C}}(\kappa) \,\} \tag{42}$$

where $Joy_{\mathcal{C}}(\kappa) \stackrel{\text{def}}{=} \exists \kappa' : \kappa' \sqsubseteq \kappa \;\&\; \mathcal{C} \models \mathbf{Joy^T}(\kappa')$. So $\prec_{\mathcal{C}}$ divides the plan base in two; those plans that are have made progress and those that have not. $ExecutePlan'$ then chooses the plan that has made the most progress. Obviously, this constraint still allows for nondeterminism, because $\prec_{\mathcal{C}}$ is only a partial order. But there are ways to extend this preference order on plans, for example by taking into account the number of subgoals having been achieved. Furthermore, $\prec_{\mathcal{C}}$ could be made to take into account goal achievements over a longer history than just the previous state.

4.4 Distress

The interleaving of the execution of plans is restricted by specifying that if a plan is executed, it is not interleaved unless it triggers distress. Thus distress will interrupt the agent's 'attention' from the current plan and allow it to consider, e.g., plan revision or executing another plan. This is accomplished by interleaving each plan with tests for distress. So Do, as used in Definition 6, is replaced by Do', which is defined as follows:

$$Do'(\alpha) \stackrel{\text{def}}{=} Do(\alpha) \tag{43}$$

$$Do'(\alpha;\pi) \stackrel{\text{def}}{=} Do(\alpha);(Distress_{\mathcal{C}}? + (\neg Distress_{\mathcal{C}}?; Do'(\pi))) \tag{44}$$

where $Distress_{\mathcal{C}} \stackrel{\text{def}}{=} \exists \overline{\kappa} : \mathcal{C} \models \mathbf{Distress^T}(\overline{\kappa})$. Observe that "if φ then π_1 else π_2" is a common abbreviation of $(\varphi?;\pi_1)+(\neg\varphi?;\pi_2)$. So $Do'(\pi)$ expands to π interleaved with $\neg Distress_{\mathcal{C}}?$ tests. The triggering of distress then effectively functions as a "stop" signal with respect to the execution of a plan.

4.5 Discussion

To summarize, we have constrained the deliberation cycle such that (1) PG-rules are only applied to goals that have triggered hope, (2) PR-rules are only applied to plans that have triggered fear, (3) plans that have triggered joy are preferred for execution, and (4) plan execution is interrupted as soon as distress is triggered. It will be clear that these constraints do not resolve all nondeterminism in the specification of $Reason\_Act$ in Definition 6. Specifically, the following nondeterminism remains:

- When different PG-rules can be applied (in the sense of rule (39)), it is not specified how many of them are to be applied and in which order.
- When different PR-rules can be applied (in the sense of rule (40)), it is not specified how many of them are to be applied and in which order.
- The order $\prec_{\mathcal{C}}$ imposed on the plan base by joy may not result in a unique most preferred plan. As noted before $\prec_{\mathcal{C}}$ can be extended to reduce nondeterministic choices.
- The triggering of distress interrupts the execution of a plan, but it is not specified what should happen after that. The current construction makes it possible for the agent to switch to another plan, but it is also allowed to continue with the same plan.
- Although $ApplyPGrule^*; ApplyPRrule^*; ExecutePlan^*$ in Definition 6 appears to suggest the order "first apply PG-rules, then apply PR-rules, then execute plans," the use of * effectively allows any order for performing $ApplyPGrule$, $ApplyPRrule$, and $ExecutePlan$.
- It is not specified how sensing of the environment is interleaved with reasoning and acting.

For future work, we will investigate principled ways to resolve this remaining nondeterminism. In particular, we will study other types of emotions for their suitability to control agent deliberation.

5. RELATED WORK

In this section we compare the presented work with related approaches at using emotions in reasoning.

Dastani & Meyer [6] have proposed to constrain the deliberation cycle of 2APL using heuristics inspired by the emotions happiness, sadness, fear, and anger, as described in the psychological model of emotions of Oatley & Jenkins [14]. These heuristics were then added on top of the deliberation cycle of 2APL. However, the existing 2APL deliberation cycle itself, which is based on ad hoc choices, was not changed in a principled way (i.e., the emotion-inspired heuristics were only used to *extend* the deliberation cycle). In contrast, our approach starts with a clean slate by assuming complete nondeterminism. Then we have added several constraints in accordance with the common psychological view of emotions as "go" and "stop" signals in task-oriented situations. Note that it is our intention that any remaining nondeterminism be resolved by investigating and adding more principled (possibly emotion-based) constraints.

Steunebrink *et al.* [19] have investigated how the options of an agent can be ordered and limited using the notion of action tendency. In psychological literature, action tendency is used to describe states of readiness to execute a given kind of action to achieve a certain end state [8]. For example, anger will subside once the removal of the obstruction that caused the anger is perceived; so anger will give one the tendency to perform actions that remove the obstruction. Steunebrink *et al.* formalized this psychological notion of action tendency in a dynamic doxastic logic such that the possible actions of an agents could be ordered by action tendency, thereby reducing possible nondeterminism in action selection. However, it was not shown how action tendencies could be used in actual agent implementations (only a logical analysis of the concept was provided). Moreover, Steunebrink *et al.* [19] only considered negative emotions.

Adam [1] proposed a formalization of the OCC model in which also the effects of emotions on behavior were considered. In particular, seven coping strategies were defined for several negative event-based emotions (distress, disappointment, fear, fears-confirmed, pity, and resentment). Some coping strategies (e.g., denial, resign) change the beliefs or desires of an agent such that the triggering conditions for the negative emotion cease to hold. Other coping strategies (e.g., mental disengagement, venting) lead to the adoption of intentions to bring about new positive emotions that "divert the individual from the current negative one" [1]. However, as in [19], only a logical analysis was provided and only several negative emotions were considered.

Gratch & Marsella [9] have been working on a computational framework for modeling emotions inspired by the OCC model, among others. An implementation, named EMA, is used for social training applications. Like Adam, their framework incorporates a number of coping strategies. However, few formal details of how emotions affect the reasoning of an agent are provided, so it is hard to assess how much of the behavior of an EMA agent results from ad hoc choices or emotion-inspired principles.

6. CONCLUSION AND FUTURE WORK

In this paper we have shown how a completely nondeterministic sense–reason–act cycle can be constrained in a principled way by using emotions. This was done by first formalizing the eliciting conditions of four emotions types (hope, fear, joy, and distress) from the psychological OCC model of human emotions. This formalization was then grounded in 2APL, such that one could reason about when emotions are triggered for a 2APL agent. The conditions giving rise to hope, fear, joy, or distress were then used as conditions for applying reasoning rules and determining which plan to execute and when to interrupt execution.

It should be noted that in the presented approach, emotions are never stored in the configuration of an agent. They only function as labels of special situations such that they can be used to guide design principles. So the constraints presented in this paper could in principle be specified without referring to emotions. However, because these constraints give rise to properties that are in line with psychological models of human emotions, it makes good sense to use emotional labels.

In this paper we have focused on just four emotion types. In future work, we will investigate how remaining nondeterminism can be further reduced by introducing more emotions.

7. REFERENCES

[1] C. Adam. *The Emotions: From Psychological Theories to Logical Formalization and Implementation in a BDI Agent*. PhD thesis, Institut Nat. Polytechnique de Toulouse, 2007.

[2] R. H. Bordini, M. Wooldridge, and J. F. Hübner. *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology. John Wiley & Sons, 2007.

[3] G. L. Clore, R. S. Wyer, B. Dienes, K. Gasper, C. Gohm, and L. Isbell. Affective feelings as feedback: Some cognitive consequences. In L. L. Martin and G. L. Clore, editors, *Theories of Mood and Cognition: A User's Guidebook*, chapter 2, pages 27–62. Lawrence Erlbaum Associates, 2001.

[4] A. R. Damasio. *Descartes' Error: Emotion, Reason and the Human Brain*. Grosset/Putnam, New York, 1994.

[5] M. Dastani. 2APL: A practical agent programming language. *International Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, 16(3):214–248, 2008.

[6] M. Dastani and J.-J. C. Meyer. Programming agents with emotions. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI'06)*, pages 215–219, 2006.

[7] P. Ekman and R. J. Davidson, editors. *The Nature of Emotion: Fundamental Questions*. Series in Affective Science. Oxford University Press, 1994.

[8] N. H. Frijda. *The Emotions*. Studies in Emotion and Social Interaction. Cambridge University Press, 1987.

[9] J. Gratch and S. Marsella. A domain-independent framework for modeling emotions. *Journal of Cognitive Systems Research*, 5(4):269–306, 2004.

[10] J. J. Gross and R. A. Thompson. Emotion regulation: Conceptual foundations. In J. J. Gross, editor, *Handbook of Emotion Regulation*. Guilford Press, 2007.

[11] K. Hindriks. Modules as policy-based intentions: Modular agent programming in GOAL. In *Proceedings of ProMAS'07*, volume 4908. Springer, 2008.

[12] R. S. Lazarus. *Emotion and Adaptation*. Oxford University Press, 1994.

[13] J. E. LeDoux. *The Emotional Brain: Mysterious Underpinnings of Emotional Life*. Simon & Schuster, 1996.

[14] K. Oatley and J. M. Jenkins. *Understanding Emotions*. Blackwell Publishing, Oxford, UK, 1996.

[15] A. Ortony and G. L. Clore, April–June 2009. Personal communication.

[16] A. Ortony, G. L. Clore, and A. Collins. *The Cognitive Structure of Emotions*. Cambridge University Press, Cambridge, UK, 1988.

[17] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: A BDI reasoning engine. In R. H. Bordini, M. Dastani, J. Dix, and A. E. F. Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*, pages 149–174. Springer, 2005.

[18] K. R. Scherer, A. Schorr, and T. Johnstone, editors. *Appraisal Processes in Emotion: Theory, Methods, Research*. Series in Affective Science. Oxford University Press, 2001.

[19] B. R. Steunebrink, M. Dastani, and J.-J. C. Meyer. A formal model of emotion-based action tendency for intelligent agents. In *Proceedings of the 14th Portuguese Conference on Artificial Intelligence (EPIA'09)*. Springer, 2009.

[20] M. Winikoff. JACK™ intelligent agents: An industrial strength platform. In R. H. Bordini, M. Dastani, J. Dix, and A. E. F. Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*, pages 175–193. Springer, 2005.

APPENDIX

A. PROOFS

PROOF (26)–(29). According to Definition 7, $\mathbf{Hope}_i^{\mathbf{T}}(\kappa)$ is equivalent to $\mathbf{Des}_i(\kappa) \wedge (\mathbf{Future}_i(\kappa) \vee \mathbf{Uncertain}_i(\kappa))$. But $\mathbf{Uncertain}_i(\kappa)$ implies $\neg\mathbf{B}_i\kappa \wedge \neg\mathbf{B}_i\neg\kappa$, which is a contradiction because of our closed world assumption. So $\mathbf{Hope}_i^{\mathbf{T}}(\kappa)$ is equivalent to $\mathbf{Des}_i(\kappa) \wedge \mathbf{New}\,(\mathbf{B}_i\neg\kappa \wedge \mathbf{B}_i\mathbf{Fut}\,\kappa)$. The same reasoning holds for $\mathbf{Fear}_i^{\mathbf{T}}(\overline{\kappa})$. The properties for $\mathbf{Joy}_i^{\mathbf{T}}(\kappa)$ and $\mathbf{Distress}_i^{\mathbf{T}}(\overline{\kappa})$ are nothing but the macros of Definition 7 expanded. □

PROOF (30)–(33). These properties are direct rewrites of (26)–(29) (see above). It should be noted that the term $\mathbf{New}\,(\mathbf{B}_i\neg\xi \wedge \mathbf{B}_i\mathbf{Fut}\,\xi)$ in property (26) and (27) expands to $\mathbf{B}_i\neg\xi \wedge \mathbf{B}_i\mathbf{Fut}\,\xi \wedge \mathbf{Prev}\,\neg(\mathbf{B}_i\neg\xi \wedge \mathbf{B}_i\mathbf{Fut}\,\xi)$. Interpreting this formula in a configuration $\mathcal{C}$ yields $\mathcal{C} \models \mathbf{B}_i\neg\xi \wedge \mathbf{B}_i\mathbf{Fut}\,\xi$ and $\mathcal{C} \models \mathbf{Prev}\,\neg(\mathbf{B}_i\neg\xi \wedge \mathbf{B}_i\mathbf{Fut}\,\xi)$, i.e., $\mathcal{C}_{-1} \not\models \mathbf{B}_i\neg\xi \wedge \mathbf{B}_i\mathbf{Fut}\,\xi$. By formula (34) this is equivalent to $Fut^+_{\mathcal{C}}(\kappa)$ & $\neg Fut^+_{\mathcal{C}_{-1}}(\kappa)$. □

PROOF (36). To obtain this property it suffices to show that $Applicable_{\mathcal{C}}(\kappa \mid \beta \rightarrow \pi)$ implies $Fut^+_{\mathcal{C}}(\kappa)$ and $G \models_g \kappa$. $Applicable_{\mathcal{C}}(\kappa \mid \beta \rightarrow \pi)$ expands to $(\kappa \mid \beta \rightarrow \pi) \in PG$ & $G \models_g \kappa$ & $B \models_b (\beta \wedge \neg\kappa)$, while $Fut^+_{\mathcal{C}}(\kappa)$ expands to $B \models_b \neg\kappa$ & $\exists r \in PG : \kappa \sqsubseteq \mathcal{H}(r)$ & $B \models_b \mathcal{G}(r)$. Assume $Applicable_{\mathcal{C}}(\kappa \mid \beta \rightarrow \pi)$. Then immediately we have $G \models_g \kappa$ and $B \models_b \neg\kappa$. But because $(\kappa \mid \beta \rightarrow \pi) \in PG$ and $\kappa \sqsubseteq \kappa$, we have that $\exists r \in PG : \kappa \sqsubseteq \mathcal{H}(r)$. Moreover, $B \models_b \mathcal{G}(r)$ is now the same as $B \models_b \beta$, which we have by assumption. So indeed $Applicable_{\mathcal{C}}(\kappa \mid \beta \rightarrow \pi)$ implies $Fut^+_{\mathcal{C}}(\kappa)$ and $G \models_g \kappa$. □

PROOF (38). To obtain this property it suffices to show that $\exists(\pi_1, \kappa_1, r_1), (\pi_2, \kappa_2, r_2) \in P : \overline{\kappa} \in PostCond_{\mathcal{C}}(r_2)$ & $\kappa \sqsubseteq \kappa_1$ implies $Fut^+_{\mathcal{C}}(\overline{\kappa})$ and $G \models_g \kappa$. Assume the antecedent above. Then $\exists(\pi, \kappa', r) \in P : \overline{\kappa} \in PostCond_{\mathcal{C}}(r)$, which is equivalent to $Fut^+_{\mathcal{C}}(\overline{\kappa})$. It was assumed (see below Definition 4) that $(\pi_1, \kappa_1, r_1) \in P$ implies $G \models_g \kappa_1$. Because $\kappa \sqsubseteq \kappa_1$, also $G \models_g \kappa$. So indeed $\exists(\pi_1, \kappa_1, r_1), (\pi_2, \kappa_2, r_2) \in P : \overline{\kappa} \in PostCond_{\mathcal{C}}(r_2)$ & $\kappa \sqsubseteq \kappa_1$ implies $Fut^+_{\mathcal{C}}(\overline{\kappa})$ and $G \models_g \kappa$. □