Journal of Information Engineering and Applications
ISSN 2224-5758 (print) ISSN 2224-896X (online)
Vol 1, No.2, 2011

An Affective Decision Making Engine Framework for Practical Software Agents

Hafiz Mohd Sarim[1*], Abdul Razak Hamdan[1], Azuraliza Abu Bakar[1], Zulaiha Ali Othman[1]

1. Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia

* E-mail of the corresponding author: hafiz@ftsm.ukm.my

The research is financed by the Fundamental Research Grant Scheme (FRGS Grant code: UKM-TT-03-FRGS0134-2010) sponsored by the Department of Higher Education, under the Ministry of Higher Education of Malaysia, and supported by the Centre for Research and Instrumentation Management, UKM.

Abstract

The framework of the Affective Decision Making Engine outlined here provides a blueprint for creating software agents that emulate psychological affect when making decisions in complex and dynamic problem environments. The influence of affect on the agent's decisions is mimicked by measuring the correlation of feature values, possessed by objects and/or events in the environment, against the outcome of goals that are set for measuring the agent's overall performance. The use of correlation in the Affective Decision Making Engine provides a statistical justification for preference when prioritizing goals, particularly when it is not possible to realize all agent goals. The simplification of the agent algorithm retains the function of affect for summarizing feature-rich dynamic environments during decision making.

Keywords: Affective decision making, correlative adaptation, affective agents

1. Introduction

Affective decision making (Picard, 1997) in artificially intelligent agents aims to buffer the availability of knowledge, by emulating the ability of emotions to substitute the deficiency of knowledge (de Sousa, 1987), or summarize knowledge derived from excessive information resources (Elster, 1998). This is a particularly challenging problem faced when designing practical software agents for autonomous operation in real world environments. Dynamism and complexity in real world environments complicate the ability of the agent to make rational decisions. These two concepts allow the introduction of new untrained features into the environment in one moment, while at the same time eliminating entire classes of trained features from the picture. As a consequence, supervised learning algorithms will fail when inputs, used for the classification of responses, disappear entirely. Unsupervised adaptive learning algorithms fare better in such open environments, as they are unconstrained by the existence of particular feature sets as inputs. Their only limitation depends on the persistence of factors used for reinforcement during adaptive learning. In affective agents – agents which employ the influence of affect in establishing behavior – adaptation is done against the agent's persistent structure of emotions. In affective computing, cognitive structures of emotions, such as the OCC cognitive structure (Ortony et al., 1988) have long been the mainstay, and often core component, for the computation of agent behaviors. In affective decision making, such structures guide the generation of affect, and subsequently provide a heuristic for the selection of actions based on affect (Steunebrink, 2006).

However, a question is posed here; can affect be generated or calculated without the benefit of a cognitive structure of emotions? Or more importantly; can software agents emulate the same benefits of affect without invoking a biologically inspired cognitive structure? These questions are asked because the necessity of using a biologically inspired cognitive structure in a synthetic software agent has never been addressed. Yet at the same time, the practicality and utility of using affect to overcome losses or excesses of knowledge in dynamic and complex problem environments are recognized and desired (Steunebrink, 2006) (Schermerhorn and Scheutz, 2009).

1.1 Related work

This research is similar to the work done on the EMAI architecture (Baillie and Lukose, 2009) which puts forward an emotion appraisal model that allows the production of autonomous adaptive behavior in software agents. Unlike EMAI, an appraisal model of labeled emotions is not as part of the software agent's framework. Neither is the OCC cognitive structure of emotions used as part of the heuristic for selecting agent actions. Rather, this research follows the approach used by Schermerhorn and Scheutz (2009) and Sharpanskykh and Treur (2010) which aims to emulate the influence of affect during decision making. The Affective Decision Making Engine outlined here emulates the cyclical nature of the *as-if body loop* (Damasio, 1994) for the propagation of affect information which has become the theoretical basis of the Somatic Marker Hypothesis (Damasio, 1996). In our work, the absence of a 'body' in a practical software agent is replaced by the use of 'goals', which are effectively the measures for evaluating the overall performance of the agent.

2. Principles of Affect in the Affective Decision Making Engine

The most atomic measure of affect used by software agents equipped with the Affective Decision Making Engine is the positive or negative correlation between the feature values of observable elements in the problem environment, against the goal values which make up the agent's performance measure. Features which have significant correlations (e.g. > +0.80 or < -0.80 correlation coefficients in a -1 to +1 coefficient scale) are suspected of having a positive or negative influence over the agent's goal values. The total sum of significant feature correlations therefore predicts whether a particular goal will rise or fall in value given the rate of changes in feature values. Subsequent to this prediction, the goals which are expected to undergo significant change can be known by the agent. In response, the agent will select actions to either prevent or promote the change in these goals' values, thereby striving to maintain or achieve the desired levels of performance as defined by the agent's designer. Using this definition, affect is principally the influence exerted by changes in the environment in determining which of the agent's priorities (i.e. goals) are currently important and require attention for ensuring the agent's operational performance.

Equation (1) shows the measure of affect for a particular agent goal G_m:

$$Affect(G_m) = \sum_n^m r(G_m, F_n) \begin{cases} = r \ if \geq 0.8 \ or \leq -0.8 \\ = 0 \end{cases} \quad (1)$$

Where **r** = correlation coefficient

The logic behind the use of correlation as units of affect is twofold:

1. Correlations are a rational indicator of the existence of a relationship between the problem environment and the agent's goals, evidenced by past observations.
2. Correlation coefficients provide a statistical quantity for measuring the aforementioned relationship and its valence, calculated from past evidence.

These two reasons are necessary to support the objectivity of decisions made by the agent. At the same time, it presents the decision making engine design with the closest statistical equivalent of psychological affect, which is both context sensitive and adaptive. A decision making engine designed around correlation does not require the user-provided specification of context, or training data sets needed to classify context, such as those used by neural network based context-sensitive agents (Turney, 1996). The Affective Decision Making Engine instead universally defines context as the predicted success or failure of the agent's goals based on correlative values. Furthermore, behavioral adaptation is made against persistent affect values for each goal, as measured by correlation coefficients in equation (1), rather than against features types and feature values belonging to elements. This would be preferable in the problem environment specified in section 1 which allows for rapid change of elements, partial visibility of feature values, and revision of the agent's available actions.

2.1 Mechanism Concepts

The following necessary concepts are defined in the design of software agents that use the Affective

Decision Making Engine:

- *Environment*: The problem/work space of observable elements in which the agent operates.
- *Elements* (E_l): Objects and events that are observed in the environment. Objects here are defined as anything that preserves the same identifier and feature types, though may undergo change in feature value. Events can either be actions performed by an object independently or in an interaction/reaction with another element (object or event).
- *Features* (F_n): A set of property classes belonging to elements, which are projected as a result of pre processing the raw input data provided by elements. The pre-processing can be any form of technical analysis done against raw data. For example, an element may display or broadcast a change of value from 100 to 200 after two iterations. The pre-processing step will project $F_{average\_element\_value} = 150$, where $F_{average\_element\_value}$ is a feature type belonging to the element, and 150 is the value of this particular feature. A single raw input from an element can be projected into any number of features, limited only by the technical analyses for data pre-processing as defined by the user. The pre-processing of raw inputs to features makes up the sensory component (preceptors) of agents based on the Affective Decision Making Engine.
- *Goals* (G_m): The agent's property class by which the performance of an agent is ultimately measured. Goals are established by the agent's designer to reflect the agent's main tasks, and serves as the component for determining whether the agent is performing within desired parameters. The agent's objective may be to maintain goal values within a particular range, or above or below a pre-defined threshold. Agents may also be tasked with ensuring positive or negative change in the goal values. Within an operating environment as proposed in section 1 it is natural and expected for a software agent to have multiple goal values, some of which may conflict with each other. For example, a day-trading investment agent may have the goals $G_{cash\ balance}$, $G_{portfolio\ valuation}$, and G_{profit}. Ideally, the agent's designer would like all three goals to increase in value. However, this cannot realistically occur, as $G_{cash\ balance}$ and G_{profit} will not increase without some form of selling, which will reduce $G_{portfolio\ valuation}$. In a bear market, liquidating stock holdings will reduce $G_{portfolio\ valuation}$ and G_{profit} but will significantly increase $G_{cash\ balance}$ and may preserve it above a set threshold. As goals may contradict each other, the role of affect will be to set focus on which goals are currently important.
- *Affect value* ($AV_{Fn,\ Gm}$): A quantity that measures importance, established against an agent's goals. This is done to divert the attention of the agent to maintaining only a subset of goal's values within each goal's predefined threshold. Essentially, the affect value frees the agent from contradictory instructions and decision conflicts, by switching goals ON or OFF past a set significance threshold. Therefore, given the current state of the environment's elements, the element's feature values alter affect value, which in turn highlights only significant goals. This allows the agent to choose actions for maintaining or improving the highlighted goals.
- *Actions*: The agent's actuators. In the problem environment established in section 1, the actions currently available to the agent may change at every iteration. To overcome this, the Affective Decision Making Engine treats actions as elements, in that every action may itself have a set of features. The added level of granularity will allow software agents to treat different actions as similar, based on the similarity in the actions' feature types.

2.2 The Affective Decision Making Cycle

Figure 1 shows the cycle of influence between an element's feature values, affect values, and goal values, and the selected actions. The cycle motivates perpetual decision-making in an ever-changing environment specified in section 1, geared by affect to highlight the operational conditions, which in turn rationalizes the biased decisions made by the agent. It is for this reason that the mechanism presented in this paper is termed the 'Affective Decision Making Engine'.

To be sure, the absence of a cognitive structure of emotions such as the OCC cognitive structure (Ortony et al., 1988) from the Affective Decision Making Engine design precludes the labeling of agent that utilizes the engine as 'emotional' agents. However, the Affective Decision Making Engine does allow the different influences exerted by different elements in the environment to permeate over the agent's goal values. This affect paints the landscape of which goals are important to the agent, leading

the agent to make a rational yet also paradoxically prejudices decisions based on these goals. With this, it can be argued that the software agents based on the Affective Decision Making Engine will display operational behavior that emulates affect.

3. The Affective Decision Making Engine

The most fundamental data structure in the Affective Decision Making Engine is a 2-dimensional matrix that cross tabulates observable feature types (F_n) belonging to elements in the environment against the goals (G_m) that are used to measure the software agent performance. This 2-dimensional matrix is called an *affect matrix* (AM), and is shown in Figure 2. Each cell in the affect matrix stores an *affect value* (AV). The affect value is a continuous, valenced, and bounded historical measure of correlation between changes in the feature values and the changes in the software agent's goal values. Each affect value is stored in the form:

$$AV_{F_n,G_n} = (r', y) \tag{2}$$

Where: ***r'*** = the historically influenced correlation coefficient between the values in feature F_n and values in the goal value G_m.

y = a historical summary of past ***r'*** values.

3.1 Calculating Affect

The actual measure of affect lies in the ***r'*** component of the affect value. Since it is possible for different features to have different data scales (e.g. nominal, ordinal, interval and ratio), any combination of statistical or clustering measure can be used to calculate the ***r'*** component of the affect value independently in each AM cell of a single affect matrix. This is done in order to accommodate the calculation of the correlation between different feature types and different goal values. Therefore, if a column in the affect matrix stores the values of an interval scale (i.e. continuous) feature, then the ***r'*** component of the affect value can be calculated using a product moment correlation coefficient. Instead, if the AM column stores a nominal scale (i.e discrete) feature, then Chi-square can be used to calculate the ***r'*** component of the affect value.

The current ***r'*** value is continuously updated at every iteration from past values of ***r'***, all the way back to the initial calculation of r. For example, if a feature observed by the software agent displays continuous values and the goal value is also continuous, then the Pearson product-moment correlation coefficient shown in (3) can be used to calculate the initial ***r***. Later we adapt the y component of the affect value to produce ***r'*** for the next iteration of agent operation.

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n(\sum x^2) - (\sum x)^2] \times [n(\sum y^2) - (\sum y)^2]}} \tag{3}$$

Using (3) as an example, ***r*** or ***r'*** establishes a linear relationship between the feature being observed the goal value, which makes up an intersecting cell of the affect matrix. A positive ***r'*** value of 0.92, for instance, is a high indication that as the feature increases in value, the goal value is also expected to increase in value. This is shown in the feature-goal value graph in Figure 3(a). Therefore, the software agent can expect that the specific goal value will be affected positively if the feature value shows a trend of increasing. Conversely, when the feature values fall, the software agent can expect the goal value to be negatively affected.

3.2 Compressing the r'-correlation information

The accurate calculation of correlation requires the preservation of all historical data points from every stage of analysis (i.e. iteration). As the software agent may be expected to observe a multitude of features for each element in the problem environment, and if we allow every element to display different numbers and types of features, then recording a history of all feature values quickly becomes intractable. For this reason, the ***y***-component of the affect value can be used to compress and summarize feature value history.

$$a = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2} \qquad b = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n(\sum x^2) - (\sum x)^2} \tag{4}$$

Where **a** = the slope of the line of best fit

b = the y-axis intercept of the line of best fit

The method for compressing feature value history in the ***y***-component depends of the data scale of the feature values and goal values for the AM cell. For interval scale data points, of which the equation in (3) is used to calculate ***r***, the ***y***-component utilizes the least-squares regression line, or line of best fit (4), to reduce the data point history from a very large number (e.g. 50 000) to only two points. The line of best fit best represents the distribution of data points on a feature-goal value graph, whereby the distance of individual data points incidentally also corresponds to the correlation coefficient. Compression of the feature value history is done by taking the left-most and right-most extreme data points on the goal value (i.e. horizontal) axis, and recording the two points on the line of best fit that have the same horizontal value. These two points can then be used to substitute the feature history when coupled with the current correlation coefficient (***r***) as they represent the extreme range of observed data points, and are within the approximate horizontal position combination for past feature value history. Figure 3(b) and (c) shows how the feature value component is compressed to feed the ***y***-component of the affect value.

In the example where the line of best fit is the ***y***-component, Figure 3(d) provides a method for recreating an approximate of the feature value history from the ***r***-component and ***y***-component of the affect value. Since the compression of the feature value history leads to the ***y***-component only approximating the feature value history, there will be some inaccuracies in the new correlation coefficient, which is done by projecting the new feature data point with the compressed data points in y as shown in Figure 3(d). This lossy derivation of ***r*** from the ***y***-component necessitates the labeling of future calculations of correlation as ***r'***, as some degradation of data occurs due to compression.

3.3 Components Necessary for the Decision-Making Process

Using the AM data structure, the Affective Decision Making Engine is made up of the following components. The connections between components are shown in Figure 4:

- An *element affect matrix* (***EAM***) for each unique element (i.e. object or event): Stores the correlation coefficient for each feature belonging to the element, against the agent's goal values. The sum of significant correlations determines the element's affect value (AV) over a particular goal. This sum is the element affect value (EAV) for each goal. Note: the definition of 'significance' is a parameter that needs to be preset by the agent designer.
- An *action affect matrix* (***AAM***) for each unique agent action: Stores the correlation coefficient for each feature produced by the action, against the rate of change in the agent's goal values. The sum of significant correlations determines the action's affect value (AV) over the change in a particular goal. This sum is the action affect value for (AAV) for each goal
- A single *goal significance matrix* (***GSM***): Cross tabulates the EAV of every element against every goal. The sum of significant EAVs produces an goal significance value (GSV) for each goal.
- A single *homeostat*: The core AM used for making decisions in the engine. The homeostat cross-tabulates significant goals found in the GSM against the agent's available actions.

The number of EAMs and AAMs expand as the number of elements increase in the problem environment. For this reason, agents based on the engine are memory hungry in unbounded operation environments. Resource use can be controlled by a fixed sized roving window that limits the number of elements currently perceived by the agent, much like an eyeball's limited degree of vision. The window essentially reduces the size of the GSM, making calculation of the GSV quicker. The Affective Decision Making Engine framework, as it applies to environments with multiple elements (i.e. objects or events), is illustrated in Figure 4. In environments with only a singular elements, the GSM can be excluded from the framework.

3.3 The Decision-Making Process

Decisions in the Affective Decision Making Engine are made in the homeostat. The homeostat used here deviates from the original homeostat designed by Ashby (1956) due to the fact that equilibrium needs to be redefined during each iteration according to the GSV of only those goals that are found to be significant in the GSM. In this regard, the homeostat used in this framework is closer to the homeostat design of the DARE architecture in (Maçãs et al., 2001), but differs in that specific emotional labels are not used as part of the design. Furthermore, the balancing component of the homeostat is derived from a similarly changing set of available actions in each iteration. For this reason, the homeostat is represented by an AM data structure in this framework. Affectively significant goals, which were previously highlighted in the GSM, populate the columns, and are cross tabulated by the agent's available actions as the rows. The GSV for each goal acts as a common multiplier for the AAV of every available action. Homeostasis is disturbed whenever there is any GSV, positive or negative, in the homeostat.

To make rational decisions, the agent selects the action that will produce the largest sum product of AAV and GSV (5). This method of action selection works very well when the affect values are defined as correlation coefficients. For example, if an goal has a high positive GSV, actions which produce the largest positive correlation for change in the aforementioned goal would also produce the largest product. Conversely, if a goal has a high negative GSV, actions which produce the largest negative correlation for change in the aforementioned goal would in turn produce the largest product of AAV and GSV. Actions that bring equilibrium to the homeostat during rational decision making are those actions that produce the highest sum product of AAV and GSV.

$$\arg\max\left(action_i^0\right) = \sum_i^n \left(AAV_i \times GSV_n\right) \tag{5}$$

4. Conclusion

The operation of software agents equipped with the Affective Decision Making Engine in empirical tests have shown that the correlation of feature values to goal values, as a measure of affect, is able to correctly identify features that exert positive or negative influences to the agent's goals. As such, the summary propagation of affect to similarly featured elements or elemental classes is possible by employing similarity measures between elements, as a future proposed direction.

Mistakes in correlation, however, occur when the element exits observation for an amount of time and the environment independently alters the element's feature values outside the view of the agent. This element's temporary absence usually creates a significant deviation from the stored ***y***-component summary of feature history when measuring affect value. Therefore an affect degradation rate, which pushes affect values to 0 (no correlation) is recommended for frequently deviating features.

Observations also show the scalability of the agent in handling feature rich elements in more populous expanded problem environments, since only significant affect values enter the EAM, AAM, and GSM affect matrices. Scalability in this case is governed by the significance threshold and the size of the roving window used to limit the agent's view.

Acknowledgements

We would like to thank the Data Mining and Optimization Research Group, UKM and the Centre of Artificial Intelligence Technology, UKM for their valuable assistance in this research.

References

Ashby, W. R. (1956). *Introduction to Cybernetics*. Methuen.

Baillie, P. and Lukose, D. (2002). An Affective Decision Making Agent Architecture Using Emotion Appraisals. *Proceedings of PRICAI '02*, 581-590.

Damasio, A. R. (1994). *Descartes' Error: Emotion, Reason and the Human Brain*. Grosset/Putman.

Damasio, A. R. (1996). The somatic marker hypothesis and the possible functions of the prefrontal cortex. *Philosophical Transactions: Biological Sciences*, *351*(1346), 1413-1420. doi: 10.1098/rstb.1996.0125, http://dx.doi.org/10.1098/rstb.1996.0125

de Sousa, R. (1987). *The Rationality of Emotion*, Cambridge: MIT Press.

Elster, J. (1998). Emotions and Economic Theory. *Journal of Economic Literature*, *36*(1), 47-74.

Maçãs, M., Ventura, R., Custódio, L., and Pinto-Ferreira, C. (2001). Experiments with an Emotion based Agent using the DARE Architecture. *Proceedings of the AISB'01 Symposium on Emotion, Cognition, and Affective Computing*, 105-112

Ortony, A., Clore, G. L., and Collins, A. (1988). *The Cognitive Structure of Emotions*. Cambridge University Press.

Picard, R. W. (1997). *Affective Computing*. MIT Press.

Schermerhorn, P. W. and Scheutz, M. (2009). The Utility of Affect in the Selection of Actions and Goals Under Real-World Constraints. *Proceedings of the ICAI*, 948-953

Sharpanskykh, A. and Treur, J. (2010). Adaptive modelling of social decision making by agents integrating simulated behaviour and perception chains. *Proceedings of ICCCI'10*, *Pt. I*. 284-295

Steunebrink, B.R., Dastani, M.M., and Meyer, J-J.Ch. (2006). Emotions as Heuristics in Multi-Agent Systems. *Proc. 1st Work. on Emo. and Comp. Impact*, 15-18

Turney, P. (1996). The Identification of Context-Sensitive Features: A Formal Definition of context for Concept Learning, *Proc. Work. Context Sensitive Domains, ICML-96*, 53-59

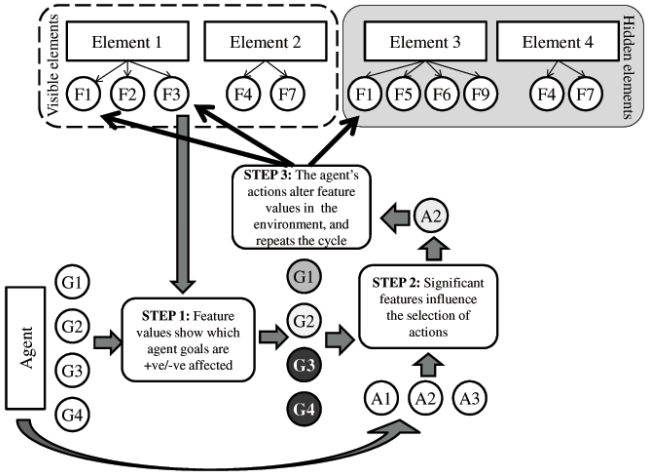

Figure 1. Cycle of the influence of affect determines significant goals (step 1) that dictate which action to perform (step 2), which then in turn influences elements' feature values.

$$\begin{array}{c} \\ G_1 \\ \vdots \\ G_m \end{array} \begin{array}{c} \begin{array}{ccc} F_1 & \cdots & F_n \end{array} \\ \begin{bmatrix} AV_{F_1,G_1} & \cdots & AV_{F_n,G_1} \\ \vdots & \ddots & \vdots \\ AV_{F_1,G_m} & \cdots & AV_{F_n,G_m} \end{bmatrix} \end{array}$$

Figure 2. The Affect Matrix. The matrix provided above is an example of the EAM (element affect matrix. The AAM (action affect matrix) uses the same cell calculations.

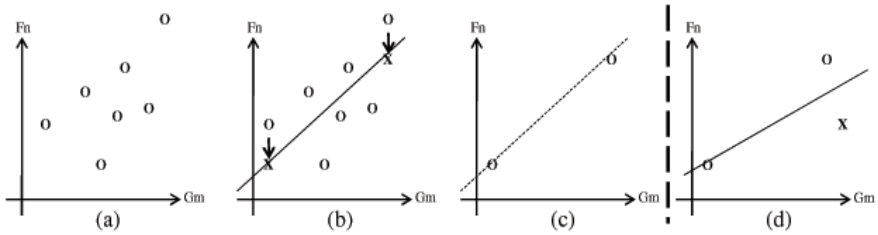

Figure 3. Affect is calculated by finding the correlation of data points in the feature-goal value graph (a). The line of best fit (b) for the historical data points can be compressed by finding the extreme horizontal axis points on the line (c) and is later stored as the ***y***-component of AV. Correlations with new data points are found by projecting these compressed points.

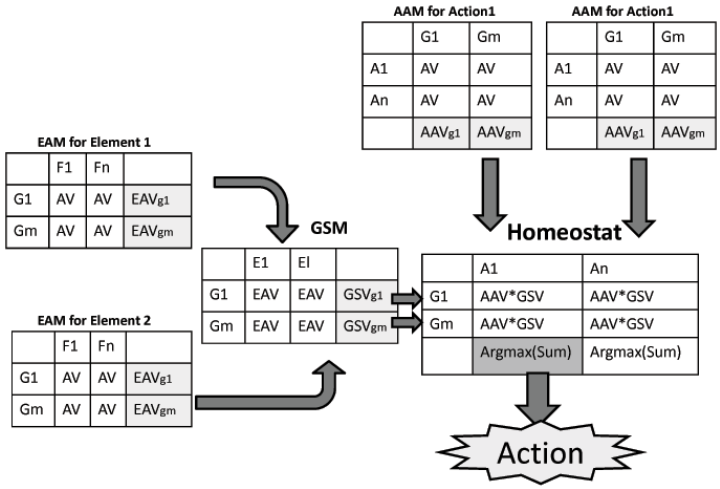

Figure 4. The interaction of the different types of affect matrices to produce a decision in the Affective Decision Making Engine