Emotionally Intelligent Agents: The Outline of a Resource-Oriented Approach

Alastair Burt
DFKI (German Research Centre for Artificial Intelligence)
Stuhlsatzenhausweg 3, 66123 Saarbrücken
burt@dfki.de

Abstract

We discuss why emotions are currently an important topic for the intelligent agent paradigm. We explain which aspects of emotion research are relevant and indicate how current architectures for intelligent agents may be extended with resource managing constructs, inspired by ideas from cognitive science, to produce *emotionally intelligent agents.*

Introduction

In the last few years, intelligent agents have become an important paradigm for software development, particularly on the Internet (Bradshaw 1997). There are several reasons to believe that emotions and related concepts will become increasingly relevant for the design of such intelligent agents.

First, agents are being used to implement lifelike characters in software for games, interactive drama, and general user interfaces. The agent-oriented paradigm, where actions are generated autonomously from high-level goals, is arguably the best way to program them (Reynolds 1987). In all forms of lifelike characters, the user is invited to suspend disbelief and treat the object on the screen as if it were a real person or animal. An important way to encourage such disbelief is to endow the character with emotion-like traits.

Second, another important use of agent software is to represent the user in networked environments or assist the user in complex interaction with such environments. For this, the agent should have detailed knowledge of the user's preferences. Although mathematical models can be used to describe preferences formally, the most natural way to describe what the users like, or would like to do, is in terms of a model of the their personality and emotional make-up. This is especially true when the agent is asked to work towards several of the user's goals in an unpredictable environment with limited feedback from the user himself.

Third, there are several interconnected trends in research into agent architectures that are related to emotions:

- A desire to accommodate more complicated goals and goal activation mechanisms.
- A movement towards integrated agent architectures that combine a broad spectrum of intelligent behaviour.
- An awareness of the pervasive nature of resource constraints.

All three trends naturally lead to an examination of the theories of cognitive scientists, psychologists and ethologists who are studying motivational and emotional mechanisms in humans and animals.

Each of these three uses of emotions imposes different design requirements on the underlying model of emotions. Lifelike characters require a model that entertains. The emotional behaviour it specifies may be a caricature of that found in real life, a *cartoon psychology.* Agents acting as a user's representative require that the emotional model matches as closely as possible the desires of the user and that the user and the agent share a common, easily intelligible vocabulary. The use of emotional mechanisms to improve the control structures in agents may use abstractions of biological models that are unintuitive to the user as long as the mechanisms can be used to engineer optimal agents. The framework we give below arises from this last use of emotional mechanisms. However, we hope that the framework may also be extended to cover the first two uses as well.

Logic and Motivation

In order show how emotional models can be used to structure resource management within an agent, we shall present a symbolic account of how the agent reacts to the world, typical of the logic-based AI tradition. We indicate how this account must be improved by taking into account resource-oriented, utility-based constructs and show how these are related to emotional models. We base our account of utility on the work of (Russell Wefald 1991). We start by outlining the resource management issues as they arise in an *individual agent* and we go on to discuss how similar resource related issues arise in a *multi-agent society.* The example we give shows how the notion of an emotional state, such as fear, influences the control flow and computation within each individual agent.

In order to show how motivational constructs play a role at all levels of a multi-agent architecture we present a scenario, which, despite its simplicity typifies the situation found when several agents, each with more than one goal, try to solve a time-critical task. The example uses the abstract view of the INTERRAP agent architecture given in Figure 1. The INTERRAP architecture is typical of today's multi-agent systems. It has low-level reactive behaviour in the behaviour-based layer; it has deliberative, goal directed planning in the plan-based later; and it reasons about the social context of an agent in the social layer. For more information see (Müller 1996). In the Figure, **SPL**, **LPL** and **BBL** represent the programs that implement the process at the social planning, local planning and behaviour-based layer, respectively. Δ^{LPL} and Δ^{BBL} represent the way an upper layer influences the layer below. The exact way one layer may influence another is discussed later. I and O are the input and output events of the agent. Γ^{LPL} and Γ^{BBL} are abstractions of the processing at the lower layer that the higher layer needs. We ignore, in this example, the way data and control flow sometimes go up rather than down the layer hierarchy.

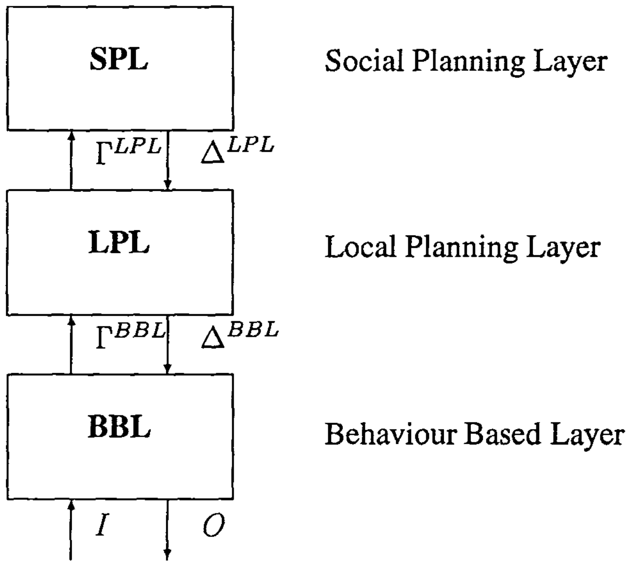

Figure 1: Intelligent Agent Architecture

Motivational Issues in the Reactive Layer of an Agent

The simplest form of an agent might not plan at all, but just react to a situation it finds itself in. It will only consist of the program for the reactive layer (in INTERRAP, the behaviour-based layer). An example of a specification of such a program is given in Figure 2. It specifies the relations that we would like to hold between the sensing and acting events of the program **BBL**. The predicate *environment* specifies an environment of an agent, and $execute(E, I, O, P)$ specifies the relation that holds between the input and output events (I and O) of the program P when it is run in the environment E. The behaviour we would like our reactive agent to exhibit is given by the predicate *bbl*. The predicates $sense\_food(i)$ and $sense\_predator(i)$ hold if the input event i is this particular type of sensing event. The predicates $eat\_food\_action(o)$ and $run\_action(o)$ hold if o is an output event of the appropriate action type. The function *time* maps events to numbers, over which the usual $<$ relation holds. The constant δ is a number.

In a framework such as (Gregory 1997) or (Fisher 1994) the definition of the predicate *bbl* could, more or less, be used as the program **BBL**. In a framework such as (Kowalski Sadri 1997) it can be seen as the specification of the temporal properties of a production rule system.

$$\begin{array}{l}\forall\, E, I, O\ environment(E) \wedge \\ execute(E, I, O, \mathbf{BBL}) \rightarrow \\ \quad bbl(I, O)\end{array}$$

$$\begin{array}{l}\forall\, I, O\ bbl(I, O) \leftrightarrow \\ \quad (\forall\, i \in I\ sense\_food(i) \rightarrow \\ \quad\quad \exists\, o \in O\ eat\_food\_action(o) \wedge \\ \quad\quad timely(i, o, \delta)) \\ \quad (\forall\, i \in I\ sense\_predator(i) \rightarrow \\ \quad\quad \exists\, o \in O\ run\_action(o) \wedge \\ \quad\quad timely(i, o, \delta))\end{array}$$

$$\begin{array}{l}\forall\, e, e', D\ timely(e, e', D) \leftrightarrow \\ \quad time(e) < time(e') \wedge \\ \quad time(e') - time(e) < D\end{array}$$

Figure 2: Simple Reactive Behaviour

The relevant motivational issues at the reactive level of an agent are:

1. The sets of input and output events, I and O, are possibly infinite, the agent could run as a non-terminating process. Our ability to reason about the consequences of the agent processes will depend on how we can chunk the input and output into finite episodes.
2. The two actions relevant to eat_food_action and run_action will sometimes lead to conflict, if the agent cannot do both actions at the same time.
3. We may well require the agent to operate in several different environments, with different characteristics, i.e.

the parameter E of *execute* may take on different values.

4. Here we do not specify all the properties of the actual predicate *bbl*. However, it is intuitively clear that we would prefer the agent to avoid a predator rather than to continuously eat food. This is specified more completely with a function that assigns a utility to the set of output events O.

5. A consequence of 2, 3 and 4 is that in certain environments it may be better for the reactive program **BBL** to delay food eating actions, when there is a high probability of a predator appearing. However, even then we do not want the agent to postpone its eating behaviour indefinitely. Within animals this dynamic control of behaviour is derived from the motivational states of hunger and fear.

6. Related to 4 and 5 is the issue of using goals in the specification that are desirable but may not be attainable: if the threat of a predator is too great then it may be better for the agent to refrain from eating food before the δ deadline.

7. We want our reactive agent to use the fewest possible resources. As (Russell Subramanian 1995) points out, we should express this in terms of bounded rationality, making explicit the machine that the agent runs on, the range of environments the agent could find itself in with their associated probability, and we need a utility function that describes how well the agent uses its resources to achieve its goals. Thus, we need:

    - to specify the abstract machine M,
    - the set $\mathcal{BBL}$ of programs for M that satisfy the specification in Figure 2,
    - a utility function U,
    - the class of environments $\mathcal{E}$, and
    - a valuation function V such that $V(p, M, \mathcal{E}, \mathcal{U})$ denotes the value according to U obtained by the program p on machine M in the environments $\mathcal{E}$, where we assume a probability distribution over the elements of $\mathcal{E}$.

    The optimal reactive agent, bbl_{opt}, is then defined by:

    $$bbl_{opt} = argmax_{bbl \in \mathcal{BBL}} V(bbl, M, \mathcal{E}, \mathcal{U})$$

    (Russell Subramanian 1995) offers some guiding principles as to how one might find the optimal program for the agent.

Flexible behaviour based on resource consumption at this level has been investigated extensively. In computer science it occurs as the problems of fairness and deadlock in concurrent systems. It is also a major subject in economics, and can be seen in the ideas that have influenced agent systems, particularly game theory, decision theory and market-oriented programming. However within agent architectures many problems are still open. General mechanisms to deal with the infinite time horizon (item above), the satisficing goals problem (item 6 above) and the optimisation over all environments and all programs to obtain bounded rationality (item 7 above) have still not been found.

Motivational Issues in the Planning Layer of an Agent

Figure 3 shows an example of the knowledge used at the planning layer of an agent. The predicate *lpl* defines the goal that the local planning layer is trying to achieve. The predicates *starve* and *be_prey* define properties that we do not want to hold in the input-output behaviour of the agent. The predicates *hunger_pang* and *sense_predator* are properties of input events of the agent. The predicates *goto_food_action*, *eat_food_action*, *run_action* and *hide_action* are properties of the output events. Thus, in addition, to the reactive behaviour of the program **BBL** the **LPL** program searches for additional behaviour to meet the goals defined by the predicate *lpl*. Here this involves actively searching for food when hunger sets in and having the option to either run or hide when a predator appears. In Figure 1 we sketched the way the local planning layer affects the behaviour-based layer through the transmission of Δ^{BBL}, i.e. the modifications that the upper layer imposes. We discuss in item 4 below the form these modifications might take.

$$\begin{array}{l} \forall\, I, O, E\ environment(E) \wedge \\ execute(E, I, O, \mathbf{BBL} || \mathbf{LPL}) \rightarrow \\ \quad lpl(I, O) \wedge bbl(I, O) \end{array}$$

$$\begin{array}{l} \forall\, I, O\ lpl(I, O) \leftrightarrow \\ \quad \neg starve(I, O) \wedge \neg be\_prey(I, O) \end{array}$$

$$\begin{array}{l} \forall\, I, O\ starve(I, O) \leftrightarrow \\ \quad \exists\, i \in I\ hunger\_pang(i) \wedge \\ \quad \neg(\exists\, o, o' \in O\ goto\_food\_action(o) \wedge \\ \quad eat\_food\_action(o') \wedge \\ \quad timely(i, o, \delta)) \end{array}$$

$$\begin{array}{l} \forall\, I, O\ be\_prey(I, O) \leftrightarrow \\ \quad \exists\, i \in I\ sense\_predator(i) \wedge \\ \quad \neg(\exists o \in O\ run\_action(o) \vee hide\_action(o)) \wedge \\ \quad timely(i, o, \delta')) \end{array}$$

Figure 3: Simple Planning Behaviour

The issues we discussed above for the reactive layer also arise at the planning level. In addition new issues arise:

1. Planning usually involves search between competing alternatives. In the example we can see that both *run_action* and *hide_action* will evade the predator. As the agent considers whether to do one action or the other, a branch opens up in the search space. The inference process must now divide resources between competing branches and resource consumption is an issue at the level of the plan execution as well as at the planning level. Does it cost more to run or to hide? And what will it cost to find out?

2. Eventually the agent must commit to one of several competing branches in the search space. It will, in general, be impossible for the agent to commit at one time to a plan representing all future behaviour. In the example when a predator appears the agent must commit itself to either running or hiding and do so quickly. It may not be able to fully work out the consequences of the action and this may even prove fatal: the agent could starve if it chose to hide and the predator were to wait that long outside the hideout. But a tradeoff must be made between waiting for better knowledge about the effect of plans and the decreasing utility of the plan.

3. Once the agent has committed to an action, there may be a further tradeoff: should the agent reclaim the resources devoted to alternative courses of action or should it keep these search branches open in case the first choice fails: that is, should it pay to hedge its bets.

There are also architectural, motivational issues between the layers:

4. How do we realise the passing of the plan to the behaviour-based layer (Δ^{BBL} in Figure 1)? In a typical hybrid agent architecture there may well be a reactive program running in the lower layer. This means, where possible, the planner should bear in mind the actions of these plans (for this Γ^{BBL} in Figure 1 provides an abstraction of the lower layer). It may also mean that the interface between the planning layer and the reactive layer should allow the planning layer to delete or restrict the resources to the processes in the lower layer. In the example, the planning layer might sense a predator and decide that hiding is better than running. If the program specified in Figure 2 is executing, the planning layer must install a new process to get the agent to hide, and inhibit the default processes that run from predators and eat any food the agent senses.

5. The principle of bounded rationality requires in this setting that the agent should divide its resources appropriately between the behaviour-based and the reactive layers. In general, the optimum will depend on the current environment. In an environment where there are no predators and food is so common that the agent continually senses it, then our example agent need not plan at all; the default behaviour embodied in the lower layer would be sufficient. But the environment may change and this happy situation may not continue. Related to this issue is that of contingency planning: When is it worthwhile for the agent to plan for situations that it is not sure will happen?

6. Given the inter-layer resource issues mentioned above, we will need cross layer categorisation of processes so that:
   - the agent recognises from the stream of primitive input events when such a dangerous situation, such as a predator appearing, is likely to be happening, and then
   - how to devote all resources in knowledge updating, planning and execution to sub-processes that will realise predator avoidance, whilst inhibiting other sub-processes.

7. To extend our definition of optimal agents, we take the set $\mathcal{LPL}$ of programs for the machine M that meet the specification of Figure 3 and form the set $\mathcal{BP} = \{(\mathcal{B}||\mathcal{P})|\mathcal{B} \in \mathcal{BBL} \wedge \mathcal{P} \in \mathcal{LPL}\}$, where $||$ represents parallel composition of programs. The above discussion shows there are many design options for the construction of the programs in $\mathcal{BBL}$ and $\mathcal{LPL}$, but we can say that we have made the right design decisions, when we have found bp_{opt}:

   $$bp_{opt} = argmax_{bp \in \mathcal{BP}} V(bp, M, \mathcal{E}, \mathcal{U})$$

   An agent that makes best use of its resources at the reactive and the planning layer is *optimal in the bounded rational sense.*

There is plenty of research in agent systems that incorporate planning and reactive behaviour. However, little attempt has been made to show that these systems are boundedly rational and no system has managed to dynamically change the resources allocated to planning and reactive processes according to the decisions carried out by the agent. This is a novel research topic and is precisely the area where motivation research in other areas outside of computing can make a contribution.

Motivational Issues in the Social Planning Layer of an Agent

The social planning layer modifies the goals of the local planning layer and handles the cooperative aspect of planning for a set of agents. An example is sketched in Figure 4. Here, the planning layer is charged with high level goals which have social implications. The predicate *spl* defines the goal for this layer of the agent, *friend* is a property of agents and *input* and *output* denote the input and output events. As a member of a social group the agent should engineer its behaviour so that not only is the agent itself free from hunger but also that its friends in the social group do not starve. Typically, the social planning layer will pass a goal, such as finding food for a group of agents, onto the local planning layer of the individual agent (via Δ^{LPL} in Figure 1). The knowledge of which agents are friends and what messages may be sent out to find out, whether a friendly agent is hungry (or will perform the social favour of giving the current agent some food) is represented at the social planning layer. The social planning layer may become aware of the need for such information from the local planning layer via the abstraction Γ^{LPL} that the lower layer offers.

$$\begin{aligned}
&\forall\, I, O, E\ \mathit{environment}(E) \wedge \\
&\quad \mathit{execute}(E, I, O, \mathbf{BBL}||\mathbf{LPL}||\mathbf{SPL}) \rightarrow \\
&\qquad \mathit{spl}(I, O) \wedge \mathit{lpl}(I, O) \wedge \mathit{bbl}(I, O)
\end{aligned}$$

$$\begin{aligned}
&\forall\, I, O\ \mathit{spl}(I, O) \leftrightarrow \\
&\quad \neg \mathit{starve}(\mathit{self}, I, O) \wedge \\
&\quad \forall\, F, I', O'\ (\mathit{friend}(F) \wedge \mathit{input}(F, I') \wedge \\
&\quad \mathit{output}(F, O') \rightarrow \\
&\qquad \neg \mathit{starve}(F, I', O'))
\end{aligned}$$

Figure 4: Simple Social Planning Goal

Exactly how social goals arise, and what sort of behaviour the agent can expect other agents to generate, depends on the society the agent finds itself in. We list below some problems that show the way inter-agent relationships affect the social planning layer:

- Multi-agent systems need social norms that describe how agents may interact. In the example, we may need food sharing mechanisms for instance. An autonomous agent will cooperate with other friends in the group if it feels the group norms guarantee that its own need for food resources is satisfied. Thus some fair means of resolving resource conflict should either be engineered by the designers of the multi-agent system or it should be evolved by the group of agents themselves.

- The control policy of the individual agents should reflect the additional constraints on resource usage as imposed by the environment. This could involve taking into account the penalties associated with unfulfilled commitments, it may also mean expending resources to maintain a friendly alliance between agents. In the example the agent should leave some food for friends. If it does not, it may expect to receive social sanctions such having to pay a fine to the group or being socially ostracised.

- The agents should be able to monitor or predict the behaviour of other agents. If an agent trusts another not to eat all the food, or food-eating is effectively policed by some other agent, then agents can go off and fulfil other goals. Reasoning about others' social behaviour is an aspect of emotional intelligence.

- Again, we can extend our definition of bounded rationality to cover reasoning at the social layer. We define $\mathcal{SPL}$ as the set of all programs for M that meet the specification in Figure 4, then define $\mathcal{BPS} = \{(\mathcal{B}||\mathcal{P}||\mathcal{S}) | \mathcal{B} \in \mathcal{BBL} \wedge \mathcal{P} \in \mathcal{LPL} \wedge \mathcal{S} \in \mathcal{SPL}\}$, and bps_{opt} makes optimal use of resources at all layers of the INTERRAP agent architecture if :

$$bps_{opt} = argmax_{bps \in \mathcal{BPS}}\, V(bps, M, \mathcal{E}, \mathcal{U})$$

Cooperation among groups of goal-oriented agents has been a major topic of research for economists, socio-biologists and mathematicians interested in game-theory. Not all of this research has found its way into architectures for multi-agent systems and an optimal way for an agent to reason about the cooperative behaviour of others has not been discovered. A aspect that remains to be covered is the affective component of inter-agent communication. Much research looks at how formal descriptions of inter-agent communication can be interpreted as speech acts. What is ignored is how almost all human speech not only has a propositional content but also conveys emotional information, such as whether the speaker is happy, upset, or surprised. This information is vital for cooperation in humans. An analogue of such information, providing an ongoing report on the state of goal directed processing within an agent, is likely to prove useful for societies of emotionally intelligent agents.

A Research Agenda

Above we have presented a logical characterisation of the processes in each layer in a typical architecture for an intelligent agent. Within the framework of logic programming we can turn these logical specifications into programs, according to the following equation (Kowalski 1979):

$$\mathit{algorithm} = \mathit{logic} + \mathit{control}$$

This has been the approach taken in recent research with the INTERRAP architecture.

Although we use logic in the specification of goals, as we indicated by our comments on the example above, logic alone does not lead to an emotionally intelligent agent. We must specify how the control component manages resources so that the resulting system is boundedly optimality. We must also specify how goals arise (through Γ^{LPL} and Γ^{BBL}), how goals are weighted and how planning and cooperation eventually lead to actions (through Δ^{LPL} and Δ^{BBL}). This combination of control within the layers of our architecture and the way the layers influence one another and generate actions we label *motivation*, resulting in the following equation:

$$agent = logic + motivation$$

When the motivational component is such that the resource consumption of the agent delivers something approaching bounded optimality, then it is our contention that we will have an emotionally intelligent agent.

There has been some work with the INTERRAP combining techniques for reasoning about resources and utilities with symbolic reasoning (Burt 1998). This has mainly concentrated on reasoning within the planning layer. The research that is now needed is based on extending this reasoning and incorporating related resource-oriented mechanisms into the control component of the architecture.

The search for bounded optimality is not, however, the sole reason we are interested in emotional mechanisms. We see animated characters playing an important part in the future computing environment and that agent-oriented techniques will be used to program them. Indeed, we think it is through programming these animated characters that agent-oriented programming will reach mainstream computing, in the same way that object-oriented programming arrived with the programming of window-based interfaces. We are thus forced to examine the emotions we are simulating in the characters being programmed. Several researchers have achieved this simulation with a separate module that reasons about emotional state according to a particular psychological model (often that of (Ortony, Clore, Collins 1988)). It is our hope that eventually these two strands of research will eventually merge: that is to say that the emotions of the character can be seen as natural extensions of the control constructs that we need in the character anyway in order to program it efficiently.

Acknowledgements

Part of this work was funded by the BMBF (German Federal Ministry for Education and Research).

References

Bradshaw, J. M., ed. 1997. *Software Agents.* MIT Press.

Burt, A. 1998. Symbolic and utilitarian reasoning in the interrrap architecture. Technical report, DFKI. In preparation.

Fisher, M. 1994. Representing and executing agent-based systems. In *Pre-Proceedings of the 1994 Workshop on Agent Theories, Architectures and Languages.*

Gregory, S. 1997. A declarative approach to concurrent programming. In *Proceedings of the 9th International Symposium on Programming Languages: Implementations, Logics, and Programs*, 79–93. Springer-Verlag.

Kowalski, R., and Sadri, F. 1997. Towards a unified agent architecture that combines rationality with reactivity. To appear.

Kowalski, R. A. 1979. Algorithm = logic + control. *Communications of the ACM* 22(7).

Müller, J. P. 1996. *The Design of Intelligent Agents*, volume 1177 of *Lecture Notes in AI.* Springer-Verlag.

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

Reynolds, C. W. 1987. Flocks, schools and herds: A distibuted behavioral model. In *Proceedings of SIGGRAPH'87.*

Russell, S. J., and Subramanian, D. 1995. Provably bounded-optimal agents. *Journal of Artificial Intelligence Research* 2:575–609.

Russell, S. J., and Wefald, E. H. 1991. *Do the Right Thing: Studies in Limited Rationality.* MIT Press.