Extending 2APL with functionality for emotions, moods and coping

by

Christiaan Floor

20 december, 2012

ICA – 3690792

Master of Science Thesis

Department of Information and Computing Sciences
Utrecht University
Utrecht, the Netherlands

Supervisors:
dr. M.M. Dastani
prof. dr. J.J.C. Meyer

Abstract

Emotions and moods play a crucial role in simulation of human-like behavior and rational decision making. For these reasons various frameworks have been developed that simulate emotion and mood generation. In some cases coping with those emotions is also incorporated in the framework. Something these frameworks have in common is that they lack a sufficient logical foundation. On the other hand we have the agent oriented programming language *2APL: a practical agent programming language* developed at Utrecht University. This language is supported by a thorough logical foundation and has been used widespread to develop multi-agent systems for all kind of purposes. In this thesis I will connect one of the frameworks that simulates emotion and mood generation to 2APL such that we have a computational model that is supported by a logical foundation, supports emotion and mood generation and is used to develop real world multi-agent systems. We will first delve into the various frameworks for emotion and mood generation and then we will see how emotion and mood generation together with a coping mechanism are incorporated in 2APL.

Contents

1. Introduction

When we talk about Artificial intelligence and agent technology there are a lot of possible applications. Autonomous systems that operate in an environment without humans intervening are not only the subjects of research and studies anymore, they are increasingly used in real world applications. In a lot of situations it is desired that an agent behaves as a human being. The last decade the focus in the area of artifical intelligence switched from behavior that is formed by pure rationality to behavior that is influenced by emotions as well. It is not only the human-like behavior that is desired, emotions can also improve the process of decision-making. The latter is especially interesting for computer scientists. As stated in [11]:

"*We are interested in agents with emotions, since we believe that emotions are important heuristics that can be used to define an effective and efficient decision-making process.*"

It was assumed that a human who did not experience emotions would be purely rational and would have better decision making capabilities with respect to humans that would experience emotions. But after the research of Damasio [8] it was proven that humans that could not experience emotions due to a physical defect in the brain performed very poorly in tests where they had to make decisions. Because of the lack of emotions they could not decide where they should focus on and make a distinction between what is important and what is not.

A problem in the area of emotions and computer science is that there is relatively little consensus about the definition of emotions, moods and coping with emotions. There are a lot of different opinions and views about what emotions and moods exactly are and what their function is. Consensus and unambiguity is something that is necessary if we want to make a computational model that incorporates emotions and moods.

Over the last couple of decades however various frameworks have been proposed [1,2,4] that claim to generate emotions and moods in a way that approximates how it works in human beings. I will look at three of these frameworks and compare them. In these three frameworks there is mutual consensus about what emotions and moods are and some of the frameworks also cover coping with emotions as we will see later in this thesis.

Motivation

The frameworks that I will treat are described in detail in multiple papers (ALMA [1,7,19], Ema [2,3,6,22,23] and Cathexis [4,20,21]). What I want to do is incorporate parts of these frameworks into a computational model, such that we can develop (multi-)agent software in which emotions, moods (and coping with those emotions) are incorporated. If we want to use emotions and moods in a computational model it should also be supported by a thorough logical foundation. Without that the behavior of a framework can not be specified in an umambiguous manner. And that would be highly undesireable if I want to incorporate a framework within a computational model. Note that some of the frameworks are also supported by a logical foundation, but this only takes into account (parts of) the generation of emotions and moods and not a completely working agent.

At Utrecht University the agent oriented language 2APL* (A Practical Agent Programming Language) [13] has been developed (and it is still being in development). This language is developed with the BDI-agent structure (Belief, Desire, Intention) [14,15,16] in mind. It is supported by a logical foundation [12] and has been used in various real world agent applications (e.g. auctions, coöperative problem solving tasks and negotiation mechanisms). In this language a clear separation is established between multi-agent and individual agent programming constructs. Multiple individual agents can be active within one or more external environments. The individual agents are programmed according to goals, beliefs, actions, plans, events and declarative rules. Based on these rules the internals of the agent can be influenced and plans for the agent can be generated. These plans are constructed with an imperative programming style. The external evironments are also implemented with an imperative programming style (the object-oriented language Java). So in this language declarative styled programming and imperative styled programming coincide. Note that 2APL is a programming language on its own and is not based on another programming language such as some other agent programming languages (Jack[24], Jadex[25] which are based on Java). So the language has been developed with the purpose of developing multi-agent systems from the beginning. Because of this I think the language feels more intuitive for developing multi-agent systems than agent programming languages which are based on other programming languages.

In this thesis I would like to connect 2APL to one of the three frameworks (ALMA, EMA, Cathexis) that I have mentioned earlier. 2APL is a programming language that is supported by a thorough logical foundation and thus can be interpreted in an unambiguous manner. This also makes formal program verification of 2APL programs possible. If I am able to incorporate (one of) the frameworks in 2APL we will have a computational model in which emotions and moods are incorporated with which we can develop (multi-) agent programs and it will also be supported by the (extended) logical foundation of 2APL.

Outline

In the remainder of this thesis the outline will be as follows. I will start by describing the problem at hand in detail in chapter 2. In chapter 3 I will explain how I approached the project and which tools I used. Chapter 4 will briefly discuss the theory for generating emotions and moods on which the three frameworks are based. Chapter 5 will give a detailed description for each framework and chapter 6 will compare the frameworks based on multiple questions. Chapter 7 will describe how I implemented appraisal, emotion generation and a coping mechanism in the 2APL programming language. In chapter 8 I will extend the logical foundation 2APL is based on such that it also incorporates the extensions made to the language to support emotion and mood generation and also coping with the generated emotions. Chapter 9 is rather technical and will discuss parts of the code that I have added to 2APL (Java knowledge required). Chapter 10 will describe the demonstration application I have developed for presenting the abilities of the extended 2APL language. Chapter 11 will describe what possible future work can be done and it will present my conclusions about this project.

* Also the language 3APL was developed, but despite your intuition do not assume that 2APL is its predecessor. It is the other way around, 2APL is an extended and modified version of 3APL. Whereas 3APL focused on a single agent, 2APL focuses on multi-agent systems.

2. Problem statement

Here at Utrecht University the agent-oriented programming language 2APL has been developed and used for implementing multi-agent systems. This language is supported by a logical foundation and we are able to implement agents with it which are based on the BDI-agent architecture [14,15,16]. Actually the language 2APL strongly relates to the BDI agent architecture where we have components like a belief base, goal base and plan base. In the (syntax of the) language we can clearly see those components, where plans are generated and adjusted according to declarative rules. In 2APL we have three types of these rules, those are Planning Goal (PG) rules, Plan Repair (PR) rules and Procedure Call (PC) rules. PG-rules are used to generate plans for achieving a specified goal, PR-rules are used for repairing a failed plan and PC-rules are used to generate plans based on abstract actions or incoming events from the external environment. In this language imperative programming coincides with declarative programming i.e., with imperative programming it is exactly specified what should happen step by step and with declarative programming it is specified which desired state should be reached, but it is not specified how this state should be reached. As mentioned above the PG, PC and PR-rules are declarative and the plans generated by those rules are imperative. For a detailed explanation of the language I refer to [12,13].

On the other hand we have three frameworks (ALMA[1,7,19], EMA[2,3,6,22,23] and Cathexis[4,20,21]) to simulate emotion and mood generation. But these frameworks are not supported by a ground logical system that covers functional agents i.e., some of the frameworks do have a logical foundation but it only covers emotion and mood generation and not a completely functional agent. If we want to incorporate a framework in a computational model it is necessary that we can reason about it in an unambigious manner. Another problem is that we do not rely on BDI-agent architectures in these frameworks, which is something we do rely on when working with 2APL.

What I would like to do is to take both sides i.e., the programming language (a new verion of 2APL) that is supported by a logical foundation and is specifically designed for developing multi-agent systems consisting of agents with the BDI-architecture on the one hand and on the other hand the three frameworks to implement emotional agent systems, and connect these two ends together. If we would succeed in doing this then we would have a complete chain from a logical system for multi-agent systems to an actual working emotional agent system. To say it more formally, our main problem is:

Connect the agent oriented programming language 2APL to (one of) the three frameworks to implement emotional agent systems.

We can split the main problem up into multiple subproblems in a straightforward manner:

- *Research how the ALMA framework works.*
  - ALMA is one of the three frameworks for emotion and mood generation that I am going to delve into in chapter 5. As you will see later ALMA will play a very important role in the project described in this thesis.

- *Research how the EMA framework works.*
  - EMA is by far the most advanced framework of the three. This will also be described in chapter 5. It is characterized by an advanced planning process and is the only framework

that incorporates coping with emotions. It is based on a large collection of theories and tries to let them all work together in one system.

- *Research how the framework Cathexis works.*
  - Cathexis is the oldest framework of the three. Although it has a formal specification it leaves a lot of holes to fill in for the user of the framework as we will see later. Although this framework is a decade older than the other two frameworks it was a good attempt at creating a framework that would simulate human emotion and mood generation.

- *Incorporate an emotional system in 2APL.*
  - With an emotional system I mean functionality that enables programmers of 2APL to use emotions and moods in their programs. Also a coping mechanism will be incorporated in the language. The extension can be split up into three parts. Those are the appraisal process, the emotion and mood generation process and the coping process. More will be explained about these three processes in the remainder of this thesis. First I will incorporate the appraisal process, followed by the emotion and mood generation process and finally I will incorporate the coping process.

3. Methodology

To develop the product of this project (a new version of 2APL in which emotions, moods and a coping process are incorporated) I have to do a lot of research and reading before I can actually implement the new functionality. The main stages of the project are doing research/reading, implementing, testing and demonstrating the new features.

Before I incorporate emotions, moods and coping into 2APL I have to get some inspiration. Therefore I read a lot of papers about emotion and mood generation and also about coping with emotions. But the focus of the first stage of the project are three frameworks for generating emotions and moods. These frameworks are:

- ALMA (A Layered Model of Affect)
- EMA
- Cathexis

When I have read the papers about the frameworks ALMA [1,7,19], Ema [2,3,6,22,23] and Cathexis [4,20,21] I will look at the similarities and differences and which framework(s) is (are) most suited for extending 2APL. First of all the theory the frameworks are based on is described in chapter 4. In Chapter 5 I will give a detailed description of all three frameworks. And in chapter 6 I will compare the frameworks based on seven questions. The questions I ask for each framework are the following:

- *Which affective phenomena are distinguished?*
  - In the theory about emotions, moods and coping with emotions various affective phenomena are distinguished e.g., an emotion is a different phenomena than a mood. More about this will be explained in chapter 4.
- *How are emotions generated?*
  - How emotions are generated in each framework is accomplished in different ways. So for each framework I will describe this process.
- *How do emotions relate to moods?*
  - In each framework there is a certain relation between emotions and moods i.e., how do emotions influence the mood of an agent (and vice versa).
- *How do emotions decay?*
  - Once an emotion is generated, it will not last forever. Over time the emotion will decay. For each framework I will explain how this decay of emotions work.
- *How do emotions and moods influence behavior?*
  - Emotions and moods are all internal phenomena. But what effect do these phenomena have on the behavior of the agent. This will be described for each framework in this part.
- *Does the framework have a coping mechanism?*
  - For each framework I will explain if it has a coping mechanism (more about this in the following chapters). And if the framework has such a mechanism, how does it work?
- *What has the user to provide for the framework?*
  - For each framework the user has to provide certain data. What data that is will be

explained in this part.

After I compare the frameworks based on the questions above I will present a small overview of the comparison in a table.

At the point where I complete the comparison I have to decide which parts of which frameworks I am going to use in the remainder of the project. What I have to do is extending 2APL with functionality for emotions and moods. This is actually splitted up into three parts (more about this in chapter 4):

- the appraisal process
- emotion and mood generation
- the coping process

I have to decide if I am going to use only one framework as inspiration or multiple frameworks. Because there are three distinct processes that have to be implemented in 2APL I could also decide to use a different frameworks as inspiration for each process. In chapter 5 I will explain which framework(s) I have chosen as inspiration and why I have made that decision.

Before I can adjust 2APL I have to study the source code of the language which has been coded in the programming language Java. After I got myself familiar with the language I can extend the language first with the appraisal process which will be implemented in the form of E-rules (more about this later). After that I can extend the language with the generation of emotions and moods (chapter 7) and finally the coping process can be implemented (chapter 7).

At some point during the project a new version of 2APL will be released (by Marc van Zee, 2012). In this version some optimalisations were done. So I have to migrate the functionality I implemented in the old version of 2APL to the new version. I expect that I can do this in a rather straightforward manner.

To demonstrate the new functionality of the new 2APL version I will create a demonstration application in 2APL. This demonstration application can also be used for test purposes during the project to validate if the new functionalities work correctly. So I will start developing this application in the early stages of the project and extend it along the way. The resulting demo application will be described in detail in chapter 10.

4. The theory behind the frameworks

The frameworks that I will explain in the next section differ in a lot of areas, but the theory on which they are based actually overlaps for a great part. Between those frameworks there is consensus (unlike the literature about the subject in general) about what emotions, moods and personality are and we can see a clear distinction of the emotion and mood generation process into three parts which will be explained in this section. We will go into as much detail as necessary to lay a foundation for the material to follow it. Keep in mind that this thesis is not about the psychological aspect of emotions and moods, but some knowledge about the subject is necessary before we dive into the details about the frameworks.

Emotions and moods

What are emotions and moods exactly and what is the difference between them? That is the question that I will try to answer here before we go into detail about the frameworks. There are a lot of different opinions in the literature about the subject but I will take the same point of view that the creators of the frameworks have taken. But keep in mind that there is not only one correct answer to this question.

Emotions are feelings that an individual can experience such as joy, anger and sadness. They are caused by some event, action or object which first will be appraised by the individual and based on this appraisal an emotion occurs. Emotions usually cover a short timespan and over time fade away. Moods on the other hand cover a longer timespan. And where we can connect emotions to a single action, event or object, you should see a mood as a general state of the individual. The mood can be influenced by occurences of emotions. But whereas an indivual can have a lot of different emotions in a minute (or even multiple emotions simultaneously), the mood is more stable and will change more slowly. It also depends on the mood how an individual will process the emotions that occur.

For example lets say that we have an individual that comes home after a good day at work, where a lot of good things happened for him: he got a raise and his boss gave him a very good review. Because of these events happening, emotions of joy were generated and those emotions influenced his mood. Therefore he is in a good mood. When he arrives home from work he steps out of his car and he steps into dog poo, this causes an emotion of anger within the individual. But because he is in a good mood, the emotion of anger will only result in a slightly less good mood. Nonetheless his overall mood is still good and he will not react very intense to the event.

But now lets assume that the same individual had a very bad day at work: he did not make his deadlines and his boss got angry towards him and even threatened to fire him. The mood of the individual got very bad because of these events. And on top of that when he comes home, he steps in the dog poo. It is the same event as in the other example and it will cause the same emotion of anger. But this time, because the mood of the individual was already bad and it is influenced by the emotion of anger, which results in a mood that is even more bad, the individual will cope differently with the event. He will scream at his dog and tell his wife she does not take care of the house.

So in the example we could see the distinction between emotions and mood. But there is a third affective phenomenom and that is personality. If we say that emotions cover a short timespan, moods a medium timespan, then personality covers a long timespan. Personality defines the character of an individual and is very hard to change. Later we will see what role these three affective phenomena, emotions, mood and personality will play in the frameworks.

Appraisal, emotions and moods

Basically when we look at emotions and moods including the consequences of these emotions and moods, the process can be divided in three subparts. The first one is the appraisal process, the second one the emotion and mood generation process (emotions influence the mood) and finally the coping process.

The appraisal is done with respect to some event, action or object. In general such an event, action or object is appraised as good or bad by the individual. The appraisal will result in some emotion and this emotion will influence the overall mood of the individual.

Note that we have a clear distinction between the appraisal and the generation of emotions. In the gross of the literature there is no distinction between appraisal and the generation of emotions. In the frameworks that we will discuss however, the authors do make this distinction. First some event, action or object is appraised. And each appraisal will be converted to an emotion in a specific way (which will be explained in detail for every framework).

Coping

Coping is about how an individual will respond to arisen emotions such that he will come in a desired/preferred emotional state e.g., an individual could feel an emotion of anger and as a coping response start screaming which will relief him of his anger. The coping part is not incorporated in all three frameworks. ALMA and Cathexis do not have this part included, but EMA does cover coping, though the authors admit that it is an unfinished part of the framework. We will see later how coping is incorporated in the EMA framework.

Note that coping is not merely the expression of emotions, it is about an individual intentionally trying to influence the appraisal process such that he will come in a desired emotional state or stay in the current emotional state if it is the state he wants to be in. Merely the expression of emotions is also covered in ALMA and Cathexis but we will not refer to this as coping in this thesis.

The OCC model

The OCC model (Orthony, Clore and Collins) is a popular model of emotions that is used in the frameworks ALMA and EMA, though it might be modified or extended versions of the model where 24 emotions instead of 22 emotions are used. EMA uses a model of Clark Elliott [3] which is based on the OCC model and ALMA uses a modified version of the OCC model together with the Big Five personality model [9,10] (more about this later). Because the OCC model is a much used model in the area of emotions and computing science I will explain it here in detail.

Emotions in the OCC model are always related to (the appraisal of) the consequences of an event, actions of agents or aspects of objects and it does not describe some overall emotional state of an individual. Note that I say the consequences of an event. An event itself will not be appraised by an individual, it are the consequences of an event that are appraised e.g., when we have an event *it is starting to rain*, an individual will probably not dislike the event itself, but the consequence of the event: becoming wet. With the term event in this context is meant: "*people's construals about things that happen, considered independently of any beliefs they may have about actual or possible causes*" [5].

The consquences of an event can be appraised as desirable or undesirable, actions of an agent as praiseworthy or blameworthy and aspects of objects can be appraised as appealing or unappealing. We call Praiseworthiness, Desirability and Appealingness (PDA) and its negative counterparts Emotion Eliciting Conditions (EECs). Note that the consquences of events, actions or objects are appraised with the internals of the individual in mind. In the case of the events those are his goals, in the case of actions his standards and in the case of objects his attitudes (tastes etc.). An overview of this can be seen in table 1 [5].

Type of percept	**Evaluated against**	**Central variable (positive/negative)**
Consequence of event	Goals	Desirability/Undesirability
Action of agent	Standards	Praiseworthiness/Blameworthiness
Aspect of object	Attitudes	Appealingness/Unappealingness

Table 1. The types of aspects of a situation that can be appraised

In the OCC model 22 types of emotion are distinguished which are listed in table 2. For each emotion a specification is given, described by five fields. Those fields are a label, a type specification, a list of tokens, the emotion eliciting conditions and an example. I will explain these fields in detail one by one. In figure 1 we can see an example of such a specification. This one is the specification for the emotion type distress.

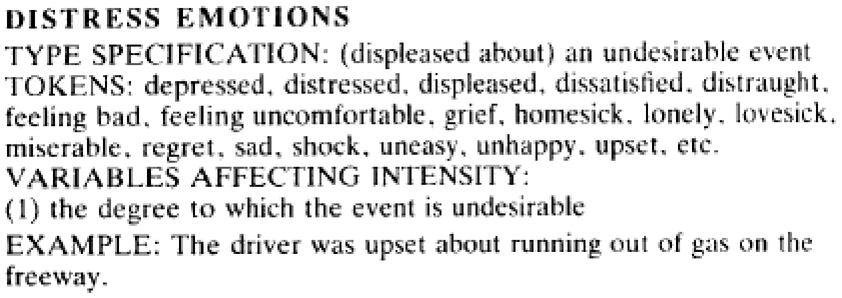

DISTRESS EMOTIONS
TYPE SPECIFICATION: (displeased about) an undesirable event
TOKENS: depressed, distressed, displeased, dissatisfied, distraught, feeling bad, feeling uncomfortable, grief, homesick, lonely, lovesick, miserable, regret, sad, shock, uneasy, unhappy, upset, etc.
VARIABLES AFFECTING INTENSITY:
(1) the degree to which the event is undesirable
EXAMPLE: The driver was upset about running out of gas on the freeway.

Figure 1. The specification of emotion type distress

Each emotion type is described with one word which is in the label field. There are several words that can describe an emotional state. But for each type a word is chosen that is "*the most free of connotations and average in typical intensity*" [5].

The type specification field describes the EECs of an emotion type. We could actually see the label as an abbreviation of these EECs i.e., what makes the emotion type described arise.

The list of tokens gives us all words that describe the emotion type at hand, though they can describe it at different intensities e.g., unhappy describes an emotion of distress at a lower intensity than the word sad. So all these tokens describe the eliciting conditions, but note that some words describe more than that. The token could also tell us something about how someone copes with the emotion. Grief for example does not only tell us that an individual is displeased about an undesirable event, but also (to some extent) the way how this individual copes with it.

The next field lists the variables affecting the intensity of the emotion. In the case of the example we see that the degree to which the event is undesirable influences the intensity of the emotion. The more undesirable the event is, the higher the intensity of distress regarding the event becomes. It should be noted that some variables only refer to specific emotions and other variables are global i.e, they influence the intensity of all emotions. An example of such a global variable that influences the intensity of all emotions is arousal. In the field described here only the variables that affect the intensity of the current emotion are listed (and not the global ones).

Finally we have the example field. Here just a simple example is given of a situation where the emotion type might arise in the form of a sentence.

The 22 emotion types in the OCC model			
Joy	Distress	Happy-for	Pity
Gloating	Resentment	Hope	Fear
Satisfaction	Fears-confirmed	Relief	Disappointment
Pride	Shame	Admiration	Reproach
Gratification	Remorse	Gratitude	Anger
Love	Hate		

Table 2. The 22 emotion types distinguished in the OCC model

As you can see some emotional types do not refer to the events or actions that relate to the individual itself, but to others. An example of such an emotional type is admiration. This type of emotion is elicited if an action that another agent performs is appraised as praiseworthy by the individual.

The overall structure of the emotion types in the OCC model is shown in figure 2. At the top line we can see that all emotions are classified as valenced reactions to some (consequence of an) event, action or object. On the second line we can see this distinction between events, actions and objects. Then we see that something is appraised e.g., an individual can be pleased about the consequences of some event. And in the next line we see a separation between (consequences of) events for the individual itself and for other agents. This separation also holds for actions. When we look for examples at the consequences for the other agent because of some event that happened we arrive in the FORTUNES-OF-OTHERS box. So when the consequences are desirable for the other, the individual itself could get the emotion happy-for (other).

If we look at the consequences for the individual itself, the diagram is splitted up in PROSPECTS RELEVANT and PROSPECTS IRRELEVANT. As you can see the latter will lead to the emotion types joy and distress i.e., here the prospects do not matter, it is the current state of affairs that matters. Thus certainty is involved. The other possibility is that prospects are relevant, and prospects always coïncide with uncertainty. This will lead to the emotion types hope or fear. At the point where the prospects are confirmed or disconfirmed, the uncertainty aspect has vanished and we get to emotion types like satisfaction or fears-confirmed.

When we look at the WELL-BEING box and the ATTRIBUTION box, residing in the event and action branch respectively, we see that these two branches converge to the WELL-BEING/ATTRIBUTION COMPOUNDS box. In this box the group of emotion types gratification, remorse, gratitude and anger are located. The convergence of the two branches is done because these emotions refer to an action happening resulting in an event with consequences. But instead of simply the coöccurrence of two emotions in that case (one for the action and one for the consequences of the event) we get another resulting emotion type. And this emotion type is more than the sum of its constituents e.g., anger is more than merely the coöccurrence of distress and reproach. The remainder of the diagram should be self explanatory. And with this information I think you know enough about the OCC model with respect to this project. In the next chapter we will see what role this model has played in the frameworks.

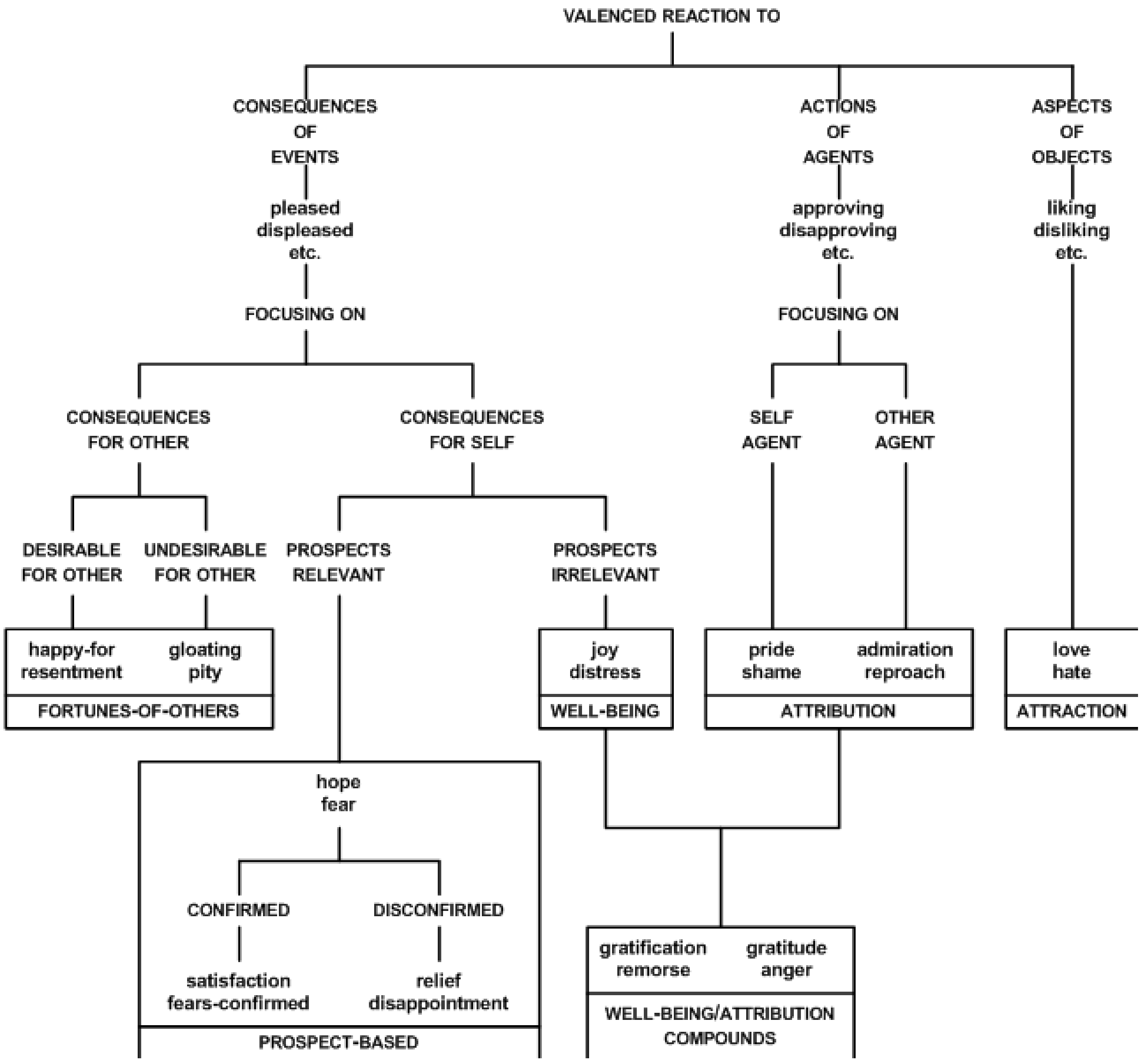


Figure 2. The overall structure of the emotion types in the OCC model

5. The Frameworks

On the one hand we have the agent oriented programming language 2APL in which we can program agents according to the BDI-architecture and on the other hand we have frameworks in which we can simulate the generation of emotions and moods. Emotions in humans are visible through body posture, facial expressions and vocal expressions and influence their behavior and decision making. But how can we accomplish this for agents? The various frameworks that we will describe here do that in different ways, but in general there are also some similarities between the frameworks.

The frameworks that will be discussed in this section are ALMA (A Layered Model of Affect) which is developed by Gebhard [1,7,19], EMA (Emotional Adaption) which is developed by Marcella and Gratch [2,3,6,22,23] and Cathexis which is developed by Velasquez [4,20,21]. Note that the former two frameworks are still work in progress. The frameworks all have a similar goal: generating emotions and moods that can support generating believable behavior in autonomous agents. A crucial difference between ALMA and Cathexis on the one hand and EMA on the other hand is that EMA also includes an advanced coping-model, whereas the other two models only concentrate on the appraisal process and the generation of emotions and moods. Of course there are a lot of other differences within the various frameworks, but there are also similarities that should be mentioned.

A first similarity that these frameworks have is that they all make a distinction between emotions, moods and personality (though it can be called differently in some literature). In general, literature about emotions is rather vague and differs in their notions of emotions and moods. Sometimes even the two terms are used interchangeably i.e., mood and emotion is treated as the same thing. But within the context of the frameworks we can see that they all agree on a distinction between emotions, moods and personality. What we can see is that emotions are feelings generated for the short-term and can be very intense. Moods are developed for the medium-term and usually last longer. They are influenced by the emotions i.e., when an emotion arises multiple times over a short period of time, this particular emotion will influence the mood of the individual drastically. So what we can see in the frameworks is that different emotions occur after every event happening and based on these emotions the mood will change more fluently and slowly than the emotions themselves. For the long term we have personality which describes properties of an individual that do not change e.g., we can have an individual who tends to be scared, so this individual will generate an emotion of fear faster than an individual who tends to be very relaxed.

Another similarity the frameworks have is a decay of emotions. When a certain emotion is generated with a certain intensity, this intensity will drop over time and eventually the emotion will disappear. In the remainder of this section we will see how this is accomplished in each model.

ALMA: A Layered Model of Affect

There is a steady increase in software that uses Virtual Characters (VCs) to make the interaction between the user and the software more natural. It is desired that these VCs have behavior similar to humans. So research in this area has focused on how to emulate typical human-like qualities such as personality and affective behavior. There are various examples of such emotional VCs. Initially focus was on systems where there was only one VC, but later the research extended to systems with multiple VCs.

In the example that I am going to describe here Gebhard uses software with multiple VCs to present something to the user. To make the VCs more believable they are extended with a personality and affective states. The current state of the VC influences the way it talks, interacts and moves.

ALMA is a computational model that can give VCs a more natural behavior. In this model we have three layers of affect, namely affects for the short-term, medium-term and long-term. For these temporal terms we have emotions, moods and personality respectively.

To demonstrate the model the Virtual Human Project is used. You can see this as a graphical 3D representation of VCs in which we can plug a separate affect generation module. And we can then see what the effect is of plugging in this module on the behavior of the VCs. In the current setting of the project we have one virtual student and one virtual teacher. The user is also seen as a student. And based on the behavior of the virtual student and the user, the teacher will act in a certain way i.e., adjust its behavior based on emotions, moods and personality. At the beginning of the program we can set the personality of the virtual student and the teacher.

The emotions, moods and personality of a VC will influence its behavior. We can see the influence of those aspects by looking at two things in general. Those are (verbal and non-verbal) expressions and the cognitive processes within the VC. Cognitive processes can be decision making, forming a motivation and the (re-)appraisal process that is based on appraisal variables and frames in ALMA as will be explained later. All the different parts of the behavior of a VC can be distinguished based on their temporal characteristics i.e., short-term, medium-term, long-term or mixed-term. An example of a short-term behavioral aspect is eye-blinking, an example of a medium-term behavioral aspect is decision making and personality is an example of a long-term behavioral aspect.

The ALMA model is based on the EmotionEngine [7,19] which is an implementation of a slightly modified version of the OCC model (see chapter 4) together with the Big five model [9]. Here the personality influences how (intensities of) emotions are calculated. Personality is described according to the Big five model, which uses five traits to describe a personality. Those traits are openness, conscientiousness, extraversion, agreeableness and neuroticism. An individual will observe the state of the world. And based on the beliefs/internals of the individual it will update its emotions according to its personality and some event, action or object in the world. On top of the emotion-layer, the EmotionEngine is expanded with a mood-layer. Moods are determined according to a model proposed by Mehrabian. In this model moods are calculated by taking an average of multiple emotional states (which are mapped to a so-called PAD point) over some period of time. So mood states change less often (are more stable) than emotional states. A mood is described by three measures, namely Pleasure (P), Arousal (A) and Dominance (D). Each value ranges from -1.0 to 1.0. If we see each measure as an axle in a three dimensional space (the PAD space) we can divide this space up into 8 octants, which are described in table 3. Here you can also see that we give each octant a mood name.

+P+A+D	**Exuberant**	-P-A-D	**Bored**
+P+A-D	**Dependent**	-P-A+D	**Disdainful**
+P-A+D	**Relaxed**	-P+A-D	**Anxious**
+P-A-D	**Docile**	-P+A+D	**Hostile**

Table 3. The 8 moods of the PAD space

So if we have a situation where all three measures are negative, we see in the table that the individual is in a bored mood. How bored he is exactly, depends on the specific P,A and D values.

The further the point described by the PAD value is from the center of the PAD space, the higher the intensity of the mood. To avoid using numbers to describe a mood, we also have three levels in the intensity of a mood. Note that each axle in the PAD space has a length of 2 (-1.0 to 1.0). We can thus see this space as a cube with planes of size 2 by 2 (figure 6). The distance from the center is at maximum $\sqrt{3}$. This distance is divided up into three (equal) parts. The first part describes a low intensity, the second part a moderate intensity and the last part high intensity. So the octant in the PAD space describes the mood the individual has and the exact PAD-value describes the intensity of the current mood.

When a VC is initiated it has to have a initial (default) mood, but how do we determine this? We do not use a same default mood for every character. Therefore the personality of a VC can be defined according to the Big Five model [10]. I assume that the creator of ALMA (P. Gebhard) has chosen this model to describe a personality because he agrees with the literature that has been written about it. As mentioned earlier, in this model the personality of an individual is described by the five factors: extraversion, agreeableness, openness, neuroticism and conscientiousness. Based on a personality described by this Big Five model we can determine the default mood an individual has with a mapping that has been devised by Mehrabian as explained in [9] i.e., we can map the Big Five to a PAD value. This mapping in shown in figure 3, as you can see the Big Five factors are variables in this mapping. So the default mood of a VC depends directly on his personality. To model mood changes in the model presented here we use emotions as the mood changing factor. In the EmotionEngine there exist 24 different emotions. We are able to map all emotions to a PAD value according to the devised mapping by Gebhard et al. as is shown in table 4 i.e., all emotions also represent a point in the PAD-space. The EmotionEngine is extended here such that it not only supports emotions, but also moods. It can now be calculated how the emotions will influence a mood according to the model of Mehrabian which calculates "an average of a person's emotional states across a representative variety of life situations". Moods are directly influenced by emotions, the reason for this is to keep the mood changing mechanism as simple as possible. The more experiences an individual has that support its current mood, the more intense that mood gets.

$$\begin{aligned}
\text{Pleasure} :=\ & 0.21 \cdot \text{Extraversion} + 0.59 \cdot \text{Agreeableness} + \\
& 0.19 \cdot \text{Neuroticism} \\
\text{Arousal} :=\ & 0.15 \cdot \text{Openness} + 0.30 \cdot \text{Agreeableness} - \\
& 0.57 \cdot \text{Neuroticism} \\
\text{Dominance} :=\ & 0.25 \cdot \text{Openness} + 0.17 \cdot \text{Conscientiousness} + \\
& 0.60 \cdot \text{Extraversion} - 0.32 \cdot \text{Agreeableness}
\end{aligned}$$

Figure 3. The mapping from the Big Five model to a PAD point

The cycle an individual goes through for updating its mood is first observing the world and notice an event, action or object. Based on appraisal (with EECs, see last section of chapter 4) of such an event, action or object an emotion is generated (which decays over a specified time). Then all currently active emotions are taken and those are mapped to a corresponding PAD point, after which the average of all those points is taken (according to the model of Mehrabian). This results in one PAD point which represents a so-called Virtual Emotion Center (VEC). This point is used to update the current mood using a function. Thus a mood is influenced by emotions, but it changes more slowly than emotions themselves. The center of the cube that represents the 3D PAD-space is called the PAD-space center i.e., the 3D point (0,0,0).

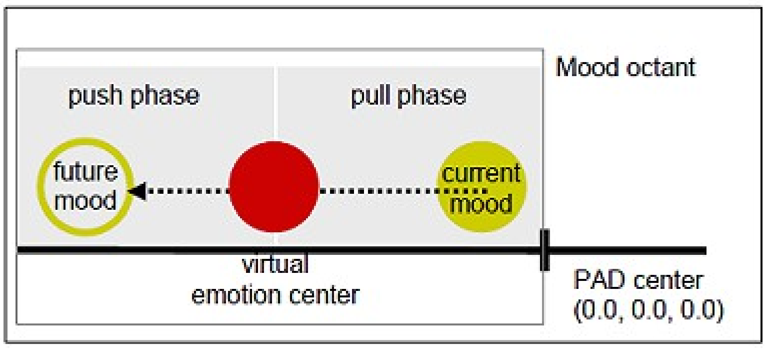


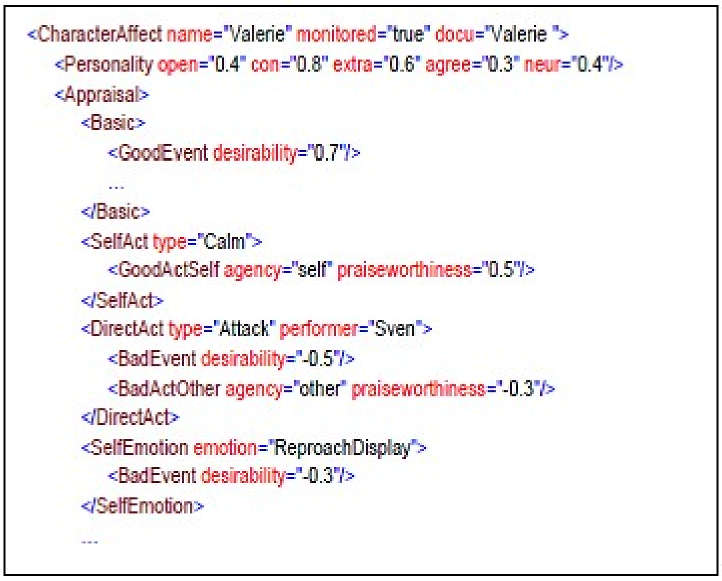
Figure 4. the push and pull mood change function

The function to determine the new mood is called *the push and pull mood change function* (figure 4). When the current mood is between the VEC and the PAD-space center, the current mood is *pulled* towards the VEC (pull phase). But when the current mood is at or beyond the VEC the current mood is *pushed* further into the current mood-octant (push phase). This push might be counterintuitive, but it represents the fact that more experiences, which support an individuals current mood, increases the intensity of that mood. The higher the intensity of the virtual emotion center, the stronger the influence will be on the current mood. To specify how long it will take to move a PAD-point from the center of one mood-octant to the center of another mood-octant given a VEC that makes the mood change function move the current mood to the center of that other octant we use the notion of *usual mood change time*. Note that we assume in that case that this VEC will exist long enough to move the current mood to the other center. Another term is the *mood return time* which specifies how long it would take to return to the default mood given a (maximum) distance from this mood of √3.

Emotion	P	A	D	Mood Octant
Admiration	0.5	0.3	-0.2	+P+A-D Dependent
Anger	-0.51	0.59	0.25	-P+A+D Hostile
Disliking	-0.4	0.2	0.1	-P+A+D Hostile
Disappointment	-0.3	0.1	-0.4	-P+A-D Anxious
Distress	-0.4	-0.2	-0.5	-P-A-D Bored
Fear	-0.64	0.60	-0.43	-P+A-D Anxious
FearsConfirmed	-0.5	-0.3	-0.7	-P-A-D Bored
Gloating	0.3	-0.3	-0.1	+P-A-D Docile
Gratification	0.6	0.5	0.4	+P+A+D Exuberant
Gratitude	0.4	0.2	-0.3	+P+A-D Dependent
HappyFor	0.4	0.2	0.2	+P+A+D Exuberant
Hate	-0.6	0.6	0.3	-P+A+D Hostile
Hope	0.2	0.2	-0.1	+P+A-D Dependent
Joy	0.4	0.2	0.1	+P+A+D Exuberant
Liking	0.40	0.16	-0.24	+P+A-D Dependent
Love	0.3	0.1	0.2	+P+A+D Exuberant
Pity	-0.4	-0.2	-0.5	-P-A-D Bored
Pride	0.4	0.3	0.3	+P+A+D Exuberant
Relief	0.2	-0.3	0.4	+P-A+D Relaxed
Remorse	-0.3	0.1	-0.6	-P+A-D Anxious
Reproach	-0.3	-0.1	0.4	-P-A+D Disdainful
Resentment	-0.2	-0.3	-0.2	-P-A-D Bored
Satisfaction	0.3	-0.2	0.4	+P-A+D Relaxed
Shame	-0.3	0.1	-0.6	-P+A-D Anxious

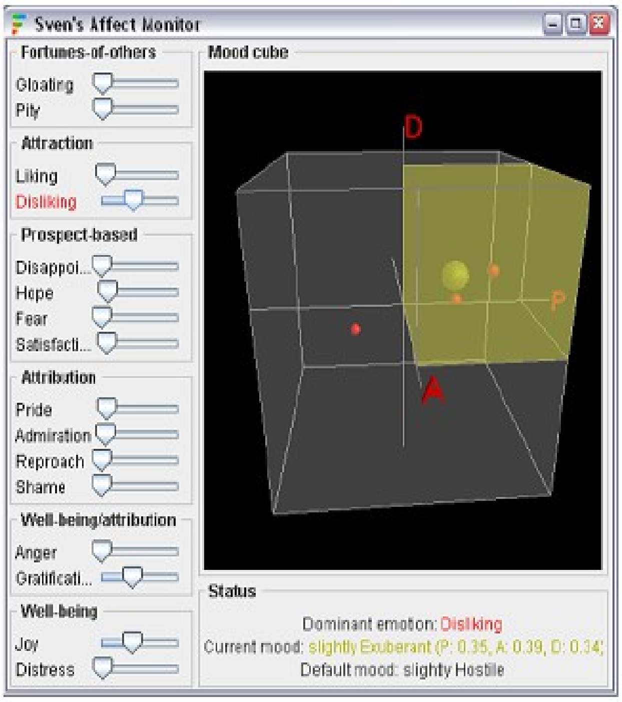
Table 4. A mapping from emotions to PAD values

In ALMA an AffectML (variation on XML) document must be specified for each agent that contains global computation parameters of the model and a personality profile (figure 5). The personality profile for an agent contains a tag that describes the agent's personality in terms of the five factors of the Big Five model and tags that describe so-called appraisal rules. These rules describe how an agent appraises its environment, i.e., how it appraises events, actions, objects, emotion displays and mood displays. Emotion and mood displays are the visual aspects of emotions and moods respectively, e.g., a tear of a sad individual is (part of) an emotion display for the emotion sadness. In general appraisal rules consist of a matching part and an action body. The matching part can either be an event, action, object, emotion display or mood display. It is compared to input from ALMA scripts (note that in ALMA we work with scripts whereas in 2APL we work with actual programs). When an event, action, object, emotion or mood display occurs in the script that matches the matching part of one of the appraisal rules, the action body of that appraisal rule is executed. The action body can be one or multiple appraisal tags or actual Emotion Eliciting Conditions (EECs). EECs are entities that cause emotions to be generated according to the OCC model (on which the EmotionEngine is based). The EECs used in ALMA are Praiseworthiness (of actions), Desirability (of events), Appealingness (of objects) and Liking (of a person) (abbreviated PDA(L), not to be confused with the PAD space). Note that appraisal tags are just abbreviations of EECs. EECs are direct input for the EmotionEngine and appraisal tags have to be converted to EECs before they are direct input for the EmotionEngine, e.g., the appraisal tag goodEvent will be converted to the EEC Praisworthiness: nil, Desirability: 0.7, Appealingness: nil, Liking: nil. In the current implementation of the model the emotions (intensities) and mood (intensity) (also called the affective profile) are updated every 500 ms. So based on an agent's personality profile (personality description and appraisal rules) events, objects, actions, emotion and mood displays are converted to appraisal tags. These appraisal tags are converted to EECs which are direct input for the EmotionEngine. The EmotionEngine will then generate new emotions (and update the intensities of already existing emotions). And based on the emotions, it will update the mood of the agent.



```
<CharacterAffect name="Valerie" monitored="true" docu="Valerie ">
    <Personality open="0.4" con="0.8" extra="0.6" agree="0.3" neur="0.4"/>
    <Appraisal>
        <Basic>
            <GoodEvent desirability="0.7"/>
            ...
        </Basic>
        <SelfAct type="Calm">
            <GoodActSelf agency="self" praiseworthiness="0.5"/>
        </SelfAct>
        <DirectAct type="Attack" performer="Sven">
            <BadEvent desirability="-0.5"/>
            <BadActOther agency="other" praiseworthiness="-0.3"/>
        </DirectAct>
        <SelfEmotion emotion="ReproachDisplay">
            <BadEvent desirability="-0.3"/>
        </SelfEmotion>
        ...
```

Figure 5. A personality profile example in AffectML



Dominant emotion: Disliking
Current mood: slightly Exuberant (P: 0.35, A: 0.39, D: 0.34)
Default mood: slighty Hostile

Figure 6. Graphical representation of a VCs affective profile

In figure 6 we can see a graphical representation of a VCs affective profile at a given time. At the left side we can see the VCs emotions and their intensities and at the right side we can see the PDA space of the VC. The highlighted octant shows the VCs current discrete mood and the big sphere shows the actual mood. The smaller red spheres show the currently active emotions.

The Virtual Human Project and the ALMA model are tightly working together. They interchange data in xml-form. We thus have a cycle between the two modules. It is also shown in demonstrations that because of the layered affect model the VCs produce more believable behavior. It should be noted that the ALMA model is not a finished model yet and is work in progress. The authors also wished to publish the model as a freely available software toolkit at the time the paper was written, so that others can use it to produce believable VCs. In the meanwhile they have indeed published an ALMA software package available at: http://www.dfki.de/~gebhard/alma/index.html. An update Gebhard gave me when I requested the package is that AffectML will be replaced by the new standard EmotionML: http://www.w3.org/TR/2010/WD-emotionml-20100729.

EMA: Emotions and Adaption

Emotions influence the behavior of humans drastically. They are triggered by events happening in the environment. Once an event has happened and triggered a certain emotion, this emotion will influence how a person thinks and acts with respect to this event i.e., emotions will influence the so-called person-environment relationship. When we talk about changes in those relations in the temporal sense we can see that those changes can happen within milliseconds, but it can also take days or weeks for a change to occur. Which emotions will be triggered does not solely depend on the environment, but also on the individual's internals.

To formalize emotional processes mainly appraisal theories have been used. With these theories the relation between an individual and its environment is described by a set of appraisal variables. And this relationship influences what kind of emotions will arise. So for each event we can specify certain appraisal variables. Examples of such variables are controllability and likelihood (of a certain event happening). But to date little attention has been given to the process that underlies the appraisal i.e., how are the values of appraisal variables determined and what are their relations. Therefore a computational model is described that tries to capture this underlying process of appraisal. Advantages of this approach are that once such a model has been realized it can be used to test how emotions influence behavior and this approach also forces to think about how this process should work. The model that is being described is called EMA (Emotion and Adaption).

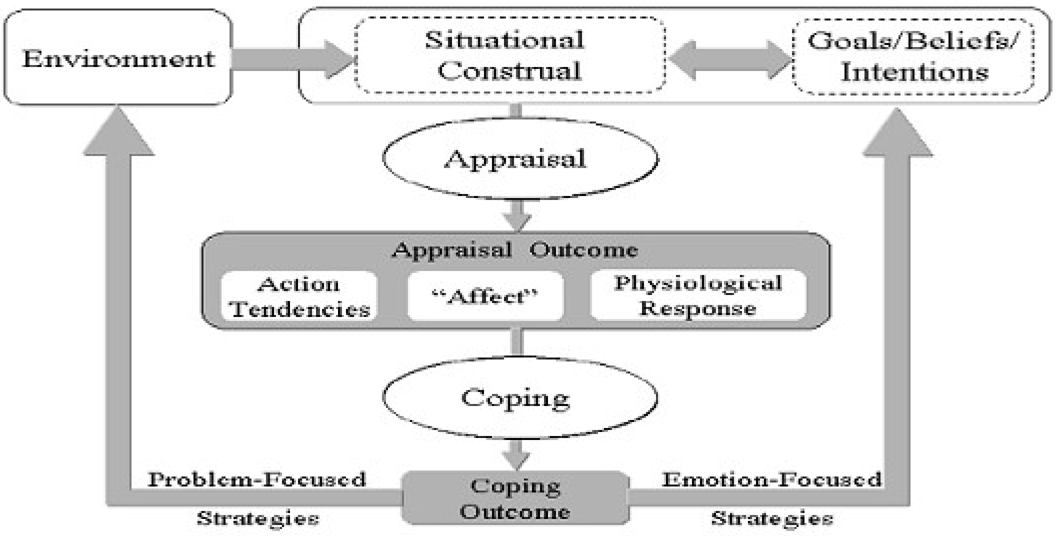


Figure 7. The cognitive-motivational-emotive model from Smith and Lazarus

In one paper about EMA an experiment is done where different kinds of emotions are triggered over a short period of time. The experiment is realized by filming two actors with 30 frames per second such that the frames can be studied separately i.e., the facial expressions and body posture. During instruction of the actors a bird flies into the room unexpectedly (for the actors). This event causes different emotions to occur rapidly. It is observed that partially the reactions of the two actors are the same, but they also differ at times. This is partially due to the fact that the second actor notices the bird later than the first actor and the second actor does not hold an object in its hand whereas the first actor does. It can be seen that appraising a situation depends on various kinds of processes within the individual i.e., perceptual processes and inferential processes. And next to processes to appraise a situation, there are also processes for determining how to react to a situation. This process is called coping. We can see coping as the counterpart to appraisal. The model of appraisal and coping is based on the model of Smith and Lazarus (figure 7). With appraisal we value the current state of the world as it is interpreted at a certain time, which includes the past, present and expected future. Appraisal will result in emotions. And each event has an appraisal frame attached to it which contains all relevant appraisal variables (this will be clarified with an example later). With coping we would like to change the appraised state to a desired state, or when a desired state is already reached, maintain it. During a situation happening re-appraisal is done, thus what we get is a cycle of appraisal, coping and re-appraisal. The exact cycle in EMA is shown in figure 8. What we can see in that figure is that first a causal interpretation is made of the world for the agent (step 1) and after that each event in the past, present and future is (re-)appraised (step 2). Note here that also events in the past can be re-appraised due to events happening in the present or future. In the next step (step 3) each appraisal frame is mapped to an emotion with an intensity and the emotions resulting from all appraisal frames are accumulated to form one general emotional state and display the current mood. The mapping from appraisal frame to emotion is based on a mapping introduced by Clark Elliott [3], which in turn is based on the OCC model. The mapping utilizes 24 different emotions. The mood of an agent is described by the set of all emotions that are used in EMA (hope, fear etc.) with respective intensities (between 0 and 1). In each update, the current appraisal frames are checked with their respective emotions and intensities. These intensities are then added to their respective emotion intensity of the mood (step 4). And the emotion with the highest intensity of the mood will determine what kind of coping strategy will be executed (step 5). This cycle keeps repeating during execution of EMA for every agent. Another thing to note about appraisal is that an agent can maintain appraisal frames from different perspectives e.g., he can have an appraisal frame for an event from his own perspective, but also from the perspective of another agent. For example imagine two people going out for dinner, and one of them eating all the food. Usually someone will not do this, even if he would like to eat it all, because he can imagine that the other person will not like this (appraise it negatively for the other person). We can see all possible appraisal variables in EMA in table 5 and all possible coping-strategies in table 6. In this paper it is a goal to capture this cycle within a computational model. There are two mentioned difficulties. The first one is that we can have short-term emotions and long(er)-term emotions (moods) which both must be captured by the model. The second difficulty is that of appraisal, because that what has to be appraised are not merely simple physical situations but can also be a more complex social situation.

Appraisal variables		
Relevance		Does the event require attention or adaptive reaction
Desirability		Does the event facilitate or thwart what the person wants
Causal attribution	Agency	What causal agent was responsible for an event
	Blame and credit	Does the causal agent deserve blame or credit
Likelihood		How likely was the event; how likely is an outcome
Unexpectedness		Was the event predicted from the past knowledge
Urgency		Will delaying a response make matters worse
Ego Involvement		To what extent does the event impact a person's sense of self (social esteem, moral values, cherished beliefs, etc.)
Coping potential	Controllability	The extent to which an event can be influenced
	Changeability	The extent to which an event will change of its own accord
	Power	The power of a particular agent to directly or indirectly control an event
	Adaptability	Can the person live with the consequences of the event

Table 5. All possible appraisal variables in EMA

Common coping strategies	
Problem-focused Coping	Active coping: taking active steps to try to remove or circumvent the stressor
	Planning: thinkg about how to cope. Coming up with action strategies
	Seeking social support for instrumental reasons: seeking advice, assistance, or information
Emotion-focused Coping	Suppression of competing activities: put other projects aside or let them slide
	Restraint coping: wating till the appropriate opportunity: Holding back
	Seeking social support for emotional reasons: getting moral support, sympathy, or understanding
	Positive reinterpretation & growth: look for silver lining; try to grow as a person as a result
	Acceptance: accept stressor as real. Learn to live with it
	Turning to religion: pray, put trust in god (assume God has a plan)
	Focus on and vent: can be function to accomodate loss and move forward
	Denial: denying the reality of event
	Behavioral disengagement: Admit I cannot deal. Reduce effort
	Mental disengagement: Use other activities to take mind off problem: daydreaming, sleeping
	Alcohol/drug disengagement

Table 6. Possible coping-strategies

In the model described for each appraisal variable a process is defined and other internal processes infer from these. We can have multiple processes, where one process determines the short-term emotions and another process determines the long-term emotions.
A computational model is "a process or processes operating on representations" [1]. So in this case the person-environment relation spoken about above is represented in a way that will be explained soon (and will be clarified with an example later), which is interpreted and altered by processes. Of course the representation of the person-environment relation must show the appraisal variable values (appraisal frames) and also the representation must make the phenoma happening, which influence the emotions, observable. The causal connection between the phenoma must also be visible. The person-environment relation as it is interpreted by an agent at some point in time is called the causal interpretation. This representation is based on a plan-based causal representation, decision-theoretic planning techniques and methods that explicitly model commitments to beliefs and intentions. Combined these methods and models can form the basics for an appraisal and coping process according to the authors (coping is discussed later). The complete causal interpretation shows beliefs, desires, intentions (and probabilities) at a point in time and it is split up in a past, present and expected future as will be clarified in an example later.

1. Construct and maintain a causal interpretation of ongoing world events in terms of beliefs, desires, plans and intentions
2. Generate multiple appraisal frames that characterize features of the causal interpretation in terms of appraisal variables
3. Map individual appraisal frames into individual instances of emotion
4. Aggregate emotions instances into a current emotional state and overall mood
5. Adopt a coping strategy in response to the current emotional state

Figure 8. The EMA algorithm

Each event that can happen is described with appraisal variables (controllability, desirability, responsible agent, etcetera). Every event will influence the emotions according to the scheme of Clark Elliott (a slightly modified version of the OCC model), which will influence the mood of the agent in the long term. Another important aspect of the EMA model is coping with the appraised events. The problem of choosing a coping strategy is handled by a separate computational model within the EMA model. This model takes care of choosing a suitable coping strategy (more about this in the section that discusses the question: "Does the framework have a coping mechanism?"). Examples of coping strategies are dropping a threatened intention (acceptance) and asking someone that is in control of an outcome for help (seek instrumental support). To have an efficient coping model there cannot be a broad distinction between problems and emotions, because problem solving is actually guided by emotions within the individual. Nevertheless we can make a distinction between emotion-focused coping strategies and problem-focused coping strategies (figure 7 and table 6). Emotion-focused strategies alter the internals of the agent (goals, beliefs and intentions) such that the agent will appraise the situation differently i.e., nothing has to change physically in the environment around him for him to appraise the environment differently. Problem-focused strategies are strategies that really alter something in the environment. This property of using both emotion-focused coping and problem-focused coping characterizes the EMA model.

In the next section an example is given of (a fraction of) the bird example described above in the EMA model which is expressed with a causal interpretation. In the example we can see all the different components that can be used within the representation of the causal interpretation.

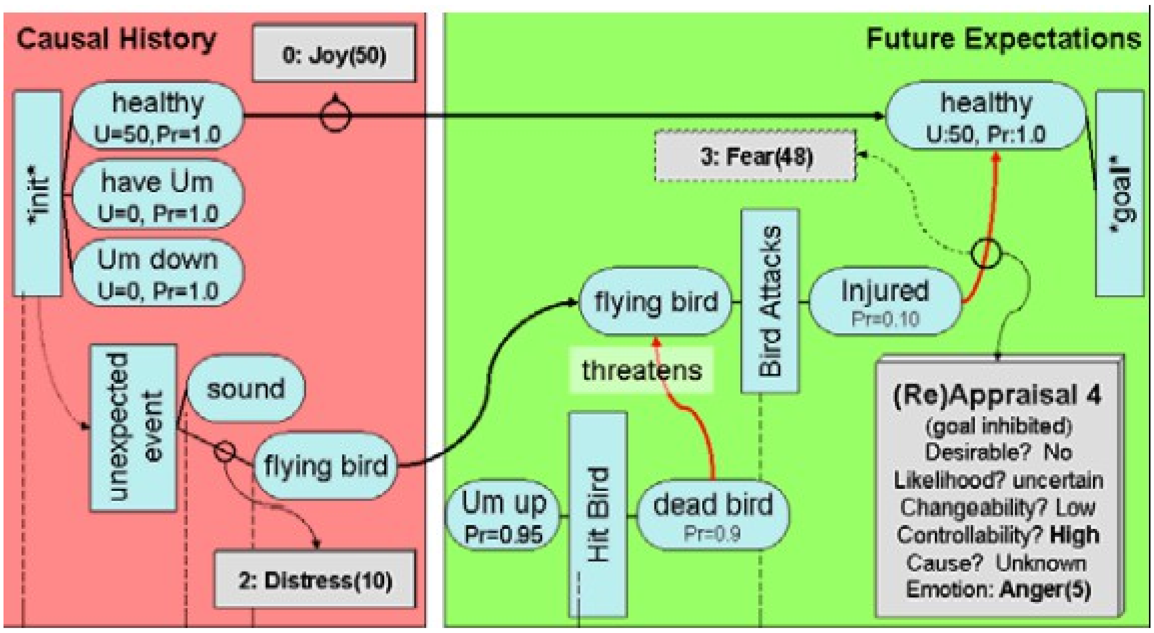


Figure 9. Causal interpretation example

In figure 9 we can see an example of such a causal interpretation at some timepoint. The first thing to notice is that this interpretation is split up into a history (the red part) and an expectation about the future (the green part) at every point in time (in other cases also a present block is visible), here the white space between the history and the future represents the present. At each timepoint an agent has only one causal interpretation about his environment. The ellipses represent predicates (beliefs) about the environment that can be true or false and the rectangles represent actions/events. From the diagram it can be seen that all events and predicates have a causal connection with each other. The causal interpretation is updated every 500 ms in this example, of course in figure 9 we see this interpretation at one timepoint. Time in the image proceeds from left to right. Note that not only the expected future can change, but also the causal history i.e., the causal interpretation is a state of the environment as the agent thinks it is, but this does not imply that it is a correct interpretation.

The init and the goal actions are special in the sense that init sets the initial state of the simulation and goal represents the case where the goal of the agent is reached. A goal is always described by one or more conditions becoming true (the probability of the condition is 1.0). Note that the goal action (the action named “goal” in the causal interpretation is referred to as the goal action) has some pre-conditions attached to it. In figure 9 the goal action has attached to it the pre-condition *healthy* being true (it is true because the appraisal variable pr(obablity) is 1.0). Reaching a goal means that some specified conditions are met. So each goal is actually described by multiple conditions that have to be met. Which conditions those are for a specific goal are specified by the goal's pre-conditions. The goal action itself is just there to display that an agent has reached its goal, it is not an action in the sense of other actions in the causal interpretation, like for example “Hit Bird”. We can also see that some predicates contain a probability and utility measure. If we look at the “Um down” (abbreviation for Umbrella down) predicate in the causal interpretation we can see that the agent is certain that this fact is true (Pr=1.0) and it has a utility of 0 (U=0) for this agent (he does not desire it but is not opposed to it either). Predicates can be linked to actions, before or after the action. These predicates denote pre-conditions and post-conditions of the actions respectively. The grey rectangles denote appraisal frames, which contains a name label, a number to indicate at which timepoint the frame is generated, an emotion the appraisal elicits with an intensity, and all appraisal variables that are relevant for it. This all has been specified only for the 4th appraisal frame due to space limitations. We can see that the healthy predicate is desired by the agent (though not explicitly denoted in this example) because the appraisal of this predicate results in an emotion of joy with intensity 50 (appraisal frame 0).

The expectation about the future that an agent has (the green part in figure 9) forms according to certain pre-defined rules that are discussed soon. So do not think this is just a randomly generated expectation, it is all inferred from rules. And of course it is denoted as “expected” future, which implies that the future does not have to be as it is defined in the causal interpretation of the agent. In other examples the expected future is also referred to as future plans, which (in my opinion) is a more suitable name for this part of the causal interpretation.

Actions an agent can perform can be specified with a time-frame i.e., how long does it take to execute an action. But of course other physical and mental events also take time. The EMA model does not have an advanced time model which is something that the authors admit and could be seen as future work.

Based on the causal interpretation, appraisal is done on the various events happened/happening or expected events in the future. After the appraisal, emotions are generated based on each appraisal frame (which corresponds to one event). And these emotions in their turn all together influence the overall mood of the agent (see figures 8 and 9). In table 7 we can see how appraisal variables are mapped to emotions for 7 of the 24 emotions. The exact value of the appraisal variables will determine the intensity of the emotion. Note that this mapping is based on a simplified version of the OCC model created by Clark Elliott as mentioned earlier.

Appraisal pattern for proposition "p"	**Emotion**
Expectedness(self,p) = low	Surprise
Desirability(self,p) > 0 & Likelihood(self,p) < 1.0	Hope
Desirability(self,p) > 0 & Likelihood(self,p) = 1.0	Joy
Desirability(self,p) < 0 & Likelihood(self,p) < 1.0	Fear
Desirability(self,p) < 0 & Likelihood(self,p) = 1.0	Sadness
Desirability(self,p) < 0 & Causal attribution(self,p) = other & Controllability(self,p) ≠ low	Anger
Desirability(other) < 0, causal attribution(p) = self	Guilt

Table 7. Mapping from appraisal pattern to emotion

Next to this causal interpretation for an agent there also exists a plan library and a set of stimulus response rules. The plan library contains action definitions and recipes that define how actions can be combined. These recipes are used for plan generation within an agent. The stimulus response rules check whether certain predicates are true at a point in time, once all the predicates in the response rule are true, an action can be executed (which is defined in the response rule). Examples of such rules are when a sound is percieved (predicate) the action the agent will perform is looking or when the agent percieves a flying bird (in his interpretation) it will try to hit the bird. The whole EMA system must be coupled with a world simulation in which it is defined how actions affect the environment (predicates that describe the world).

Cathexis

Maybe when you first think about it, emotions and rationality do not really have a lot to do with each other. In some areas of research it can be desirable that agents behave like humans and it has also been proven that for humans to behave rational, emotions are critical. Without emotions humans would fail to act rational in a lot of situations i.e., effective decision-making by humans is guided by emotions. Would we thus want to create agents that behave more natural and have an effective decision-making model, it seems that emotion(like) mechanisms are necessary.

Most literature about emotions is rather vague in its definition of emotions and there is not a lot of consensus about the subject as already has been noted before. Though there are some elements of emotions where most researchers in this area agree on.

An important question is that of what it is that makes an emotion arise i.e., what elicits an emotion? An emotion can have multiple elicitors that activate it, most research (at the time this framework was developed) has focused on cognitive elicitors of emotions. But there are also non-cognitive elicitors of emotions, which will also be covered in the computational model that this section discusses.

Another aspect of emotions is that of its dynamic behavior. Once an emotion is active, it should not stay active forever e.g., when an individual is happy because he won a bet for 10 dollars, he will probably not be happy eternally. This is why it should be taken into account for each active emotion with which rate the emotion will decay over time.

Something where most researchers also agree on is that we have to make a distinction between emotions and moods, though there is a very strong connection (interaction) between the two phenomena. The main aspect in which they differ is temporal i.e., moods last longer than emotions.

Once emotions are active they will influence the behavior (action selection, goal generation etcetera) of an agent or system. How emotions influence behavior exactly is not the main focus of the model that is going to be described, but it is covered. For this reason it is a separate module in the model which can be adjusted very easily, without changing other parts of the model.

The computational model discussed in this section is called Cathexis and is designed to simulate the generation of emotions and moods. Within the model the author has defined 6 basic emotions, which are anger, fear, distress/sadness, enjoyment/happiness, disgust and surprise. We should see each of those basic emotions as a family of emotional states and not as a single state in the model. All the states in one family have certain similarities e.g., the behavioral response to the emotion. But the difference between states in one family can be their intensity or expression.

Not all possible affective states that are emotions are captured by the basic emotion families. We can also have emotion blends, which means that we can have two emotions active at the same time. A good example of such an emotion blend is jealousy, with this blend an individual can feel anger and fear at the same time. Another possibility is to feel mixed conflicting emotions simultaneously. If an individual has won a car and a friend of him is ill, he can be happy because he won the car but at the same time also be sad because his friend is ill. Within the Cathexis model it is possible for multiple basic emotions to exist simultaneously i.e., the situations described above can be captured in the model.

A clear distinction is made between emotions and moods as mentioned earlier. There is interaction between the two though. On the one hand emotions can cause moods to arise and on the other hand moods can influence the probabilities of certain emotions being generated. Each emotion has a so-called activation threshold here and after referred to as α in this section. This means that when the intensity of a certain emotion becomes greater than α, it will be activated. It seems that for emotions that occur most often during a certain mood, their α decreases e.g., when an individual is in a bad mood, events that would normally (in another mood) not make him angry can cause him to get angry. Moods in humans can be generated in two different ways. The first way of a mood arising is that of an individuals biochemical state (hunger, tiredness). The second cause can be the occurrence of a specific high intensity emotion over a short period of time. Within Cathexis a distinction is made between emotions and moods by the level of arousal. Though moods and emotions exist in the same emotional system, emotions are characterized by high arousal and moods are characterized by low arousal. Also temperaments can be set in the model for each agent to describe its personality as we will see later.

As mentioned earlier Cathexis covers both cognitive and non-cognitive elicitors of emotions. The elicitors are divided over four categories within the emotion generation system of Cathexis. Those are neural, sensorimotor, motivational and cognitive. The first three categories cover non-cognitive elicitors.

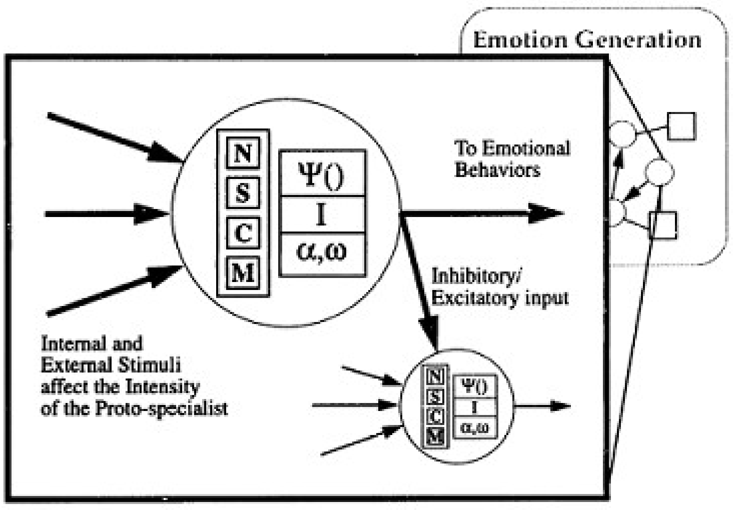

Emotion Generation
To Emotional Behaviors
Inhibitory/ Excitatory input
Internal and External Stimuli affect the Intensity of the Proto-specialist
N
S
C
M
Ψ()
I
α,ω

Figure 10. A proto-specialist

Each basic emotion in Cathexis is represented as a so-called proto-specialist (Minsky) (figure 10), where all proto-specialists reside in a Emotion Generation network. This network consists of all proto-specialists. Figure 10 can be misleading because of an incorrect notation, but in the rectangle in the front you can see two circles that each represent a proto-specialist such that a connection between these two can be made obvious. It is thus not a zoom of one proto-specialist as the background of the diagram implies. A proto-specialist can have output signals that are input signals for other proto-specialists in the form of excitatory or inhibitory inputs (Later more about this). This way connections between proto-specialists are established. It is not predefined by the framework which connections exist and which inputs/outputs are in this network. This can be defined by the user. Some emotions can strenghten other emotions, but it could also be the other way around i.e., an emotion is weakened by another emotion. This is modeled by excitatory inputs and inhibitory inputs respectively. How these inputs influence an emotion is not prescribed by the model but can be specified by the users of the model i.e., it can be specified by the user which proto-specialist influences other proto-specialists and also the degree of impact can be defined.

Each proto-specialist has sensors that monitor whether the right conditions are met that would elicit the emotion represented by the current proto-specialist. The sensors are divided into the same groups as the emotion elicitors (neural, sensorimotor, motivational and cognitive). So the sensor-input determines for a proto-specialist whether the intensity for its represented emotion should be increased or decreased. As mentioned earlier each emotion (thus each proto-specialist) has an activation threshold α. So when the intensity of the emotion surpasses this value the emotion is activated and its output signal is released to other proto-specialists. There is also a threshold value called the maximum intensity level ω, which denotes the maximum intensity an emotion can reach. Another element each proto-specialist has is the decay-function $\psi()$, which determines how long an emotion stays active once it is activated. The letter *I* denotes the intensity of the emotion represented by the proto-specialist.

As said before, emotions are represented by high arousal and moods are represented by low arousal. So the proto-specialists can model emotions (high intensity) as well as moods (low intensity). That moods will last longer than emotions is modelled in the system by letting the decay-functions decrease the intensity more slowly at low intensities than at high intensities of emotions. Temperaments are simply modelled by different threshold values for each emotion e.g., someone who tends to be fearful has a relatively low α for fear.

The intensity of an emotion is determined by several factors such as the old intensity, all the elicitors for that emotion and the inhibitory and excitatory inputs of the other emotions. The intensity at time t for emotion i is given by equation 1.

$$EI_{it} = \chi\left(\left(\Psi(EI_{i(t-1)}) + \sum_k EL_{ki} + \sum_l (EX_{li} \cdot EI_{lt}) - \sum_m (IN_{mi} \cdot EI_{mt})\right), 0, \omega_i\right)$$

Equation 1. The equation that gives the intensity of emotion i at time t

EI_{it} represents the intensity of emotion i at time t. The function $\chi()$ constrains the value of an intensity between 0 and ω_i. Remember that ω_i is the maximum intensity emotion i can have. $\chi()$ can thus be seen as a sigmoid function, where the last two parameters specify the minimum and the maximum value. $EI_{i(t-1)}$ gives us the intensity of emotion i in the previous timestep and $\psi(EI_{i(t-1)})$ gives us the intensity that is left in the current timestep. Note here that $\psi()$ is the decay function as is explained before. The part of the formula $\Sigma_k EL_{ki}$ adds all the k values of the emotion elicitors for emotion i. $\Sigma_l (EX_{li} * Ei_{lt})$ denotes all the l emotions that add excitatory gain to emotion i i.e., EX_{li} gives the excitatory gain for emotion i from emotion l and multiplies this with the current intensity of that emotion l, Ei_{lt}. And it does the same for all the inhibitory gain from emotions m with the part $\Sigma_m(IN_{mi} * EI_{mt})$. So summarized this function takes the old (decayed) emotion intensity, adds the values of all emotion elicitors, adds the excitatory gain, subtracts the inhibitory gain and puts it in the sigmoid-like function. The value of a specific emotion elicitor is determined by the conditions it depends on. Note that each elicitor can have totally different conditions.

We spoke about the activation threshold α earlier. A specific proto-specialist representing an emotion will only produce an output-signal other than zero when the intensity is higher than α. This is formalized in equation 2, where O_{it} is the output signal for emotion i at time t.

$$O_{it} = \begin{cases} EI_{it}, & \text{if} \quad (EI_{it} > \alpha_i) \\ 0, & \text{otherwise} \end{cases}$$

Equation 2. The equation for the output-signal of emotion i at time t

The decay function $\psi()$ is specific for each basic emotion. It is a task of the user of the model to implement these functions, the model does not prescribe anything for this. The only thing that is modelled is that this function is called for every emotion in every update-cycle.

We have spoken about the Cathexis model with respect to generation and the dynamics of emotions, moods and temperaments. In the next part it will be discussed how these affective phenomena influence the behavior of an individual. This functionality is covered by the emotional behavior system of the model. It is the behavior of the individual that expresses the influences of emotions and moods. Note that this is a separate module of the model.

The emotional behavior system takes care of choosing the right behavior given the current emotion of the individual at a certain time. You can see this system as a network of behaviors, where the emotional behavior with the highest value is chosen to control the agent. The value of an emotional behavior is determined by various factors and is updated in every update-cycle. Such behaviors consist of two parts: the experiential component and the expressive component. The expressive component takes care of all the expressions caused by an emotion i.e., facial expressions, body posture and vocal expressions. And the experiential component determines how an emotion is experienced by the agent and how it affects its action decision process and cognition process i.e., actions and cognition of an agent are heavily influenced by their emotions.

$$EB_{it} = \sum_{k} BR_{ki}$$

Equation 3. The equation for determining the value of emotional behavior i at time t

The factors that influence the value of an emotional behavior are called (behavioral) releasers. Examples of releasers are motivations (emotions, moods etc.) and external stimuli. The value of an emotional behavior is determined by equation 3. Here EB_{it} is the value of emotional behavior i at time t, and BR_{ki} is the value of behavioral releaser k of emotional behavior i. How the value of a releaser is determined is not specified and can be done in various ways, it depends on the nature of the releaser in question. The emotional behavior system is kept simplistic intentionally such that it can be used for various action-selection mechanisms.

6. Comparison of the frameworks

In this part I will compare the three frameworks based on multiple questions. So in every part I will discuss a question and answer it for each framework. The order in which I will answer these questions is first answering it for ALMA, then EMA and finally Cathexis. I assume in this part that the reader has read chapter 5 in which the frameworks are discussed.

Which affective phenomena are distinguished?

ALMA

In ALMA we have three affective phenomena. The first one is emotions, these relate to events, actions or objects in the environment and are short-term affective phenomena. For the medium-term moods are used and for the long-term we have personality of an individual. Personality is something that does not change regularly, but it is possible over a long period of time and is described with the Big Five model (more about this in chapter 5 in the section about ALMA).

We have 24 different emotions in ALMA. The EmotionEngine (which is an implementation of a modified version of the OCC model in combination with the Big Five personality model) is the core of the emotional system. The emotions can influence the mood to be one of eight different moods. For the details about the exact emotions and moods I refer to chapter 5.

EMA

EMA distinguishes the phenomena: emotions and the overall emotional state. The latter corresponds to a mood. Here emotions are directly related to some event that has happened or an expected event. Moods are a longer lasting more global phenomenon which are formed by multiple emotions.

Here the emotions are (indirectly) based on a model of Clark Elliot which is also inspired by the OCC model. We have 24 different emotions. Moods are described as a set of all emotion-types with a respective intensity for each emotion-type, we thus have an enormous set of possible mood-states.

Cathexis

In Cathexis so-called basic emotions are used, which is taken from the ideas of Ekman [31]. We should not look at a basic emotion as a single affective state here, but rather as a family of affective states. The affective states within a family have certain similarities (behavioral response, expressions etc.), but they can differ in other areas such as intensity. The six basic emotions defined in Cathexis are: anger, fear, distress/sadness, enjoyment/happiness, disgust and surprise.

Apart from these basic emotions, we also have emotion blends in Cathexis. So two emotions can be active simultaneously. An example of such an emotion blend is jealousy, this is actually a blend of the emotions anger and fear (and perhaps sadness). Another case where we can have emotion blends is when an individual feels mixed emotions. There could be a situation in which an individual experiences anger and happiness at the same time.

As in the other two models also moods are defined as an affective phenomenon which are longer lasting and more stable than emotions. Moods are formed by emotions and moods influence the probabilities of certain emotions being triggered e.g., when an individual is already in a bad mood it is more easy to make this individual angry than it would be if he was in a good mood.

To define an individual's characteristics the term temperaments is used in Cathexis. We can see this as a personality description. This comes down to altering the activation-thresholds (see chapter 5) for emotions e.g., we can have someone who becomes frightened easily, thus this individual will have a low activation threshold for fear.

How are emotions generated?

ALMA

Each event, action or object in the environment defined for ALMA has a so-called appraisal tag. In ALMA an appraisal tag is put after each utterance. An example of such an utterance is:

John: The weather is getting better [=good_event]

This utterance done by agent John is appraised as a good event (by John). All possible appraisal tags are shown in table 8. These appraisal tags are abbreviations of so-called Emotion Eliciting Conditions (EECs). EECs are entities that cause emotions to be generated according to the OCC model (on which the EmotionEngine of ALMA is based). In ALMA the EECs Praiseworthiness (of actions), Desirability (of events), Appealingness (of objects) and Liking (of persons) are used. Do not confuse these PDA(L) values with PAD values i.e., PDA(L) values are aspects that make emotions arise and PAD values describe an actual mood (and also emotions in the case of ALMA). To give an example, the appraisal tag good_event will be converted to the following PDA(L) value in ALMA:

P: nil, D: +0.7, A: nil, L: nil

Of course we could explicitly set all these values after each utterance directly, but to work with these PDA(L) values is not practical for users of the framework. Therefore we will use appraisal tags instead of PDA(L)-values i.e., appraisal tags are just abbreviations for PDA(L)-values. The PDA(L)-values can also be called Emotion Eliciting Conditions (EEC).

Class	Appraisal tag	
event	good_event	bad_event
	good_unlikely_future_event	bad_unlikely_future_event
	good_likely_future_event	bad_likely_future_event
action	good_act_self	bad_act_self
	good_act_other	bad_act_other
Object	nice_thing	nasty_thing

Table 8. All possible appraisal tags in ALMA

How emotions are generated in ALMA is defined by the appraisal rules which can be one of four types: basic appraisal rules, act appraisal rules, emotion display appraisal rules and mood display appraisal rules. In table 9 an example of each type of rule is given.

The basic appraisal rules describe how each event, object or action related to an agent himself is appraised by this agent. Such rules have the form: <basic appraisal input> := <EEC variables>.

Act appraisal rules denote how an agent appraises acts of himself, but also acts from other agents. When the act is performed by another agent, it must be specified whether it is a direct or indirect act. Direct acts are directly addressed to the agent himself and indirect acts are acts where the agent himself is just a listener. These rules are of the form: <act input, {self, {direct|indirect}|other}>:= {basic appriasal input}+.

Emotion display appraisal rules define how an agent appraises its own emotion display (visible signs of an emotion) and those of others. Also in these rules the agent in question has to be specified, which can be the agent himself or another agent. These rules are of the form: <emotion display input, {self|other}> := {basic appraisal input}+.

Mood display appraisal rules define how an agent appraises its own mood display (visible signs of a mood) and those of others. Here also the character in question has to be specified. These rules are of the form: <mood display input, {self|other}> := {basic appraisal input}+.

Type of rule	**Example(s)**
Basic appraisal rules	• good_event := P: nil, A: nil, D: +0.7, L:nil
Act appraisal rules	• Calm, self := good_act_self • Attack, Sven, direct := bad_event, bad_act_other
Emotion display appraisal rules	• ReproachDisplay, self := bad_event • SatisfactionDisplay, Sven := good_event
Mood display appraisal rules	• HopeDisplay, self := good_event • HappyDisplay, self := good_event

Table 9. Types of appraisal rules with examples

So appraisal rules will convert actions, events or objects to PDA(L)-values (directly or indirectly), which are EECs. These EECs will cause ALMA to generate emotions. And these emotions will influence the overall mood of an agent.

EMA

In EMA the causal interpretation plays a central role. This is an instance of the person-environment relation an agent has at some point in time, which is formed by observation and inference of the agent. It is actually a product of all the cognitive processes going on inside the agent. In table 10 all the operations that can be done on the causal interpretation to update it are listed. Events that (are expected by the agent to) happen or events that have already happened are appraised by the agent. This appraisal is done with multiple appraisal variables. To each event an appraisal frame is attached, which contains all the appraisal variables and as a label an emotion with an intensity to which it is mapped according to a modified version (by Clark Elliott) of the OCC model.

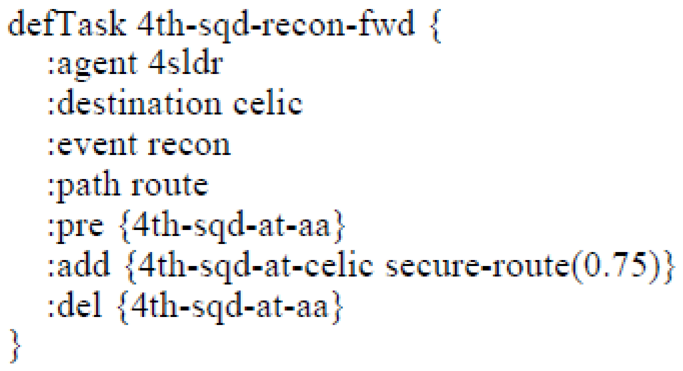
The basic element to form the causal interpretation is the plan formation model. This is a modified model of the SOAR reasoning model (Newell, 1990). Because the used model must represent beliefs, goals, plans and possibilities some modifications were made to the SOAR model. The plans that are formed consist of tasks, or as they are called in EMA: actions. The set of actions (with its constraints) is called a plan network.

An example of such an action is shown in figure 11. In the action description a lot of abbreviations are used. In words this action is a reconnaissance action in the forward direction for the 4th squadron. The action can be executed by the 4th squadron leader (4sldr). The destination of the reconnaissance is Celic. The event-type is recon along the path route. A precondition for this action is that the 4th squadron is at the assembly area (4th-sqd-at-aa). And when the action is executed the squadron is at Celic and the route is then secure with a probability of 75% (add {4th-sqd-at-celic secure-route(0.75)}). The fact 4th-sqd-at-aa is removed after executing the action.

Cognitive Operators		
Cognitive	Update belief	Add/drop a commitment to truth value of a proposition
	Update intention	Add/drop a commitment to achieve state/perform action
	Update plan	Add/drop a plan step or resolve-conflicts in a current plan
	Understand speech	Interpret incoming speech act
	Output speech	Produce speech act
	Wait	Default action if no other operator applies (busy wait)
Perceptual	Monitor goal	Orient to observe truth value of goal proposition
	Monitor expected effect	Orient to observe consequence of executing action
	Monitor expected act	Orient to observe initiation of pending action
	Listen to speaker	Orient to speaking agent
	Expect speech	Orient to agent that is expected to speak
	Monitor unexpected event	Orient to event location (e.g. Attend to a sound) and record any unexpected change in truth value of proposition
Motor	Initiate-action	Start action (or record start of external observed act)
	Terminate-action	Terminate action (or record end of external observed act)

Table 10. All the cognitive operations possible in EMA to update the causal interpretation

The preconditions of an action are just a conjunction of simple facts (predicates) that have to be true in the environment before the action can be executed. The effects of an action are described by the *add* list and the *del* list. They describe which predicates have to be added and removed respectively. In EMA actions take time to be executed i.e., the effects of actions do not have to occur immediately. And actions can also fail i.e., only a part (or none) of the effects of an action are realised. Therefore the STRIPS formalism is adjusted slightly. The adjustment is that each predicate in the add list is satisfied individually at some point during the execution of an action (so not all effects in the add list of an action hold instantaneously as is the case in standard STRIPS). Once a predicate holds it will hold until it is negated by another effect (this means that this effect has to be in the del list). Note that a predicate in the add-list is satisfied at some point during the execution of the task, if the exact point in time is also specified is not mentioned by the authors in the paper [6].



```
defTask 4th-sqd-recon-fwd {
    :agent 4sldr
    :destination celic
    :event recon
    :path route
    :pre {4th-sqd-at-aa}
    :add {4th-sqd-at-celic secure-route(0.75)}
    :del {4th-sqd-at-aa}
}
```

Figure 11. An action in EMA according to the STRIPS formalism

The same properties hold for the delete-list, only here all the predicates that become false caused by executing the task are listed. Because there is an open world assumption, predicates can only be known to be false if they are explicitly negated. It is assumed that plans can execute in parallel, but the planner does not take this into account. So plans can alter predicates in parallel, but the planner does not take this into account when constructing plans. Eventually when the world has changed in unexpected ways (according to the planner) it can cope with this change. Finally a task can have three states: pending, executing and executed. Before the action starts it is pending, when it starts it is executing and when it is done (terminated) it is executed.

To construct plans some constraints have to be met. The model that is responsible for checking these constraints within the planner is the CFOR model, which is a constraint-posting planner. One type of constraints we can have are ordering constraints. So the actions within a plan can have a partial ordering relationship. It is a binary relation between tasks. We could have the constraint for some plan: before(T1,T2). This denotes that action T1 has to start before action T2. Interval protection constraints (IPCs) denote that some predicate has to hold in the interval between two tasks. IPC(T1,P,T2) denotes that predicate P has to hold between the initiation of action T1 and the initiation of action T2. When this is not the case a signal is produced by the planner that the IPC is violated.

Likelihood, unexpectedness and future expectation are all important appraisal variables in EMA and for each event the value of these variables is determined with probability calculus. We can make a distinction between three types of probability. Those are achievement probability, execution probability and repair probability. Achievement probability represents the probability that a goal or pre-condition can be achieved (denoted as P_{ACH}(pre) where pre is the precondition or goal), which also depends on the goal or pre-condition being true or not. If it is already true the probability is one. Otherwise this initial probability is determined a priori. Actions can have multiple effects. The execution probability (denoted as P_{EX}(eff|I) where eff is the effect and I denotes if the action was intended or not) is assigned to such an effect and represents the probability that this effect will arise given that the action that causes it is executed. Plans can also fail and the repair probability (denoted as P_{REP}(pre) where pre is a precondition or goal) gives the probability that a plan (to achieve precondition or goal pre) can be repaired given that the plan has failed. These types of probabilities for effects, preconditions and goals are calculated according to the rules in figure 12. In EMA we say that a goal is established when the effects that are contained within the goal are true. A goal is threatened when there is an effect that can undo the goal before this goal is needed. And we say that an effect is satisfied when it is simply true in the current interpretation of the world. We call an effect an establisher when it establishes a goal.

Probability of an effect: P(eff)

IF State(action(eff)) = Pending THEN

$P(\text{eff}) = P_{EX}(\text{eff}) * P(\text{intend}(\text{action}(\text{eff}))*\prod P(\text{precondition}(\text{action}(\text{eff})))$ (1)

IF State(action(eff)) = Executing AND -satisfied(eff) THEN $P(\text{eff}) = P_{EX}(\text{eff})$ (2)

IF State(action(eff)) = (Executing OR Executed) AND satisfied(eff) THEN $P(\text{eff}) = 1$ (3)

IF State(action(eff)) = Executed AND -satisfied(eff) THEN $P(\text{eff})=0$ (4)

Probability of a goal/precondition: Pr(goal)

IF -established(goal) THEN $P(\text{goal}) = P_{ACH}(\text{goal})$ (5)

IF established(goal) AND -threatened(goal) THEN $P(\text{goal}) = P(\text{establisher}(\text{goal}))$ (6)

IF established(goal) AND threatened(goal) THEN

$P(\text{goal}) = P(\text{establisher}(\text{goal}))[1 - P(\text{threat}(\text{goal}))] + P_{REP}(\text{goal})P(\text{threat}(\text{goal}))$ (7)

Figure 12. The rules to determine probabilities

I will explain some of the equations for effects and all of the equations for goals and preconditions in words. Equation 12.1 (figure 12, equation 1) says that if we have an effect that is caused by an action that is in a pending state then the probability that that effect will arise is the probability that the associated action will be executed times the probability that the action is intended by the agent times the probability that all the preconditions of the action are true. Equation 12.5 says that if a goal has not been established the probability for that goal is equal to the a priori probability that the goal will be achieved. Equation 12.6 says that if a goal has been established and it is not threatened then the probability of that goal being kept established is equal to the probability that the effect that established the goal is being kept satisfied. Equation 12.7 shows the case when we have an established goal that is being threatened by some effect. It could be the case that the threat will simply not occur, that chance is 1 – P(threat(goal)). The other case is the case where the threat does occur, in that case the probability that the goal is kept being established is equal to the probability of the plan being repaired (P_{REP}(goal)P(threat(goal))).

Next to likelihood another important appraisal variable is utility. When we have a plan it can consist out of multiple actions, and those actions may achieve subgoals. If all these subgoals are achieved, it can lead to the final goal of the plan. This final goal can have some utility for the agent. But also the subgoals along the path of the plan can have utility. Lets now take a plan where we have two subgoals s1 and s2 and one final goal f1, this means that the agent first has to achieve s1, then s2 and then it can achieve f1. So it is necessary for the agent to achieve s2 before achieving f1. S2 itself can have some utility for the agent, but next to this direct utility for s2 there is gain in utility because if s2 is achieved it increases the chance of achieving f1. Therefore in EMA a distinction is made between the extrinsic (intermediate) and intrinsic (essential) utility of a goal (based on Sloman 1987 and Beaudoin 1995 as stated in [6]). Instrinsic utility is the utility of the (sub)goal itself, independent of other goals it might help achieving or not. This intrinsic utility can be determined a priori. Determining the extrinsic utility of a subgoal is the sum of the intrinsic utility of goals it might help achieving weighted by the increase in probability for each of these goals given that the subgoal is achieved. The overall utility of the subgoals is then the extrinsic utility plus its intrinsic utility (would the subgoal have this). The equation for determining the extrinsic utility for some subgoal *s* is given in equation 4. We have to take two steps for each goal *g*, where subgoal *s* influences the achievement probability. First we have to obtain the intrinsic utility for each goal *g* whose achievement probability is influenced by achieving subgoal s. And secondly, we have to determine the impact that achieving subgoal s has on the achievement probability of each goal *g*.

$$Extrinsic\_Utility(s) = \sum_{g \in Impact(s)} Intrinsic\_Utility(g)\frac{P(g)[1-P_{REP}(s)]}{P(s)}$$

Equation 4. equation for determining the extrinsic utility for some subgoal s

In equation 4 we can see that we take the set of all the goals whose probability is impacted by achieving subgoal s (g $\in$Impact(s)). For each goal g in that set we take the intrinsic value (*Intrinsic_Utility(g)*) multiplied by the the gain in achievement probability if subgoal s is true ($\frac{P(g)[1-P_{REP}(s)]}{P(s)}$) i.e., P(g) gives us the probability of goal g and P(s) gives us the probability of subgoal s. If we would only have $\frac{P(g)}{P(s)}$, then we separate the contribution of subgoal s from the contribution of all other subgoals that help achieving goal g. So now we have filtered out the contribution of subgoal s from attaining goal g, we want to know the difference of the probability of goal g if subgoal s was currently true (probability of subgoal s is 1) and if subgoal s was not currently true (1-P_{REP}(s)). We could also write this explicitly as:

$$\frac{P(g)*1}{P(s)}-\frac{P(g)*[1-P_{REP}(s)]}{P(s)}$$

$\frac{P(g)*1}{P(s)}$ gives us the case where subgoal s is true and $\frac{P(g)*[1-P_{REP}(s)]}{P(s)}$ gives us the case that subgoal s is false. When we subtract the first from the latter we get the difference in probabilistic contribution for goal g. In equation 4 it is merged as:

$$\frac{P(g)[1-P_{REP}(s)]}{P(s)}$$

Note that P_{REP}(s) does not simply denote the achievement probability of subgoal s, but it represents the repair probability as mentioned before. It comes from probability calculus:

"*Repair probabilities represent the probability that a plan can be repaired given that the current plan fails, denoted P_{REP}(pre) where pre is the precondition of some action or a top-level goal. If there is no existing plan, the repair probability defaults to the achievement probability for that precondition.*" [6, Page 5]

That is why we multiply $\frac{P(g)}{P(s)}$ with [1 – P_{REP}(s)].

To give an example lets say that the P(g) = 0.6 and P(s) = 0.3. Now lets split up the case where we assume that subgoal s is true (1) and the case where subgoal s is false (P_{REP}(s) which is 1 - P(s)): 0.6 * 1 = 0.6 and 0.6 * (1 - 0.3) = 0.42. The difference between the two cases is 0.18 (0.6 – 0.42). This we have to divide by 0.3 (P(s)) which is 0.6. And this is the impact that subgoal s has on the achievement probability of goal g. Now we could do this faster by not separating the two cases, but just do 0.6 * [1 – (1 - 0.3)] in the first step and then divide it by 0.3 (P(s)). And finally we add all of these values calculated for g $\in$Impact(s).

In EMA it is possible to do appraisal from various perspectives, we can do appraisal from the perspective of the agent himself, but we can also do appraisal for an agent from the perspective of another agent i.e., the agent can determine how he thinks another agent will appraise an event. It is only possible to determine the extrinsic utility from the agent's own perspective, for other perspectives the agent can only determine the intrinsic utility. This is for efficiency reasons as stated in [6].

The causal interpretation is an interpretation about the state of the world as the agent thinks it is (person-environment relation). When a state (past,present or future state in the causal interpretation) has enough (dis)utility for the agent himself or for another agent known to him (utility > 1.0) this means the state is relevant for the agent. Therefore a frame is created that describes characteristics about the state (see table 11 equation 9 and 10). In each state (past, present or future) there are events that will help achieving the goal in that state (facilitators) and there are events that will inhibit the goal in that state (inhibitors). For each of those events a subframe is created to appraise these events individually for the agent (see table 11 rule 11 to 15). And each subframe will be converted to an emotional instance according to the mapping in table 12. The exact values of the appraisal variables will determine the intensity of the generated emotional instance. In EMA it is possible to have different emotional instances simultaneously e.g., an agent can have both hope and fear at the same time. In normal decision theory, from these emotional instances an average would be calculated, but in EMA this explicit separation is kept intact when an overall emotional state is calculated. The appraisal rules in table 11 are implemented as Soar elaboration rules. So to summarize: for every (relevant) event in the causal interpretation a subframe is created, which results in an emotional instance. These emotional instances are used to generate an overall emotional state and mood. More details about how this overall emotional state and mood are determined based on the emotional instances is in the next section.

Appraisal Rules		

∀ state∈ causal_interpreation	IF $\|\text{Utility}_{SELF}(\text{state})\| > 1.0$ THEN
	    Create frame F:
	        F:Relevant := True
	        F:State := state;
	        F:Perspective := Self
	        F:Utility := Utility_{SELF} (state)
		(9)

	IF $\|\text{IntrinsicUtility}_{OTHER}(\text{state})\| > 1.0$ THEN
	    Create frame F:
	        F:Relevant := True
	        F:State := state;
	        F:Perspective := Other
	        F:Utility := $\text{Intrinsic\_Utility}_{OTHER}$ (state)
		(10)

∀ F ∈ Frames,
∀ S ∈ F
	IF established(F:State) THEN
	    Create subframe F.S
	        F.S:Type := facilitator;
	        F,S:Cause := establisher(state)
	        F.S:Likelihood := Pr(establisher(state))
	        F.S:Desirability := F:Utility
	        F.S:Coping_Potential := Pr(establisher(state))
		(11)

	IF established(F:State) AND threatened(F:State) THEN
	    Create subframe F.S
	        F.S:Type := inhibitor;
	        F.S:Cause := threat(state)
	        F.S:Likelihood := Pr(threat(state))
	        F.S:Desirability := 0.0 – F:Utility
	        F.S:Coping_Potential := Pr_{REP}(state)
		(12)

	IF unestablished(F:State) THEN
	    Create subframe F.S
	        F.S:Type := inhibitor;
	        F.S:Cause := plan-blockage;
	        F.S:Likelihood := Pr_{ACH}(state)
	        F.S:Desirability := 0.0 – F:Utility
	        F.S:Coping_potential := Pr_{ACH}(state);
		(13)

	IF F.S:Desirability > 0 AND responsibility(F.S:Cause) = agent THEN
	    F.S:Causal_attribution := praiseworthy
	    F.S:Causal_agent := agent
		(14)

	IF F.S:Desirability < 0 AND responsibility(F.S:Cause) = agent THEN
	    F.S:Causal_attribution := blameworthy
	    F.S:Causal_agent := agent
		(15)

Table 11. All the appraisal rules in EMA

Appraisal pattern for proposition “p”	**Emotion**
Expectedness(self,p) = low	Surprise
Desirability(self,p) > 0 & Likelihood(self,p) < 1.0	Hope
Desirability(self,p) > 0 & Likelihood(self,p) = 1.0	Joy
Desirability(self,p) < 0 & Likelihood(self,p) < 1.0	Fear
Desirability(self,p) < 0 & Likelihood(self,p) = 1.0	Sadness
Desirability(self,p) < 0 & Causal attribution(self,p) = other & Controllability(self,p) ≠ low	Anger
Desirability(other) < 0, causal attribution(p) = self	Guilt

Table 12. Mapping from appraisal pattern to emotion label in EMA

EMA also has an attentional model integrated (Frijda and Zeelenberg, 2001; Smith and Kirby, 2001). Whenever a cognitive operation is done on some part of the causal interpretation, the emotional instances related to that part of the causal interpretation come into focus. There could be multiple activities within the causal interpretation e.g., one could refer to an activity that elicits an emotion of anger and another activity could elicit an emotion of happiness. It depends on which activity is in the part of the causal interpretation that the agent accesses, what emotional instance will be in focus. It is the emotion in focus that impacts how an agent will behave at that moment (more about this later).

Cathexis

In Cathexis each basic emotion is represented by a proto-specialist. So in totall we have six proto-specialists (remember the basic emotions in Cathexis are: anger, fear, distress/sadness, enjoyment/happiness, disgust and surprise). Those proto-specialists are in a network with each other and have various kinds of sensors. Emotions can influence other emotions positively or negatively once their intensity reached the activation-threshold. Once the intensity of an emotion reaches this threshold value it becomes active and can influence the other (proto-specialists that represent) emotions. This is realized in Cathexis with excitatory and inhibitory inputs respectively. For each emotion so-called emotion elicitors can be defined. The value of an emotion elicitor is determined by pre-defined conditions for each emotion elicitor. Note that these conditions can be different for each elicitor. A condition is represented by a numeric value multiplied by a weight for that condition. Based on the old intensity of an emotion, the excitatory and inhibitory inputs and the values of the emotion elicitors, the new intensity of an emotion is determined. The equation for determining this intensity is shown in equation 5. Note that it is not prescribed by the framework which emotions will negatively or positively other emotions, and it is thus a task of the user to define this.

$$EI_{it} = \chi\left(\left(\Psi(EI_{i(t-1)}) + \sum_k EL_{ki} + \sum_l (EX_{li} \cdot EI_{lt}) - \sum_m (IN_{mi} \cdot EI_{mt})\right),\ 0,\ \omega_i\right)$$

Equation 5. The equation that gives the intensity of emotion i at time t

The explanation of this equation is also present in chapter 4 of this thesis, but I will repeat it here. In this equation EI_{it} represents the intensity of emotion i at time t. The function χ constrains the value of an intensity between 0 and ω_i. Remember that ω_i is the maximum intensity emotion i can have. $\chi()$ can thus be seen as a sigmoid function, where the last two parameters specify the minimal and the maximal value. $EI_{i(t-1)}$ gives us the intensity of emotion i in the previous timestep and $\psi(EI_{i(t-1)})$ gives us the intensity that is left in the current timestep. Note here that $\psi()$ is the decay function as is explained before. The part of the formula $\Sigma_k EL_{ki}$ adds all the k values of the emotion elicitors for emotion i. $\Sigma_l (EX_{li} * Ei_{lt})$ denotes all the l emotions that add excitatory gain to emotion i i.e., EX_{li} gives the excitatory gain for emotion i from emotion l and multiplies this with the current intensity of that emotion l, Ei_{lt}. And it does the same for all the inhibitory gain from emotions m with the part $\Sigma_m(IN_{mi} * EI_{mt})$. So summarized this function takes the old (decayed) emotion intensity, adds the values of all emotion elicitors, adds the excitatory gain, subtracts the inhibitory gain and puts it in the sigmoid-like function. The value of a specific emotion elicitor is determined by the conditions it depends on. Note that each elicitor can have totally different conditions.

How do emotions relate to moods?

ALMA

In ALMA every agent has an initial default mood that is described by the Big Five model and is cast to a PAD-value. The mood is influenced by emotions. In the ALMA system there is an update process every 500 ms. This process starts by looking at the currently active emotions (PAD-values) which were elicited by events, actions or objects. The PAD-values for the active emotions are input for a so-called push and pull function (See the section about ALMA). This function determines how the current mood is influenced by the currently active emotions. Recall that the mood is also a PAD-value. How the current mood is influenced determines on the currently active emotions and the intensities of those emotions. The push and pull function will determine an average of all active emotions called the virtual emotion center and based on that center the current mood will be moved.

EMA

In EMA an overall emotional state is calculated which corresponds to a mood. The mood is represented by a set of emotion types (Hope, fear etc.). To calculate the overall emotional state we have to take all the emotional instances created (as explained in the previous section) in the current causal interpretation. For each emotion type defined in EMA the intensities of all emotional instances are added to their corresponding emotion type. So an emotional state is described by all the emotion types with their aggregated intensities. These aggregated intensities are called the mood-intensities and are the input for a sigmoid function which maps all intensities to a value from zero to one. So a mood is represented as an aggregate of all the emotional instances in the current causal interpretation. The mood will have an effect on appraisal. When we have an appraisal frame that has, for example, a label hope with intensity X, the mood influences this intensity by taking the mood-intensity of hope and add this to the appraisals intensity: X + mood(hope). As we will see later the (coping) behavior and expressions of an agent are determined by the recently accessed appraisal frame (model of attention) with the highest intensity.

Cathexis

As explained in the previous part about Cathexis, the basic emotions and emotion blends are represented by the proto-specialists in a network. The proto-specialists for an emotion becomes active if the intensity for that emotion reaches or exceeds the activation threshold. Emotions are represented by high levels of intensity in this network. Moods are represented in the same network of proto-specialists, but only by the low intensity proto-specialists. In Cathexis high intensity emotions will decay faster than low intensity emotions. This represents the fact that moods are more stable. Another thing to notice is that the possibility of an emotion becoming active also depends on the mood. Because moods are represented by low intensity levels in the proto-specialists they actually lower the distance to the activation threshold for certain proto-specialists i.e., the distance between the current intensity of a proto-specialist (because of the current mood) and the activation threshold becomes smaller than it would have been if the intensity of the proto-specialist was zero. You can imagine that we have an enormous amount of possible moods in this way.

How do emotions decay?

ALMA

How emotions decay in ALMA can be specified by the user. The user can choose to decay the intensity of an emotion according to three functions: linear, exponential or tan-hyperbolic. And also the time it will take for the emotion to decay completely can be specified i.e., the time it will take for the emotion intensity to drop to zero. This time can vary between 0 and 4 minutes.

EMA

In EMA decay of emotions is very different from the way ALMA realizes it. The intensities of emotions are determined by an aggregate of emotional instances (which are formed by appraisal frames) currently in the causal interpretation. Over time emotional instances will be added and removed from the causal interpretation because the causal interpretation changes over time. So decay of emotions is accomplished by removal of appraisal frames e.g., we could have a causal interpretation at time t where there are two appraisal frames with emotional instances Joy(0.2) and Joy(0.4). At time t the aggregated intensity of Joy is 0.6. But at time t+1 a change has happened in the causal interpretation which removes the appraisal frame with Joy(0.4), thus at t+1 Joy has decayed from 0.6 to 0.2.

Cathexis

Once an emotion becomes active in Cathexis i.e., the intensity of the emotion reaches or exceeds the activation threshold, it will not be active forever. So the emotion will decay over time. Each proto-specialist also contains a decay function $\psi()$. This function defines how an emotion will decay once it is activated. The decay function can be defined for each proto-specialist individually. It is not prescribed by the framework how this function should be implemented and it is the task of the user to define a function to realize realistic decay of emotions.

How do emotions and moods influence behavior?

ALMA

In ALMA we can see the effect of moods in the following forms:

- wording and phrasing
- selection of dialog strategies
- idle gestures
- type and characteristics of conversational gestures and postures
- facial expressions

In the next question of this chapter we will see that ALMA does not include a coping process.

Merely the effect of emotions and moods cannot be seen as coping. Coping comprises more than simply the expression of emotions. Coping is about trying to come or stay in a desired emotional state.

EMA

In EMA the emotions will infuence the expressions of the agent and the actual values of the appraisal variables will determine which coping strategy the agent will choose. I made a distinction between behavior simply as an expression of emotions and coping behavior. The reason for this is that this is one of the most important differences between EMA and the other two frameworks. In ALMA and Cathexis emotions and mood will result in behavior but in EMA coping strategies are used as the behavior. And coping (in EMA) is the reverse of the appraisal process. More about this topic will be explained in the answer to the next question for EMA.

Cathexis

In Cathexis it is the emotional behavioral system that takes care of the influence that emotions and mood have on the behavior of an individual. The different behaviors that result from the emotions and mood are called emotional behaviors. The emotional behaviors are also in a network with each other and each emotional behavior consists out of two components. Those are the experiential component and the expressive component. The expressive component takes care of facial expressions, body postures and vocal expressions. The experiential component covers the motivations for an emotional behavior becoming active or inactive and the action tendency i.e., an emotional behavior will influence which actions the individual will tend to do. An emotional behavior can be represented by various behaviors. If an individual is in an irritable mood his action tendency could be described by various behaviors such as fighting and screaming for example. There are also other components of the behavior that are influenced by emotions and mood, but these are not covered in Cathexis because of complexity reasons and lack of fundamental studies on these subjects according to the authors.

In the network of emotional behaviors at every update the emotional behavior with the highest value is selected as the emotional behavior the individual will have.

$$EB_{it} = \sum_{k} BR_{ki}$$

Equation 6. The equation for determining the value of emotional behavior i at time t

The factors that influence the value of an emotional behavior are called (behavioral) releasers. Examples of releasers are motivations (emotions, moods etc.) and external stimuli. The value of an emotional behavior is determined by equation 6. Here EB_{it} is the value of emotional behavior i at time t, and BR_{ki} is the value of behavioral releaser k of emotional behavior i. How the value of a releaser is determined is not specified and can be done in various ways, it depends on the nature of the releaser in question. The emotional behavior system is kept simplistic intentionally such that it can be used for various action-selection mechanisms.

Does the framework have a coping mechanism?

ALMA

Coping is not covered in ALMA and falls outside the boundaries of this framework. We can see the effect of a mood or emotions on the behavior of an agent, but this is not coping. That is simply expressions resulting from the emotions and moods.

EMA

EMA does have a coping mechanism (which is a work in progress and the authors themselves mention that it is far from complete). As mentioned before coping is influenced by the recently accessed appraisal frames. The appraisal frame with the highest intensity then determines how coping will be done by the agent. In EMA coping has a different meaning than it has in the general sense: "coping strategies are proposed to maintain desirable or overturn undesirable in-focus events (appraisal instances)" [2].

The process of choosing a coping strategy is done in five steps. The first step is identifying a coping opportunity. Whenever a cognitive operation is done such that the causal interpretation changes, a check is done for appraisal that may motivate coping. Therefore a coping elicitation frame is constructed which contains three fields: focus agency, interpretation-objects and max-interpretation. The focus-agency field contains the agent or object that elicited the cognitive operation. The interpretation-objects are all the actions, states and agents in the causal interpretation that were relevant for the cognitive operation. The max-interpretation field contains the interpretation-object with the strongest appraisal intensity. Whenever this intensity is larger than some pre-specified value, the corresponding coping elicitation frame is marked as a coping opportunity.
The second step is elaborating the coping elicitation frame with the social relations between the agents involved and the actual or possible responsibilities of the agents for the event for which the elicitation frame is constructed. The third step is proposing coping strategies based on the contents of the coping elicitation frame. In the fourth step the coping potential for each proposed strategy is assessed by taking the abstract effect a strategy would have on the causal interpretation and check what (expected) influence it would have on the appraisal frame. The strategies are then ranked according to the impact they would have on the appraisal frame. And finally in the fifth step a coping strategy is chosen. It could be the case that multiple strategies can be applied i.e., we have a tie between strategies. To solve the ties EMA adheres to a preference over strategies based on the appraisal variables e.g., when controllability of an event is high an agent will prefer problem-directed coping strategies. When a strategy has been chosen it is actually applied and the causal interpretation is updated accordingly. And if after the application still other strategies are applicable another one is chosen according to the same preferences until no strategies are applicable anymore.

We distinguish four types of coping in EMA: attention-related coping, belief-related coping, desire-related coping and intention-related coping. Attention-related coping makes the agent focus on certain propositions in the causal interpretation i.e., it focuses its attention on specific parts of the causal interpretion (seek information) or it could suppress certain parts (suppress information. Belief-related coping makes the agent alter its beliefs: the agent can shift responsibility of some event (shift responsbility) or he can increase/decrease the probability of some event happening (wishful thinking). Desire-related coping strategies can be mental disengagement or silver lining. The former means that the agent distances itself from some goal such that its utility will drop and the latter means that the agent will add utility to a positive side-effect of a negative action. Finally intention-related coping strategies can be forming an intention to execute a plan or action to achieve a desired goal (planning/action selection), forming an intention to get help from some other agent (seek instrumental support), forming an intention to make something the agent did wrong right again (make amends), delay an intention to some point in the future (procrastination), dropping an intention to achieve a desired state (resignation) and finally avoidance of a threat for the agent by taking some action.

In EMA the appraisal process and coping process are interleaved with each other i.e., appraisal influences which coping strategies are chosen and coping influences the appraisal process again. We thus in general we get a cycle of appraisal and coping.

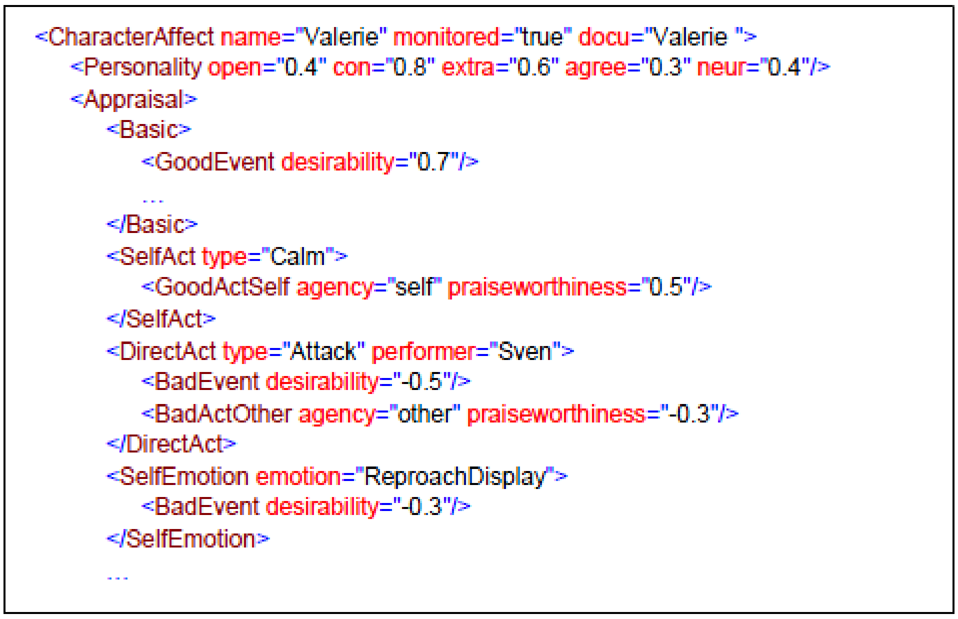
Cathexis

In Cathexis there is no coping mechanism and just as in ALMA it falls outside of the scope of the framework. We only have emotional behaviors that are influenced by emotions and moods, but there is no coping involved.

What has the user to provide for the framework?

ALMA

In ALMA the user of the framework has to provide a character personality specification for each character in the system. It has to be written in AffectML (an xml-based language), which will soon be replaced with EmotionML of the w3 foundation [32]. This file contains a personality description according to the Big Five model and the appraisal rule definitions. In figure 13 we can see a fragment of such a file.



```
<CharacterAffect name="Valerie" monitored="true" docu="Valerie ">
    <Personality open="0.4" con="0.8" extra="0.6" agree="0.3" neur="0.4"/>
    <Appraisal>
        <Basic>
            <GoodEvent desirability="0.7"/>
            ...
        </Basic>
        <SelfAct type="Calm">
            <GoodActSelf agency="self" praiseworthiness="0.5"/>
        </SelfAct>
        <DirectAct type="Attack" performer="Sven">
            <BadEvent desirability="-0.5"/>
            <BadActOther agency="other" praiseworthiness="-0.3"/>
        </DirectAct>
        <SelfEmotion emotion="ReproachDisplay">
            <BadEvent desirability="-0.3"/>
        </SelfEmotion>
        ...
```

Figure 13. A fragment of a character personality specification

EMA

Next to this causal interpretation for an agent there also exists a plan library (Soar is used for planning) and a set of stimulus response rules. The plan library contains action definitions and recipes that define how actions can be combined (preconditions, action description, time-ellapse, effects). These recipes are used for plan generation within an agent (plans can only be generated when controllability is high). The stimulus response rules check whether certain predicates are true at a point in time, once all the predicates in the response rule are true, then an action can be executed (which is defined in the response rule). Examples of such rules are when a sound is percieved (predicate) the action the agent will perform is looking or when the agent percieves a flying bird (in his interpretation) it will try to hit the bird. The whole EMA system must be coupled with a world simulation in which it is defined how actions affect the environment (predicates that describe the world).

Cathexis

Cathexis leaves a lot of space for the user of the framework to fill in. The most important things the user has to define are how emotions are influenced by other emotions (the inhibitory and excitatory inputs) and the emotion eliciting conditions. Also the user has to define the decay functions which can be defined for each emotion individually. The emotional behaviors have to be defined in terms of the expressive component and the experiental component. In Cathexis six basic emotions are defined, but it is mentioned by the authors that the users of the framework are free to add other basic emotions.

Which framework(s) have I decided to use?

In general EMA is by far the most advanced framework for generating emotions and moods. It is influenced by a lot of models (Soar, decision theory, CFOR, STRIPS etcetera) and keeps track of the past, present and future. It even has a planning module to construct plans for the future. These plans are also taken into account when emotions are determined. In EMA also coping is covered, whereas the other two frameworks do not cover a coping mechanism. ALMA and Cathexis are more basic frameworks for generating emotions, but they are more suited to be coupled with 2APL in my opinion. In EMA the appraisal process and the generation of emotions and moods is not that complex and the same holds for the coping process. Most parts of EMA individually are in my opinion satisfying as to how they accomplish their task.

All those parts are also inspired by existing models, but the most significant problem with EMA is that everything is put together in one model. And the authors do not fundamentally justify why this should work. You can not take all the models that have properties you like, put them together and (almost) blindly expect that the result will be a system that has all those desired properties without consequences. I know that the authors altered the models where they based their EMA system on, but the whole system just lacks a fundamental explanation about why a composition as they constructed it should work. After the comparison I decided to use ALMA for the appraisal process and the generation of emotions and moods. And for the coping part I used EMA as inspiration.

I shall give a couple of reasons why I have decided to use ALMA as the basis of my work instead of Cathexis and EMA for appraisal and the generation of emotions and moods. First of all Cathexis only defines what is the relation between emotions and moods. How emotions are generated is something the user has to define on his own i.e., the appraisal process is not defined. There are conditions that influence the elicitation of emotions. But what exactly those conditions are has to be filled in by the user of the framework.

EMA is the most advanced framework. But a problem (apart from the already mentioned problems) is that it is all build around the causal interpretation. Which is something we are not going to use in 2APL. Another problem is that you need a lot of information and you have to define a lot of rules for the planning. In practice I think there is too much overhead and without the causal interpretation and the advanced planning system I do not see a crucial benefit of using the techniques that EMA utilizes as the basis instead of ALMA. The only added value in my opinion that EMA has opposed to ALMA is the coping mechanism. But this can be separated from the appraisal process and the generation of emotions and moods. As I told earlier I will use EMA as an inspiration for the coping process.

With ALMA as the basis for appraisal, emotion and mood generation, I can imagine that the user just has to tag each (internal or external) event, message or action with an appraisal tag. This can be done with a new kind of rule (the E-rules, more about these rules in chapter 7). For each appraisal tag we define an appraisal rule e.g., good_event → P:+0.5;A:+0.0,D:+0.0 (in some cases also L(ikelihood) is used as mentioned earlier). So each appraisal tag will correspond to a PAD-point. The appraisal rules are stored in a rule base (also more about this later). So each time appraisal is done, the appraisal tag is compared to the heads of all appraisal rules and when a match is found, the corresponding PAD-point is stored in the emotion base. In each deliberation cycle we will store all PAD-points. From the PAD-points we calculate the virtual emotion center. And based on this virtual emotion center and the current mood we calculate a new mood with the push & pull function. Note that the emotions and thus the corresponding PAD-points decay over time i.e., the PAD-points intensity will decay until it reaches an intensity of zero. The higher the intensity of a PAD-point the bigger the influence will be on the virtual emotion center. Properties about the agents (personality, decay time of emotions etc.) are defined in a file called Character.xml. As will be described later the deliberation cycle of a 2APL agent will be adjusted. Note that initially an agent will have a default mood (personality). When we do not specify an appraisal tag for an (internal or external) event, message or (abstract) action, it does not influence the mood. This way the programmer can decide which events, messages and actions are relevant for the mood of the agent and which ones are not. Over time the mood will always go back to the default mood for that agent if nothing happens that influences the mood. In chapter 7 we will see how an agent configuration has to be adjusted such that we can generate emotions. Remember that in ALMA we have various appraisal tags (Table 13), which we can use directly in 2APL. Later we will see how these appraisal tags are converted to emotions.

Class	**Appraisal tag**	
event	good_event	bad_event
	good_unlikely_future_event	bad_unlikely_future_event
	good_likely_future_event	bad_likely_future_event
action	good_act_self	bad_act_self
	good_act_other	bad_act_other
object	nice_thing	nasty_thing

Table 13. The appraisal tags used in ALMA

Overview

Appraisal & Emotions	Questions	ALMA	EMA	Cathexis
	Which affective phenomena are distinguished?	Emotions, moods, personality	Emotions, overall emotional state	Basic emotions, emotion blends, moods
	What are the key elements of appraisal/emotion generation?	Appraisal tags	Appraisal frames	Emotion elicitors
	Number of emotions distinguished?	24 (based on modified OCC model)	24 (based on a mapping from Clark Elliott)	6 basic emotions + emotion blends
	Number of discrete moods distinguished	8 (which can vary in intensity)	No discrete moods	No discrete moods
	How are moods represented	A PAD-value	The set of all emotions with intensities	The set of all emotions which have low intensities
Behavior	Is emotional behavior covered	Yes	Yes	Yes
	Is there a coping mechanism?	No	Yes	No
	Is there a distinction between internal and external coping?	No	Yes	No

Table 14. An overview of a comparison between ALMA, EMA and Cathexis

7. Implementation in 2APL

In this chapter I will first describe how appraisal and emotion generation is implemented in 2APL and after that how the coping mechanism is implemented. Note that in this chapter I will not discuss the code. I will do that in chapter 9.

Implementing appraisal and emotion generation in 2APL

As explained earlier, ALMA is used to generate emotions and moods in 2APL. In ALMA scripts are used to generate the appraisal input for the system. These scripts contain utterances from agents where each utterance is tagged with an appraisal tag. The appraisal tags are actually abbreviations of PDA(L) values (see chapter 6 for an explanation about PDA(L)-values). So each appraisal tag can be mapped to a PDA(L)-value e.g. The appraisal tag *good_event* can be mapped to:

- Praiseworthiness: nil
- Desirability: +0.7
- Appealingness: nil
- Liking: nil

If we want to use ALMA with 2APL, we are not going to use scripts to generate appraisal input. Instead we are going to use 2APL programs and that is where the appraisal input should be generated. Below I will explain in detail how and when appraisal data can be generated in a 2APL program.

In a 2APL application from the point of view of an agent various things can happen. External events can arise in the environment, external actions can be performed by the agents themselves or other agents can alter the environment, messages can be send or received and internal events in the agents can happen.

It is already possible in 2APL to react to a certain external event. As an example we can have the following simple PC-rule:

```
event(rainStarted(),env) | true < – { print(“rain started”)}
```

In this rule when the external event rainStarted() is received, the agent will print the line “rain started”. We do not have a test condition in this rule, just true. What we want now is not performing some actions when the event is received, but generate appraisal input i.e., how does an agent appraise the event that it has started raining? For this to be possible I introduce Emotion-rules (abbreviated as E-rules) in 2APL. To give a simple example of such a rule I use the example of the PC-rule above and adjust it such that it becomes an E-rule. Lets assume that we have an agent Lena, and this agent does not like rain i.e., she would appraise this as bad_event. Then we get the following E-rule:

```
externalEvent: event(rainStarted(),env) => bad_event(1.0)
```

So when agent Lena receives a signal that the event rainStarted() has occurred in environment env, then the appraisal input bad_event(1.0) will be generated and passed on to ALMA to be processed. ALMA then generates an emotion based on the input (which influences the mood). Note the number after the appraisal tag (1.0). This number represents the intensity of the tag which can vary from 0.0

to 1.0. The higher (lower) this intensity the greater (smaller) the influence of this tag on the mood of the agent will be. What I presented here for external events, we can also do for internal events, abstract actions, goals and belief queries. For the E-rules to be interpreted correctly I have distinguished five cases which are listed below. If I would not make this distinction the interpreter could not always interpret the rules correctly e.g., a goal could not always be distinguished from an abstract action.

- external event (for the PC-rules)
- internal event (plan failure, for the PR-rules)
- abstract action (for the PC-rules)
- goal (for the PG-rules with head)
- belief (for the headless PG-rules)

External event E-rules can appraise all external events (also incoming messages). Internal event E-rules can appraise all internal events (which are plan failures). Goal E-rules can appraise goal queries and belief E-rules can appraise a belief query. To clarify this I will give an example of all five cases.

- externalEvent: event(rainStarted(),env) => badEvent(1.0)

- internalEvent: @env(east(),_);REST => badEvent(0.4)

- abstractAction: abstractAct(X) => goodEvent(1.0)

- goal: reachHome => goodEvent(0.8)

- belief: racesWon(X) and X > 1 => goodEvent(0.5)

The first rule appraises the external event rainStarted() as a bad event with intensity 1.0. The second rule appraises the failure of the plan @env(east(),_);REST as a bad event with intensity 0.4. In the third rule the abstract action abstractAct(X) is appraised as a good event with intensity 1.0. Note here that if this abstract action is executed for the first time, it is not appraised yet. So we can only use the appraisal data in cases where the abstract action has already been executed at least once. The fourth rule appraises the goal reachHome as a good event with intensity 0.8. So when the goal reachHome is added to the goal base, at each deliberation cycle an appraisal tag of goodEvent(0.8) will be passed to the emotion engine of ALMA until the goal is removed from the goal base. The last rule appraises the query *racesWon(X) and X* > *1* as a good event with intensity 0.5. So the query of the rule is checked against the belief base. As long as the query evaluates to true, at each deliberation cycle an appraisal tag goodEvent(0.5) is produced and passed to ALMA. With these five kinds of E-rules we can appraise all the necessary internal and external events, goals, beliefs and abstract actions.

So now we have seen examples of E-rules where the head of the rule is an external event, internal event, abstract action, goal or belief query. And we have seen how we can use these E-rules to generate appraisal input for ALMA. But there should also be a way to generate appraisal input based on the goals and beliefs of agents i.e., there should be a coupling between the possible cognitive elements and the beliefs and goals of the agent.

Lets assume that agent Lena does not like rain except when it is very hot outside, then she likes to cool down a bit. So lets say that in the belief base the agent keeps track of the current temperature: Celsius(X), where the X represents the current temperature in degrees Celsius. If we take the external event rainStarted() again and let Lena appraise it as bad_event when it is colder than 15 degrees Celsius and appraise it as a good_event when it is warmer than 30 degrees Celsius. We could use the following two E-rules to define this:

externalEvent: (rainStarted(), env) | B(Celsius(X)) & B(X<15) => bad_event(1.0)

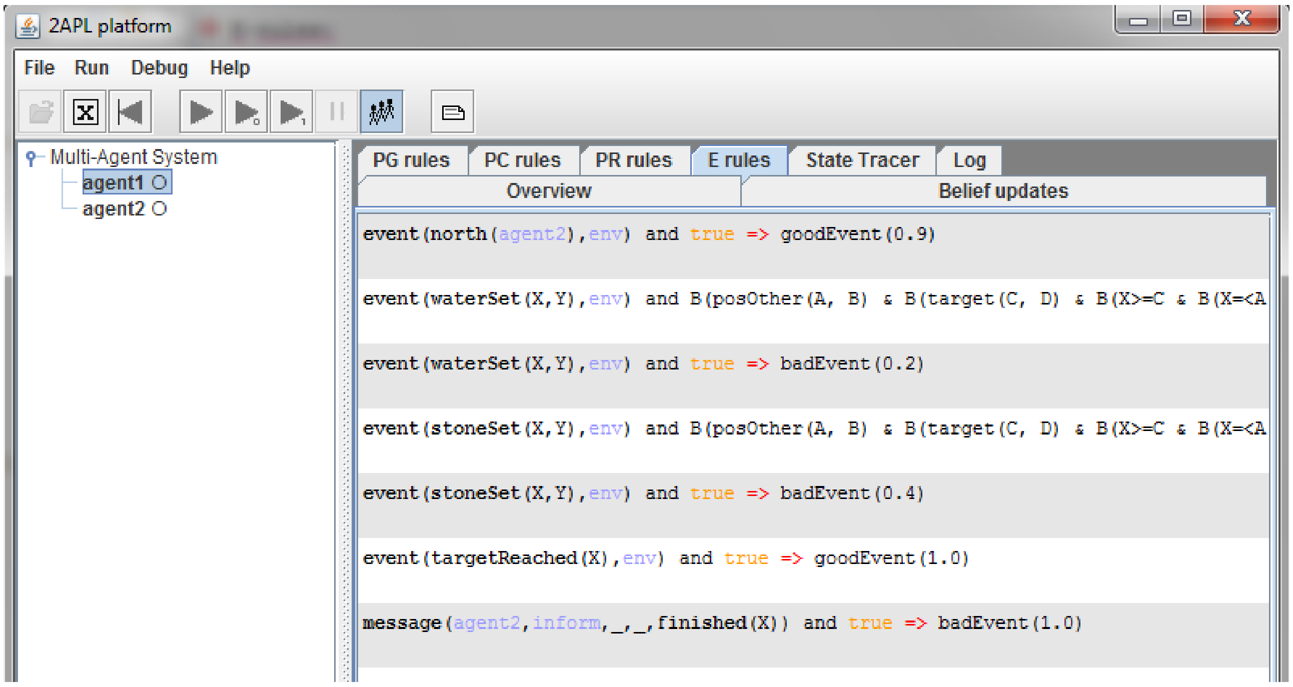
externalEvent: (rainStarted(), env) | B(Celsius(X)) & B(X>30) => good_event(1.0)

As you can probably already see, the first rule specifies the case where it is colder than 15 degrees Celsius and the second one specifies the case where it is warmer than 30 degrees Celsius.

Another example is one where we also incorporate a goal of an agent. Now lets assume that agent Lena needs to reach some target at point (X,Y). We can represent this by G(reach(X,Y)) and an obstacle Stone(A,B) is placed between Lena her own position pos(C,D) by another agent and the target(X,Y), then Lena appraises this as bad_act_other(1.0). For this example we can use the following E-rule:

```
externalEvent: (Stone(A,B),env) | B(pos(C,D))
                                  & G(reach(X,Y))
                                  & B(A >= C)
                                  & B(A <= X)
                                  & B(B >= D)
                                  & B(B <= Y) => bad_act_other(1.0)
```

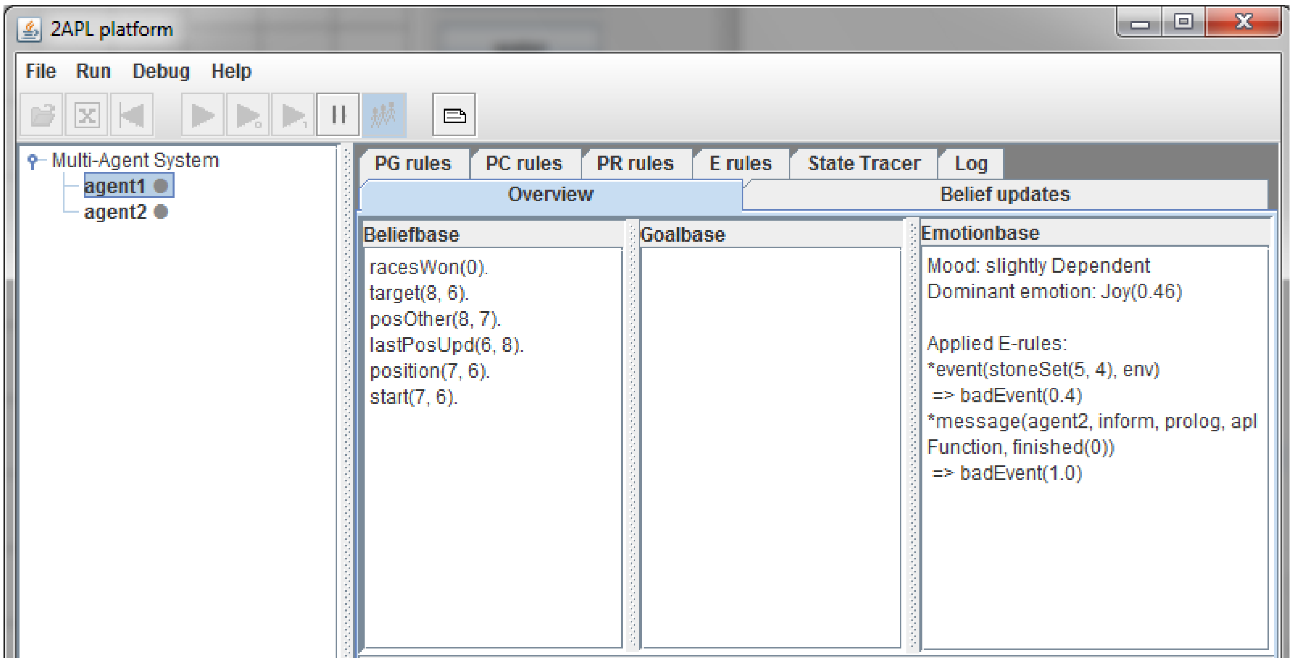
All the E-rules for an agent will be put in the E-rule base of that agent. In figure 14 we can see an example of such a base in the GUI of 2APL.



```
event(north(agent2),env) and true => goodEvent(0.9)
event(waterSet(X,Y),env) and B(posOther(A, B) & B(target(C, D) & B(X>=C & B(X=<A
event(waterSet(X,Y),env) and true => badEvent(0.2)
event(stoneSet(X,Y),env) and B(posOther(A, B) & B(target(C, D) & B(X>=C & B(X=<A
event(stoneSet(X,Y),env) and true => badEvent(0.4)
event(targetReached(X),env) and true => goodEvent(1.0)
message(agent2,inform,_,_,finished(X)) and true => badEvent(1.0)
```

Figure 14. Example of an E-rule base in the 2APL gui

Next to the E-rule base we also have the emotion base. In this base all the applied E-rules are gathered and the current dominant emotion and mood of the agent are also stored here. An example of this base during execution of a 2APL program is shown in figure 15.



```
racesWon(0).
target(8, 6).
posOther(8, 7).
lastPosUpd(6, 8).
position(7, 6).
start(7, 6).
```

```
Mood: slightly Dependent
Dominant emotion: Joy(0.46)

Applied E-rules:
*event(stoneSet(5, 4), env)
 => badEvent(0.4)
*message(agent2, inform, prolog, apl
Function, finished(0))
 => badEvent(1.0)
```

Figure 15. Example of the Emotionbase

We can see that the mood of the agent is slightly dependent and the dominant emotion is joy with intensity 0.46. We can also see which E-rules have been used during execution of the 2APL program. In this case we can see that two E-rules have been applied. The first one is instantiated when a stone was set on location (5,4) and the second one was instantiated when a message was received from agent 2.

Implementing a coping mechanism in 2APL

This section will describe how I incorporated the coping process in 2APL. Next to appraisal and generation of emotions and moods, we have the coping process i.e., how will the agent cope with emotions and thus how will emotions influence the behavior of an agent. Instead of adding an extra step to the deliberation cycle that a 2APL agent iterates through, the coping process is an overarching process over multiple deliberation steps as can be seen in figure 16. Note that the coping system always connects an emotion to a specific event, action or object. Thus coping strategies will always be with respect to one of these phenomena i.e., coping strategies are not influenced by the mood in this mechanism. In most coping theories choosing a coping strategy depends on the emotion connected to some specific phenomenon. I got my inspiration from the EMA framework and [26]. Note that coping can have influence on the mood. Depending on the coping strategies executed by the agent different emotions can be generated which will influence the mood.

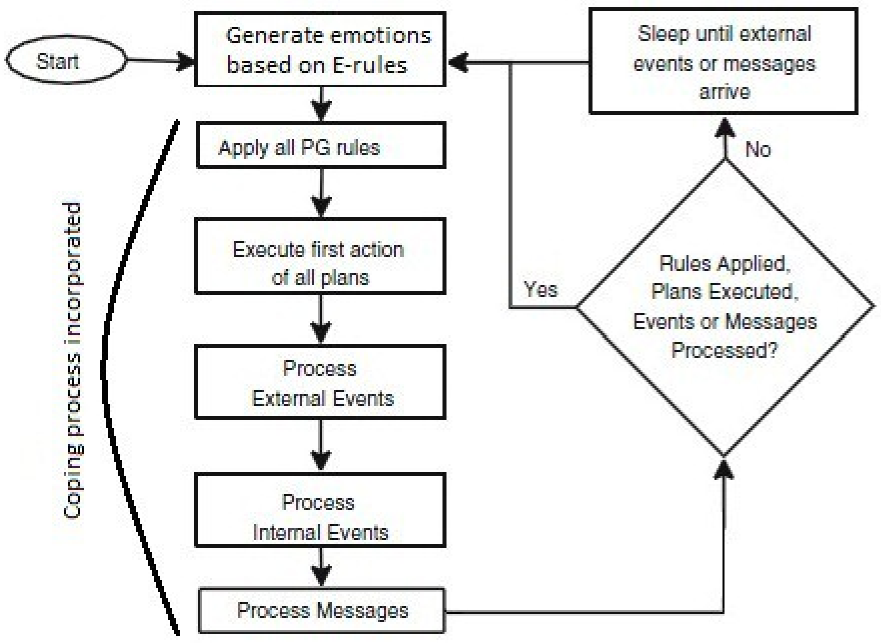

Start
Generate emotions based on E-rules
Apply all PG rules
Execute first action of all plans
Process External Events
Process Internal Events
Process Messages
Coping process incorporated
Sleep until external events or messages arrive
No
Yes
Rules Applied, Plans Executed, Events or Messages Processed?

Figure 16. The deliberation cycle of 2APL with emotion extension

What basically happens is that in the first step “Generate emotions based on E-rules” all external and internal events, abstract actions, goals and beliefs are appraised according to the E-rules (see previous section for a thorough explanation of E-rules). And based on this appraisal, emotions are generated in the ALMA emotion engine. Thus each emotion is connected to some event. This data is stored so it can be used in the next deliberation steps. Note again that the generated emotions will influence an overall mood of the agent, but we will not use the mood for deciding which coping strategy to use.

To clarify I will give a simple example. With the E-rules we can appraise internal and external events, goals, beliefs and abstract actions. A simple example of such a rule is the following:

externalEvent: event(rainStarted(),env) => badEvent(1.0)

This E-rule states that if the external event rainStarted() is received by the agent from environment env, this should be appraised as a bad event with intensity 1.0. This appraisal tag is passed on to the ALMA emotion engine which will generate an emotion based on this tag (distress in this case). The emotion is then passed back to 2APL and is stored as a mapping:

<event(rainStarted(),env) → distress>

We now have the connection between the event and the emotion it has caused within the agent. I call this collection of mappings the coping data (more about this later).

So if we arrive at the deliberation step “Process External Events” we can use this data. In the remainder of this chapter lets assume that we are working with an agent that has an introvert personality (the personality of an agent is specified in an xml-file that has to be given for each agent in the multi-agent system). Now lets have a look at the following PC-rules:

event(rainStarted(),env) :: introvert, joy ← true | { print(“rain started, which caused joy”); π} event(rainStarted(),env) :: introvert, distress ← true | { print("rain started, which caused distress"); π'}

event(rainStarted(),env) ← true | { print("rain started, no emotion generated for this event"); π''}

In the previous version of 2APL without emotions, when the deliberation step "Process External Events" was executed, the collection of all received external events was iterated over and for each event it was checked if the head of a PC-rule matched with this event i.e., with match in this case we mean that the head of the rule could be unified with the actual event. If this was the case then the guard-condition was evaluated against the belief base. If this guard evaluated to true, then the plan of the PC-rule was added to the plan base. Of course in the above examples, all the guards are just true. Nothing has changed in 2APL about this part. It works the same as before.

The difference is that an extra check is done at the evaluation of each rule. The first check is the same, namely the head is checked for a match with the event. After that (if it is specified in the rule, which is optional) it is checked if the personality in the rule matches the personality of the agent. Another new check is that the emotion in the rule (which is also optional) is checked against the collected coping data. So what this means is that we search the head of the rule in the coping data. When this head is found, the corresponding emotion is retrieved (an example of this will be given later). When this emotion matches the emotion specified in the PC-rule, only the guard has to be evaluated as before and when this evaluates to true, the rule is applied. So to sum it up here I will give an overview of the steps that are taken. Blue marks the steps taken in the old situation and red marks the added steps in the new situation.

1. Check the head for a match (with possible substitutions)
2. Check if the guard evaluates to true (it is tested against the belief base)
3. Check if the personality of the rule matches the personality of the agent (optional)
4. Check if the emotion matches the emotion in the coping data (optional)
5. If everything above is true, add the plan to the plan base with possible substitutions

It is possible that multiple rules are applicable, but only the first applicable rule is applied in each deliberation cycle. So a programmer should keep in mind that the rules are checked from top to bottom. Thus it should always be the case that the default rule (the one without a personality and emotion specification) is listed under the corresponding rules that have a personality and emotion specification.

In the example we only covered E-rules in relation to PC-rules. But E-rules can also be used for PG-rules and PR-rules. For both cases I will give an example. I will start with an E-rule for the PG-rules.

goal: reachHome => goodEvent(1.0)

This rule is applied when the goal reachHome is encountered in the goal base. Note that the rule is being applied in each deliberation cycle as long as reachHome is encountered in the goal base. The appraisal tag goodEvent(1.0) will cause the emotion engine of ALMA to generate an emotion instance of joy. With this in mind lets define four PG-rules:

reachHome :: introvert, fear ← true | { print("go home in fear"); π}

reachHome :: extravert, joy ← true | { print("go home with joy and being extravert"); π'} reachHome :: introvert, joy ← true | { print("go home with joy and being introvert"); π''}

reachHome ← true | { print("go home without emotions about this goal"); π'''}

As told before, the agent has an introvert personality and in the coping data the emotion joy is connected to the goal reachHome. Recall that the coping data is a collection of mappings from a head of a PG, PC or PR-rule to an emotion. A mapping can be one of five types:

- goal query → emotion (PG-rule)
- belief query → emotion (Headless PG-rule)
- external event → emotion (PC-rule)
- abstract action → emotion (PC-rule)
- internal event → emotion (PR-rule)

For each type of mapping I specified between brackets to which kind of rule they are connected e.g., a PC-rule can have either an abstract action or an external event in the head of the rule. Note that a headless PG-rule does not have a head, in that case we store the guard (which is a belief query) in the mapping instead of the head (which is always an instance of true in the case of a headless PG-rule).

To check if a rule in which an emotion is specified is applicable we have to check if the emotion in the rule matches the emotion in the coping data. What I mean with this is that we look at the (instantiated) head of the rule, and we try to find this head in the collection of mappings. When we find this head as a key of some mapping, we retrieve the corresponding emotion of that mapping. An example of the coping data for goals could be the following:

{reachHome → joy, reachSchool → distress, goSwimming → joy}

So if we now have the following PG-rule:

goSwimming :: introvert, joy ← true | π

And we want to check if the emotion in the rule matches the emotion in the coping data, we look in this data for a mapping with the key goSwimming and check what the corresponding emotion is, which in this case is joy. So in this case the emotion in the coping data matches the emotion in the PG-rule. Thus the PG-rule will be applied.

If we now go back to the four PG-rules defined above for the goal reachHome, the first rule will fail at the emotion check, which is fear and this does not match with our coping data. In the second rule the emotion does match with the coping data, only the personality of the rule does not match the personality of the agent so this rule is also not being applied. When we arrive at the third rule we see that the personality matches and the emotion also matches the coping data. As can be seen the guard is just an instance of true and so the third rule is being applied in this case. Would none of the three rules match the personality and emotion we always have the default rule, which is the fourth case. This one is always being applied if no other rule matches. And the programmer should always keep in mind to specify such a default rule for each case. Otherwise the behavior of a 2APL program can become very unpredictable.

The last example will be about the PR-rules. I first give an example of an E-rule for appraisal of an internal event (plan failure) and then I will give some PR-rules and explain which one will be selected. The E-rule is as follows:

internalEvent: B(winner);REST => badEvent(1.0)

If a plan failure will arise of a plan that matches (can be unified) the head of this E-rule an emotion instance of distress will be produced by ALMA as output for the input badEvent(1.0). So in the coping data a mapping is added <B(winner);REST → distress>. Now lets take four PR-rules for this example:

B(winner);REST :: extravert, joy ← true | { π }

B(winner);REST :: introvert, distress ← true | { π' }

B(winner);REST :: introvert, distress ← true | { π'' }

B(winner);REST ← true | { π''' }

Maybe you are a little bit confused by the fact that I have defined two PR-rules that have the same head, personality and emotion (the second and the third rule), but this is not an error. I think it is clear by now that one of these two rules will be applied. But which one, the second or the third rule? Well, all the rules are tried from top to bottom and only the first applicable rule will be selected. So in this case the second rule will be applied. When we have a tie between rules as is the case here, it is the first rule in the program that is being evaluated that is being applied. So note here that the third rule could never be applied.

With these examples I have also covered the deliberation steps “Apply all PG rules” and “Process Internal Events”. Note that we can also only specify the personality or the emotion in a rule like this:

reachHome :: introvert, _ ← true | { print(“go home with an introvert personality”); π}

reachHome :: _, fear ← true | { print(“go home in fear”); π}

In the above examples of PG-rules we can see that in the first case we left out the emotion. This means that it is only checked if the personality matches the agent's personality, it is not checked which emotion the head of the rule is mapped to in the coping data. In the second rule we left out the personality. So in this case we do not check the personality of the agent, we only look at the emotion of the rule and if it matches the mapping in the coping data.

As you may have noticed I have not discussed explicit coping strategies yet. What I have described is a mechanism that adjusts the behavior of the agent according to its personality and emotions with respect to events, goals, beliefs and abstract actions. The next step is to devise templates (*.2apl files) in which coping strategies are described. In this template it can be described for example that if an agent has fear for a certain internal event, what his coping strategy will be (with a PR-rule). The 2APL programmer can use these templates as guidance, but of course he or she can also decide to devise their own coping strategies. An example of such a template will be described in chapter 10 where a demonstration application is discussed.

8. Updating the operational semantics of 2APL

In this chapter I will describe the adjustments I have made to the existing transition rules for applying PG-rules, PC-rules and PR-rules in 2APL. Note that a lot that has been written here has already written in [12], but I added explanation about the additions I made to the transition rules together with the already explained parts for clarity. Everything that has been added to the transition rules is marked with the color green.

Planning goal rules

Let $\kappa :: \eta, \omega \leftarrow \beta \mid \pi$ be a variant of a PG-rule for agent ι. Where κ is the head of the rule, η the personality of the rule, ω the emotion of the rule, β the guard of the rule and π the body of the rule. For PG-rules the head can be either an abstract goal or it can be an instance of true. When the head of the rule is an instance of true, we call the rule a headless PG-rule. Because beliefs and goals can persist through time, a decision had to be made when to apply these PG-rules. I have decided to apply the PG-rules with a goal in the head only when the following holds:

- The goal in the head can be unified with a goal in the goal base
- The personality in the rule matches the personality of the agent (optional)
- The emotion in the rule matches the emotion from the coping data (optional)
- The guard of the rule evaluates to true
- There is not already a (partial) plan in the plan base generated for the same goal by the same rule

But note that if we have a PG-rule like:

goToPos(X,Y) ← true | π

Then we can have two plans in the plan base generated by the same rule for goToPos(3,5) and goToPos(6,7), because these are two different instantiations of the same abstract goal (thus they are two different goals).

When we are looking at headless PG-rules, the rule can only be applied when the following holds:

- The personality in the rule matches the personality of the agent (optional)
- The emotion in the rule matches the emotion from the coping data (optional)
- The guard of the rule evaluates to true
- There is no (partial) plan in the plan base generated by this PG-rule (independent of the instantiation of the rule)

So when we have a headless PG-rule like this:

true ← temperature(X) | π

And it is instantiated with the substitution [X/25], a plan π is added to the plan base. So as long as this plan is in the plan base, the PG-rule cannot be applied for any instantiation of the rule i.e., so also not for temperature(10) or temperature(30). Note that this choice has not been made by me, but by the original creators of 2APL as described in [12]:

"A PG-rule of the second type (i.e., a PG-rule with `true` *as the head) can be re-applied only if the plan generated by this rule is executed and removed from the plan base, regardless of the specific instantiation of its belief condition."*

If we now look at the transition rules for applying PG-rules we can see multiple checks. The first one is checking if κ is entailed by the goal base, then we look if the personality matches the personality of the rule, we look if there is coping data available for the specific head in the rule and β is entailed by the agent's belief base. And finally it is checked if there does not exist a plan in the plan base π* that has been generated by the same rule for the same instantiated goal $\kappa'\tau_1$. In this transition rule P denotes the set of all possible plans, id is a unique identifier for a plan in the plan base and r' = κ' :: η, ω ← β' | π' is a variant of r i.e., the same rule but with fresh variable names to avoid problems with identical variable names when applying substitutions. Also note Equals, which is a boolean operator for comparing two personalities.

In the remainder of this section we assume that ρ is the personality of the agent, σ the belief base, γ the goal base, Π the plan base, θ the list of substitutions, ε = <E,I,M> the event base, Equals is a boolean operator that compares two personalities and χ is the coping data in the form of a mapping from (instantiated) abstract action/goal/belief/internal event or external event to emotion and Σ is the emotion base to which all appraisal tags of applied E-rules are added. In ε, E represents the external event base, I the internal event base and M the message base. Assume also in the remainder of this section that GetEmotion is a function that takes as arguments the coping data and the head of a rule. This function searches through the coping data and checks if it can find a mapping of which the key (head of a rule) matches with the current rule. If a match is found in the coping data, the corresponding emotion of that mapping is returned. Otherwise if no match can be found in the coping data, this function returns ⊥.

$$\frac{\gamma \models_g \kappa'\tau_1 \;\&\; \mathrm{Equals}(\eta,\rho) \;\&\; \mathrm{GetEmotion}(\chi,\kappa'\tau_1) = \omega \;\;\&\; \sigma \models \beta'\,\tau_1\,\tau_2 \;\&\; \neg\exists\pi^*\in P : (\pi^*, (\kappa'\tau_1 :: \eta, \omega \leftarrow \beta \mid \pi), id') \in \Pi}{<\iota,\rho.\sigma,\gamma,\Pi,\theta,\varepsilon,\chi,\Sigma> \rightarrow <\iota,\rho,\sigma,\gamma,\Pi \cup \{(\pi'\tau_1\,\tau_2, (\kappa'\tau_1 :: \eta, \omega \leftarrow \beta \mid \pi), id)\},\theta,\varepsilon,\chi,\Sigma>}$$

Note that after application of a PG-rule the instantiated plan $\pi'\tau_1\,\tau_2$ is added to the plan base, together with the rule $\kappa'\tau_1$ :: η, ω ← β | π and a unique plan id. Note also that in the rule κ' is instantiated and β and π are not (and η and ω are never instantiated). For each plan we keep track of the rule that generated it. At every deliberation cycle when a PG-rule is applicable it is checked whether a plan generated by the same rule is already in the plan base, that is why we keep track of the rule for each plan in the plan base. Because the head of the rules in the plan base are instantiated the same PG-rule can be applied for different instantiated goals because the rules have different heads, but it cannot be applied when a headless PG-rules' plan is already in the plan base. To clarify this I will give an example.

win(X) ← true | { π }

true ← won(X) & X > 2 | { π' }

Above we see two PG-rules. The first one will be activated when an instantiated version of win(X) will be added to the goal base (the guard is just true). Now lets say that two separate goals are added to the goal base: win(2) and a goal win(3). Two separate plans will be added to the plan base with the rule that generated the plans and a plan id:

$$\Pi \cup \{(\pi[X/2], (win(2) \leftarrow true \mid \pi), 0)\} \cup \{(\pi[X/3], (win(3) \leftarrow true \mid \pi), 1)\}$$

Note that the rules in the plan base are two different rules because the heads of the rules have been instantiated. If we look at the headless PG-rule and we assume that the guard evaluates to true because we have won(3) in the belief base, the following entry will be added to the plan base:

$$\Pi \cup \{(\pi'[X/3], true \leftarrow won(X)\ \&\ X > 2 \mid \pi',3)\}$$

Because the guard and the plan of the rule are not instantiated, the same PG-rule cannot be applied for any other instantiation of that rule.

So each entry $(\pi'\tau_1\tau_2, (\kappa'\tau_1 :: \eta, \omega \leftarrow \beta \mid \pi), id)$ stays in the plan base until the plan $\pi'\tau_1\tau_2$ is fully executed or removed from the plan base. Also note that personality and emotions are not affected by substitutions.

Procedure call rules

abstract actions

Let $r = \varphi :: \eta, \omega \leftarrow \beta \mid \pi$ be a variant of a PC-rule of agent ι, φ the head of the rule in the form of an abstract action, η the personality of the rule, ω the emotion of the rule, β the guard of the rule and π the plan of the rule. And let α be an abstract action, Unify an operator that returns the most general unifier of two abstract actions if they are unifiable and otherwise it will return $\perp$. The χ represents the coping data, in this case it is a mapping from abstract action to emotion.

$$\frac{Unify(\alpha\theta,\varphi) = \tau_1\ \&\ Equals(\eta,\rho)\ \&\ GetEmotion(\chi,\varphi\tau_1) = \omega\ \&\ \sigma \models \beta\tau_1\tau_2\ \&\ \gamma \models_g G(r)}{<\iota, \rho, \sigma, \gamma, \{(\alpha, r, id)\}, \theta, \varepsilon,\chi,\Sigma> \rightarrow <\iota, \rho, \sigma, \gamma, \{(\pi\tau_1\tau_2, r, id)\}, \theta, \varepsilon,\chi,\Sigma>}$$

Since the abstract action may contain variables that are bound by the substitution θ, we apply θ to it before unifying it with the head of the PC-rule.Note that the resulting substitution τ_1 is applied to the belief condition of the PC-rule before testing it against the agent's belief base. This test may result in a second substitution τ_2, which together with τ_1 is applied to the plan of the PC-rule before replacing the abstract action with the plan. The applications of substitutions to the belief condition and the plan of the PC-rule ensures that the value of the shared variables are passed from the head of the rule to the belief condition, and subsequently to the plan of the rule.

When none of the PC-rules of an agent are applicable (i.e., a PC-rule is not applicable if either its head cannot be unified with the abstract action or the personality of the agent does not match the rules' personality or the emotion of the rule does not match the coping data emotion or its belief condition is not entailed by the belief base), then we assume the abstract action has failed to execute. Whenever an abstract action fails, it remains in the plan base of the agent and an exception is generated which is added as an id to the internal event base I. Let α be an abstract action and PC be the set of PC-rules of agent ι.

$$\frac{\forall(r'=\varphi' :: \eta, \omega \leftarrow \beta' \mid \pi') \in PC : (Unify(\alpha\theta,\varphi') = \perp \text{or } \sigma \not\models \beta' \text{ or } \\ \neg Equals(\eta,\rho) \text{ or } GetEmotion(\chi,\varphi') = \perp)\ \&\ \gamma \models_g \varphi'}{<\iota,\rho,\sigma,\gamma,\{(\alpha,r',id)\},\theta,<E,I,M>,\chi,\Sigma> \rightarrow <\iota,\rho,\sigma,\gamma,\{(\alpha,r',id)\},\theta,<E,I \cup \{id\},M>,\chi,\Sigma>}$$

external events and messages

Let φ :: η, ω ← β | π be a variant of a PC-rule of agent ι, φ the head of the rule in the form of an external event or a incoming message, η the personality of the rule, ω the emotion of the rule, β the guard of the rule, π the plan of the rule, χ be a mapping from event or received message to emotion (also referred to as coping data) of agent ι.

$$\frac{\psi \in E \cup M \;\&\; Unify(\psi, \varphi) = \tau_1 \;\&\; Equals(\eta,\rho) \;\&\; GetEmotion(\chi,\psi) = \omega \;\&\; \sigma \models \beta\, \tau_1\, \tau_2}{<\iota,\rho,\sigma,\gamma,\Pi,\theta,\varepsilon,\chi,\Sigma> \rightarrow <\iota,\rho,\sigma,\gamma,\Pi',\theta,\varepsilon',\chi,\Sigma>}$$

where id is a new unique plan identifier, r = (true :: η, ω ← β | π), Π' = Π ∪ {(π τ$_1$ τ$_2$,r,id)} and ε' = <E \ {ψ}, I, M> if ψ = event(ϕ,env) or ε' = <E,I,M\{ψ}> if ψ = message(s,p,l,o,e).

If there is no PC-rule of which the head is unifiable with an event or message from the agent's event base (*E* or *M*), then the event or message will be removed from *E* and *M*, respectively.

$$\frac{\begin{array}{c}\psi \in E \cup M \;\&\; \forall(\varphi :: \eta, \omega \leftarrow \beta \mid \pi) \in PC : (Unify(\psi,\varphi) = \bot \text{or } \sigma \not\models \beta \text{ or} \\ \neg Equals(\eta,\rho)) \text{ or } GetEmotion(\chi,\psi) = \bot)\end{array}}{<\iota,\rho,\sigma,\gamma,\Pi,\theta,\varepsilon,\chi,\Sigma> \rightarrow <\iota,\rho,\sigma,\gamma,\Pi,\theta,\varepsilon',\chi,\Sigma>}$$

where PC is the set of PC-rules of agent ι, ε' = E \ {ψ}, I,M if ψ = event(φ, env) or ε' = E, I,M \ {ψ} if ψ = message(s, p, l, o, e).

Plan repair rules

Let π$_1$ :: η, ω ← β | π$_2$ be a variant of a PR-rule of agent ι, η the personality of the rule, ω the emotion of the rule, π$_1$ is the abstract plan that should be matched at a plan-failure, π$_2$ is the abstract plan that should be instantiated with the substitutions that results from matching π$_1$ with the actual failed plan and β is the guard of the rule. PlanUnify(π,π$_1$) is a plan unification operator for unifying plan π with abstract plan π$_1$. The plan unification operator returns a tuple *(τ_d, τ_p, π^*)* where τ_d is a substitution that binds domain variables, τ_p is a substitution that binds plan variables, and π^* is the postfix of π that did not play a role in the match with π_1 (e.g., *PlanUnify(α(a)*; *α(b)*; *α(c), X*; *α(Y)) = (*[*Y/b*], [*X/α(a)*]*, α(c))*). The χ represents the coping data, in this case it is a mapping from a plan to an emotion.

$$\frac{PlanUnify(\pi,\pi_1) = (\tau_d, \tau_p, \pi^*) \;\&\; Equals(\eta,\rho) \;\&\; GetEmotion(\chi,\pi_1\tau_d\tau_p) = \omega \;\&\; \sigma \models \beta\, \tau_d\, \tau_p \;\&\; id \in I}{<\iota,\rho,\sigma,\gamma,\{(\pi,r,id)\},\theta,\varepsilon,\chi,\Sigma> \rightarrow <\iota,\rho,\sigma,\gamma,\{(\pi_2\tau_d\tau\tau_p;\pi^*,r,id)\},\theta,<E,I\backslash\{id\},M>,\chi,\Sigma>}$$

Note that substitutions τ_d, τ_p (resulted by the plan unification operator) and τ are applied to abstract plan expression π_2. The resulted instantiated plan $\pi_2\tau_d\tau\tau_p$ followed by the unmatched rest plan π^* form a new plan that replaces the failed plan.

Let PR be the set of plan repair rules of agent ι. If there is no rule in PR applicable to repair a failed plan (i.e., the head of the rule cannot match with the failed plan or its belief condition is not entailed by the belief base), then the exception is deleted from the event base and the failed plan remains in the plan base.

$$\frac{id \in I \ \&\ (\pi,r,id) \in \Pi \ \&\ \forall(\pi_1 :: \eta, \omega \leftarrow \beta \mid \pi_2) \in PR : (\text{PlanUnify}(\pi,\pi_1) = \perp \text{or } \sigma \not\models \beta \text{ or } \neg\text{Equals}(\eta,\rho)) \text{ or GetEmotion}(\chi,\pi_1) = \perp)}{<\iota,\rho,\sigma,\gamma,\Pi,\theta,<E,I,M>,\chi,\Sigma> \rightarrow <\iota,\rho,\sigma,\gamma,\Pi,\theta,<E,I\backslash\{id\},M>,\chi,\Sigma>}$$

Emotion rules

External events/Messages

Let $\varphi \mid \beta => \lambda$ be a variant of an E-rule, where φ is an external event, β is the guard and λ is an appraisal tag. The ζ represents the emotion generated by ALMA based on the appraisal input λ. Let UnifyEEvent be an operator that checks if two external events are unifiable, if so it results in the substitution τ_1, otherwise it results in $\perp$. Remember that χ represents the coping data i.e., a mapping from abstract action/belief/goal/internal event or external event to emotion.

$$\frac{\psi \in E \cup M \ \&\ \text{UnifyEEvent}(\varphi,\psi) = \tau_1 \ \&\ \sigma \models \beta\tau_1\tau_2}{<\iota,\rho,\sigma,\gamma,\Pi,\theta,\varepsilon,\chi,\Sigma> \rightarrow <\iota,\rho,\sigma,\gamma,\Pi,\theta,\varepsilon',\chi',\Sigma'>}$$

Where $\varepsilon' = <E \setminus \{\psi\},I,M>$ if ψ = event(x, env) or $\varepsilon' = <E,I,M \setminus \{\psi\}>$ if ψ = message(s, p, l, o, e), $\chi' = \{\varphi_{\tau 1} \rightarrow \zeta\} \cup \chi$ and $\Sigma' = \{\lambda\} \cup \Sigma$. In the transition rule above we see that if for an external event/message ψ there is a unification possible with the head of an E-rule and the guard is entailed by the belief base, then the E-rule is applied i.e., the appraisal tag of the E-rule is added to the emotion base.

Internal events

Let $\varphi \mid \beta => \lambda$ be a variant of an E-rule, where φ is an internal event, β is the guard and λ is an appraisal tag. The ζ represents the emotion generated by ALMA based on the appraisal input λ. Let UnifyIEvent be an operator that checks if two internal events are unifiable, if so it results in the substitution τ_1, otherwise it results in $\perp$.

$$\frac{\psi \in I \ \&\ \text{UnifyIEvent}(\varphi,\psi) = \tau_1 \ \&\ \sigma \models \beta\tau_1\tau_2}{<\iota,\rho,\sigma,\gamma,\Pi,\theta,\varepsilon,\chi,\Sigma> \rightarrow <\iota,\rho,\sigma,\gamma,\Pi,\theta,\varepsilon,\chi',\Sigma'>}$$

Where $\varepsilon' = <E,I \setminus \{\psi\},M>$, $\chi' = \{\varphi_{\tau 1} \rightarrow \zeta\} \cup \chi$ and $\Sigma' = \{\lambda\} \cup \Sigma$. The transition rule here is almost the same as the one for external events, only here we get the event from the internal event base I instead of E or M.

Goals

Let $\varphi \mid \beta => \lambda$ be a variant of an E-rule, where φ is an abstract goal, β is the guard and λ is an appraisal tag. The ζ represents the emotion generated by ALMA based on the appraisal input λ. And let UnifyGoal be an operator that checks if two goals are unifiable, if so it results in the substitution τ_1, otherwise it results in $\perp$.

$$\frac{\gamma \models \psi \;\&\; \mathrm{UnifyGoal}(\varphi,\psi) \models \tau_1 \;\&\; (\sigma,\gamma) \models \beta\tau_1\tau_2}{<\iota,\rho,\sigma,\gamma,\Pi,\theta,\varepsilon,\chi,\Sigma> \rightarrow <\iota,\rho,\sigma,\gamma,\Pi,\theta,\varepsilon,\chi',\Sigma'>}$$

Where $\chi' = \{\varphi_{\tau 1} \rightarrow \zeta\} \cup \chi$ and $\Sigma' = \{\lambda\} \cup \Sigma$. In the above transition rule we see that if some goal follows from the goal base and it is possible to unify this goal with the head of the E-rule and the guard of the rule follows from the belief base and goal base then this E-rule is applied.

Beliefs

Let $\varphi \mid \beta \Rightarrow \lambda$ be a variant of an E-rule, where φ is a belief query, β is the guard and λ is an appraisal tag. The ζ represents the emotion generated by ALMA based on the appraisal input λ. And let UnifyBelief be an operator that checks if two belief queries are unifiable, if so it results in the substitution τ_1, otherwise it results in $\perp$.

$$\frac{\sigma \models \psi \;\&\; \mathrm{UnifyBelief}(\varphi,\psi) \models \tau_1 \;\&\; (\sigma,\gamma) \models \beta\tau_1\tau_2}{<\iota,\rho,\sigma,\gamma,\Pi,\theta,\varepsilon,\chi,\Sigma> \rightarrow <\iota,\rho,\sigma,\gamma,\Pi,\theta,\varepsilon,\chi',\Sigma'>}$$

Where $\chi' = \{\varphi_{\tau 1} \rightarrow \zeta\} \cup \chi$ and $\Sigma' = \{\lambda\} \cup \Sigma$. In the transition rule we see that if some belief-query ψ follows from the belief base and this query can be unified with the head of the E-rule and the guard of the rule follows from the belief base and goal base then this E-rule is applied.

9. Modifications to the code of 2APL

In this part I will explain some of the most important extensions and modifications I have done to the source code of 2APL. Note that a lot of code that I have added is not explained here. If you want to know everything about the extensions you should look in the source code of 2APL. Knowledge of the 2APL language and Java are required before reading this chapter and knowledge about the 2APL source is also recommended. It is only necessary to read this part of the documentation if you want to know the technical details about the modifcations I made to 2APL.

Extending the deliberation cycle

The deliberation cycle in 2APL is a cycle that every agent in a 2APL program iterates through. To clarify it figure 17 gives an overview of the original deliberation cycle. As you can see there we start with the step "Apply all PG rules". In general in this step we look at the PG rules of the agent and check which PG rule is applicable. And all the plans of the applicable PG rules are added to the plan base. In the next step "Execute first action of all plans", the first action of every plan in the plan base is executed. Note that plans are not executed consecutively, but the first action of every plan is executed in one cycle. In the step "Process External Events" all the received external events of that cycle are handled according to the PC-rules of the agent. And in the step "Process Internal Events" the same is done for internal events according to the PR-rules of the agent. Note that internal events are plan failures. In the step "Process Messages" all the received messages of the agent in that cycle are processed also according to the PC-rules. After this step as long as rules were being applied, plans are in the plan base or event or messages are received we go to the next iteration. Otherwise the agent's delibration cycle will go into a sleeping-state. For more details about each step I refer to [12].

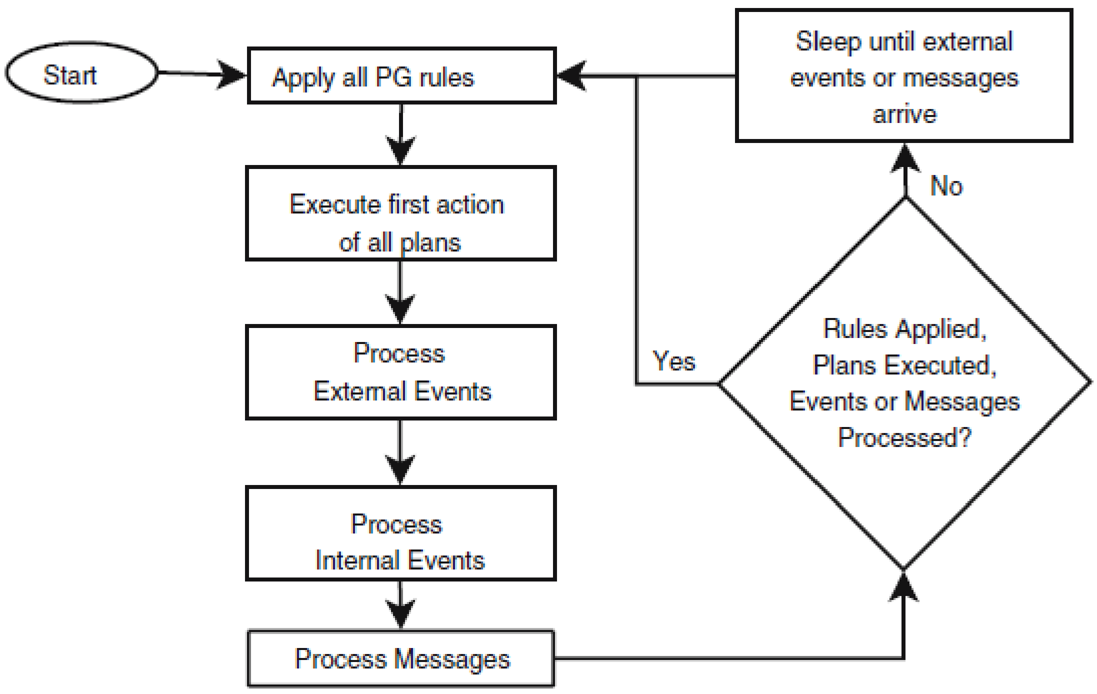

Start
Apply all PG rules
Execute first action of all plans
Process External Events
Process Internal Events
Process Messages
Rules Applied, Plans Executed, Events or Messages Processed?
Yes
No
Sleep until external events or messages arrive

Figure 17. The original deliberation cycle of 2APL

Now we have discussed the orignal deliberation cycle I will explain which adjustments I have made to this cycle (also see chapter 8 for more details). In figure 18 we see this extended deliberation cycle. The first thing to notice is that a step is added, called “Generate emotions based on E-rules”. In this step abstract actions, external events, internal events, goal updates and belief updates can be appraised with the E-rules. The appraisal data will be stored for later use. As can be seen from figure 18, the coping process influences multiple deliberation steps (again see chapter 8).

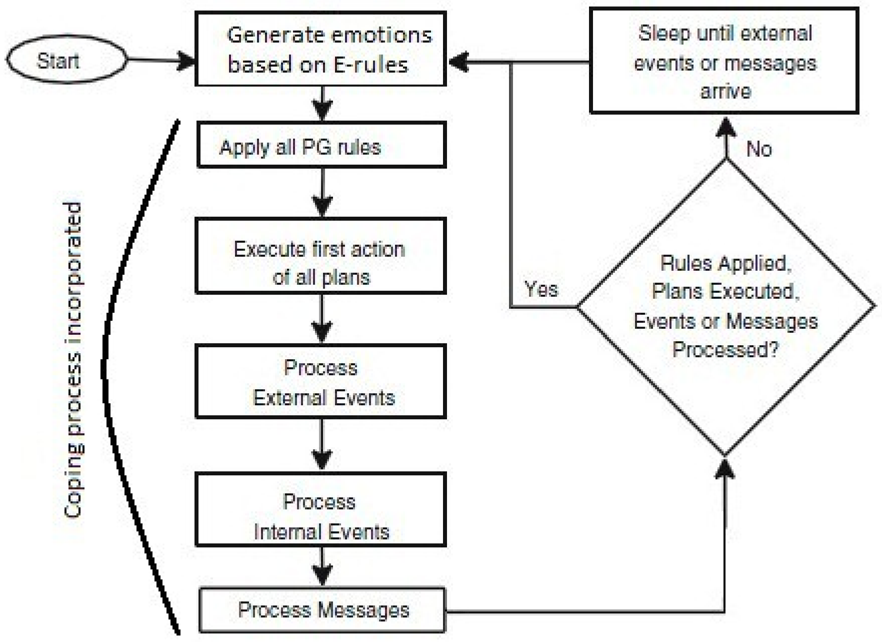

Start
Generate emotions based on E-rules
Apply all PG rules
Execute first action of all plans
Process External Events
Process Internal Events
Process Messages
Coping process incorporated
Sleep until external events or messages arrive
No
Yes
Rules Applied, Plans Executed, Events or Messages Processed?

Figure 18. The extended deliberation cycle

As told before I added an extra deliberation step to the cycle for 2APL agents, called “Generate emotions based on E-rules*”*. Every deliberation step in the cycle is implemented with a class that extends an interface called *DeliberationStep*. The only purpose of this interface is guaranteeing that every deliberation step has a method *execute(APLModule module)*. In this function the real work of the deliberation step is done. An APLModule represents the internals of one agent (belief base, goal base etc.) in the 2APL application. As explained before, each individual agent in a 2APL application goes through the deliberation cycle.

When the deliberation step is done, all processed and unprocessed E-rules are stored in a result object of type “*GenerateEmotionsResult”*. But lets continue with explaining the code in the execute-method of the step “Generate emotions based on E-rules”. The function is very large, so I left some parts out of the fragment (fragment 1). As already explained in chapter 7, there are five types of E-rules:

- external event E-rules
- internal event E-rules
- abstract action E-rules
- goal E-rules
- belief E-rules

What basically happens in the execute function is that we check for each type of E-rule, which ones can be applied. The order in which we do this is first checking it for goals, then beliefs, internal events, external events (also messages) and finally for abstract actions. I will explain how this E-rule selection works for goals and external events. In the other cases it works the same in general.

```java
public DeliberationResult execute( APLModule module )
{
         //Initiation of various bases etc.
         [...]

         //First check which goal E-rules can be applied
         try
         {
                  ArrayList<Erule> rules = erules.selectRules
                                           (module,beliefs,goals,theta);
                  if (rules != null && rules.size() > 0)
                  {
                        for(Erule r : rules)
                        {
                              Erule copy = r.clone();
                              copy.applySubstitution(theta);

                              result.addProcessed(r.getCondition(),r,theta);

                              emo = emotions.addAppraisalTags
                                   (module.getAgentname(),copy.getHeadQuery()
                                   ,copy.getCondition(),copy.getAppTags());

                              module.addAppraisedGoal(
                              new CopingData<Query>(copy.getHeadQuery()
                              ,new Emotion(emo.getName().toString())));
                        }
                  }
            }
            catch(NoRuleException exc){}

            //Check which belief E-rules can be applied
            try
            {
                  [...]
            }
            catch(NoRuleException exc){}

            //Process all logged plan-failures of the last cycle
            for(PlanSeq plan : module.getFailedPlans())
            {
                  [...]
            }
            module.getFailedPlans().clear();

            APLFunction e = null;
            //Process the external events of the last cycle
            while((e = module.getEEvents().poll()) != null)
            {
                  theta = new SubstList<Term>();
                  rule = null;

                  unfreshVars = new ArrayList<String>();
                  try
                  {
                        rule = erules.selectRule(module,beliefs
                                    ,goals,e,unfreshVars,theta);
                  }
                  catch (NoRuleException exc) {}
60.
61.                  if (rule != null)
62.                  {
63.                        result.addProcessed(e,rule,theta);
64.                        Erule copy = rule.clone();
65.                        copy.applySubstitution(theta);
66.                        emo = emotions.addAppraisalTags
67.                              (module.getAgentName(),e,copy.getCondition(),
68.                               copy.getAppTags());
69.                        module.addAppraisedEEvent(new CopingData<APLFunction>
70.                              (e,new Emotion(emo.getName().toString())));
71.                  }
72.                  else
73.                  {
74.                        module.addAppraisedEEvent
75.                              (new CopingData<APLFunction>(e,null));
76.                        result.addUnprocessed(e);
77.                  }
78.            }
79.
80.            APLMessage m = null;
81.            Messenger msgr = module.getMessenger();
82.            String name = module.getLocalName();
83.            //Process the received messages of the last cycle
84.            while((m = msgr.receiveMessage(name)) != null)
85.            {
86.                  [...]
87.            }
88.
89.            Plan p = null;
90.            /*Process all other executed actions (abstract actions, belief
91.              updates, messages sendings etc.) of the last cycle*/
92.            while((p = module.getActions().poll()) != null)
93.            {
94.                  [...]
95.            }
96.            return(result);
97.         }
```
Fragment 1. The execute method of the deliberation step " Generate emotions based on E-rules"

I will start with explaining the application of goal E-rules (fragment 1, line 7-30). In line 9-10 the function selectRules is called which will check which goal E-rules are applicable. In this case that means that it simply checks for each goal E-rule if the goal in the head of the rule is entailed by the goal base. If this is true, the rule will be put in the resulting ArrayList "rules". So selectRules will result in a possible collection of E-rules, together with a list of substitutions in the variable theta. The substitutions are the result of matching the goals in the head of the E-rules with the actual goal base and matching the test-query of the E-rule with both the goal base and the belief base (see chapter 9 for the operational semantics). If it is the case that at least one goal E-rule is applicable (fragment 1, line 11), we will iterate through this list of rules (fragment 1, line 13) and clone the current E-rule (line 15). This is because we do not want to instantiate the original E-rule (we might want to use it later on again), but a copy of it.

Then we apply the substitutions theta generated in the selectRule method to this copy (line 16). In each deliberation step some result is produced, which contains processed and unprocessed goals, external events etcetera e.g., with a processed goal I mean a goal for which an E-rule is applied and an unprocessed goal a goal for which no E-rule is applied. In line 18 the goal together with the applied E-rule for that goal and the substitution generated by the application of the E-rule are stored in the processed part of the result. In line 20-22 we add the applied (instantiated) E-rule to the emotion base, so that we can display it in the GUI of 2APL. The emotion base in its turn delivers the appraisal tag of the applied E-rule to ALMA as input, which will update the intensities of the emotions of the agent. The emotion that is affected the most by the input (this is determined within the function emotions.addAppraisalTags(..)) is the return-type of the function call and is stored in the variable *emo*. In line 24-26 we add the appraised goal to the coping data in the form of a mapping from goal to emotion (the mapping is represented by the CopingData-object). Note that we will use this coping data in later deliberation steps of the cycle.

Another type of E-rule application that I will explain, is that of external event E-rules (fragment 1, line 48-77). In line 48 we poll the external event collection of the module. External events are logged in a (linked) list which we obtain with calling module.getEEvents(). And note that we poll this list, which means that we take the first element of the list, store it in variable e and remove it from the list. As long as an external event is in this list we iterate through the while-loop. In line 50 we refresh the substitution list to an empty list and in line 51 we set the variable rule to null and in line 53 we set the variable unfreshVars also to a new empty list. Note that this list keeps track of already used variable names, that way we do not get duplicate variable names which could lead to errors in the substitution process. Then we arrive at a try-catch block (fragment 1, line 54-59) in which we will try to find an applicable external event E-rule for the current external event (note that we go through the received external events one by one). We will do this with a call to the function selectRule (line 56-57). If an applicable E-rule is found for the current external event this rule will be returned and stored in the variable *rule*. Another result from the function is the list of substitutions *theta* resulting from the application of the E-rule. If no applicable E-rule can be found for the current event the catch block is reached (line 59), but note that no consequences follow from this. Then we arrive at an if-else block (line 61-76) in which we check if indeed an applicable E-rule has been found for the current external event. If this is the case we go into the if-branch (line 63-69). In the first line of this branch (line 63) we add the external event to the processed part of the deliberation step result. We do this by adding the actual external event, the applied E-rule for this external event and the resulting substitution theta. In line 64 we create a copy of the E-rule because we do not want to instantiate the original E-rule and in line 65 we apply the list of substitutions theta to the copy of the current E-rule. In line 66-68 we add the applied E-rule to the emotion base and the emotion base will pass the corresponding appraisal tag to ALMA which will generate an emotion as output based on the appraisal tag. This emotion will be stored in the variable *emo*. In line 69-70 we add the external event together with the generated emotion to the coping data in the form of a CopingData-object. If no applicable E-rule was found for the current external event, the external event is also added to the coping data, but with the value null attached to it instead of an emotion (line 74-76). And also the external event is added to the unprocessed part of the deliberation step result (line 77). The application of E-rules for beliefs, internal events, messages and abstract actions works similar to the way it works for goals and external events. You can look at the source code of this project for more details about this.

The coping process

Note that the coping process is incorporated in multiple steps of the deliberation cycle (figure 18) and as can be read in the previous section about the application of E-rules, coping data is collected in that step. In this section I will explain what happens with this collected coping data during the continuation of the deliberation cycle. I will not explain it for every deliberation step because the implementation of the coping process in each step is very similar. The most important part of the coping mechanism is an if-statement. I will explain it for the step “Apply all PG-rules” that can be split up in application of reactive PG-rules (headless PG-rules) and the application of non-reactive PG-rules (PG-rules with a head). I will explain it for the former. Note that the following if-statement is executed for every PG-rule.

```
if(!containsQuery(appliedQueries,pgrule.getGuard())
&&((rulePers == null && ruleEmotion == null)
|| (rulePers != null && ruleEmotion == null &&
    rulePers.equals(currPersonality))
|| (rulePers == null && ruleEmotion != null && emoBelief != null &&
    ruleEmotion.equals(emoBelief))
|| (rulePers != null && ruleEmotion != null && emoBelief != null &&
rulePers.equals(currPersonality) && ruleEmotion.equals(emoBelief))))
{
      plans.add(p);
      planbase.addPlan(p);
      appliedQueries.add(pgrule.getGuard());
      if (onlyone) return plans;
}
```

Fragment 2. The if-statement that is part of the coping process in the deliberation step "Apply all PG-rules"

As you can see in fragment 2 the if statement is of the form *A && (B || C || D || E)*. I will explain the variables A to E one by one. Before I explain this note that A has to be true in any case and either B, C, D or E has to be true. The variable A is replaced with:

```
!containsQuery(appliedQueries,pgrule.getGuard())
```

The function *containsQuery* checks if the guard of the current PG-rule (*pgrule.getGuard()*) is already in the array list *appliedQueries*. If this is the case, this means that there is already a plan in the plan base generated by an identical instantiation of the current PG-rule and we will not add it again to the plan base. Thus we use the "!" symbol to check if there is not already such a plan in the plan base. If this is the case we go to the next check. And that is checking if B, C or D is true. We start with B:

```
rulePers == null && ruleEmotion == null
```

If in a PG-rule a personality is not specified (*rulePers == null*) nor an emotion (*ruleEmotion == null*), the current PG-rule can be applied. If this is not the case we proceed by checking C:

```
rulePers != null && ruleEmotion == null && rulePers.equals(currPersonality)
```

If a personality is specified in the PG-rule (*rulePers != null*) and not an emotion (*ruleEmotion == null*) and this personality is equal to the personality of the agent (*rulePers.equals(currPersonality)*), then the PG-rule can be applied. If this is not the case we proceed to D:

```
rulePers == null && ruleEmotion != null && emoBelief != null &&
ruleEmotion.equals(emoBelief)
```

If a personality is not specified in the PG-rule (*rulePers == null*) and an emotion is specified (*ruleEmotion != null*) and there is an emotion coupled in the coping-data to this particular PG-rule (*emoBelief != null*) and this emotion equals the emotion specified in the rule (*ruleEmotoin.equals(emoBelief)*), the rule can be applied. Otherwise we proceed to E:

```
rulePers != null && ruleEmotion != null && emoBelief != null &&
rulePers.equals(currPersonality) && ruleEmotion.equals(emoBelief))
```

If both a personality and an emotion are specified (*rulePers != null && ruleEmotion != null*) in the rule and an emotion is specified in the coping-data for this PG-rule (*emoBelief != null*) and the personality of the rule equals the personality of the agent (*rulePers.equals(currPersonality)*) and the emotion in the rule equals the emotion from the coping-data (*ruleEmotion.equals(emoBelief)*), the rule can be applied. In any other case the (reactive) PG-rule cannot be applied.

If it is the case that a PG-rule can be applied, the plan of the current PG-rule is added to the plan base (fragment 2, line 10-11). The belief-query (guard) of this PG-rule is added to appliedQueries such that the plan is not added multiple times (line 12). In line 13 we see that if the boolean value *onlyOne* is true the function will return the plans constructed. There are cases where it is only necessary to construct one plan. But in most cases the *onlyOne* variable is false and we will just continue iterating through the PG-rules.

The Affect Engine

The Affect Engine is the part of ALMA where all the input in the form of appraisal tags is gathered and the output in the form of emotions is delivered. The input is produced within 2APL and sent to ALMA which processes the input and delivers output. This output is then sent to 2APL again (figure 19).

The most important function in the Affect Engine class is *update(AffectUpdateEvent event)*. In the Affect Engine an instance of the Affect Manager class is created. This is the core of ALMA that calculates the mood of an agent based on the input it gets. The Affect Engine is declared as a listener of the Affect Manager. So every time the Affect Manager throws an *AffectUpdateEvent*, the *update* method of the Affect Manager is executed. What exactly happens in this method will be explained below.

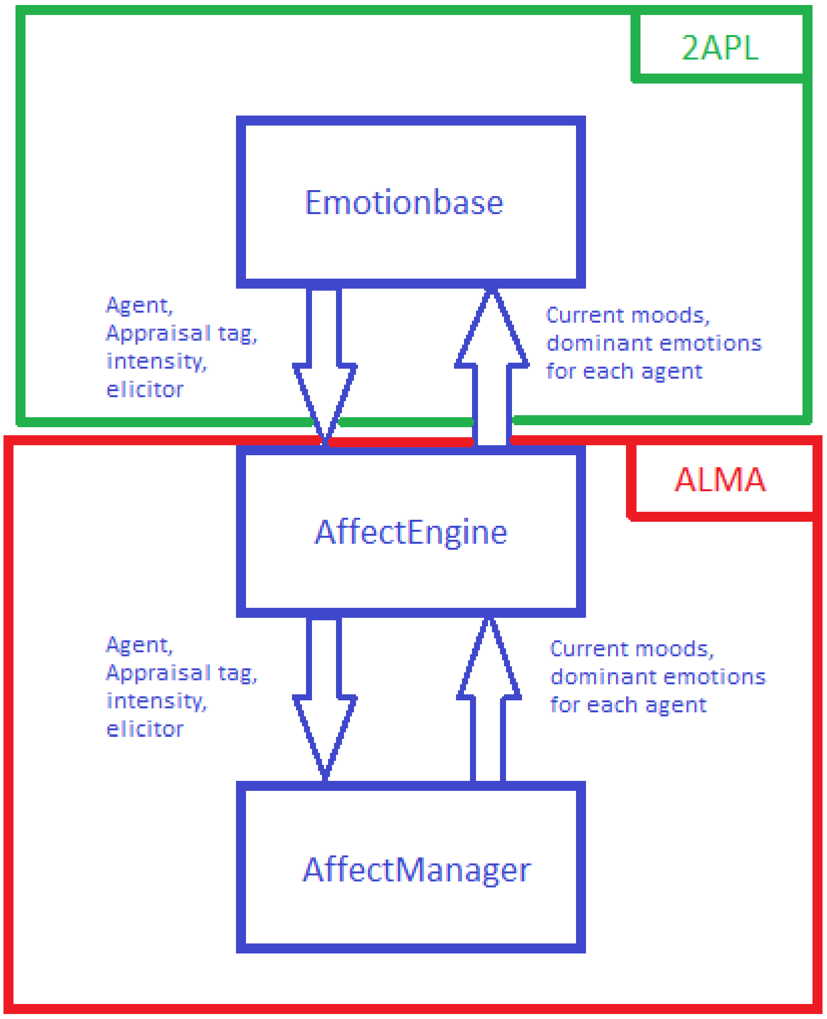

2APL
Emotionbase
Agent, Appraisal tag, intensity, elicitor
Current moods, dominant emotions for each agent
ALMA
AffectEngine
Agent, Appraisal tag, intensity, elicitor
Current moods, dominant emotions for each agent
AffectManager

Figure 19. Connection between 2APL and ALMA

```
        public synchronized void update(AffectUpdateEvent event)
        {
            aod = event.getUpdate();
            try
            {
              int i = 0;
              for(Iterator<CharacterAffect> it = aod.getAffectOutput().
                  getCharacterAffectList().iterator(); it.hasNext();)
              {
                  CharacterAffect character = it.next();
                  Emotionbase ebase = ebases.get(i);

                  String name = character.getName();
                  String emotion = character.getDominantEmotion()
                                       .getName().toString();
                  double eIntensity = Double.parseDouble
                                     (character.getDominantEmotion().getValue());
                  String mood = character.getMood().getMoodword().toString();
                  String mIntensity = character.getMood().
                                         getIntensity().toString();

                  log.info(name + " has dominant emotion " + emotion + "(" +
                           eIntensity + ")");
```

```
23.                 ebase.updateMood(mIntensity + " " + mood);
24.                 ebase.updateDomEmotion(emotion + "(" + eIntensity + ")");
25.                 i++;
26.             }
27.         }
28.         catch (Exception e)
29.         {
30.                 e.printStackTrace();
31.         }
32.     }
```
Fragment 3

In line 3 of the function-block the update-date of the event is put in the variable *aod*. Then we iterate through all the defined characters of ALMA (one for each agent in the 2APL application) in line 7 and 8. Each character is described by a CharacterAffect object which is stored in the variable *character* at each iteration (line 10). After that the corresponding emotion base for that character is retrieved. And we get to the point where the name, dominant emotion, the intensity of that emotion, the mood and the intensity of the mood are retrieved (line 13-20). This data is then passed on to the Emotionbase with the functions *updateDomEmotion* and *updateMood* (line 23-24). Finally the variable *i* is incremented to keep track of each agent's emotion base (line 25).

The emotion base

The emotion base is the part of 2APL where all the applied E-rules are collected and the appraisal input is passed on to ALMA. And from ALMA the emotion base gets input in the form of a dominant emotion (and also the intensities of all other emotions) and mood. The functions that collect the applied E-rules and pass the appraisal data to ALMA are called *addAppraisalTags.* There are five versions of this function (one function for each type of E-rule, page 46). But I will only explain the function for E-rules of type external event:

```
public EmotionType addAppraisalTags(String agent, APLFunction event, Test
                                    condition, ATagSeq tags)
```

I will explain the code in this function in detail and the other four functions are the same except for the type of the elicitor that is collected in the emotion base and sent to the AffectEngine (marked in red). Note that an elicitor is the head of an E-rule.

```
1.          public EmotionType addAppraisalTags(String agent, APLFunction event,
2.                                                   Test condition, ATagSeq tags)
3.          {
4.                EmotionType result = null;
5.                ruleData rd = new ruleData();
6.                rd.condition = condition;
7.                rd.appTag = tags;
8.                emotions.add(new KeyValuePair(event.toString(), rd));
9.                Iterator<ATag> it = tags.iterator();
10.               while(it.hasNext())
11.               {
12.                     ATag t = it.next();
13.                     result = affectEngine.processInput
14.                                 (agent,t,t.getIntensity().toString(),
15.                                  event.toString());
16.               }
17.               return result;
18.         }
```
Fragment 4. The function addAppraisalTag

We start with defining a result object of the type EmotionType and an object rd of type ruleData (line 4-5). In rd we store the condition of the applied E-rule and the appraisal tags of this rule (line 6-7). To the global variable (of the emotionbase-class) emotions we add a new key-value pair where the event is the key and the rule data the value (line 8). The emotions-variable is used to display the E-rules in the emotion base. Then we iterate through all the appraisal tags of the current E-rule (note that a single E-rule can have multiple appraisal tags here, though this has not yet been tested, it is marked as future work) and pass it as input to the affectEngine of ALMA (line 12-15). The functino processInput will return the emotion that has been generated based on the appraisal tags as input. And finally this result is returned (line 17).

The E-rule base

We also have an E-rule base where all the defined E-rules for an agent are stored. In this base we have a function *selectRule.* For the same reason as in the emotion base we have five variants of this function. I will only explain it for one variant:

```
public Erule selectRule(APLModule module, Beliefbase beliefbase, Goalbase
goalbase, APLFunction a, ArrayList<String> unfreshVars, SubstList<Term> theta)
```

Just as in the part about the emotion base I will explain the function only for a subset of the E-rule types. In this case I will explain it for E-rules for external events and E-rules for abstract actions as this function is almost the same for the other variants. They only differ in the red areas in fragment 5. The function in fragment 5 covers both types (external event and abstract action E-rules).

```
1.    public Erule selectRule(APLModule module, Beliefbase beliefbase, Goalbase
2.    goalbase, APLFunction a, ArrayList<String> unfreshVars, SubstList<Term>
3.    theta)
4.    throws NoRuleException
5.    {
6.          boolean norulefound = true;
7.          for (Erule erule : rules) {
8.                if(erule.getType() == ruleType.EEVENT || erule.getType() ==
9.                   ruleType.PLANACTION)
10.               {
11.                     SubstList<Term> theta2 = new SubstList<Term>();
12.                     Erule variant = erule.getVariant(unfreshVars);
13.                     APLFunction head = variant.getHeadAPLFunc();
14.                     Test condition = variant.getCondition();
15.
16.                     if (Unifier.unify(head,a.clone(),theta2))
17.                     {
18.                           norulefound = false;
19.                           if(condition instanceof TrueTest)
20.                           {
21.                                 theta.putAll(theta2);
22.                                 return variant;
23.                           }
24.                           else
25.                           {
26.                                 condition.applySubstitution(theta2);
27.                                 SubstList<Term> theta3 =
28.                                      condition.test(module);
29.
30.                                 if(theta3 != null)
31.                                 {
32.                                       theta.putAll(theta2);
33.                                       theta.putAll(theta3);
34.                                       return variant;
35.                                 }
36.                           }
37.                     }
38.               }
39.         }
40.
41.         if (norulefound) throw new NoRuleException();
42.               return null;
43.   }
```
Fragment 5. The selectRule function of the class EruleBase

In this function we iterate through all the E-rules and only look at E-rules that are of type ruleType.EVENT or ruleType.PLANACTION (fragment 5, line 7-9). E-rules can be of type EEVENT, IEVENT, PLANACTION, GOAL or BELIEF. So when the E-rule is of type EEVENT or PLANACTION, a new substitution list is created and a variant of the current E-rule is created (line 11-12). With variant it is meant that the E-rule is cloned with fresh unique variables, this is necessary for the substitution process. If no fresh variables would be used, duplicate variable names could be used and that can lead to problems. Then the head and the test-condition of the E-rule are stored in *head* and *condition* respectively (line 13-14). With the function *Unify.unify* it is tried to unify the head of the E-rule with the instantiated head of the E-rule given in the parameter list of the function selectRule (in the form of an APLFunction object, line 16). When the unification is possible we go to the next step, otherwise we iterate to the next E-rule. When a unification was possible the next step is looking at the test-condition of the E-rule. If the test-condition is not specified or was just “true”, then we return the variant of the E-rule together with possible substiutions in theta2 (line 20-23). Otherwise the substitutions of unifying the head of the E-rule with the instantiated E-rule are applied to the test-condtion and the test-condition is evaluated (line 26-28). If the evaluation does not result in null it means that the test-condtion results in true and produces new substitutions for the test-condition (line 30-33). Therefore we have two substitution lists i.e., the one produced by the unification of the heads of the rules theta2, and the one produced by evaluating the test-condition theta3. These two substitution lists are thus stored in theta and the variant of the E-rule is returned by the function (line 34). When no applicable rule is found the function just returns null (line 42).

Extending the parser

The parser of 2APL is generated by JavaCC [28]. The input files for JavaCC are *.jj files. In these files the EBNF notation [29] is used to describe which input the parser should process. To be able to parse E-rules in *.2apl files I had to extend the parser.

The first addition to the parser is the description of the syntax of the E-rule base. In the code fragment below we can see that the Erulebase consists of one or more E-rules (Fragment 6).

```
void Erulebase(Erulebase erules) :
{Erule erule;}
{
      (erule = Erule() {erules.addRule(erule);})+
}
```
Fragment 6. Description of the E-rule base with EBNF

The E-rules themselve can be parsed in six different ways. In code-fragment 7 I only show two ways of parsing E-rules. If you understand that it is not difficult to understand they other ways.

```
1.     Erule Erule() :
2.     {Query headQuery; APLFunction headEvent; Plan headPlan; PlanSeq
3.     headPlanFail; Test condition = null; ATagSeq tags;}
4.     {
5.           LOOKAHEAD(<EXTERNALEVENT> PlanAtom() (<VERT> TestConjunction())?
6.                     <DOUBLESTRIPEDARROW> ATagSeq())
7.           <EXTERNALEVENT> headEvent = PlanAtom() (<VERT> condition =
8.            TestConjunction())? <DOUBLESTRIPEDARROW> tags = ATagSeq()
9.           {
10.                if(condition == null)
11.                  return new Erule("externalEvent",headEvent,new
12.                                    TrueTest(),tags);
13.                else
14.                  return new Erule("externalEvent",headEvent,condition,tags);
15.          }
16.        | LOOKAHEAD(<PLANACTION> PlanAtom() (<VERT> TestConjunction())?
17.                    <DOUBLESTRIPEDARROW> ATagSeq())
18.          <PLANACTION> headEvent = PlanAtom() (<VERT> condition =
19.          TestConjunction())? <DOUBLESTRIPEDARROW> tags = ATagSeq()
20.          {
21.                if(condition == null)
22.                   return new Erule("planAction",headEvent,new
23.                                     TrueTest(),tags);
24.                else
25.                   return new Erule("planAction",headEvent,condition,tags);
26.          }
27.          […]
28.    }
```
Fragment 7. Description of the syntax of E-rules with EBNF

If we look at the green part in fragment 7 (line 5-6) we can see that an E-rule for an external event consists of the keyword externalEvent followed by a plan atom, which is the external event itself. This plan atom is optionally (note the "?") followed by a vertical symbol "|" and a test conjunction. The test conjunction is an optional test on the belief base and goal base of the agent. The E-rule ends with a double striped arrow "=>" and a sequence of appraisal tags. The other code makes sure that parts of the E-rules are assigned to their respective variables and send to the correct functions. The LOOKAHEAD function is used to make a distinction between E-rules before they are parsed. Sometimes this is necessary because the start of an E-rule type can overlap with the start of another E-rule type. It actually tells the parser to look ahead literally, so that it is able to make the distinction. In the blue section of fragment 7 (line 16-17) we use this same LOOKAHEAD function. Here we can see that an E-rule for a plan action consists of the keyword planAction followed by a plan atom and an optional "|" symbol and test conjunction. It ends with an "=>" symbol and a sequence of appraisal tags. For clarity the rules described above in fragment 7 are of the following form, where <test-condition> is optional:

externalEvent: <external event> | <test-condition?> => <appraisal-tags>
planAction: <abstract action> | <test-condition?> => <appraisal-tags>

New data structures

A few new data structures have been added to 2APL to describe the different types of data in the package *apapl.data*. We start with the *AffectInputData*-class. Input data for ALMA is described by four fields: character, input, intensity and elicitor. The character describes from which agent the appraisal comes, the input describes which appraisal-tag is used, the intensity describes the intensity of the appraisal tag and elicitor what elicited the appraisal i.e., which event, external action etc. caused the appraisal. All this data is stored in an AffectInputData object and we can retrieve it from this object. Just as in any other data structure class there is no complicated functionality in this class.

Because in an E-rule multiple appraisal tags can be defined, we use the class ATagSeq to represent those sequences of appraisal tags. In this class a LinkedList is created where the appraisal tags are stored.

The AppraisalTag-class is part of ALMA and contains Enums that enumerate all possible appraisal tags. This class also contains functions that convert data to a form such that it is ready to be input for ALMA's Affect Manager. These functions are *makePADInput* and *makeAffectInput*.

```
1.      public AffectInput makePADInput(String character, String p, String a,
2.                                String d, String intensity, String description)
3.      {
4.          AffectInput aiInput = AffectInput.Factory.newInstance();
5.          Character perfCharacter = Character.Factory.newInstance();
6.          perfCharacter.setName(character);
7.          aiInput.setCharacter(perfCharacter);
8.
9.          PAD pad =PAD.Factory.newInstance();
10.         pad.setPleasure((new Double(p)).doubleValue());
11.         pad.setArousal((new Double(a)).doubleValue());
12.         pad.setDominance((new Double(d)).doubleValue());
13.         pad.setIntensity((new Double(intensity)).doubleValue());
14.         pad.setDescription(description);
15.         aiInput.setPAD(pad);
16.
17.         return aiInput;
18.     }
```
Fragment 8. The function AffectInput of the class AffectEngine

In fragment 8 we can see the function makePADInput. This function recieves the characters name, the PAD values (See documentation about ALMA) in String format, the intensity and a short description about this value as parameters. In lines 4-5 of fragment 8 an AffectInput object and Character object are created. Then the name of the character is set to the name in the parameter list of the function (line 6). This is set as the performing character for the PAD-input (line 7). A new instance of a PAD value is created and the P, A and D values are set and so is the intenstiy and description according to the parameter values given to the function (line 9-14). This PAD object is then coupled to the AffectInput object and the object is returned at the end of the function (line 15-17). The data is now ready to be presented to the Affect Manager of ALMA so that it can be processed internally.

```
1.      public AffectInput makeAffectInput(String character, String tag, String
2.                                          intensity, String elicitor)
3.      {
4.          AffectInput aiInput = AffectInput.Factory.newInstance();
5.
6.          Character perfCharacter = Character.Factory.newInstance();
7.          perfCharacter.setName(character);
8.          aiInput.setCharacter(perfCharacter);
9.
10.         if (isEventAppraisalTag(tag)) {
11.           // Building the Event element
12.           Event event = Event.Factory.newInstance();
13.           event.setType(EventTypes.Enum.forString(tag));
14.           event.setIntensity(intensity);
15.           event.setElicitor(elicitor);
16.
17.           aiInput.setEvent(event);
18.         } else if (isActionAppraisalTag(tag)) {
19.           // Building the Action element
20.           Action action = Action.Factory.newInstance();
21.           action.setType(ActionTypes.Enum.forString(tag));
22.           action.setIntensity(intensity);
23.           action.setElicitor(elicitor);
24.
25.           aiInput.setAction(action);
26.         } else if (isObjectAppraisalTag(tag)) {
27.           // Building the Object element
28.           Object object = Object.Factory.newInstance();
29.           object.setType(ObjectTypes.Enum.forString(tag));
30.           object.setIntensity(intensity);
31.           object.setElicitor(elicitor);
32.
33.           aiInput.setObject(object);
34.         } else {
35.           System.err.println("Error creating affect input - no such
36.                                   appraisal tag: " + tag);
37.         }
38.
39.         return aiInput;
40.     }
```
Fragment 9. makeAffectInput function of the AffectEngine class

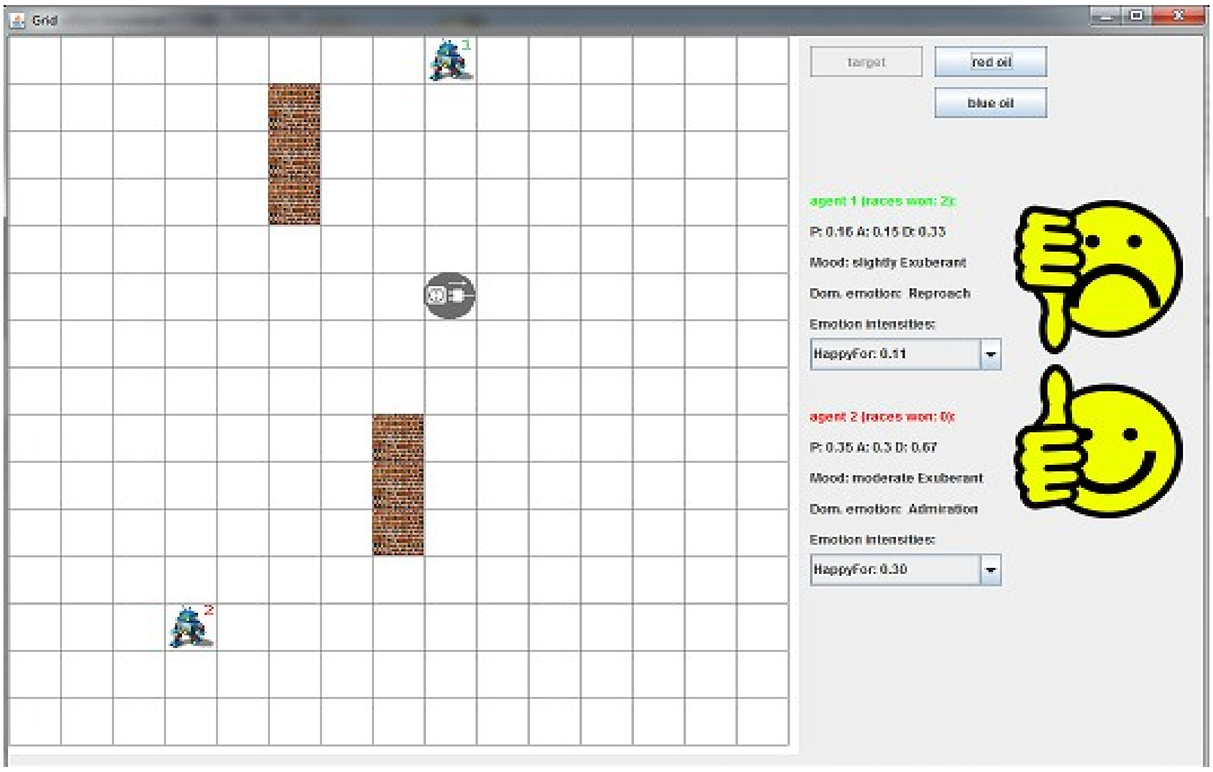
The function in fragment 9 also converts data to be ready for the Affect Manager of ALMA. But the input is different. Here we do not construct a PAD-value, but an appraisal tag. Internally when the data is processed the tag will also be converted to a PAD-value. So in the end it does not matter how you deliver the data to the system, though appraisal tags are easier to work with for programmers. A PAD-value {-0.5,-0.5,-0.5} is the same as the appraisal tag badEvent. As a programmer you should easily see that the latter is more convenient to work with. The first part of this function is the same as the former (line 4-8). The AffectInput object is created and the character is coupled to the input. But then a distinction is made between event appraisal tags, action appraisal tags and object appraisal tags. Based on the type of appraisal tag an Event object, Action object or Object object is created respectively (line 10-33). Then the tag, intensity and elicitor are coupled to it and finally it is attached to the Affect Input object. The AffectInput object is returned by the function (line 39).

10. Demonstration

During the project I had to test the new functionalities that I have added to the 2APL platform. Because of this I developed a 2APL application that exploited all new functionalities of the language. Each time a new functionality was added, I also made sure the test application utilized this new functionality. When the project reached its end the application was developed in such a way that it could demonstrate all new functionalities of the language. So this application could be used for both testing and demonstration purposes. In this chapter I will explain the demonstration application in detail.

I had to come up with an application in which emotions, moods and coping with emotions would play an important role and it had to be in a multi-agent setting. The scenario in the application is one in which we have two robots located in a grid (figure 20). One robot is located in the upper left corner of the grid and the other robot is placed in the lower right corner of the grid. In this grid we can place a target. Once the target has been placed each robot wants to get to this target before the other robot gets there. Once one of the robots has reached the target the race is over and both agents stay at the position where they are. The target can be placed at some other point which initiates a new race. Before the race obstacles can be placed on the grid for the robots. The obstacle can be either barrels, water or walls. If a wall is placed somewhere in the route to the target of a robot, the robot has to come up with a route around the wall. When water is placed on the route of the robot, the robot does not necessarily have to come up with a new route. But the robot does not like water, so if the robot could find a route that is not that much longer than his current route through the water he will probably pick the new slightly longer route. Another aspect that can change the environment where the robots reside in is the placing of oil cans. Before a race starts we can place oil cans in the grid. An oil can can be either red or blue. Robot 1 does like blue oil and dislikes red oil, for the other robot it is the other way around. When a robot comes across oil that he likes, he will consume it. If a robot comes across oil that he dislikes he will leave it where it is. Later we will see how this (coping) behavior is implemented.

The two robots are both implemented with a 2APL agent. Thus we generate appraisal tags with E-rules, the appraisal tags are input for ALMA and it will generate an emotion and calculate the effect on the agent's mood. This data (emotion intensities, dominant emotion, current mood) will be send back from ALMA to 2APL. In the environment we can see the PAD value, mood, dominant emotion and intensities of all emotions for each agent. Within ALMA 24 different emotions are used, but in this demo application I will only use six emotions to keep things clear and simple. The emotions used in the application are joy, distress, admiration, reproach, liking and disliking. Joy and distress will always be caused by events, admiration and reproach are caused by actions of other agents and liking and disliking are caused by objects in the environment. For each of those six emotions I can display an emoticon that represents the current dominant emotion. The six possible emoticons together with a neutral emoticon for the case that there is no dominant emotion are shown in figure 21.



agent 1 (races won: 2):
P: 0.16 A: 0.15 D: 0.33
Mood: slightly Exuberant
Dom. emotion: Reproach
Emotion intensities:
HappyFor: 0.11
agent 2 (races won: 0):
P: 0.35 A: 0.3 D: 0.67
Mood: moderate Exuberant
Dom. emotion: Admiration
Emotion intensities:
HappyFor: 0.30

Figure 20 . The environment of the demonstration application for the new functionalities in 2APL

I will now explain which E-rules have been defined for the agents. Each agent has the same set of E-rules. For each of the six emotions I have defined one or more E-rules. I will start with joy. When an agent reaches the target position an external event targetReached() will be received by the agent. This implies that he has won the race, so this external event will be appraised as a good event (with intensity 1.0). We thus define the following E-rule:

```
externalEvent: event(targetReached(), env) => goodEvent(1.0)
```

So when this E-rule is being applied an appraisal tag goodEvent(1.0) will be delivered to ALMA which will generate the emotion joy. When an agent wins a race he will send a message to the other agent to let him know that he reached the target. The other agent then knows that he has lost the race. Received messages are also external events and of course in this case the received message will be appraised as a bad event, because the agent does not like to lose a race. We thus get the following E-rule:

```
externalEvent: message(agent2,inform,_,_,finished(X)) => badEvent(1.0)
```

When this E-rule is applied the appraisal tag badEvent(1.0) is passed on to ALMA which will generate the emotion distress. Note that this is the E-rule for agent 1, the corresponding E-rule for agent 2 is identical except that the first argument of the message is *agent1* instead of *agent2*. This first argument denotes the addressee of the message.

When the score (number of races won) of an agent is at least two points lower than that of his opponent he will get distressed by this. For this situation the following E-rule has been defined:

```
belief: racesWon(X) and racesLost(Y) and (Y-X) >= 2 and target(C,D) and posOther(A,B)
      and not wallPlaced() => badEvent(0.5)
```

In this E-rule we see that we compare the belief racesWon to the belief racesLost. If the number of races lost is 2 points less than the number of races won the agent will appraise this as a bad event. In a reaction (defined by a PG-rule) he will try to make it harder for the opponent to get to the finish in the next race by placing walls in the squared area formed by the opponent's location and the target location. When robot 1 notes that a wall has been placed between him and the target he will appraise this as a bad act of the other robot. This is defined with the following E-rule:

```
externalEvent: event(wallReceived(X,Y), env) => badActOther(1.0)
```

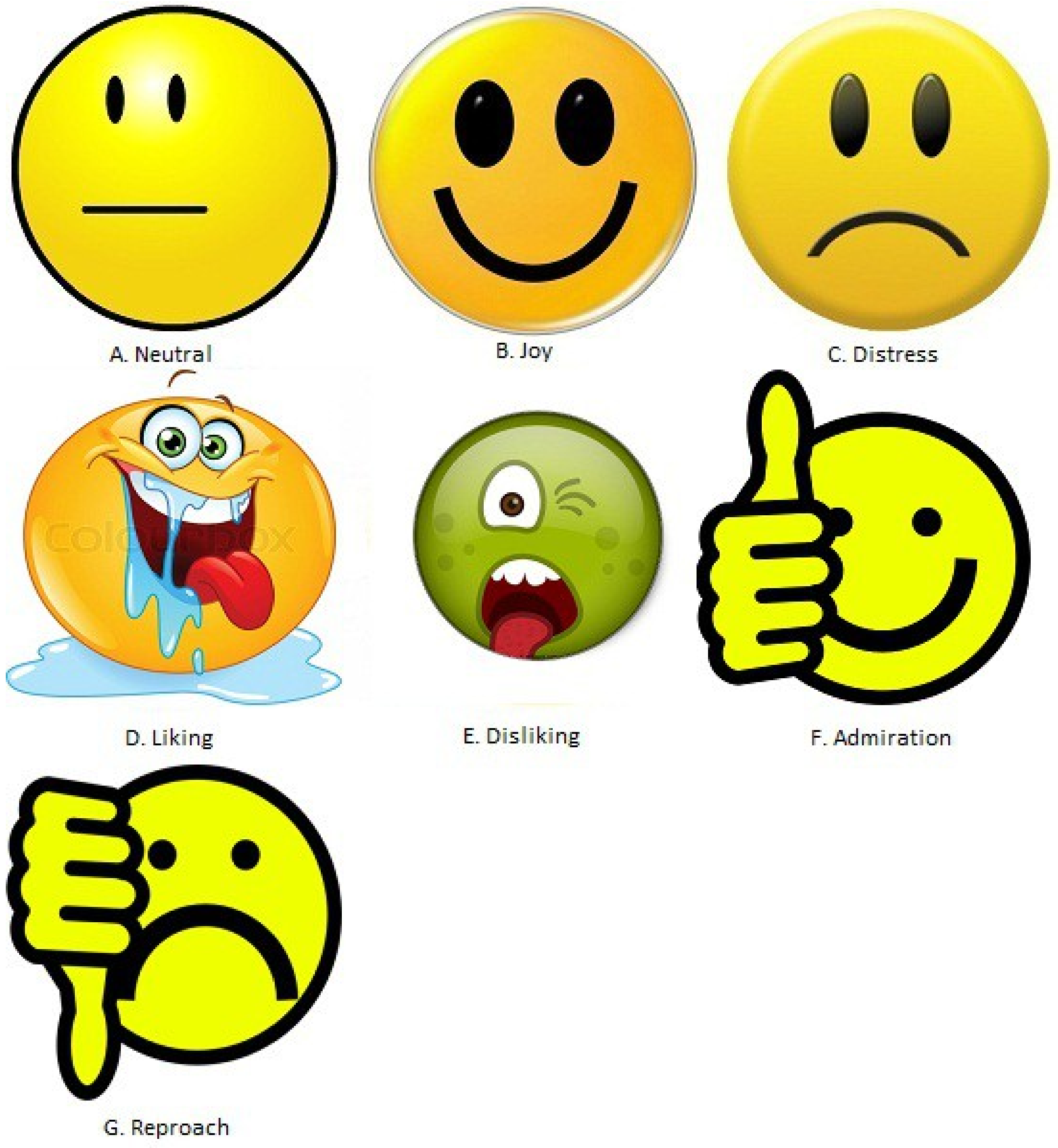


Figure 21. The possible emoticons in the demo application

The appraisal tag badActOther(1.0) will result in the emotion reproach towards the opponent (robot 2). Robot 2 likes a challenge and will appraise the placing of a wall by robot 1 as a good act which will lead to the emotion admiration:

```
externalEvent: event(wallReceived(X,Y), env) => goodActOther(1.0)
```

When a robot comes across an oil barrel he will either leave it or consume it depending on if he likes the oil or not. For this we define two E-rules for each robot. I will show the rules for robot 1:

```
externalEvent: event(redBarrelFound(X,Y), env) => nastyThing(1.0)

externalEvent: event(blueBarrelFound(X,Y), env) => niceThing(1.0)
```

As you can see here robot 1 will appraise red oil barrels as nasty things and blue barrels as nice things. The appraisal tag nastyThing(1.0) will result in the emotion disliking and the appraisal tag niceThing(1.0) will lead to the emotion liking. For robot 2 it is the other way around:

```
externalEvent: event(redBarrelFound(X,Y), env) => niceThing(1.0)

externalEvent: event(blueBarrelFound(X,Y), env) => nastyThing(1.0)
```

Robot 2 will appraise red barrels as nice things and blue barrels as nasty things.

We have now discussed how appraisal and emotion generation is utilized in the application. Another aspect that we need to demonstrate is coping with those generated emotions. The coping strategies used in this demo application are loosely based on a classification created by Frijda [26]. He states that every emotion has a function and will lead to an action tendency. In table 15 I show a fragment of this classification. Here we can see again that a coping strategy is connected to an emotion and not to a mood.

Emotion	**Function**	**Action tendency**
Disliking/Disgust	Protection	Rejection of object
Liking	Consume	Accept object
Joy	Readiness	Free activation

Table 15. A fragment of the classification from Frijda [26]

To give an example, the emotion disliking has the function to protect the agent. The action tendency of the agent will therefore be to reject the object that he dislikes such that his emotional experience of disliking will decay. As mentioned before we can place oil barrels in the environment that the robots can pick up or leave depending on if they like the oil or not. Robot 1 likes blue oil barrels and dislikes red oil barrels i.e., if robot 1 comes across a red oil barrel he will leave it where it is and if he comes accross a blue oil barrel he will pick it up. This is implemented with two PC-rules (and the two E-rules shown above):

```
event(redBarrelFound(X,Y), env) :: _, liking <- true |
{
        @env(pickupRedBarrel(X,Y), _);
}

event(blueBarrelFound(X,Y), env) :: _, liking <- true |
{
        @env(pickupBlueBarrel(X,Y),_);
}
```

In the first E-rule we can see that when a red oil barrel is found at position (X,Y) in the grid, the robot will pick up the barrel at this location. But he will only do this if this event is connected to the emotion *liking*. For robot 1 this is not the case, because the E-rule specified for this event has the appraisal tag nastyThing(1.0) at the end which will result in the emotion disliking. In the second PC-rule we can see that if a blue barrel is found and the emotion liking is connected to the external event specified in the rule, the barrel will be picked up. For robot 1 this is the case because with the E-rules it is defined that if the robot comes accross a blue barrel he will appraise this as a niceThing(1.0), which will lead to the emotion liking. Note that we have not specified the personality in this rule, so it does not depend on the personality if one of the PC-rules will be executed or not. It depends only on the fact which emotion is connected to the events redBarrelFound(X,Y)/blueBarrelFound(X,Y).

When a robot has lost at least two more races than the other robot he will appraise this as a bad event which will lead to the emotion distress to be generated as explained above. As a coping behavior he will try to slow the opponent down by placing a wall between this opponent and the target. We have defined this with the following (headless) PG-rule:

```
:: _, distress <- raceStarted() and racesWon(X) and racesLost(Y) and (Y-X) >= 2 and target(A,B) and
                  posOther(C,D) and not wallPlaced()|
{
        PlaceWall();
        @env(placeWall(A,B,C,D),_);
}
```

Again here we do not look at the personality of the robot, we only look if the belief-query in this rule caused the robot to generate the emotion distress. And indeed this happens if we look (again) at the E-rule for this belief-query:

```
belief: racesWon(X) and racesLost(Y) and (Y-X) >= 2 and target(A,B) and posOther(C,D)
        and not wallPlaced() => badEvent(0.5)
```

As you can see the appraisal tag badEvent(0.5) has been connected to this belief-query. And as mentioned before this will lead to the emotion distress. In the body of the PG-rule we see a wall is placed by executing the belief update action PlaceWall() and by executing the external action placeWall(A,B,C,D). The latter function will put a wall between position (A,B) and (C,D). Note that this is only executed if the robot is distressed with respect to the specified belief query.

When a wall is placed robot 1 will appraise this as a bad act of the opponent and robot 2 will appraise this as a good act of the opponent as has been defined with the E-rules above. So for robot 1 this will lead to the emotion reproach directed at the opponent and for robot 2 this will lead to the emotion admiration directed at the opponent. So for this we have defined the coping behavior of the agents with the following two PC-rules:

```
event(stoneReceived(X,Y), env) :: _, reproach <-
target(A,B) and posOther(C,D) and raceInProgress() |
{
        PlaceWall();
        @env(placeWall(A,B,C,D),_);
}

event(stoneReceived(X,Y), env) :: _, admiration <- raceInProgress() |
{
        print("robot has admiration for placing the wall");
}
```

The first rule of these two PC-rules will be executed if the event stoneReceived(X,Y) has led to the emotion reproach. Then the robot will take revenge on the opponent by also placing a wall between this opponent and the target. If the robot admires the act, he will do nothing, only a line will be printed that the robot admires the act. In the demo application we have show how events, actions and objects are appraised and how this appraisal will lead to emotion generation. The emotions will have influence on the behavior of the agent. And indirectly this behavior will influence the mood of the agent. Of course this demo application is just an example of how the coping mechanism in ALMA can be utilized. In this case I have based it on a theory of Frijda, but a programmer can decide for himself if and how he will use the coping mechanism.

11. Conclusion, contributions and future work

During the project I discussed in this thesis I did research about three frameworks for generating emotions and moods. The way each framework tries to accomplish the emotion and mood generation is different and the frameworks also differ in what is incorporated i.e., ALMA covers appraisal and emotion and mood generation, EMA covers appraisal, emotion and mood generation and coping and Cathexis covers emotion and mood generation but leaves holes for the users of the framework to fill in such that appraisal can also be done.

EMA is by far the most extensive framework and is based on a lot of different theories which are put together in one model. The advanced planning model in combination with the causal interpretation is the most notable feature of the framework. It takes a prediction of the future into account when it has to be decided which coping strategy should be executed. In the framework a lot of theories are used. In my opinion the authors of this framework do not sufficiently explain why al these theories can work together. Every theory may individually work, but that does not imply that all the theories together work without a problem. Nonetheless I think the authors have created a very impressive framework to say the least.

As mentioned earlier Cathexis leaves a lot of holes in the framework for the user to fill in. It mainly provides formulas for determining intensities of emotions (which also form the mood) and which behavior should be picked based on those emotion intensities. Because the user has to fill in all the holes, it depends on the user if the framework will be an accurate model for emotion and mood generation. I think a lot of time has to be spend to obtain an accurate model by filling in those holes.

ALMA is the framework that I have chosen to use during the implementation of the appraisal process and emotion and mood generation. Although ALMA does not cover coping I think the appraisal process and the emotion and mood generation process are all based on solid theories. And it does show that we can have a fairly good simulation of appraisal and emotion and mood generation without becoming extremely complex. Taken into account that it does not have all the advanced features that EMA has I think it does a good attempt at the simulation of appraisal and emotion and mood generation.

The main problem in this thesis was:

Connect the agent oriented programming language 2APL to (one of) the three frameworks to implement emotional agent systems.

In this thesis I have connected the framework ALMA to 2APL. ALMA incorporates the appraisal process, emotion generation and mood generation in 2APL. The coping process is based on our own ideas (myself, Meyer and Dastani) and I got a bit of inspiration from EMA and Frijda's ideas.

For the coping process we have devised templates. Therefore it is relatively easy to adjust the coping strategies. So if someone would like to put his or her own thoughts into it, they can adjust the coping part without actually adjusting the source code of 2APL. A disadvantage of this approach is that the coping process is error prone. That coping works correctly is the responsibility of the 2APL programmer. The templates serve as a guideline of how to use the coping process within a 2APL program.

What I have created is a new version of 2APL in which emotions and moods are incorporated. Because 2APL relies on a solid logical foundation and is used to develop multi-agent systems we have a platform that connects the two ends we spoke about in the beginning of this thesis i.e., one end is the three frameworks that seem to work in practice but are not sufficiently supported by a logical foundation and the other end is the 2APL platform which is sufficiently supported by a logical foundation and is used to develop multi-agent systems.

Future work could be devising another coping mechanism that is completely or partially implemented in 2APL itself. An advantage of this would be that the 2APL programmer does not have to program the coping process himself (entirely) and thus it is less error prone. A disadvantage would be that this solution is less flexible. The logical system described in [30] would be a great starting point for this modification of the coping mechanism.

It is also possible to extend the language with a different kind of appraisal process, emotion generation process. Because we can make a distinction between the appraisal process, the emotion generation process and the coping proces we can take one of them and adjust it without adjusting one of the other two processes as long as the output of one process can be the input for the next process. I have used ALMA for appraisal and emotion generation. But also other frameworks can be used for this in the future. One could try to implement Cathexis or EMA for example.

I would also like to use the Philips iCat to demonstrate the new functionalities of 2APL. Lack of time caused this to be marked as future work.

Acknowledgements

I would like to thank Mehdi Dastani and John Jules Meyer for their guidance and support during this project. A lot of ideas described in this thesis follow from the conversations we had at the university. Additionally I would also like to thank them for the courses they provided. Among other courses it were these courses that got my attention and got me interested in artificial intelligence. Another person who has helped me during this project was Patrick Gebhard, I would like to thank him for adjusting the ALMA framework in such a way that it could be integrated with 2APL more smoothly. I would also like to thank Bas Testerink for introducing me to the Philips iCat and helping me with acquiring the necessary software tools for this device. And last but not least I would like to thank my family and friends for their support during this project.

Bibliography

1) Gebhard P, ALMA – A Layered Model of Affect, AAMAS'05

2) Marsella S and Gratch J: EMA: A computational model of appraisal dynamics, Agent Construction and Emotions 2006

3) Marsella S and Gratch J: A Domain-independent Framework for Modeling Emotion, Journal of Cognitive Systems Research, Vol. 5, Issue 4, 2004

4) Velásquez J D, Cathexis: A Computational Model for the Generation of Emotions and their Influence in the Behavior of Autonomous Agents, Massachusetts Institute of Technology, 1996

5) R. Steunebrink B: The Logical Structure of Emotions, 2010

6) Marsella S and Gratch J; Technical Details of a Domain-independent Framework for Modeling Emotion; Technical Report ICT-TR-04, 2004

7) Gebhard P, Kipp M, Klesen M and Rist T; Adding the Emotional Dimension to Scripting Character Dialogues

8) Damasio A R, Descartes; Error: Emotion, Reason and the Human Brain, Grosset / Putnam Press, New York, 1994

9) Mehrabian A. Analysis of the Big-five Personality Factors in Terms of the PAD Temperament Model. *Australian Journal of Psychology*, vol. 48, 2, 1996, 86-92.

10) McCrae R.R. and John O.P.; An introduction to the five factor model and its implications. *Journal of Personality*, vol. 60, 1992, 171–215.

11) Dastani M and Meyer J.J.C; Programming Agents with Emotions; Utrecht University 2004

12) Dastani M., 2APL: a practical agent programming language, University Utrecht 2008

13) Dastani M., 2APL: a practical agent programming language user guide, University Utrecht 2008

14) Dastani M. and Steunenbrink B.R., Operation Semantics for BDI Modules in Multi-Agent Programming, Utrecht University 2009

15) Rao A.S., Modeling Rational Agents within a BDI-architecture, Australian Artificial Intelligence Institute 1991

16) Rao A.S. and Micheal P.G., BDI Agents: From Theory to Practice, Australian Artificial Intelligence Institute 1995

17) Gebhard P. and Kipp K.H., Are computer-generated emotions and moods plausible to humans?, German Research Center for Artificial Intelligence and Experimental

Neuropsychology Unit, Saarland University

18) Ortony A., Clore G.L. and Collins A., The Cognitive Structure of Emotions, Cambridge University Press

19) Gebhard P., Klesen M. and Rist T., Coloring Multi-Character Conversations through the Expression of Emotions, DFKI GmbH

20) Velásquez J.D., An Emotion-Based Approach to Robotics, MIT Artificial Intelligence Laboratory 1999

21) Velásquez J.D. and Maes P., Cathexis: A Computational Model of Emotions, Autonomous Agents 1997

22) Marsella S.C. And Gratch J., EMA: A process of appraisal dynamics, Journal of Cognitive System Research vol. 10 2009

23) Marsella S.C. And Gratch J., Evaluating a computational model of emotion, University of California 2006

24) Winikoff M., JACK intelligent agents: An industrial strength platform, Multi-Agent programming: Languages, platforms and applications. Kluwer 2005

25) Pokahr A., Braubach L. and Lamersdorf W., Jadex: A BDI reasoning engine, Multi-agent programming: Languages, platforms and applications. Kluwer 2005

26) Frijda N.H., The Emotions, Cambridge University press 1987

27) http://en.wikipedia.org/wiki/A*_search_algorithm

28) http://javacc.java.net/doc/docindex.html

29) http://en.wikipedia.org/wiki/EBNF

30) Dastani M. and Lorini E., A logic of emotions: from appraisal to coping, Proceedings of the 11th International Conference on Autonomous Agents and Multi agent Systems, 2012

31) Ekman P., Moods, Emotions and Traits. In: The Nature of Emotions: Fundamental Questions, Oxford University Press, 1994

32) http://www.w3.org/TR/2010/WD-emotionml-20100729