An Emotional Agent

The Detection and Control of Emergent States in Autonomous Resource–Bounded Agents

Thesis Proposal
Research Progress Report 3

Ian Wright
The Cognition and Affect Project
Supervisor: Aaron Sloman, Thesis Group: Glyn Humphreys & David Murphy
August 1994

Abstract

In dynamic and unpredictable domains, such as the real world, agents are continually faced with new requirements and constraints on the quality and types of solutions they produce. Any agent design will always be limited in some way. Such considerations highlight the need for self-referential mechanisms, i.e. agents with the ability to examine and reason about their internal processes in order to improve and control their own functioning.

Some emotional states, such as debilitating grief, could be considered as disruptive responses to new circumstances. In other words, the design is unable to 'cope' adaptively. This research wishes to investigate the generation of such 'emotional' states, their detection by self-referential mechanisms, and possible coping, prevention or solution strategies within a unifying framework of an extant broad agent architecture.

There are many theories of emotionality. Sloman's (1981) 'attention filter penetration' theory provides an architectural framework for the understanding of 'emotional' states that involve partial loss of control of attention – so-called *perturbant* states. Beaudoin (1994) has designed an agent architecture (NML1) that, when implemented, should exhibit such states. Smith (1986) states three requirements for self-referential systems: a theory of the self, the use of this theory in guiding the behaviour of the system, and the ability to swap between action in the world and reasoning about the self. The NML1 design only meets the final requirement. Also, the design requirements for self-control of problematic processing states have yet to be systematically addressed.

Therefore, this work will implement a prototype agent architecture, exhibiting some kind of perturbant state, based on a revised NML1 design. The design will be extended to meet the requirements for self-referential systems. The aim is to design and partially implement an NML1 that is self-referential, can exhibit 'emotional' states, detect such states, and try to do something about them.

Results from this research will contribute to autonomous agent design, emotionality, internal perception and meta-level control; in particular, it is hoped that we will i. provide a (partial) implementation of Sloman's theory of perturbances within the NML1 design, ii. investigate the requirements for the self-detection and control of processing states, and iii. demonstrate the adaptiveness of, the need for, and consequences of, self-control mechanisms that meet the requirements for self-referential systems.

Contents

1 Introduction

The title of this proposal indicates that the aim of the research is to build an autonomous agent, functioning in a simulated world, that enters into processing states that can be characterised as 'emotional' from an information processing perspective. The word 'emotional' can cause terminological confusion, and is here used to locate the research within a particular field. Its use should not imply that emotions constitute a single phenomenon, or that the research will address the full variety of emotional phenomena. A number of theoretical positions are assumed, namely: an abstract machine ontology of the processes implemented in human and animal brains[1], Sloman's 'attention filter penetration' theory of emotionality, and the design-based approach to the investigation and design of intelligent artifacts. None of the positions are defended here as such a defence would lie outside the scope of the thesis proposal.[2]

The subtitle – the control of emergent states in autonomous resource-bounded agents – states the purpose of this research with greater precision. A type of emotional state – a *perturbant* state – is considered as one form of emergent state arising from a complex architecture meeting the requirements for autonomy in a dynamic and unpredictable environment. The economy of such problematic emergent states (their origins, effects, detection and control) is the topic of concern.

For self-control of emergent states an agent needs self-referential mechanisms, i.e. the ability to examine its own internal processing, compare its current functioning to a theory of normative functioning, and, if possible, control itself to further its achievement of current intentions. An extant agent architecture – the NML1 design – will be used as a starting point in design-space to explore these issues.

Therefore, the basic aim of the research is to design and build an autonomous agent functioning in a dynamic and unpredictable domain that can exhibit a type of perturbant state, which, depending on various factors, it may detect and attempt to control.

The following section considers autonomous agents, perturbances and their connection with self-control. The next section states the goals of the research. The final section of the introduction summarises the contents of the thesis proposal.

[1] See (Sloman, 92b).

[2] However, see the appendix I for a summary of the design-based process and section 3.1.1 for an exposition of the 'attention filter penetration' theory.

1.1 Autonomy, emotionality and self-control

Agent architectures are abstract machines designed for embodiment and autonomous behaviour within dynamic environments. The agents considered here have multiple motives, gain information about the state of the environment via sensory modalities, and alter their environments to achieve goals via efferent action. An autonomous agent should maintain itself over time and be the source of its own control.

Autonomous behaviour implies that an agent has its own agenda within a world. For example, consider a word-processing program that passively accepts commands from a user and only occasionally attempts to control its own input by posting error messages or rejecting illegal input. In contrast, an autonomous agent will have a set of concerns that it will attempt to satisfy via action that affects its own input, i.e. it is an *active* program, as opposed to reactive, and does not need a user to control its actions.

Humans and animals conform to this definition of autonomous agents. Humans, and perhaps many animals, also exhibit emotionality. Indeed, the importance of emotional behaviour can be understood by considering literature, which is replete with examples of emotional behaviour – for instance, consider Greek tragedy with its grand themes of loss, pain, jealously and suffering. Two questions immediately arise – what is 'emotionality', and why is it such an important aspect of many autonomous agents, particularly humans?

The answers to these questions are interconnected. Consideration of the requirements for autonomy and analysis of what constitutes emotional phenomena converge to an architectural answer (Beaudoin & Sloman, 91 & 93). Resource-bounded autonomy in a sufficiently rich, dynamic and unpredictable environment requires some type of selection process that protects resource-limited processes from unnecessary interruption and computation. And an essential characteristic of many emotional states is that they cause partial loss of control of attention. For example, a mother grieving for her departed daughter is unable to think about anything else, despite the fact that she may have many pressing goals to achieve. In other words, by performing a computational abstraction (physiological responses and reports of subjective feelings are temporarily placed to one side) we can posit that an important subset of emotional phenomena is closely connected with the interruption of a resource-limited control mechanism. Autonomy and perturbances are, therefore, architectural cousins; hence the importance of emotions in autonomous agents such as ourselves. (These issues are developed further in later sections.)

In dynamic and unpredictable domains, such as the real world, agents are continually faced with new requirements and constraints on the quality and types of solutions they produce. Any agent design will always be limited in some way. It is finite, whereas the world can potentially provide infinite combinations of sensory inputs – new situations – that expose design flaws or limitations. For example, consider long-term debilitating grief that can prevent the person from carrying on with their life. This could be an example of an architecture responding maladaptively (with regard to pursuing its concerns[3] to a new, unexpected eventuality. The need for learning arises here. However, learning will be placed to one side in this proposal and instead we will concentrate on another type of mechanism, probably formed by some kind of learning process, which is a design solution to the problem of achieving greater adaptivity from an extant design: self-referential mechanisms.

An immediate problem is what constitutes a 'self'. Without an answer we will be unable to distinguish between a self-referential mechanism and other types of mechanism. 'Self' is a notoriously abstract concept and has different meanings within different research fields. For our current purposes we will define the 'self' as the totality of architectural elements at time T, and a self-referential mechanism as a new architectural element formed after time T designed to overcome a perceived limitation of the earlier design and whose object of reference is either the earlier design or some subset of the earlier design. Therefore, different self-referential mechanisms may refer to different 'selves'. This is an architecturally dependent, *genealogical* definition of the self, and allows us to use such useful, intuitively graspable concepts such as 'self-control' without commiting a reification error with the concept 'self'. [4]

Investigation of dynamic, evolving, broad architectures is beyond the reach of current research efforts. However, agent architectures are evolving even as these lines are read. An examination of current designs in the AI literature for autonomous agency reveals a number of common elements, namely: mechanisms for goal generation, goal management and execution, i.e. the formation of intentions from beliefs and desires. For example, the NML1 (Beaudoin, 94; Sloman et al., 94) design has generactivators (generate and reactivate goals), an interpreter that forms intentions from goals, and an execution subsystem. Therefore, we can define this stage in the design of autonomous agents as what we mean by the 'self'. The self, therefore, is that which forms intentions from beliefs and desires (and will differ in detail from architecture to architecture).

We can now link autonomy and perturbances with self-control. To recap, the requirements for resource-bounded autonomy pose design problems. One design solution posited is that of a filtering mechanism that protects resource-limited processing. Perturbances are deemed to be closely connected to interruption of attention. Often perturbant states can be considered disruptive rather than interrupting, as in the case of debilitating grief. In other words, a complex architecture can enter into states that are maladaptive, i.e. limitations of the extant design are revealed. A type of self-referential mechanism – a self-control mechanism that can detect problematic states and attempt to do something

[3]The word 'concern' is used in a similar way to Frijda (1986), i.e. concerns are that which cause the generation of goals. For example, the concern to prevent hunger may cause the generation of a goal to eat fruit.

[4]A distinction is made between a self-referential mechanism and a self-referential system. The former is a mechanism, embedded within a self-referential system, that refers to the 'self', whereas the latter is a system that contains self-referential mechanisms referring to 'itself'.

about it – would therefore be adaptive and improve the functioning of the extant design.

Some people experiencing grief are 'stronger' than others, can lay aside their pain and make a supreme effort to rebuild their lives – they have better self-control.

1.2 Goals of research

The research is intended to contribute to three related areas: first, Sloman's 'attention filter penetration' theory of emotionality; second, the extension and implementation of (probably a subset) of the NML1 agent architecture design; and third, consideration of the requirements for self-control, in particular, how an agent could detect its own problematic processing states, e.g. perturbant states, and try to do something about them. This work, therefore, builds directly on previous work in the Attention and Affect project (Wright, 93) and aims to extend the requirements, design and implementation of the extant broad agent architecture. The design and implementation of a self-referential agent, able to detect its own global processing states and employ self-control strategies for improved functioning will be a new contribution to the field.

A number of questions need to be addressed, primarily:

- Does the NML1 design meet its requirements, i.e. will a prototype implementation actually *work* in the domain?
- If not, in what ways will the design need to be improved, refined or altered?
- Does a working implementation enter into the expected problematic processing states theoretically postulated? E.g., perturbant states that involve repeated interruption of ongoing processing by an insistent goal.
- How does this implementation relate to Sloman's 'attention filter penetration' theory of emotionality?
- In what ways the theory will need to be clarified and revised after the process of implementing a demonstration of an 'emotional' agent?
- What are the requirements for an agent to control its own (problematical) global processing states?
- What set of designs for self-control mechanisms could satisfy these requirements?
- What consequences do self-control mechanisms have for the rest of the architecture and how do they effect emotionality?

- Can any new theoretical conjectures be obtained, especially with regard to pathologies of human motive processing and extant psychological theories?

The consideration of such questions will encompass theories of emotion, agent architectures, self-referential mechanisms such as self-modelling and self-control, internal perception, and meta-level control. The research will attempt to unify disparate concerns within the design and implementation of a broad agent architecture.

However, it is important to have definite research aims that can be achieved in the time available. And it is better to begin with modest aims and add new goals later rather than be overly ambitious from the start. Consequently, I'll summarise two research goal scenarios, one modest and the other ambitious.

Modest goals. Build a working prototype of the NML1 design, add a plausible emotional scenario to the agent's domain in order to explore the concept of perturbant states, and relate this implementation to the 'attention filter penetration' theory of emotionality. It is expected that the design of NML1 will be extended during this process. Requirements for and design of self-control mechanisms will be presented. Achievement of the modest research goals will directly contribute to previous work in the Attention and Affect project, the design of broad agent architectures, and to the understanding of emotionality.

Ambitious goals. Achievement of the modest goals, plus: implementation of self-control mechanisms within the agent that can detect a class of problematic, global emergent states, perform a diagnostic analysis based on a theory of the self, and attempt to improve functioning by either preventing the emergent state, coping with it, or eradicating the state entirely. The implementation of self-control mechanisms in the agent will be related to the space of possible designs for self-controlling mechanisms, including an analysis of the various design options that meet the requirements. How self-control is learnt and mechanisms formed in a dynamic architecture will be considered, and such considerations related to extant psychological theories of self-control (for example, (Kuhl, 92; Kuhl & Kraska, 89; Heckhausen & Kuhl, 85; Rachlin, 94)). The varieties of self-control of emotional episodes will be investigated, including how emotions may decay over time. Achievement of the ambitious research goals will, in addition to the contributions made by the modest goals, add to the understanding of the mechanisms for self-control in people, especially with regard to emotional episodes, and help clarify the terminology and concepts within psychological theories from a design-stance.

The title of this thesis proposal refers to the modest research aims and the subtitle to the ambitious research aims. It is my expectation that the actual achievements of this research will fall somewhere between these two scenarios.

1.3 Outline of proposal

The structure of the thesis proposal is as follows. Section 2 presents a literature review of existing agent architectures from the field of AI. A distinction between reactive, deliberative and hybrid architectures is made, and the problem of decision-theoretic control flagged. Critical comments are provided where appropriate and any similarities between architectures highlighted. Requirements and constraints for autonomous agency are briefly summarised. The review concludes with a summary of the NML1 architecture, which will serve as the architectural basis of the research. Section 3 presents an overview of emergent states, concentrating on problematic emergent states in a NML1 (or PRS) architecture. The consideration of emergent states is linked to a discussion of emotions. A few emotion theories are briefly discussed. Simon and Sloman's 'interrupt' theories are reviewed and the notion of a perturbant state introduced. The requirements for detecting such global states by self-controlling mechanisms are discussed, followed by design options for the control of such states and how this may relate to emotional behaviour. Section 4 presents research aims, including a plausible emotional scenario that can be implemented in the nursery domain to explore the issues outlined in section 3. Criteria for success are given and how the research is to be evaluated. Finally, a high-level plan is provided for the production of the final thesis within the time available.

2 Agent Architectures

The agent architectures reviewed in this section are attempts to provide mechanisms for coherent and effective behaviour in complex, unpredictable and dynamic domains, such as the real world. The requirements necessary to achieve such behaviour, and some constraints on any proposed solutions, are outlined in section 2.1. In section 2.1.1 an outstanding problem in the design of autonomous agents is flagged as an illustration of the problems involved in such an endeavour. The following sections describe agent architectures selected from the extant literature, providing critical comments where appropriate. Agent architectures are classified according to which aspect of the requirements for autonomy they are designed to address. For example, reactive architectures are concerned with immediate responses to the current situation, deliberative architectures[5](not reviewed here) are concerned with goal-oriented planning, and hybrid architectures (both uniform and layered) are concerned with combining both reactivity and deliberation. It is assumed that any sophisticated, autonomous agent will need to combine both reaction to unforeseen events and forward planning for effective operation; therefore, it is existing hybrid architectures that are of most interest and relevance.

2.1 Requirements and constraints for agent design

A dynamic, unpredictable, real-time environment entails certain requirements for autonomous agent functioning. For example, there will be too much information (both externally in the environment, and internal within the agent) in any one current situation (Hayes-Roth, 90). The agent will need to focus its attention and ignore irrelevances. Information will be widely distributed both in time and space, requiring the agent to search for relevant information and remember and recall past information. The quality of sensory input will vary; in other words, the agent will need to be able to cope with incomplete or partially incorrect information. Over time the environment will place diverse demands on agent functioning: for example, at one moment the agent may have very little to do and have the luxury of deliberation while at the next moment the agent may need to perform many complex tasks quickly and efficiently. The unpredictability of the environment renders complete planning prior to action impossible. Instead, opportunities and threats will need to be constantly monitored for. Also, some planning will need to be sketchy, the precise details chosen during execution. Therefore, an autonomous agent needs to be robust, flexible and have the ability to cope with variable stress to survive in a 'real-world' environment (Beer et al., 90).

A number of conditions need to be met for autonomous activity. The agent must be able

[5]Such as IRMA (Bratman), AUTODRIVE (Wood), Behaviour Hierarchies (Durfee & Montgomery), Agent-Oriented programming (Shoham) and Homer (Vere and Bickmore). For an overview, see (Ferguson, 92).

to react in a timely manner – acting at opportune moments and curtailing its reasoning processes if necessary. The agent's behaviour needs to be coherent, i.e global coordination of actions, otherwise the agent will be unable to successfully complete a plan. This requires the ability to select between multiple motives (Sloman, 85), prioritise goals and decide on a level of commitment towards current intentions. But to react to new, motivationally relevant events in the environment the agent will need to interrupt its ongoing processing and switch its attention to new contingencies (Sloman, 87). And to detect such events it must be able to generate motivations asynchronously to current processing. A level of coarse-grained parallelism is therefore likely to be necessary (Maes, 90) to enable execution of current intentions and at the same time check for new information that may entail intention revision. The agent will need to be adaptable, i.e. alter its behaviour in different and new situations. It will need to learn from mistakes and successes. Therefore, architectures that model autonomous agency will need to integrate a wide range of behavioural capabilities. (Bates et al., 91) term such architectures as 'broad'.

Finally, the agent is constrained in various ways – it is not omniscient or omnipotent. Its computational resources are assumed to be bounded. For example, humans find it difficult to listen to more than one conversation at once. Also, the agent will be physically constrained – it will only be able to move at a certain pace, manipulate a finite number of objects and so on. Good design solutions will manage an agent's finite resources as efficiently as possible. The following section highlights one aspect of this: the problem of resource-bound practical reasoning.

2.1.1 Decision-theoretic control

The need to meet temporal deadlines coupled with resource-boundedness requires that an agent be able to control its search for solutions, trading quality of solution for speed of response if need be. A number of methods have been proposed to deal with this problem. (Boddy & Dean, 89) have described an *expectation-driven iterative refinement* framework. Planning within this framework uses a set of decision procedures called *anytime algorithms* that can return answers at any time but return better answers with more time. The use of anytime algorithms together with a deliberation-scheduling algorithm, which allocates resources to anytime algorithms according to projected expectations regarding their performance, is put forward as a solution to time-dependent planning problems. (Lesser et al., 89) provide an alternative to anytime algorithms.[6] They state that a problem solver should use different types of approximate processing to arrive at solutions, i.e. depending on the time constraints and the nature of the problem the agent can employ different reasoning strategies with different computational costs. They provide a taxonomy of approximations and dimensions of solution quality. For example, data approximations can provide an abstract view of data resulting in a simpler search space. These two approaches are particular cases of the more general problem of decision-theoretic control

[6]Although their framework can include iterative refinement of solutions.

(rational deliberation under resource limitations). (Horvitz et al., 89) provide a model of normative rationality under scare resources based on utility theory (or decision theory). Reasoning about the utility of acting now or deliberating further, and controlling this decision-theoretic inference itself, is a problem that needs to be solved for effective autonomous functioning. They maintain that decision theory provides the foundations for a principled approach to meta-level decision making and use measures of value and utility to achieve this. They utilise Good's (1968) distinction between type I and type II rationalities: type I is rationality that is consistent with the axioms of decision-theory regardless of the cost of inference, and type II is behaviour that takes into account the costs of the reasoning process itself. An autonomous agent in a dynamic environment needs type II rationality. However, (Horvitz, 87) moves the emphasis away from providing a complete normative analysis of decision-making. He states that limited resources for engineering and computation make such an analysis impossible. Also, in many situations normative reasoning (i.e., basing one's decisions on expected utility) is inadequate due to agent resource limitations. In other words, knowledge about the reasoner itself needs to be included in the reasoning problem. (Doyle, 89) proposes a theory of *rational self-government* (a theory of efficient thinking) based on decision-theoretic rationality. Truly rational thought is, in this conception, defined as each decision step having the maximum expected value with respect to the agent's expectations and preferences. Doyle argues the need for focus of reasoning (selective rationality), and effective allocation of mental resources; and he provides an analogy between political superstructures and the management of rationality. (Bratman et al., 88) address the same problem of resource-bounded reasoning and argues for the efficacy of a *filtering mechanism* that constrains the overall amount of practical reasoning necessary. This is an architectural solution to the problem of limited rationality, as opposed to using a revised utility theory. They provide an abstract design for the practical reasoning component of an autonomous agent, including a filter that protects the reasoning system and a filter override mechanism that allows interruption when opportunities or threats to current intentions are detected.

Investigation of the utility of utility theory, and the proposal of various design solutions to constrain reasoning, are current areas of research concern. Many unsolved problems remain.

2.2 Reactive architectures

In reaction to the unrealistic requirements of classical planning a new design paradigm has been proposed, notably by Rodney Brooks at MIT, that seeks to situate and test agents within real world domains. Brooks has labelled this research paradigm 'nouvelle AI' (Brooks, 90). Nouvelle AI declares that the physical symbol system hypothesis of traditional AI is fundamentally flawed. An alternative hypothesis is counterposed: the physical grounding hypothesis, which states that intelligence is decomposed into behaviours rather than functional, information processing modules. In Nouvelle AI each module produces a behaviour whereas in GOFAI (Good Old-Fashioned AI) combinations of modules produce behaviour.

The subsumption architecture (Brooks, 91a) is an instance of Nouvelle AI. The behaviours of a physical robot are the building blocks of the architecture. Brooks has built a sequence of subsumption architecture robots that operate successfully in the real world. Each behaviour subsystem is a computational unit connected to others via wires that form a layered architecture. There is no explicit representation of goals or other symbolic structures. Some behaviours inhibit or prevent others, 'subsuming' the lower level behaviour into a higher level synthesis. For example, the behaviour *avoid-obstacles* could be subsumed by the higher level behaviour *chase-light* to synthesise a behaviour of chasing a light while avoiding obstacles. By adding layers of behaviour producing subsystems Brooks hopes to engineer increasingly complex robots that are robust enough to operate in the real world (Brooks, 91b).

The work of Agre and Chapman is very much in this vein. In (Agre & Chapman, 87) they describe Pengi, an agent that plays a commercial arcade video game called Pengo, a domain that places real-time demands on agent functioning. The design of Pengi is motivated by a theory of activity that emphasises the importance of routines in the dynamics of everyday life. A routine is a pattern of interaction between an agent and its world, and is not represented within the agent. In addition to routines, a new way of representing the world is introduced, variously called indexical-functional aspects, indexical representation (Chapman, 89) or deictic representation. A deictic representation represents only what is necessary, immediate and functionally important to the agent in its current situation. For example, instead of representing the location of a block as, say, **(AT BLOCK-213 427 991)**, Pengi will employ unitary entities such as *the-block-I'm-pushing* that gain semantic significance within a particular action context.

In addition to Pengi, Chapman has developed the Sonja system (Chapman, 90) that uses instructions in the course of visually-guided activity. Sonja is an agent operating in the Amazon domain; she has various tasks to perform and a human instructor can help and guide Sonja via natural language commands. Sonja has the ability to interleave various courses of action at once but, like Pengi, uses no explicit representation of plans. The system, a lá Brooks, is implemented as a digital circuit with explicit connections between sub-elements. Underlying this system is the belief that the world is its own best model, i.e. it is more efficient to look at the current, concrete situation for action selection than relying on deliberation and complex, internal models of the real world.

The behaviour-based approach of Brooks and the concrete-situated approach of Agre and Chapman share the fundamental concern of building architectures that are robust and successful in real world domains. Both eschew traditional plan representation and execution, and both believe that for AI to progress it must first solve the problems of situated activity. Brooks' robots are a practical demonstration of the efficacy of such an approach for low-level behaviours. The strength of behaviour-based architectures lies in their ability to use local patterns of activity in their current surroundings as a basis for generating pre-compiled or simple to compute action responses. However, claims and theoretical assertions are made for Nouvelle AI that are untenable. As the behaviour-based approach could be construed as a direct challenge to top-down research approaches in particular and the catholicity of the design-based approach in general it is worthwhile taking some time providing a critical overview.

2.2.1 Reactive architectures in context

First let us consider representations. Chapman states that diectic representation is a new form of representation based on relationships, not identity, i.e. diectic representations refer to relations between objects whereas identity representations refer to the objects themselves (in Chapman's view at least). This claim can easily be weakened: a strong tradition of representing elements of the world in terms of their functionality in relation to the viewing subject exists in philosophy, finding its definitive formulation in such Existentialist thinkers as Heidegger, Husserl, Merleau-Ponty and Sartre (Cooper, 90). Heidegger's concept of ready-to-hand (*zuhanden*) is precisely the idea of constituting the world in terms of functionality.

Diectic representations undoubtedly play a role; indeed, they may be a phylogenetically older form of representation more suited to routine activity and simple stimulus-action responses on the level of insect-like intelligence. However, explicit representation schemes are needed for more complex control tasks and behaviours. Recent work in planning concerning the detection of opportunities ((Pryor, 94) see later) could provide an intermediate representational layer between diectic representations and traditional plan representations via 'reference features' that link plans to functionally relevant features of the environment.

Chapman criticises identity-based representation schemes as they 'take for granted that the world comes already neatly divided into objects.' This attacks traditional representation schemes from a differing set of assumptions. 'Objective' representations of the world, the starting point of many AI implementations, can be the result of functional differentiation of concepts through activity. That this formative process is ignored or assumed by identity-based representation schemes is not an argument against their use or existence.

Finally, there is an implicit assumption contained in the statement that the 'world is its own best model'; namely, that perception is the sole source of information about the world. A 'concrete' situation for an agent also includes its own internal environment: beliefs, desires, memories and so on. In many instances action selection will be more easily achieved by 'looking' at past experience rather than the immediate, external environment; for example, deciding to press the right button rather than the left because the previous time you pressed the left button you received a shock.

Chapman claims that 'the problem of action selection is easy and has been overemphasised' as 'Sonja performs actions because they make sense in concrete situations, not because they are the next step in a program'. This is clearly a case of solving a problem by avoiding it: the requirements for Sonja are such that it is not faced with the problem of multiple motives, or even multiple strategies for achieving the same action outcome. For example, Sonja has no long-term goals and therefore the problem of such a goal conflicting with the demands of the immediate situation cannot arise. The problem of action selection is effectively factored out. Also, some action selection requires forward planning using hypothetical reasoning to search for solutions. The Sonja architecture is incapable of providing this functionality.

Let us now examine the physical grounding hypothesis of Brooks. The first thing to note is that it is not a hypothesis in any real sense, rather it is a design recommendation that systems should be built bottom-up using behavioural decomposition and thereby generate representations that are causally connected to the environment, i.e. physically 'grounded'. Counterposing this design attitude to the physical symbol system hypothesis, which is essentially an ontological statement concerning the embodiment of abstract symbols in physical machines, is misleading.

Brooks can be viewed as directly challenging a top-down design-based approach in the following quote:

> I, and others, believe that human level intelligence is too complex and little understood to be correctly decomposed into the right subpieces at the moment and that even if we knew the subpieces we still wouldn't know the right interfaces between them. Furthermore, we will never understand how to decompose human level intelligence until we've had a lot of practice with simpler level intelligences.

This argument can be summarised as bread today, jam tomorrow and falls prey to *reductio ad absurdum*. Science would need to drop any pretensions to economic theory, social theory, psychology and so on, and instead concentrate on non-complex phenomena; however, even a simple Hydrogen atom is a rich, complex and unpredictable structure not entirely captured by quantum theory. Nothing is simple. What Brooks ignores is a fundamental idea underlying all scientific endeavour – that of levels of abstraction. A top-down decomposition, taking as its starting point the external behaviour of a complex system, can make strides towards a theory of intelligence at a high level of abstraction. Undoubtedly, such a theory may be wrong and miss many significant details, but it may also contain important elements that provide an explanatory framework that aids investigation of the phenomena; and, more importantly, we can learn from our mistakes. There is a need for both approaches. Bottom-up approaches need top-down theories to constrain possible design solutions and *vice versa*. It should be noted that the design-based approach recognises this and subsumes both top-down, middle-out and bottom-up approaches within a unifying framework.

Brooks provides a further argument, based on evolution, for concentrating on perception and action within a bottom-up design approach. He states that it took much longer for nature to solve the design problems of situated activity when compared to the time taken to evolve higher level reasoning abilities; hence, the design problems of situated activity must be harder and consequently AI should concentrate its efforts in this area. (Etzioni, 93) makes the point that there is no correlation between time taken to evolve and design complexity; in fact, it is well known that evolution occurs in fits and starts (the notion of punctuated equilibria) and is heavily dependent on factors such as climactic change.

There are many more arguments that can be marshalled against the behaviour-based approach and Nouvelle AI in general. For example, it is a trivial matter to demonstrate that humans employ plans and think ahead. The assumption that non-trivial behaviour can be strictly situationally determined has yet to be demonstrated. It is inconceivable how any behaviour-based architecture could cope with explicit global task constraints, such as deadlines. (Pryor, 94) makes the point that behaviour-based architectures make use of world regularities at design-time (by pre-compilation of action selection in the connectivity of wires) but makes no provision for discovering such regularities at execution-time. Brooks recognises there may be a problem when attempting to scale up the subsumption architecture to accommodate many more behaviours. Therefore, the problem of coordination and control of behaviour producing subsystems remains. Such systems may lack flexibility; for example, the reaction time of the system may diminish as the number of behaviours increases. (Norman & Long, 94) illustrates how such behaviour-based systems become inefficient when a multiple-goals requirement is introduced and argues that a symbol manipulating mechanism is necessary to overcome this drawback. (Etzioni, 93) shows that simulated domains, such as the UNIX operating system, can satisfy all the real world requirements that Brooks deems necessary for situated activity; and he also makes the important point that designing and implementing 'softbots', or simulated, software robots, is a much speedier process than building real, physical machines.

The claims of greater validity made by Nouvelle AI over GOFAI cannot be supported. Many of the arguments stem from designers starting from different sets of requirements. It is highly likely that subsumption-like, behaviour producing subsytems exist in nature and have evolved to solve the problems of situated activity outlined by Brooks, Chapman and Agre. Different designs satisfy different regions of niche-space. To argue that one approach is intrinsically superior to another is unnecessarily divisive. The design-based approach views differing research paradigms, be they bottom-up, top-down or middle-out, as essentially complementary. AI can afford to advance on all fronts, not just the one.

2.3 Hybrid architectures

Hybrid architectures are concerned with combining reactivity and deliberation[7], i.e. event-driven and goal-driven activity.

Layered hybrid architectures are designs that address the need for reactivity and deliberation by proposing separate systems for each task and a layered control framework for their integration within a single agent. Firby's RAP system, Ferguson's TouringMachine, Pryor's PARETO and Bates' Hap are all examples of layered hybrid architectures, although this list is by no means exhaustive.

2.3.1 Combining Reactive Action Packages with deliberation

(Firby, 87) makes the point that having to choose actions at execution time is unavoidable in a complex, dynamic domain; i.e., reactive planning must occur at some level in any autonomous system to maintain robustness. An agent cannot plan completely in advance: uncertainty prevents correct reasoning, and urgency constrains the time available for such reasoning. Reactive planning of some kind will therefore be needed. A more deliberative, classical planning scheme could then be implemented upon such a reactive ground.

Consequently, Firby has proposed a model of purely reactive planning based on the concept of Reactive Action Packages, or RAPs. Each RAP can be viewed as an independent entity embodying a goal that competes for processing resources with other RAPs. It is important to note that there is no prediction of future states within a RAP system – plan selection is done entirely at execution time. A RAP may contain explicit sensory tasks within its plan allowing the same plan execution mechanism to deal with action execution and sensory guidance. Each RAP obeys three principles while running: which action to execute next is based only on the current world state, when a RAP finishes successfully it is guaranteed to have satisfied its goal, and a RAP will only fail if it does not know of any way to reach its goal from the current state. Each RAP has a pre-defined set of methods for achieving a goal and only need choose between these paths (called the task net) rather than construct new ones. A RAP, therefore, is a structure that links a goal, a success test for achievement of the goal, a collection of methods to achieve the goal applicable in different contexts, and invocation conditions that determine when a particular RAP is appropriate.

The RAP control algorithm is designed to address the problem of execution monitoring

[7] Not to be confused with systems that combine connectionist and symbolic mechanisms. The hybrid architectures outlined here are designed to combine reactive and deliberative capabilities, and could employ connectionist and/or symbolic methods to achieve this aim.

and replanning in uncertain domains. A RAP interpreter and execution queue provide a mechanism for coordinating competition between RAPs. Such a scheme allows interleaved RAP execution as, for example, when a running RAP stops and returns to the execution queue to wait for a subgoal to complete; in this situation the interpreter can choose another RAP to run in its place. The problem of goal selection, or choosing which RAP to run next, is based on temporal deadlines and an ordering on RAPs made by task nets.

Two limitations of the RAP system are outlined by Firby. A RAP may fail without preventing the original conditions in the world from re-generating the failed RAP. This could lead to an indefinitely long loop. Also, the RAP system cannot think ahead. Both these limitations point to the need for an extra layer of control that places constraints on RAP behaviour prior to execution; in other words, neither urgency or uncertainty obviate the need for more deliberative decision making. Hence, in (Hanks & Firby, 90) the RAP system is extended by considering the extra, deliberative layer of control needed in an effective, autonomous agent.

The addition of planning ahead and reasoning abilities generates two new design problems: how to deliberate, and how to coordinate deliberation and reactive execution. This problem is further divided into the representation problem (how to model a complex and dynamic world) and the control problem (how to manage such information so that the agent acts effectively and efficiently). Any solution to the control problem must be able to curtail the deliberation process at any time in order to guarantee reactivity.

The combination of a RAP execution system with a deliberative layer forms a layered architecture. The execution system has the task of generating atomic actions from a plan using no projection, confirming the results of each action to ensure a step has been achieved, and watching the environment for any threats and opportunities to its current project. A deliberation system needs to incrementally improve the agent's plans, make predictions about the future state of the world, and react to new information and assess how it affects the agent. Any agent architecture must coordinate an execution system and a deliberative system. Hanks and Firby believe that the combination of an execution system based on RAPs and a deliberation system based on a probabilistic world model manager and projector will meet this criteria. Their layered architecture is still in development and they provide few details of the deliberation system.

2.3.2 TouringMachines

Another example of a layered hybrid architecture is Ferguson's TouringMachine (Ferguson, 92). The TouringMachine is an integrated software control architecture designed for controlling the actions of autonomous agents operating in complex environments; in particular, the TouringWorld, a multi-agent traffic domain. The design consists of separate activity-producing behaviours in a layered control framework, and in this sense resembles the behaviour producing subsystems of Brooks. However, there the similarities end: the TouringMachine uses explicit goal and plan representation, and each activity producing layer is not a simple control system connected via wires, but a more or less sophisticated control algorithm.

The TouringMachine has three different control layers: a reactive layer, a planning layer and a modelling layer. All layers operate concurrently and are independently motivated, each with its own internal computational mechanisms. Each layer is also connected independently to sensory input and effector output. These three layers are mediated by an enveloping control framework.

Each layer is intended to model the agent's environment at a different level of spatio-temporal abstraction. For example, the reactive layer provides the agent with reactive capabilities for immediate or short-term contingencies that higher level layers would have insufficient time to compute responses to. In Ferguson's implementation the reactive layer is hard-wired, domain-specific and reactive, i.e. reminiscent of a behaviour-based subsystem. The planning layer generates and executes plans of action to achieve the agent's goals. Use is made of sketchy, procedural plan structures and the system can defer commitment to a specific subplan until run-time. The modelling layer's function is to attempt to detect and predict potential goal conflict situations; it achieves this by monitoring execution (observation), abductive inference (explanation), and temporal and counterfactual reasoning (prediction). The modelling layer will then attempt to avoid the detected or foreseen goal conflicts. Also, using knowledge about its own functioning the TouringMachine is able to project this knowledge onto other entities in the environment and model their behaviour. The modelling layer, therefore, provides the agent with reflective and predictive capabilities (what Ferguson calls metaplanning), i.e. a meta-level layer of control. The integration of the three layers to produce consistent behaviour is achieved using a control framework. This control framework consists of a set of control rules (suppressors and censors) that resolve perception and action command conflicts arising from instructions sent from different layers.

The TouringMachine architecture has been implemented and extensively tested in the TouringWorld testbed. However, many of the planning problems in the TouringWorld are problems of navigation: how to get from A to B, how to avoid collisions, how to arrive on time and so on. The proposed architecture may be robust in such a domain but may not generalise to richer domains. For example, the control framework, consisting of control rules, shunts the control problem into a large collection of exception rules that could prove unwieldy when the architecture is developed. The focus of attention mechanism (i.e., deciding on what to concentrate processing power) relies on a 'relatively static focusing rule set'. In other words, there is no disciplined way of controlling attention in this architecture. A final point is that the problem of decision-theoretic control (deciding whether to deliberate further or act now) is avoided by having three layers running concurrently, but the problem resurfaces within the mediating control framework.

2.3.3 PARETO and reference features

PARETO (Pryor, 94) is a plan execution system that operates in an uncertain and dynamic environment. Its embodiment and autonomous activity within such a simulated domain entitles it to be classified as an agent architecture. The design of PARETO is motivated by the view that information gathering and opportunity taking are essential aspects to planning in an unpredictable domain. In addition, PARETO makes decisions on the fly, filling in the details of sketchy plans at execution time. The system recognises opportunities and threats to its current goals during plan execution by using a heuristic based on reference features.

Pryor makes a distinction between the types of decision a planning system will need to make – deferred and unforeseen. Deferred decisions arise due to the information needed to make a decision not being available during plan formation. Deferred decisions, therefore, will require information gathering at execution time. The Cassandra system (Pryor & Collins, 93) was designed to solve this problem in a traditional planning system by including explicit decision and information gathering steps within its plan representation. PARETO, however, was designed to address the problem of unforeseen decisions, which are due to the unpredictability of the environment. Such decisions require opportunity taking, i.e. detecting when an unforeseen situation is helpful for achieving a goal (an opportunity) or harmful to a goal (a threat). PARETO recognises opportunities (and threats) by using a filtering process based on reference features (Pryor & Collins, 92). Reference features label functional stability, i.e. they mark the functionally important aspects, relative to the agent's goals, of the elements that comprise a situation. For example, an agent may have an interview for a job and notice that a thread is showing on their jacket. Before entering the interview room the agent spots a pair of scissors on a desk; immediately, the agent uses the scissors to cut the thread. Thus an opportunity is detected and taken. The reference feature used in such a transaction could have been the functional label *sharp*. The goal to cut the thread would have been labelled with *sharp* as one of its reference features, i.e. something sharp would be connected to the goal of cutting the thread; and, in addition, the perceptual recognition of scissors would have generated the reference feature *sharp*. A simple matching process would then detect the opportunity. Such reference features are argued to be cheap to infer and readily available to the agent as they are computed anyway in the normal course of perception and situated activity.

A hypothesis – *the critical factor hypothesis* – underlies the use of reference features. It states that the presence of a single factor is often crucial for the existence of an opportunity in a given situation (the presence of *sharp* in the above example). This hypothesis can be directly contrasted with the behaviour-based programming paradigm that does not attempt to use the inherent stability of the world. The critical factor hypothesis maintains that many situation elements remain constant across many different situations. As stated earlier, the use of reference features or something similar would provide a mediating representational scheme between, for example, Agre and Chapman's diectic representations and traditional goal representations. Pryor adumbrates such an idea:

> Reference features form the basis of an effective filter for opportunities because they constitute an intermediate level of conceptualising the world between the physical vocabulary provided by perception and the functional vocabulary required to reason about goals.

The PARETO plan execution system is based on Firby's RAP system, extending it via a layered control architecture consisting of a robot control, plan execution and reasoning layer. It operates in the TRUCKWORLD domain consisting of a road system, various storage buildings, building sites in need of materials and a delivery truck. The system has two kinds of goals: preservative goals (such as maintaining the truck's fuel supply) and delivery goals (such as requests for materials from workmen at the building sites). For example, the delivery truck may be travelling to **SITE 1** to deliver bricks; on the way it notices a petrol station and takes advantage of the opportunity to satisfy its fuel preservation goal.

In PARETO reference features are used not only to detect opportunities but also for resolving goal conflicts and as a principle for goal selection (deciding what to do next). An additional filter mechanism is used to detect goal conflicts using a simple characterisation of goal interactions using the reference features involved in RAPs and a table of known, problematic interactions between the reference features (with a number of additional conditions). For example, a goal could be tagged with the label *urgent* and possible interactions with other goals detected using this. This strategy for detecting goal conflicts is justified with the notion of *effective independence*, i.e. there are no significant interactions between an agent's goals, unless there is evidence to the contrary. Therefore, reference features help to focus the agent's reasoning when attempting to detect goal interactions.

PARETO also utilises reference features to tackle the problem of deciding what to do next. It chooses tasks for which there are known or predicted opportunities. Chapter 8 of Pryor's thesis is dedicated to providing heuristic principles for goal selection based on ideas such as main and side tasks, task priorities and opportunities. These principles could be extended and perhaps provide domain-independent heuristics for goal selection.

A few critical comments are in order. No methodology is provided for choosing the reference features of situation elements, and it appears that they may have been chosen in an *ad hoc* fashion. However, addressing this problem is an instance of the more general problem of integrating learning and the acquisition of reference features – an area Pryor highlights for future research. A central claim for the opportunity taking filter process based on reference features is that it is computationally inexpensive; however, no mathematical argument is provided to support this claim. Such an analysis could show, for instance, that there is no combinatorial explosion of reference feature matching. A reference feature mechanism for opportunity taking will need to be augmented by more sophisticated, deliberative reasoning; PARETO's reasoning layer (although not implemented) is designed to address this issue. But how successful are reference features? – The PARETO implementation is able to operate autonomously and detect and take opportunities in a dynamic domain. The use of reference features is an elegant and simple solution to the problem of opportunity taking, and, along with the critical factor hypothesis, is an important contribution to the field of planning and autonomous agency.

2.3.4 HAP and interactive drama

Hap (Loyall & Bates, 91) (Reilly, 93) is the reactive architectural component of Tok, a broad agent architecture (i.e., exhibits a wide range of behavioural capabilities) designed to operate in an environment that supports user participation in interactive drama (Bates, 94). The aim of the Hap architecture is to provide robust, reactive behaviour (like the hard-wired architectures of Brooks and Agre and Chapman) but also allow a sophisticated range of goal-directed, higher level actions.

Hap provides many of the mechanisms found in other agent architectures – such as arbitrary demons, multiple active goals, and situated, run-time interpretation of plans – but also claims to provide three unique features: mechanisms for detecting opportunities and threats, an easily extendable plan library, and flexible mechanisms for determining goal success without explicit characterisation of what constitutes success.

Loyall and Bates state that Hap is similar to Firby's RAPs and Georgeff's PRS by providing explicit representation of goals, supporting reactivity and handling multiple active goals. As these other architectures are reviewed elsewhere I will concentrate on the three unique features of Hap.

Hap can recognise the spontaneous achievement of a goal or subgoal by associating a *success test* with each goal node. If this test evaluates to true then the goal has been achieved and need not be pursued. This is opportunity recognition of a sort (but note it is an impoverished opportunity detection mechanism compared to Pryor's PARETO). Additionally, threats can be detected (but not predicted) by use of a *context condition* associated with each plan. If this condition fails to evaluate to true at any time during plan execution then the plan can no longer be pursued, i.e. a threat to the plan has occurred.

Plans can be added to a plan library at any time without the need to change existing plans. In order to maintain coherent behaviour the notion of *plan specificity* is associated with each plan. Specificity of plans imposes a partial ordering on plans. If more than one plan is applicable to a given goal the Hap architecture will choose the more specific plan.

A proposition need not be evaluated to true or false in order to determine whether a plan has succeeded. Instead, plan success occurs when a plan finishes. This subsumes the case of an explicit plan success test by allowing a success predicate to be the final step in the plan. This weakened plan success criterion was motivated by considering types of plans that afford no immediate success criterion, such as attempting to communicate with a friend by posting a letter.

It is debatable whether the above three features of the Hap architecture are unique – other systems can detect opportunities and threats, and allow extension of plan libraries.

Uniform hybrid architectures are designs that address the need for reactivity and deliberation by proposing a single control framework that integrates these capabilities within an agent, i.e. do not have separate subsystems dealing with reactive and deliberative functionality. Hayes-Roth's Guardian system, Fehling's Heuristic Control Virtual Machine and Georgeff's Procedural Reasoning System are examples of uniform hybrid architectures.

2.3.5 Intelligent control

In (Hayes-Roth, 90) the characteristics of real-world environments are outlined and an attempt made to formalise the requirements that an agent operating in such a world must satisfy. (Hayes-Roth, 91b) presents a design based on a blackboard architecture with a 'satisficing' execution cycle designed to meet these requirements. It is an architecture that can trade quality for speed of response under dynamic goals and resource and performance constraints.

The proposed agent architecture has three subsystems that operate in parallel: the perceptual, cognitive and action subsystems. Each subsystem may have an arbitrary amount of internal differentiation. A communications interface routes data among the various input-output buffers of the different subsystems. For example, environmental stimuli enter the perceptual subsystem which selectively filters the input under attentional parameters determined by the cognitive subsystem; the communications interface routes this information either directly to action subsystems (fast reactions) or via the cognitive subsystem where the stimuli compete with other perceptions or internally generated events for processing time. The cognitive subsystem performs all knowledge-based reasoning and can send effector commands to the action subsystem. All input-output buffers of the subsystems have limited capacity with best-first retrieval (defined along dimensions such as relevance, importance, recency and urgency) and worst-first overflow. All reasoning within the cognitive subsystem utilises a global memory that represents all information using a conceptual graph formalism. Such a global memory could include important knowledge such as differing reasoning strategies for differing circumstances and perhaps meta-knowledge about the differences between the various strategies.

The cognitive subsystem contains three major data structures: a cognitive buffer holding information about events produced by reasoning actions; an agenda that holds executable reasoning operations; and a control plan representing the agent's intended course of action as determined by reasoning operations. The cognitive subsystem satisficing cycle consists of three major processes: an agenda manager identifies operations and rates the importance and urgency of the operations with regard to the current control plan; a scheduler determines which identified operations to execute and when and records each successive operation as the next operation to be performed; and the executor executes the next operation, making all necessary changes to global memory. The satisficing cycle, instead of triggering all possible actions and executing the best one, uses the current control plan to focus decision-making and trigger only a few important actions. The control plan, in addition, focuses the attention of perception subsystems and imposes a temporal order on plan execution (Hayes-Roth, 91a, 93a & 93b).

The GUARDIAN system (Hayes-Roth, 90 & 91b), an implementation of the above design, monitors intensive care patients and has been shown to successfully respond to urgent patient needs and also perform longer-term diagnostic reasoning.

There are a number of unresolved problems with Hayes-Roth's architecture. Such a system may encounter difficulties dealing with highly urgent goals that require deliberation. As it stands, the architecture can either respond quickly via reflex-like connections from perception to action, or via a cognitive subsystem. Only in the latter case could deliberation occur and it is not clear how the architecture could guarantee reactivity in this case. Best-first retrieval from input-output buffers occurs across four dimensions (see above). It is not clear how these dimensions interact, or if they are indeed orthogonal, nor how they are computed. In addition, there is no good reason given why the input-output buffer capacity is limited. The executor drivers compute the urgency and importance of effector commands in order to prioritise them for execution. The goals that generate these commands are also analysed in terms of urgency and importance (and other factors). In other words, the computation of these dimensions is distributed throughout the architecture and inefficient repetition may occur. The prioritisation of effector commands by an execution subsystem may cause problems when attempting to synchronise many execution commands across many execution subsystems; for example, synchronising hand and arm movements. The control plan and satisficing cycle operate in a mysterious way: it would not be possible from any of the above referenced papers to implement the proposed design – there are too many gaps and unanswered questions and lack of an explicit design methodology. It may be the case that the architectural solutions Hayes-Roth proposes are domain dependent due to the application-driven nature of the work.

2.3.6 The Heuristic Control Virtual Machine

The Heuristic Control Virtual Machine (HCVM) (Fehling et al., 89) is a particular implementation of the Schemer model, which is a general computational model of problem-solving systems. The Schemer model is intended to model systems that are capable of autonomous, real-time performance via event-directed, reflective control of reasoning and action. Each Schemer implementation is a domain independent architectural framework for building problem-solving systems especially suited for resource-bounded problem solving. The architecture combines features of blackboard architectures with modularity, information hiding and abstract data machines normally found in object-oriented systems. A number of successful applications of Schemer have been developed, including automated performance management of avionics systems, monitoring and task management in a distributed information-processing system, and intelligent, real-time control of a material composition process.

There are four major components of the HCVM: a collection of modular procedures called handlers, an object-oriented global blackboard called the data space, a distributed triggering mechanism that determines which handlers should be executed (based on occurrences of certain patterns in the data space), and an agenda-based scheduler for determining which procedures are executed and when. The HCVM recognises two different kinds of control regimes: event-driven procedure invocation and direct procedure invocation. (These two control regimes are analogous to top level goal creation and subgoal creation). Invoked handlers may be placed on the agenda, assigned priorities and scheduled for execution.

Handlers contain procedural knowledge. There are two types of handlers – task handlers and knowledge handlers. Task handlers are explicitly called by other handlers whereas knowledge handlers are called whenever certain patterns of data are present. A simple priority scheme is used for ordering tasks (and their constituent handlers) based on the idea of different levels of importance. The same mechanism – an agenda-based scheduler – can accommodate meta-level layers of control; for example, scheduling capabilities may be enhanced by using 'control-knowledge' handlers that respond to patters in data space, in particular the state of the agenda and various history-list structures of recent processing. When executed these 'control-knowledge' handlers can directly manipulate the agenda. Therefore, control knowledge (or meta-level control) can be distributed within many 'control-knowledge' handlers whose object is the scheduling mechanism. The inclusion of the agenda and other structures as part of the data space allows for a limited kind of 'self-perception'. Interruption or non-interruption of current processing is achieved by the body of a handler explicitly specifying its interrupt conditions. When an interrupt condition is met control can pass to a higher level handler or the agenda-based scheduler. The degree of reactivity of the architecture is limited by the grain sizes of the parallel problem-solving components. Also, the body of a handler may be an entire, embedded HCVM system or systems. Consequently, a handler may be an arbitrarily complex abstract machine.

The HCVM architecture shares many striking similarities with the PRS system and NML1 (next sections), in particular the use of procedural knowledge and meta-level control procedures. One problem with HCVM as it stands is the need to explicitly specify interrupt conditions, which could cause problems of how to detect when the conditions are met (and the computational expense of such a process) and possible interactions between complex hierarchies of handlers each with their own interrupt conditions. However, it appears that Schemer-II (sketched in (Fehling et al., 89)) has begun to address this problem.

2.3.7 The Procedural Reasoning System

Georgeff's Procedural Reasoning System (PRS) (Georgeff & Ingrand, 89) aims to achieve a balance between acting and decision making. It has many features in common with Firby's RAP system but is designed to provide more powerful mechanisms for balancing decision making requirements against the constraints on time and information that are typical of complex domains.

PRS consists of a database containing the current beliefs or facts about the world, a set of current goals to be realised, a set of plans (called *knowledge areas*) describing processes for achieving goals, and an intention structure containing plans chosen for execution. An interpreter manipulates these components, selecting appropriate plans based on the system's beliefs or goals, and places these plans on the intention structure for execution.

PRS is an example of a BDI-architecture (Belief–Desire–Intention architecture) (Rao & Georgeff, 91 & 92), which is a formalisation of an autonomous agent based on a branching-time, possible-worlds model of behaviour. Although this formalisation restricts the flexibility of the agent by requiring the agent's intentions to be consistent, it is useful for characterising the major components of an autonomous agent. Informally, Beliefs are all the statements the agent believes are true in the world, Desires are all the states-of-affairs the agent would like to bring about in the world (including its own processing) and Intentions are commitments to action formed from the interaction of Beliefs and Desires. Any architecture that has explicit representations of Beliefs, Desires and Intentions can be viewed as an instantiation of a BDI-architecture.

Goals in PRS are specified using procedural logic. The use of a procedural logic is motivated by the belief that much commonsense knowledge about the world is in the form of procedures or sequences of actions for achieving goals. In (Georgeff, Lansky & Bessiere, 85) a formalism is provided for a procedural logic that serves as an executable program specification language suitable, it is claimed, for constructing complex systems. It extends existing logics, such as temporal, dynamic or interval-based, by being able to express sequencing, conditional selection, nondeterministic choice, iteration and hierarchical abstraction. This expressive power is used to automatically generate behaviours for achieving goals. In other words, the procedural logic offers the same advantages of some programming languages, such as Prolog, but is applicable in a dynamic rather than a static domain. The interpreter, in this scheme, can then be viewed as the operational semantics of the procedural language, i.e. takes the knowledge representation and uses it in a world. In brief, the procedural logic is concerned with processes, making use of such temporal operators as !p (make proposition p true), ?p (test p) and #p (preserve p), which can be combined with other conditions in an arbitrarily complex way. The operators of many standard planning systems, such as NOAH, SIPE etc., are restricted forms of process descriptions.

Each element of procedural knowledge, or knowledge area (KA), is a belief of the system about the utility of performing certain action sequences in particular contexts. The KA consists of an invocation condition and a body. The interpreter matches system beliefs to KA invocation conditions; if unification occurs the KA can be chosen for execution (i.e., become an intention) and its body run. At any one moment, the intention structure can contain a number of intentions, some of which may be suspended, conditionally suspended or deferred. Subgoals of executing intentions are posted as new goals of the system; otherwise, primitive actions are directly executed.

Goal descriptions are not restricted to specifying desired behaviours in an external world but can also apply to the internal behaviour of the system. These descriptions are called metalevel goal specifications and have corresponding metalevel KAs – that is, information about the manipulation of the beliefs, desires and intentions of the PRS itself. For example, metalevel KAs could include various methods for choosing amongst multiple applicable KAs, modifying and manipulating intentions, and computing the amount of reasoning that can be undertaken given the real-time constraints of the problem domain (i.e., decision-theoretic control and goal selection knowledge can be encoded within KAs). This is why PRS is probably the best example of a uniform hybrid architecture – no extra architectural element or mechanism is required to support deliberative processes as they are treated as another form of procedural knowledge.

The interpreter is relatively inflexible and stringently bounded in execution time, yet can be overridden whenever the system can bring more powerful decision-making knowledge to bear. Such knowledge is encoded in metalevel KAs and can be invoked when needed; however, these processes are themselves interruptible; therefore, reactivity is maintained.

The PRS architecture has been implemented within a physical robot concerned with navigation and emergency tasks (Georgeff & Lansky, 89) (i.e., must interrupt its ongoing intentions at certain junctures) and used for fault isolation and diagnosis in the Space Shuttle.

The features of PRS that Georgeff believes have contributed to its success are its partial planning strategy (not examined here), its reactivity, its use of procedural knowledge and its metalevel, or reflective, capabilities.

One problem with the PRS system as it stands is the metalevel KA invocation problem. The interpreter checks for invocation conditions once every cycle and problems can arise when invocation conditions for a particular KA are asserted during the cycle. The KA, therefore, may be invoked late or – if the conditions cease to hold before the next unification process – not at all.

One attraction of the PRS architecture is the simplicity of its interpreter and the addition of extra control knowledge in metalevel KAs. It also presents a flexible, architectural framework for the development of autonomous agency.

2.4 NML1

The NML1 architecture is based on Georgeff's Procedural Reasoning System, but extends PRS by allowing asynchronous goal generation, richer goal representation, a structured goal database, greater parallelism and a goal filtering mechanism. It is briefly summarised here; for a detailed description see (Beaudoin, 94).

The main components of NML1 are as follows: it has a perceptual module that records information about the environment; a perceptual control module that directs sensing operations based on control strategies and current activities; various goal *generactivators* (i.e., generate or re-activate generated goals) that respond to motivationally relevant information in a world model; an interrupt filter that can suppress generated goals; an interpreter that finds management procedures (similar to Georgeff's knowledge areas) that are applicable to goals, selecting some for execution; and an effector driver that performs physical actions within the environment. The NML1 architecture can be divided into three aspects: vigilation, management and meta-management.

Vigilation processes include perceptual processes, asynchronous goal generation and goal filtering. Perceptual processes take information from the environment and store this information in NML1's world model. Various asynchronous, parallel goal generactivators, expressing agent concerns (e.g., eat food), search the world model for firing conditions; if met, the goal generactivator constructs a goal representation and proposes it as a candidate for filtering. The filtering mechanism suppresses goals or allows them to 'surface' to the management system based on current filtering thresholds and the heuristic insistence measure of the candidate goal. The insistence of a goal is an inexpensively computed measure of the urgency and importance of a goal. The filter mechanism is designed to protect resource-bounded processing from unnecessary computation and interruption. For example, the goal to eat food would not surface had the agent recently eaten because the goal's insistence measure would be very low. The filter threshold level can be dynamically altered by management processes according to the current processing situation. For example, if the management system is processing many highly important goals the filter threshold level could be raised to prevent disturbance. If a goal surfaces it is managed.

Management processing refers to goal economy once a goal has surfaced (Sloman, 94a). A surfaced goal needs to be decided, i.e. whether it should be rejected, scheduled for execution, scheduled for deciding, postponed, conditionally suspended, expanded, partially expanded and so on. The interpreter performs these tasks by invoking applicable management procedures, which can contain various forms of procedural knowledge as per knowledge areas in PRS, and runs them in parallel. The interpreter itself does not engage in any of the management functions; rather, management procedures are considered as independent processes with dedicated processors. The goal database is a data structure that facilitates management of goals; it contains decisions made about goals and other relevant information. Once a management procedure for a goal has been scheduled for execution and is running it can dispatch effector commands to an effector module that performs actions in the agent's environment.

As in PRS there can be meta-level control, called *meta-management* processes in NML1. (Sloman, 93c) has provided a recursive definition of meta-management:

> A meta-management process is any goal-directed process whose goal refers to either a management or meta-management process. E.g., deciding whether to decide whether to adopt a goal, deciding when to decide whether to adopt a goal, deciding whether to expand a goal, deciding whether which management process to run now, deciding whether there's too much goal-switching going on or too many interrupts of management processes, etc.

Meta-management is concerned with assessing the priorities of various management functions, deciding when to 'think' about certain tasks, deciding whether the agent needs to change its current processing, determining whether all relevant issues have been considered, deciding whether management processes are taking too long to arrive at decisions, and detecting problematic processing states or episodes.

Beaudoin claims the strengths of the NML1 architecture are as follows: it benefits from the advantages of PRS, the interpreter and management processes are interruptible, planning and physical action can occur simultaneously or interleaved, asynchronous goal generators can respond to opportunities in the environment, and meta-management processes allow more 'reflective' control than, for example, behaviour-based systems.

2.4.1 An alternative view of the architecture

The figure on the opposite page is an abstract conception of the NML1 architecture. It is inspired by a cell analogy: management and meta-management processes are regarded as the cell 'nucleus' and the world model together with goal generating mechanisms are regarded as a surrounding 'cytoplasm'. Below the analogy is outlined, providing an alternative view of NML1.

The NML1 architecture is analogous to a cell body – both are autonomous systems, both communicate with an external world and both have internal processes. The nucleus of a cell contains genetic material that codes for amino acids that fold into proteins; in other words, the nucleus contains the genetic 'program' for cell functioning. The management system in NML1 controls agent functioning: it manages intentions and also forms them from candidate goals. Both the nucleus and the management system are the real 'brains' of the system. The nuclear membrane in a cell partitions the nucleus from the surrounding cell cytoplasm while allowing transfer of RNA material to and from the nucleus. The filter mechanism in NML1 protects the management system from candidate goals formed by generating mechanisms, but allows some goals to surface into the management system. The cytoplasm of a cell is the protoplasm excluding the nucleus, i.e. the surrounding chemical 'soup' and various organelles. The belief-desire substrate of NML1 is a 'soup' of beliefs (a globally accessible world model[8]), various desires (agent concerns) and a set of generating mechanisms that form candidate goals from beliefs and desires (generactivators in the current design). A generactivator, for example, could be compared to a cell organelle, such as a mitochondrion, that forms energy by utilising the chemical energy stores in the surrounding cytoplasm. Both the mitochondria and generactivators are autonomous entities, functioning in a substrate that they use to form output. The enclosing cell membrane can both ingest and egest material. NML1 can be thought as having an enclosing perception-action surface that can receive information from and perform actions in the environment. The following table summarises the analogy.

Animal cell	**NML1**
nucleus	management processes
nuclear membrane	filter mechanism
cytoplasm	belief-desire substrate
organelles	goal generating mechanisms
cell membrane	perception-action surface

[8]Management procedures can also be classified as beliefs as they express a declarative fact about the utility of certain actions in certain contexts.

3 Emergent States

What is an emergent state? – An emergent state could be defined as a global pattern of behaviour arising from a set of interacting architectural elements that have not been explicitly programmed to exhibit the global pattern. There are many analogous examples of such emergent phenomena to be found in science and philosophy. For example, the amoeba-like undulating of a school of fish or the flight pattern of a flock of birds, or the political superstructure that arises from the economic base in Marxist socio-philosophy. It is not our aim to investigate the concept and varieties of emergent phenomena; rather, it is sufficient to realise that, for our purposes, an emergent state is a global state of affairs resulting from the dynamic interaction of many architectural elements.

The following subsection contains a discussion on what may constitute an 'emotion' (although we do not assume that this concept refers to a single phenomenon) and how this may relate to emergent states in an architecture. The next subsection reviews the theoretically postulated *problematical* (from a control perspective) emergent states that could possibly occur in an NML1 (or PRS) –like architecture, followed by an outline of the design problems that need to be solved in order to detect such states. The concluding subsection discusses self-control and how self-referential mechanisms could be used to avoid, prevent or solve the postulated problematic emergent states and thereby achieve increased agent adaptivity.

3.1 Emotions and some theories

The literature on emotion is extensive and it would not be possible – or useful – to provide a full review here. Instead, a brief, general overview is given and Simon's and Sloman's design-based theory reviewed more fully.

However, it should be noted that there are difficulties contained within the literature on emotion. (Read & Sloman, 93) highlights the problem that different researchers (e.g., psychologists, biologists, cognitive scientists etc.) often use different vocabulary for the same phenomena or use the same vocabulary for differing phenomena. In other words, there is much terminological confusion in the literature. In addition, (Pfeifer, 92) makes the point that no consensus has been reached on what actually constitutes emotion, and that the study of this area is riven by differing 'schools of thought'.

(Sloman, 92a) distinguishes three types of emotion theories: semantics -based, phenomena -based, and design-based. Semantics-based theories analyse language use for meanings; for example, analysis of the word 'anger' into constituent elements. Phenomena-based theories assume that the object of theory is known and attempt to correlate other phenomena with it; for example, psychologists assume they know what emotions are and investigate contemporaneous phenomena, such as physiological causes and effects. In contrast, design-based theories take the stance of an engineer attempting to build a system that exhibits the phenomena to be explained, i.e. it here serves as a methodology for discovering generative mechanisms. The design-based approach is deemed essential, for without an understanding of architectures it will be difficult, if not impossible, to resolve terminological confusions and provide clearer distinctions and new vocabularies for the explanation of the phenomena we currently label as 'emotional'. Some examples of semantics-based and design-based theories are briefly presented below.

Examples of semantics-based theories are (Wierzbicka, 92) and (Dyer, 87).[9] Wierzbicka defines emotion concepts in terms of 'universal semantic primitives' such as 'good', 'bad', 'know', 'want', 'happen' and 'do'. For example, frustration can be defined as: I *want* to *do* something, I can't *do* this, therefore I feel *bad*; or amazement can be defined as: something *happened*, I didn't *know* before that this can *happen*, if I *knew* about this I would have thought this cannot *happen*, therefore I feel something. Wierzbicka claims that her prototypical scripts for each emotion state define necessary and sufficient conditions for the use of the applicable emotion concept. One consequence of the definitions that this theory provides is that such apparent synonyms as 'sad' and 'unhappy' can be distinguished as different conceptual structures.

(Dyer, 87) describes BORIS, a system that takes as input a number of narratives and reasons about what kind of emotional states the protagonists will be in. For example, BORIS may reason that a character is frightened if this character is alone and being followed by a shadow. To achieve such reasoning BORIS has domain knowledge about particular scenarios built-in. BORIS includes definitions of emotion concepts in terms of abstract goal situations based on the theoretical claim that all affects can be represented in terms of a positive or negative state of arousal coupled with intentional information. For instance, the affect 'happy' can be represented as a positive feeling of arousal coupled with the achievement of a goal. In other words, BORIS reasons about emotional labels based on an operational semantics of emotion concepts.

Frijda and Swagerman's ACRES system and Pfeifer's FEELER system are examples of design-based models of emotionality. (Frijda & Swagerman, 87) present the ACRES (Artificial Concern REalisation System) model built to explore Frijda's theory of emotion. It is a query database system with a number of concerns it wishes to fulfill. For example, ACRES wants correct input, does not like to be kept waiting and wishes to continue to operate. And, to confuse matters, ACRES' knowledge domain is about emotions.

Frijda regards emotions as 'manifestations' of a system operating in an uncertain environment that attempts to satisfy multiple concerns. Designing a system that realises concerns will reveal the functional need for emotional phenomena. (Frijda, 86) proposes a seven stage emotion process consisting (very briefly) of an analyser (codes the stimulus

[9] Although Dyer's BORIS system is really a mixture of semantics-based and design-based approaches – often research projects are 'fuzzy' and do not fit perfectly into the classificatory scheme.

event), a comparator (stimulus event is evaluated with regard to the agent's concerns), a diagnoser (determines what the agent can do about the current situation), an evaluator (determines the urgency, difficulty and importance of the stimulus event), an action proposer (including physiological change generators) and, finally, an actor that executes physical or cognitive actions. This seven stage process is reproduced in ACRES. Frijda and Swagerman claim that the program's behaviour exhibits much of human and animal emotional behaviour; for example, response interference (normal responses are changed in emotional situations), goal changing and social signalling of emotional states.

(Pfeifer, 92) provides a critical review of his FEELER (Framework for Evaluation of Events and Linkages into Emotional Responses) (1988) model of emotional processes. A situation is presented to the system which arrives at an appropriate emotional response via production rules. The left-hand side of the rule corresponds to the dimensions of the classification of various emotions (such as positive or negative evaluation) and the right-hand side generates the 'emotional' response. Pfeifer makes the point that FEELER is not a model of the generative mechanisms underlying emotional responses but rather a program that reasons about emotions. Meuller's (1990) DAYDREAMER program can similarly be criticised.

3.1.1 Simon and Sloman's design-based theory

H. A. Simon (1967), in a seminal paper, presents the beginnings of a design-based theory of emotionality. Simon outlines a number of important themes and ideas, primarily: that human behaviour is characterised by having multiple goals, that motivation can be thought of as that which controls attention at any given time, and that human behaviour, in many circumstances, can be interrupted. These ideas form the basis of his proposal that there is a close connection between the operation of an interrupt system and much of what is generally called emotional behaviour.

Any interrupting stimulus (e.g., the presence of a predator) [10] is supposed to interrupt ongoing goals in the central nervous system and substitute new goals to deal with the altered situation producing, amongst other things, emotional behaviour (e.g., the flight–fight–fright response). Other effects of an interrupting stimulus include arousal of the autonomic nervous system and production of feelings of emotion. Simon's computational theory is primarily concerned with goal interruption and consequent change of behaviour, rather than physiological and subjective affects. In other words, this theory views cognition (computational processes) as determining, i.e. any response program (to the interruption) controls the activation of the autonomic nervous system producing the feelings of emotion, and not *vice versa*.

[10] Environmental stimuli are not the only entities that can activate emotion (cause interruption of ongoing 'programs'): 'memory images' and 'drives' may also interrupt current goals.

Interruption is needed to serve the real-time needs of the organism. Simon distinguishes three types: needs arising from uncertain (or unpredicted) environmental events (as in the predator example), various physiological needs (internal stimuli; for example, hunger), and 'cognitive associations' (e.g., memory associations that cause anxiety).

Simon also flags the issue of adaptivity versus non-adaptivity of interruption. The emotional stimulus is to be considered as generally more *interrupting* than *disrupting* serving the real needs of the organism based upon some normative evaluation. However, in certain cases, the emotion-producing stimulus may be persistent and intense causing the invoked program response to be repeatedly interrupted resulting in maladaptive behaviour.

Issues of learning emotional behaviour are also raised. Learning can change the efficacy of certain stimuli to cause interruption, or allow new associations to cause previously uninterrupting stimuli to interrupt. In addition, the organism may acquire new or modified response patterns to interrupting stimuli. Learning will tend to reduce emotionality of response as situations become more familiar.

Simon summarises his own theory as follows:

> The theory explains how a basically serial information processor endowed with multiple needs behaves adaptively and survives in an environment that presents unpredictable threats and opportunities. The explanation is built on two central mechanisms: 1. A goal-terminating mechanism [goal executor] ... 2. An interruption mechanism, that is, emotion, allows the processor to respond to urgent needs in real time.

A. Sloman's 'attention filter penetration' theory (Sloman & Croucher, 81; Sloman, 85; Sloman, 87; Sloman, 92c) is an extension of Simon's theory. It introduces new, architectural detail implicit in Simon's paper. A variable threshold interrupt filter (or filters) is proposed that controls the ability of new motivators, thoughts or percepts to disturb or divert attention. The need for a filter mechanism relies on the assumption that ongoing activities use resources, both cognitive and physical, that are limited; therefore, these activities will need protecting from interruption to ensure adaptive behaviour. The variability of the threshold allows the level of protection to be dependent on context.

Sloman's theory focuses on interruption of current cognitive processing rather than Simon's focus on interruption of current *goals*. Emotional states involve a disposition to divert attention without necessarily disturbing ongoing goals or actions. Three dimensions along which motivational states can vary are distinguished: insistence, importance and urgency. The insistence of a motivator is its propensity to generate emotional states (interrupt 'attention'), and is a heuristic calculation of the importance and urgency of the motivator, i.e. only important or urgent (or both) motivators need to interrupt attention. It is heuristic as it needs to be computed inexpensively without diverting the very resources the filter mechanism is designed to protect.

Insistence, therefore, is a dispositional state – it has a tendency, or potential, to disturb and divert attention but need not actually surface through the filter or disturb ongoing processing. Sloman describes the strong potential for disturbance and diversion of attention as a characteristic of many of the states called emotional. These states can exist without diversion of attention; for example, jealousy can persist even though other activities occupy attention for some time. One consequence of this theory is that there are only differences of degree between emotional and non-emotional motivational states; and, in addition, states that have high insistence but do not have any motivation or positive or negative evaluation, such as a very catchy tune, have much in common with emotional states. The theory also implies that many different emotional states can co-exist.

In this theory 'emotions' are emergent from an architecture meeting the requirements for autonomy and real-time response in a dynamic and unpredictable environment. Unlike, say, Dyer's BORIS system that explicitly represents emotional states within data structures, the 'attention filter penetration' theory views emotions as a dynamic state of an architecture.[11]

3.2 Problematic emergent states

Architectures are complex abstract machines. The intricate clockwork mechanisms of nineteenth century automata are mere simplicities when compared to the complexity of, for example, a modern day operating system. It is not an easy task to posit the possible states such architectures can enter from an analysis of the design alone. Hence the need for implementation. Despite this, (Beaudoin, 94) has outlined some possible problematic control states that could occur in the NML1 architecture. Undoubtedly this list will not be exhaustive. Also, as the architecture is developed and new features added the variety of control states it can exhibit will increase.

I will make a small change in terminology: Beaudoin's problematic control states will be renamed problematic control *episodes* to emphasise that they are dynamic processes occurring over (perhaps lengthy) periods of time.[12] Here follows a brief summary of Beaudoin's postulated emergent states. Note that they all can be considered maladaptive with regard to the agent satisfying its set of concerns.

- **Oscillation between decisions.** This is the case where management processes are dispatched that are incompatible with previous management decisions. This

[11]However, this does not preclude the possible representation of this dynamic, global state for communicative and control purposes.

[12]However, 'state' and 'episode' can be considered synonymous for the purposes of this proposal.

is not to be confused with the problem within planning of a new action violating previous sub-goals. Here we are concerned with decision making, not the execution of those decisions. Beaudoin gives the example of a person who is faced with the choice of wearing a green or a red tie and selects a green tie only to change his mind and select a blue tie and so on repeatedly. Meta-management needs to detect such processing states and arbitrate. This category of processing states is also intended to cover such situations where a physical action commences only to be interrupted by another goal only to be interrupted by (perhaps) the original goal and so forth. Empirically, these kinds of states bring to mind the experimental paradigm of laboratory rats in a shuttle-box (Mackintosh, 83) that are presented with food at one end of a corridor. On each occasion food is taken the rat is shocked, which results in the rat, depending upon its desire to eat, oscillating about a point near the food.

- **Perturbance.** A perturbant episode is when a goal has been postponed or rejected but nevertheless keeps resurfacing and disrupts ongoing processing. This disruption is likely to interfere with the management of other, important goals. Strategies for dealing with this situation could involve satisfying the perturbing goal or suppressing its activation. A perturbant episode is the technical name for the type of information processing state that the 'attention filter penetration' theory posits as characteristic of many emotional states. The above definition of a perturbant episode needs to be extended to include entities other than goals that can cause disruption of ongoing processing, for example Simon's physiological drives and 'cognitive associations'.

- **High busyness.** Busyness can be viewed as a measure of management stress or load; it is defined as 'the extent to which there are important, urgent, and adopted unsatisfied (but potentially satisfiable) goals that require management and/or action relative to the amount of time which is required to manage or execute the goals' (Beaudoin, 94). High busyness implies greater likelihood of meta-management processing – for example, goals will be more likely to be postponed, or decisions made such as 'think about this later' and so on. Also, the filter threshold may need to be increased when busyness is high in order to suppress goal surfacing.

- **Maundering.** Maundering occurs when a goal is being managed without having decided, at a meta-management level, to manage it. The agent, therefore, could manage a goal that is not urgent or important relative to other pressing goals. Colloquially, this could be understood as daydreaming or even 'fiddling while Rome burns'.[13]

It might be argued that the above problematic control episodes will be unique to the NML1 architecture and therefore have no wider significance. A number of responses can be given to such objections. A perturbant episode, which is the control episode

[13] I have not described *digressions* here, which are similar to maundering; see (Beaudoin, 94) for details.

of most interest to us, has psychological validity (i.e., appears to be congruent with our own subjective experience of some emotional states) and, in addition, is deemed to be a common aspect of many architectures satsifying the requirements for autonomy in a dynamic environment.[14] High busyness is architecturally *independent*, i.e. is a result of the constraints of the specified niche-space; however, the actual details of high busyness will differ from design to design. Oscillating between decisions, high busyness and maundering also have psychological validity with correlates in our own subjective experience of mentality. In other words, such states or episodes are not unique to NML1 (or PRS) –like architectures. On the contrary, it appears that NML1 may share these features with people, albeit on an abstract level of conceptualisation.

The above control episodes have yet to be demonstrated to occur in an implementation of NML1. It may be that a prototype implementation will lack the architectural richness to exhibit some of the episodes. However, it should be possible to demonstrate a relatively simple form of perturbant episode using the idea of an *other-model*, which is a process that attempts to predict the behaviour of other agents in the environment. This idea will be extended in section 4.1 where details of the proposed implementation will be given.

It is expected that implementation work will reveal limitations in the above characterisations of problematic control episodes. For example, a perturbant episode may refer to many types of perturbance varying along dimensions such as the episode length, the control oscillation frequency (the rate at which ongoing processing is repeatedly interrupted), semantic content (what the perturbing goal is about), intensity (which may be interconnected with subjective evaluation of the episode as positive or negative, e.g. contrast a grieving mother with the athlete who cannot stop thinking about the Olympic gold medal he/she has just won), the source of the perturbance and also diagnosis of the episode by a self-referential mechanism and the control strategies elicited, if any.

The NML1 architecture will need to control its own global processing states otherwise it will not be well adapted to its niche. An architecture that cannot detect its own busyness and respond accordingly will be at a competitive disadvantage to architectures that can. Evolutionary pressure may have forced designs to become self-modifying and develop some type of learning that can form self-referential mechanisms to perform this meta-level control task. The detection of such control episodes and efficacious strategies to deal with them is undoubtedly an important component of intelligence.

3.3 The detection of emergent states

Before discussing the requirements for the detection of emergent states (and therefore problematic control episodes) it will be instructive to review some related work on self-

[14] Whether perturbant episodes are a *necessary* feature of resource-bounded architectures with multiple motives in dynamic environments is a research question.

reference from the literature.

(Smith, 86) distinguishes three camps within AI that are concerned with investigating the idea of a 'self': the *autoepistemic* tradition, mainly concerned with formal ideas of introspection and how systems reason about their own knowledge and beliefs; the *circumstantial* tradition, arising mainly from philosophy and linguistics and concerning itself with the self-relativity of thought and language; and, finally, the *control* camp that investigates meta-level reasoning and inference about control. (The research described here can be classified as belonging to the control camp.) Importantly, he states that no clear, single concept of what constitutes a 'self' has emerged capable of unifying these disparate efforts. Neither has an adequate explanation been provided for why self-reference is important for systems that fully participate in a world. In other words, why is it that humans have the concept *I*, why can they introspect, and why can they reflect about themselves and their relation to the world in a detached manner? Smith's partial answer is based on the assertion that the regularities underlying self-reference arise from necessary architectural requirements for any embedded system.

Smith states that there is a 'tension' between the need for effective situated activity and the detachment necessary for general-purpose reasoning. For example, an indexical representation, such as *there's-something-to-the-right* gains unambiguous meaning only in a certain context but, by being so, facilitates local inference and close connection to action; however, such representations are limited in their expressive power, possibilities for general-purpose use and communication with other architectural elements.

Self-referential mechanisms are proposed as a *design solution* to the need for local effectiveness together with general-purpose use of representations in detached reasoning. He distinguishes three self-referencing mechanisms – autonymy, introspection and reflection – that overcome different kinds of representational relativity (e.g., *there's-something-to-the-right* is relative to the system that asserts it). Each mechanism is based on distinct notions of the self: self as unity, self as complex system, and self as independent agent.

A system must represent its own relativity in order to causally connect abstract generalisations to indexical representations to action, i.e. represent its own finitude and particularness. For example, *there's-something-to-the-right* could not be communicated unless a notion of self was introduced and what is implicit in the representation is rendered explicit, i.e. *there's-something-to-the-right* becomes generalised to **RIGHT(something, I)**; this could then be communicated as 'there's something to the right of me!'. This example relies on the notion of self as unity, or *I*. Another reason given for the need for self-reference is that, typically, as long as some aspect of internal architecture isn't represented the system will behave in the 'standard' way with respect to that aspect. Explicit representation of implicit information paves the way for more flexible behaviour; without it, a system is locked into its primitive ways of doing things.

Smith provides three requirements for self-reference. The system must possess a theory of the self, i.e. what it thinks it is, how it should behave, what structure it has and so forth. This theory must be embedded within the system so that it can play a causal role in guiding the behaviour of the system. And finally a mechanism is needed for swapping between action in the world and detached reasoning about the self. NML1, at the present stage, meets only one of the requirements for self-reference. It can swap between action and detached reasoning due to the design of its interpreter and management procedures. The architectural mechanism to satisfy this requirement exists. However, NML1 does not have a theory of itself, nor does such a theory play a causal role in determining its behaviour.

A system is said to be autonymic if it is capable of using a name for itself in a causally connected way (Smith states that some email systems are capable of this). A system is introspective if it possesses causally connected self-referential mechanisms that render explicit some of the otherwise implicit internal structure of the system. And a reflective system is one that can represent the external world, including itself and its circumstances so that it renders explicit its own particularity, i.e. it can reason about itself as an agent within a world.

How do Smith's ideas relate to the research undertaken here? We are not interested in linking abstract representations to indexical representations; therefore, we can put the notion of autonymic systems to one side. However, an NML1 that can function in the world, detect its own global processing states, classify and diagnose such states, and attempt to do something about them, would be an introspective and reflective system. Therefore, we need to consider Smith's high-level requirements for self-reference. The need for a theory of the self is of importance: without knowledge of how the system works or is supposed to work it will not be possible for a self-referential mechanism to be casually efficacious. In particular, if we wish to design mechanisms that detect problematic control episodes we need a theory of the normative functioning of the self – for example, how is it possible to detect when things go wrong without knowing when things are going right? Also, to attempt to do something about such problematic episodes it will be necessary to have a theory of what junctures are suitable for intervention by a self-control mechanism.

We are now in a position to sketch out high level requirements (and not design solutions) for the detection of problematic control episodes.

- A theory of the self, including a theory of normative functioning and a theory of problematic control episodes. The question of how this knowledge is obtained will be placed to one side. Any such theory must be partial otherwise there will be the problem of infinite regress of a theory of the self that includes a mechanism with a theory of the self and so on. Smith provides three reasons why self-reference will be limited and perfect self-knowledge impossible: i. the complexity of the calculations involved, ii. theory-relativity (no theory can render everything explicit), and iii. circumstantial relativity[15] is beyond the causal reach of the agent; in other words, implicit aspects of the architecture will remain opaque to other mechanisms (consider abstract data structures). Sloman (1994b) has also made the point that complete self-knowledge would be of little use anyway: 'perception, inner or outer, needs to give affordances, not complete factual information', i.e. functional requirements determine what need be known (and any extra information would be superfluous).

- A mechanism, or set of mechanisms, that uses a theory of the self for detection of problematic control episodes, in a manner analogous to the role of knowledge in external perception.
- Such a mechanism must sample (sense) information from the self. This is a kind of 'internal perception' where the agent is the self-referential mechanism and the environment is the internal structure of the agent. Such information gathering or data sampling must not affect the functioning of the self to any great degree; or, more precisely, the operation of the self-referential mechanism must not invalidate its theory of the self. For example, visual perception in the physical world is passive: i.e., looking at a scene does not change the scene. Is it possible to mimic this process within an architecture and have mechanisms that can inspect ongoing processing without interfering with the object of inspection?
- A self-referential mechanism requires that critical junctures of the self be transparent. For example, to detect 'busyness' it would be necessary to sample the intention structure of the self; therefore, information about the intention structure, such as the number of pending, satisfiable goals, needs to be available to the mechanism. How this transparency is achieved is a design question.
- The opacity of some architectural elements (and the problem of circumstantial relativity) may need to be avoided or overcome by extra reasoning. This case is analogous to inferring the shape of an occluded object in visual perception.
- Facilities for recording sampled data over a period of time.
- 'Unobtrusiveness' – the mechanism must not interfere with the limited resources of other systems.
- A lexicon of problematic control episode concepts.[16]
- The detection mechanism must provide a mapping from the space of perceivable system states (which, for example, would be an n-dimensional space if n quantitative architectural junctures are sampled) to a space of labelled problematic control

[15] Circumstantial relativity states that a 'great deal of the full significance of a representational system will not, in general, be directly or explicitly represented by any of the representational structures of which it is composed. Instead it will be contributed by the attendant circumstances.'

[16] (Sloman, 94b) has pointed out that children can get into certain states yet be unable to classify them properly due to lack of appropriate concepts. The child who maintains that he has a 'fizzing' in his foot only to be told that he is suffering from pins-and-needles is an amusing example of such lack.

episodes.[17] Note that the space of perceivable system states will not be identical with the space of system states. What is 'looked' for will be determined by the theory of the self. There is likely to be information loss during such a mapping process.[18] Also, some system states may not map onto a label, or may map onto many labels, or be at the boundary of two or more labels. This raises the possibility of *anoetic* states – states in which the system is unsure of its own state.

The above is merely a sketch of high-level requirements, not a full requirements analysis for the self-detection of problematic control episodes. Such a task will be an object of research and will be aided by prototype implementation.

Many possible designs could satisfy the above requirements. Many design decisions need to be made, which makes this a fruitful area of research. The research will attempt to explore generic designs for self-detection; however, implementation solutions will be constrained by what we take the 'self' to be – in this case the NML1 design.

Two examples of possible design solutions readily come to mind. Many development environments provide tracing facilities that allow the control path of a program to be recorded and viewed. Perhaps similar techniques could be used in our agent. Or, alternatively, the self-referential mechanism could model the self and run a simulation in parallel. When the normative model and reality diverge the mechanism could then attempt to label the unexpected control state. Another design option concerns whether the self-referential mechanism could be a specialised meta-management process or an entirely separate mechanism.

3.4 Self-control: coping with emergent states

Self-control needs more than self-detection of global problematic control episodes – it requires the selection of *remedies* and their *application* within the system. The medical analogy is clear: self-control can be viewed as a three-stage process consisting of diagnosis (detecting what the problematic control episode is), selecting strategies for dealing with the episode (selection of medication and prognosis), and their application within the system (treatment).

The possible strategies for controlling such states will be numerous, the exact details differing from architecture to architecture. Enumerating and classifying such strategies will be a research aim. Below I will simply sketch some high level requirements for self-control intermixed with issues that will need to be addressed.

[17] Or, mappings from qualitative descriptions to labels, or parsing of system states into complete interpretations, or complex stages of analysis at successively higher levels of abstraction and so forth.

[18] Investigation of the process of learning finer-grained concepts for system states could be an area of research.

- A set of remedies/strategies will be needed that express knowledge about the utility of control actions for each problematic emergent state. There may be many strategies for each control state that apply in different contexts. For example, raising the attention filter in states of high busyness may be an efficacious strategy under some circumstances but would be disastrous if the agent needed to be continually aware of new, highly important motivationally relevant events.[19] The representation of such control actions is a design question.

- *Strategies could treat symptoms.* Consider the strategy of counting sheep in order to fall asleep at night. If we assume that the person cannot sleep because he is continually ruminating over things that have occured during the day, the counting sheep strategy could be viewed as a meta-level decision to concentrate deliberative resources on a task that requires full attention. This is an example of treating symptoms, i.e. the cause of the insomnia remains but its ability to attract processing resources is diminished due to the conscious selection of an 'attention-expensive' task. Or, alternatively, consider the grieving Mother who turns to drink in order to dull her painful thoughts and emotional responses (whether this is an effective strategy is by the by).

- *Strategies could treat causes.* For example, a control strategy could reduce the intensity of a perturbance, or it could turn off the generating mechanism. However, it appears that the latter option is not always immediately possible. Grief is an example where, for whatever reason, the source of the perturbance cannot be removed. There could be various reasons for this, such as a meta-level control mechanism not having the necessary causal powers due to opacity, circumstantial relativity, or due to the architectural design; or the source of the perturbance may be widely distributed across an information store; or, in the case of the loss of a loved one, many of the grieving person's concepts, goals, opinions, beliefs are saturated with and grounded upon the other's existence (and in this case the complexity of the adjustment would require time). Another possibility is that, for example, the *other-model* of the departed person could have its rationale for generating candidate goals removed, and thereby have its *direct* route to management closed; however, such a model may have effects on many other parts of the architecture that will, in turn, generate candidate goals. Therefore, the model could still have *indirect* causal powers. Analysis of these issues will also be an object of research.

- *Strategies could cure.* It may be possible to eradicate the production of certain problematic emergent states entirely. Take the example of *oscillating between decisions.* It may be that an architecture can learn to recognise this state of affairs quickly and easily and build new architectural links that 'hard-wire' this control knowledge in the form of a cognitive reflex. In other words, the self-control mechanism effects

[19]Consider a child-minder who is extremely busy caring for three babies in a room. She knows that another child is in the kitchen, which she can view through an open door. The three babies are so demanding that she concentrates all her attention on them, so much so that she fails to notice the child in the kitchen reach for a hot iron.

a design change instead of invoking control strategies. Or it may be possible to construct deeper design changes that will prevent the production of the state in the first place. Curative strategies will involve dynamic architectures, design modification and learning. This problem can itself be analysed into many sub-problems.[20] Psychological evidence exists for some kind of self-modelling, reflection and self-repair, e.g. (Kuhl & Kraska, 89).

- *Strategies could utilise.* Another type of self-control strategy is to use the problematic control state for normative ends. (Sloman, 94b) has pointed out the example of a teacher who discovers that he can control his class the angrier he becomes – in this case, the relaxation of self-control helps achieve the teacher's goals. In other words, a new use is found for the state and strategies employed to allow or cause that state to happen when the use is needed. However, investigation of this type of control strategy would require, I believe, a richer typology of problematic control episodes.

- A decision mechanism will be required that can select strategies for application.

- The selected control strategy will need to be applied to the self. The varieties of *modes of application* may cover the full range of causal properties in software systems. As stated before, there will be constraints on the application of control strategies that will depend on the causal powers of the self-control mechanism and the structure of the surrounding architectural environment. For example, (Smith, 86) states that, given psychological self-knowledge, it is hard for humans to become the person they can so easily represent themselves to be. This is the a more general case of the possibility of diagnosing and representing what needs to be done (the goal of the control strategy) contrasted with the difficulty of actually achieving the control aim within the architecture. Let us briefly consider some varieties of causal connection that self-control mechanisms could employ. At a first approximation I can distinguish three types of causal connection that could be used by a self-control mechanism on the self: *structural, procedural* and *direct.* Structural changes would include such strategies as altering the causal connections between mechanisms (for example, preventing an other-model from generating goals that are candidates for filtering), replacement of mechanisms (for example, replacing an interpreter with one more suitable for the current processing situation), or addition of mechanisms (for example, the construction of a cognitive reflex). Procedural changes would include strategies that alter the inner details of mechanisms (i.e., intra-mechanism change as opposed to structural/inter-mechanism change), such as altering procedural constants, using parameterised procedures to alter functioning in useful ways, biasing the outputs or inputs from procedures, and so on. Direct changes would include strategies that employ extra processing to alter global functionality (for

[20] For example, the design problem would need to be identified by assigning blame, a modification to the design selected, repair work effected on the design followed by some kind of verification process to check whether the new design performs better than before, i.e. a real improvement in the design has been achieved.

example, the cessation of maundering by the firing of a meta-management process that decides whether a goal, which is currently being deliberated to the exclusion of others, should actually be adopted for deliberation). I am not at all happy with the above typology of casual connections from self-control mechanisms to the self. More thought, requirements and design work will help clarify these concepts and ideas.

The requirements and design options for self-controlling mechanisms are only touched upon here, which is why this topic is suitable for sustained research. I hope that consideration of the effects of self-control may shed some light on the processes underlying the decay of emotions.[21] Emotional decay may be a particular case of how self-referential mechanisms may affect affect.

3.5 Pathologies of self-control

Without some knowledge of the designs and design options for self-control mechanisms it is difficult to determine why and how errors in self-control may occur, and how these errors may relate to human pathologies. It may be that we are trying to run before we can walk. - It will be necessary to review extant psychological theories of pathologies of self-control before this subject can be broached in earnest. However, even the sketchy requirements for self-control outlined above suggest possibilities for errors in self-control. Such errors in the architecture could be caused by physical damage, chemical imbalance, developmental aberrations, design errors and so on. Consider the following.

- Cognitive reflexes embodying learnt control knowledge could have been built incorrectly, or built unnecessarily.
- The theory of the self may be incorrect. Depending on the complexity and structure of this knowledge there will be more or fewer different types of theory errors. For example, the theory of normative behaviour may be faulty causing self-control mechanisms to produce maladaptive behaviour; or errors in the theory of what techniques are causally efficacious for self-control may cause the pursuit of improved adaptation to result in its opposite. There will be many subtleties.
- The theory of the self may be used incorrectly. The self-control mechanism may use the theory in an erroneous way.
- Errors may occur during internal perception. The self-control mechanism may believe the self is in a state that it is not. This could cause inappropriate control strategies to be fired.

[21] Frijda, in a recent paper (in Cognition and Affect, vol. 8, no. 4, 1994), highlights this as an area of emotion research so far ignored.

- There may be a lack of architectural transparency, i.e. incomplete self-knowledge. A little knowledge can be dangerous – if there is insufficient information it will not be possible for self-control mechanisms to decide which control strategies to elicit. This type of error will be dependent on design solutions for architectural transparency.

- Control strategies may be poorly formed, or dysfunctional. This is analogous to the case of faulty plans in planning.

The use of such words as 'incorrect', 'errors', 'poor', 'dysfunctional', 'faulty' all comprise value judgements on the behaviour of mechanisms within the architecture. However, how are such evaluations to be made without criteria for normative behaviour? The agent's theory of the self serves, for self-control purposes, as such a normative basis. However, how can we state that the self-theory itself is erroneous: on what basis do we make this claim? The evaluation will have to be made from an observer's standpoint and a knowledge of the requirements of the agent's niche. This is a particular case of the more general problem within psychology of making value judgements on human behaviour. Ignorance of this theoretical problem can produce a tendency to medicalise difference and interpret it as aberrance. If this research proceeds in a theoretical direction and approaches problems in human psychology it will be important to be aware of such issues.

However, the above possibilities for errors in self-control only scratch the surface. Much more work needs to be done, both in terms of reviewing extant literature and analysis.

4 Research Aims

In the introduction the goals of the research were split into modest and ambitious research aims. In this section the aims will be considered in greater detail followed by criteria for evaluation of the work. First, however, a scanario for implementation will be presented that will be used to investigate perturbance and self-control.

4.1 An emotional scenario

A framework is needed for the implementation of prototype NML1s. Initial implementation (modest aims of the research) is to investigate the generation and effect of perturbant states. Therefore, we need a plausible scenario from the real world that can be modelled.

> A child-minder has spent all week looking after Jack and Jill. She now knows that both children like to be fed every two hours. She leaves the room for a few minutes to answer the phone, but when she returns she discovers that Jack has disappeared. She searches the room but there is no sign of him. It is time that the children are fed, and Jill begins to wail. Shortly, Jill's mother returns to collect her. The child-minder tries to converse with Jill's mother, but her thoughts are continually disrupted by the thought that Jack has disappeared and will be wanting his food. The child-minder is in a terrible state.

The child-minder has ongoing processing (talking to Jill's mother) continually disrupted by the negatively valenced thought that Jack has disappeared. This is an example of a perturbant state. How can we model a similar process using the chosen agent architecture?

NML1's current environment is the nursery domain (Sloman, 94c; Sloman & Humphreys, 92). The nursery domain consists of a number of rooms separated by walls and connected via doors. Babies wander around the nursery and need to be cared for. For example, they will need feeding at regular intervals, or given medical care if they catch certain illnesses, or can get injured by bumping into objects, falling down ditches or by having fights with other babies. It is the task of the child-minder, NML1, to ensure that the babies are well cared for. For example, a concern of NML1 might be to grab babies when they are near ditches and remove them to a safe distance. The nursery domain provides a rapidly changing and unpredictable environment that places variable stress on the agent architecture. NML1 will need to arbitrate between multiple motives while maintaining reactivity to external events.

The above emotional scenario can be modelled in the nursery domain. The idea of an *other-model* will be introduced as a stop-gap solution to the difficulty of ensuring early prototypes of NML1 enter perturbant states. An other-model simulates some aspect of the behaviour of another agent in the environment. For our implementation, we will devise an other-model[22] that can predict when a baby will need recharging at a recharge point. After a period of time a particular baby will be removed from the nursery. The other-model for this baby will remain and generate a motivator to recharge the missing baby. This is likely to be (depending on the current processing state) a highly insistent motivator that surfaces. A management process will attempt to satisfy the motivator by finding the baby and taking it to a recharge point. This management process will fail because the baby is missing. However, the other-model will continue to generate the motivator, which may now be rejected after deliberation or scheduled for execution again. In this way, current, ongoing processing will be repeatedly interrupted and the agent will be in a type of perturbant state.

A prototype implementation conforming to the above description would be a good base point to begin a more detailed implementation of perturbance, problematic emergent states and their detection and control. The scenario would need to be extended to include self-referential mechanisms and strategies for dealing with certain global processing states.

An important point needs to be made here about a prototype implementation. In section 2.? the problem of decision-theoretic control and goal selection was highlighted. For a working agent architecture these issues need to be addressed, but it is not the aim of the research to investigate, in any great detail, such problems. However, solutions to the problems of decision-theoretic control will have relevance to emergent states and perturbance. Here we need to re-state the theoretical basis of the research – that we need to employ a 'divide and conquer' strategy and factor out some aspects of autonomous agency in order to investigate and build broad architectures. The difficulties of decision-theoretic control will be one such factor we will underemphasise. In prototype implementations we will use heuristic techniques and domain-dependent solutions. The principles that Pryor (1994) provides in Chapter 8 ('Deciding What to Do Next') of her thesis, or similar, could be modified and used in prototype NML1s.

At the time of writing a nursery domain has been implemented and a child-minder with vigilational functionality. The agent has simulated coarse-grained parallelism including a perceptual process, motivator generactivators and an effector (claw) that can manipulate objects in the environment. The child-minder can notice when a baby is too close to a ditch, update its world model and dispatch commands to the claw to pick up the baby and remove it to a safe distance. There is no plan formation, motivator filtering, interpreter or management processes. The next step in any prototype implementation will be to provide these features.

[22] Ferguson's TouringMachines have models of other entities in the environment and it may be that some techniques from this work can be modified and used with NML1.

4.2 Aims and evaluation

As stated earlier, the research wishes to investigate three related areas – the 'attention filter penetration' theory of emotionality, the extension of the design and implementation of an agent architecture, and the self-control of emergent states in autonomous agents. Perturbance is considered as one form of emergent phenomenon that can be the object of self-control. A design-based methodology can unify these research concerns within the design and implementation of a broad agent architecture.

Many models of emotional processes exist in the literature (e.g., see section 3.1); however, Sloman's 'attention filter penetration' theory has yet to be tested by computer modelling. Also, the final NML1 design has yet to be implemented. There has been much work on 'self-control' in psychology; however, this term often refers to different phenomena (e.g., abstinence now for greater reward later, as opposed to controlling fear or embarassment in social situations). It is my belief that extant theories of self-control will benefit from a design-based analysis grounded in the requirements for autonomous agency. Therefore, this work can build upon and potentially contribute to three areas of research. In particular, a systematic investigation of the requirements for the detection and control of global processing states and an exploration of the possible design options would be a new contribution to the field.

It will be useful, in addition to distinguishing modest and ambitious research aims, to separate theoretical and practical aims. The implementation goal of the research will be relatively modest; depending on time and success the implementation will be extended to incorporate more of the developed theory. This is a flexible research strategy but with defined aims. Consequently, the research should aim to build a prototype NML1 that can operate autonomously in a (relatively simple) nursery domain with the ability to switch attention between goals successfully and exhibit some form of perturbant state. This is the 'base-level' modelling aim of the research.

Below, two research scenarios are provided, one modest and the other ambitious. It is my expectation that the actual achievements of this research will fall somewhere between these two scenarios. A decision will need to be made at some future point whether to consider perturbant states as one form of emergent state that can be the object of self-control, and therefore direct the research towards an investigation of self-control, or, alternatively, direct the research towards an investigation of emotionality and regard self-control only to the extent it may affect emotional states. It is a question of emphasis. The decision is postponed because the fruitfullness of each approach will only become apparent after preliminary work.

Modest goals. Build a working prototype of the NML1 design, add a plausible emotional scenario to the agent's domain in order to explore the concept of perturbant states, and relate this implementation to the 'attention filter penetration' theory of emotionality. In particular, consider how extensions to the design could increase the variety and type of perturbant states and how this may relate to people. It is expected that the design of NMl1 will be extended and refined during this process. The requirements for and design of self-control mechanisms in autonomous agents will be presented. Achievement of these modest aims will build on previous work in the Attention and Affect project, the design of broad agent architectures, the understanding of emotionality and mechanisms for self-control.

Ambitious goals. Achievement of the modest goals, plus: implementation of self-control mechanisms within the agent that can detect a class of problematic, global emergent states, perform a diagnostic analysis based on a theory of the self, and attempt to improve functioning by either preventing the emergent state, coping with it, or eradicating the state entirely. For example, the agent may detect that it is in a state of high busyness that is adversely affecting its normal functioning; therefore, the agent could employ the strategy of raising its filter threshold level to prevent further motivators from surfacing. The implementation of self-control mechanisms in the agent will be related to the space of possible designs for self-controlling mechanisms, including an analysis of the various design options that meet the requirements. It is my hope that this aim will be furthered by making design-space exploration less analytic and more automatic by using methodological techniques from system design (see thesis timetable). How self-control is learnt and mechanisms formed in a dynamic architecture will be considered, and such considerations related to extant psychological theories of self-control (for example, (Kuhl, 92; Kuhl & Kraska, 89; Heckhausen & Kuhl, 85; Rachlin, 94)). Such considerations may have some bearing on cognitive development theories. The varieties of self-control of emotional episodes will be investigated, including how emotions may decay over time. Achievement of the ambitious research goals will, in addition to the contributions made by the modest goals, add to the understanding of the mechanisms for self-control in people, especially with regard to emotional episodes, and help clarify the terminology and concepts within psychological theories from a design-stance.

How is this research to be evaluated? From my own perspective I will determine whether the research is successful by comparing what I achieve with what my original aims were. Achievement of the modest aims would constitute modest success and so forth. However, how is the research to be evaluated from the outside?

The work, as stated above, aims to contribute to three current areas of research. By how much the research provides new and useful contributions to these fields is one, and the most important, dimension of evaluation.

However, there is the difficulty of evaluating computer models in AI. (Chapman, 90) provides a weak justification of implementations: he considers the process of implementation as a special *form of understanding*, which renders explicit what is implicit in any design. An implementation, therefore, serves as an exemplary illustration of a theory. It can demonstrate that a design meets its requirements, i.e. that the theory can explain the chosen phenomena. The design-based approach, in addition to the above, considers implementations not as end-points but as moments in the process of exploration of design-space. Knowledge gained through implementation will feed back into the investigation of requirements and designs. In this conception, therefore, implementation is a necessary part of the methodological whole. Designing computer models of cognitive processes without implementation would be like an engineer providing successive designs for bridges without ever testing those designs against reality. However, the design-based approach shares with software engineering the same difficulties of 'proving' that an implementation meets its requirements. Design validation is a difficult problem. If i. the implementation has provided a deeper understanding of the requirements, design and various design options, and ii. exhibits the required behaviour (i.e., successful operation within the chosen domain based on some commonsense yardstick) or *fails* but provides knowledge concerning *why* the design failed then, I believe, the computer model can be judged worthwhile.

The requirements and design work of the research are partly motivated by perceived inadequacies of previous agent designs. For example, the problematic emergent episodes postulated as occurring in NML1 are non-adaptive – hence the need for additional control mechanisms. Proposed solutions to these problems will contribute to agent design in general. Discovering design options will contribute to an understanding of a region of design-space. And it is likely that new problems and new solutions will be uncovered in the course of research.

In addition to the above, there are at least three ways to determine whether the research will provide an original contribution to knowledge.

- New predictions from theory. For example, it may be that an understanding of self-control mechanisms from a design-stance will predict certain behavioural phenomena (in humans or animals) not so far noticed, or currently unexplained, or so far confused with other, similar phenomena.

- New possibilities. New ways of designing agents or mechanisms may be provided that extend our understanding of the design possibilities and the respective behavioural correlates. (This is exploration of design-space).

- New (superior) ways of talking about known phenomena. For example, an extant psychological theory of self-control could be refined, clarified or modified using the design-based approach. Reducing terminological confusion and providing clearer concepts would be a step forward.

Success on any of these fronts would constitute an original contribution.

4.3 Thesis Plan

All timetables are provisional and subject to revision. The following table sketches out how I hope the research will progress. Two prototype stages are distinguished. The prototype I NML1 will have rudimentary goal filtering, a simple interpreter and a small collection of management processes that can be used to perform straightforward behaviours in the domain, such as recharging babies and preventing them from falling into ditches. Also, there will be simple heuristics for goal selection. The prototype II NML1 will have the functionality of the first prototype, plus a more sophisticated interpreter and management processes, and an other-model that predicts aspects of baby behaviour. Prototype II should generate a type of perturbant state as per the emotional scenario. In addition, a number of ideas for papers and technical documents have been included in the timetable. Hopefully, these will act as landmarks and serve to summarise the results from different aspects of the research. The titles of these papers are subject to revision also.

From talking to other research students it seems that their thesis proposal timetables have been shown to be, with the benefit of hindsight, overly optimistic given the time available. Therefore, I have allowed six months for the writing of the thesis, which, if things do not go according to plan, can be 'eaten into'. It is my aim to achieve the modest aims by the end of 1995 leaving nine months to attempt the ambitious aims and thesis write up.

	Month	Research
Year 2		
	October November December	Continue coding prototype NML1; write a paper entitled *Exploring Design-Space with QOC*. By December should have prototype I completed.
	January February March	Literature review of concepts of self-control in AI, cognitive science and psychology; write a document entitled *The Varieties of Self-control.*
	April May June	Continue coding NML1: extend functionality, and implement emotional scenario. By the end of June should have prototype II completed.
	July August September	Requirements analysis for self-control of global states. Discuss various design options. Summarise this work in a paper entitled *A Proposal for a Design-Based Study of Self-control.*
Year 3		
	October November December	Review relevant literature on emotions. Investigate the varieties of perturbant state in theory and practice; write a document: *The Varieties of Perturbant State.*
Modest goals achieved.		
	January February March	Add self-control mechanisms to NML1 implementation. Provide a design-based critique of a theory of self control from the field of psychology.
	April May June July August September	Tie up odds and ends and begin writing thesis. Write papers if appropriate. ... ... Aim to complete thesis. Submit. Proposed thesis title: *A Design-Based Study of Self-Control*

5 Appendix: The Design-Based Process

The design-based approach, drawing its inspiration from software engineering[23] and conceptual analysis in philosophy[24], is a methodology that enables exploration of an abstract space of possible designs for functioning agents (Sloman, 93a). Individual humans and animals are loci in design-space, along with other systems yet to exist. Such a space is difficult to picture – it is hyperdimensional, structured, highly complex, with both qualitative and quantitative dimensions. We do not possess sufficiently powerful formalisms or techniques to automatically search design-space. Evolution took billions of years and as many design decisions to arrive at the functioning systems we see today – squeezing such a natural process, in all its richness and complexity, into a few years research remains a task beyond us. The design-based approach helps us conceptualise and simplify the task to arrive at preliminary answers. When used as a methodology to adumbrate the virtual machines implemented in existing organisms (including humans) it relies on the assumed congruity between design decisions 'taken' by evolution under environmental and competitive pressures and the design decisions taken by the designer when moving from requirements to design.

The design-based approach is a *unifying* methodology. It views the disparate research methods and aims within Artificial Intelligence (and other computational sciences, such as cognitive science) as different but related ways of arriving at designs. For example, a bottom-up approach may be useful for deriving designs for insect-like robots (see section 2.2) whereas a top-down, requirements analysis for autonomous agency may be useful for arriving at a broad, high level functional decomposition of important elements of an architecture. Both research paradigms are applications of the design-based approach.

5.1 A summary

The following is a summary of an idealised conception of the design-based process.

The process can begin by specifying initial, general requirements normally written in natural language.[25] There is a highly complex mapping from requirements to designs (Sloman & Cognition and Affect Group, 94). The set of requirements can be viewed as a *topography* of a region of niche-space, which defines a *type* of designs over a region of design-space. Each design is a member of the type, satisfying the specified requirements. To illustrate: biologists classify environmental niches and the organisms that inhabit them; an environmental niche can be thought of as specifying a set of general requirements

[23] c.f. (Bell et al., 92).

[24] See (Sloman, 78).

[25] Formal specification techniques could be used but would conflict with the rapid prototyping approach used in this project.

that all designs within that niche must meet in order to survive – for example, a marine habitat could specify the type marine, and include such organisms as sharks, octopii, crabs and so on. Supersets of the requirements, i.e. collections of different requirements, specify a *typology* over design-space; and supersets at different levels of abstraction can form a hierarchical *taxonomy* of types of designs as per the classificatory schemes of plant and animal life.

The actual requirements will be determined by the aims of the research, i.e. what is to be investigated, and depending on their specificity will encompass fewer or more designs meeting the requirements. Requirements can be *functional* – specifying what the system should do; or *data* requirements – inputs and outputs to the system, data storage within the system and so on; or various *constraints* – such as performance constraints, target language limitations and hardware limitations; or *guidelines* – for example, heuristics for resolving requirements with more than one implementation strategy[26]. After requirements analysis[27] we have a requirements list $Niche = \{r_1, r_2, ..., r_n\}$ where r_i is one requirement statement[28]. The requirements are decomposed[29] and a high level, architectural design is derived. That is, from $Niche$ we get the list $Design = \{d_1, d_2, ..., d_n\}$ where d_i is one design decision. Further functional decomposition and analysis, and recursive application of requirements analysis and design for sub-components of the architecture (e.g. subprograms, subroutines, procedures, functions, processes etc.) leads to lower level design decisions being added to $Design$ that approach the limit of implementation details. In other words, the structure of $Design$ increases in complexity forming a hierarchical structure. There are three categories of design decision that follow.[30]

- Decisions linked directly to initial requirements.
- Decisions linked indirectly to requirements via higher level design decisions (higher level decisions become 'requirements' for lower levels).
- Decisions that are arbitrary, but placed within $Design$ at a functional level (decisions made where previous decisions or requirements do not prescribe a unique

[26] Or encompass such considerations as evolvability, empirical observation etc. See (Read, 93) for an exposition of Systemic Design, a particular form of the design-based approach, that emphasises these factors.

[27] Requirements analysis may proceed bottom-up or middle-out as well as top-down. Empirical data can constrain design-space by making requirements more specific and so on.

[28] What constitutes one requirement statement is problematical. Only a general guideline can be given: the statement should be simple, concise and clear, and not contain conjunctives. E.g. the requirement that the agent's environment is complex and dynamic could be split into the following requirement statements: r_1 = 'environment is rich enough to provide multiple sources of motivation', r_2 = 'environment is constantly changing with time' etc.

[29] Normally by using the functional decomposition method; but other approaches could be used, such as object-oriented design or the data flow method.

[30] These three cases map onto the six categories of design decision listed by the Attention and Affect project (Sloman et al., 92). Case 1 maps onto decisions made wrt requirements, empirical data and to test a theory; case 2 is equivalent to decisions made wrt previous decisions; and case 3 maps onto arbitrary decisions or decisions made due to hardware or software limitations.

decision to be made.) These decisions are unlinked, i.e. not derived from requirements, but placed in the functional hierarchy (note also that they do not invalidate or contradict any requirement statement). $Arbitrary$ is defined as the list of all arbitrary design decisions in $Design$.

What is the nature of design decision linkage? – As requirements and design decisions are not specified formally linking a requirement to a design decision, or a design decision to a lower level decision, rests solely on analysis. The quality of such analysis will determine the validity of the design with regard to the initial requirements. A $d_i \in Design$ is *strongly* linked to a $d_j \in Design$ or $r_k \in Niche$ if d_j or r_k *uniquely* determines d_i, i.e. no other design decision would meet the requirements.

Prototype implementation can begin once a sufficiently specific design is derived from the requirements. Implementation will reveal holes in the design and suggest new requirements. This is a feedback process with progressive refinement of requirements, design and implementation.

5.2 Exploring design-space

Producing an implementation is not sufficient for interesting science – we need to explore neighbouring designs to explore trade-offs between design options and alternative implementation strategies (Sloman, 93b; Beaudoin, 93). Requirements do not uniquely determine a design, instead they specify a region of niche-space that maps onto a region of design-space in a non-trivial way.

There are at least two ways of exploring design-space: i. exploring different designs within a general type specified by $Niche$ by considering design options for $d_i \in Arbitrary$, and combinations of $d_i \in Arbitrary$ (exploring possible systems designed to function in the same region of niche-space); or ii. exploring niche-space by considering changes to $r_i \in Niche$, or adding to or deleting from $Niche$ (exploring systems designed to function in different regions of niche-space). Case i. is of most interest in this research ($Niche$ remains fixed).

A significant (with regard to the aims of the exploration) $d_i \in Arbitrary$ is chosen.[31] For example, it may be that a number of methods exist for updating a memory store within an architecture; but the actual method used was chosen on an arbitrary basis. Choosing different design options for the update method may have 'knock-on' effects for

[31] In other words, design options emanate from arbitrary design decisions as we assume that all other $d_i \in Design$ are linked and therefore uniquely specified by $Niche$ or some previous $d_j \in Design$. But as linkage relies on argument strong linkage will be rare. Therefore, this is a simplifying assumption. Actual exploration of design-space for an existing design may well discover design options for some $d_i \in Design$.

other elements of the architecture. To discover what these effects may be as many possible replacements for d_i that still satisfy requirements (be they initial requirements or previous design decisions) are considered and a replacement set for the chosen design decision, $Replacements = \{d_1, d_2, ..., d_n\}$, is formed that is unlikely to be complete, i.e. it is likely that some replacement options are overlooked. For each significant d_j in $Replacements$ (e.g., qualitatively different update methods), $d_i \in Arbitrary$ is replaced with d_j and the consequences for every $d_k \in Design$ linked to d_i is calculated. In other words, the consequences of the replacement decision d_j for the rest of the design are considered, i.e. all parts of the architecture that are 'dependent' on the memory update method. This process of 'checking the consequences' can either be achieved by re-implementation of dependent parts of the design (time consuming), by intuitive analysis or perhaps mathematical analysis. When all consequences have propagated through the design a neighbouring design is formed with different properties but still satisfying $Niche$. The propagation of consequences is complicated by the fact that replacement design decisions may be structural or quantitative, and that quantitative changes may cause qualitative changes in other parts of the design. Exploration of the designs satisfying $Niche$ need not proceed by considering the options for one design decision at a time – combinations of alternative design decisions can be considered. Discovering which combination of alternatives is significant will be problem dependent. The exploration is open-ended, and the size of the region of design-space satisfying $Niche$ will be extremely large, even when the problem requirements are well specified. The extent and scope of the exploration will also be problem dependent. [32]

This method requires the designer to explicitly categorize each $d_i \in Design$ within one of the three types of design decision. This categorization facilitates reasoning about the consequences of design options. Without such categorization design-space exploration becomes a 'hit and miss' affair. Choosing replacement sets for design decisions at higher functional levels (e.g. architectural layer or sub-component layer) will explore fewer and more significant design options than exhaustively finding $Replacements$ for every $d_i \in Arbitrary$.

One of the uses of prototyping in traditional software engineering is to aid the requirements analysis and design stages of the software development process. Knowledge gained through prototype implementation can clarify requirements and criticise design decisions. For example, initial design decisions can be shown to be inefficient, unwieldy, incompatible and incomplete etc. when attempts are made at implementation.

Research resource limitations (time, number of people, money) entail pragmatism. Hence, prototype development is used in conjunction with the design-based approach. But problems can arise: for example, prototypes are usually developed in a 'dirty' manner – incomplete requirements analysis, implicit design decisions, unstructured code and so on.

[32] If $Arbitrary = \{\}$ then $Niche$ uniquely specifies a design $Design$; however if $Arbitrary \neq \{\}$ then the size of the region of design-space satisfying $Niche$ will be some function of $Arbitrary$ and the respective replacement sets $Replacements_i$ for every $d_i \in Arbitrary$.

Implicit design decisions made during prototype development can make the subsequent exploration of design-space incomplete and the consequences of changing an aspect of the design may not be fully explored. Retrospective theoretical analysis can tend to rely on the researcher's memory if design decisions are not explicitly logged. Due to the use of natural language specification the analysis of design-space remains informal. – Such difficulties will not be solved overnight. Much research and theoretical effort will be needed to tackle these problems. But any research that attempts to investigate an area of design-space will need to be aware of the pitfalls and attempt to avoid them.

5.2.1 A note on QOC

(MacLean et al., 91)[33] present a semiformal notation called QOC (Questions, Options, and Criteria) that is intended to represent the design space surrounding an artifact. I have yet to study their paper in any depth, but from a quick perusal it appears that such a technique could be very useful for design-space analysis in the design-based approach. It may help to avoid or solve some of the pitfalls highlighted above. Below is the abstract quoted in full:

Abstract

> Design Space Analysis is an approach to representing design rationale. It uses a semiformal notation, called QOC (Questions, Options, and Criteria), to represent the design space around an artifact. The main constituents of QOC are *Questions* identifying key design issues, *Options* providing possible answers to Questions, and *Criteria* for assessing and comparing the Options. Design Space Analysis also takes account of justifications for the design (and possible alternative designs) that reflect considerations such as consistency, models and analogies, and relevant data and theory. A Design Space Analysis does not produce a record of the design process but is instead a coproduct of design and has to be constructed alongside the artifact itself. Our work is motivated by the notion that a Design Space Analysis will repay the investment in its creation by supporting both the original process of design and subsequent work on redesign and reuse by (a) providing an explicit representation to aid reasoning about the design and about the consequences of changes to it and (b) serving as a vehicle for communication, for example, among members of the design team or among the original designers and later maintainers of the system. Our work to date emphasises the nature of the QOC representation over processes for creating it, so these claims serve as goals rather than objectives we have achieved.

I intend to write a document that examines whether QOC will be of use in the design-based approach.

[33]Thanks to Russell Beale for pointing out this work to me.

6 References

Agre, P. E., & Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Aritificial Intelligence,* (pp. 268-272). Seattle: AAAI.

Bates, J. (1994). The role of emotion in believable agents. *Communications of the ACM, 37*(7), 122-125.

Bates, J., Loyall, A. B., & Reilly, W. S. (1991). Broad agents. *SIGART Bulletin, 2*(4), 38-40.

Beaudoin, L.P. (1993). Analysis of the design space surrounding a model of motivator processing. Notes for a talk at the *Workshop on Architectures Underlying Motivation and Emotion - WAUME 93*, University of Birmingham.

Beaudoin, L. (1994). *Goal Processing in Autonomous Agents.* PhD Thesis, School of Computer Science, University of Birmingham.

Beaudoin, L. P., & Sloman, A. (1991). *A Proposal for a Study of Motive Processing* Unpublished manuscript. School of Computer Science, University of Birmingham.

Beaudoin, L. P., & Sloman, A. (1993). A study of motive processing and attention. In *Proceedings of AISB93*, A. Sloman, D. Hogg, G. Humphreys, A. Ramsay and D. Partridge (Eds), pp 229–238, Oxford: IOS Press.

Beer, R. D., Chiel, H. J., & Sterling, L. S. (1990). A biological perspective on autonomous agent design. In P. Maes (eds.), *Designing autonomous agents: Theory and practice from biology to engineering and back* (pp. 169-186). Amsterdam: Elsevier Science Publishers.

Bell et al. (1992). *Software Engineering: a programming approach.* Second Edition, Prentice Hall.

Boddy, M., & Dean, T. (1989). Solving time-dependent planning problems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* Vol. 2.

Bratman, M. E., Israel, D. J., & Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. In *Computational Intelligence,* Vol. 4.

Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems, 6,* 3-15.

Brooks, R. A. (1991a). Intelligence without representation. *Artificial Intelligence, 47,* 139-60.

Brooks, R. A. (1991b). Integrated systems based on behaviours. *SIGART Bulletin, 2*(4), 46-50.

Chapman, D. (1989). Penguins can make cake. *AI Magazine, 10*(4), 45-50.

Chapman, D. (1990). *Vision, instruction and action* (Technical report No. 1204). Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

Cooper, D. E. (1990). *Existentialism, a reconstruction.* Oxford, Basil Blackwell.

Doyle, J. (1989). Reasoning, representation, and rational self-government. In Z. W. Ras (Eds.), *Methodologies for intelligent systems* (pp. 367-380). New York: Elsevier Science Publishing.

Dyer, M. G. (1987). Emotions and their computations: Three computer models. *Cognition and Emotion 1*(3), 323-347.

Etzioni, O. (1993). *Intelligence without robots (a reply to Brooks).* To appear in AI Magazine.

Fehling, M. R., Altman, A. M. & Michael Wilber, B. (1989). The Heuristic control virtual machine: An implementation of the schemer computational model of relflective, real-time problem-solving. *Blackboard Architectures and Applications.* Academic Press, Inc.

Ferguson, I. A. (1992). *TouringMachines: An architecture for dynamic, rational, mobile agents.* PhD thesis, University of Cambridge Computer Laboratory, University of Cambridge (technical report No. 273).

Firby, R. J. (1987). An investigation into reactive planning in complex domains. *Proceedings of the Sixth National Conference on Aritificial Intelligence,* (pp. 202-206). Seattle: AAAI.

Frijda, N. H. (1986). *The emotions.* Cambridge: Cambridge University Press.

Frijda, N. H., & Swagerman, J. (1987). Can computers feel? Theory and design of an emotional system. *Cognition and Emotion, 1,* 235-257.

Georgeff, M. P., & Ingrand, F. F. (1989). Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* 2 (pp. 972-978). Detroit, MI: IJCAI.

Georgeff, M. P., & Lansky, A. L. (1989). Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Aritificial Intelligence,* (pp. 677-682). Seattle: AAAI.

Georgeff, M. P., Lansky, A. L. & Bessiere, P. (1985). A procedural logic. *International Joint Conference on Aritifical Intelligence.*

Hanks, S., & Firby, R. J. (1990). Issues and architectures for planning and execution. In *Proceedings of a Workshop on Innovative Approaches to Planning, Scheduling and Control,* San Diego, CA: DARPA.

Hayes-Roth, B. (1990). Architectural foundations for real-time performance in intelligent agents. *Journal of Real-Time Systems, 2,* 99-125.

Hayes-Roth, B. (1991a). Evaluation of integrated agent architectures. *SIGART Bulletin, 2*(4), 82-84.

Hayes-Roth, B. (1991b). An integrated architecture for intelligent agents. *SIGART*

Bulletin, 2(4), 79-81.

Hayes-Roth, B. (1993a). *An architecture for adaptive intelligent systems* (KSL Report No. 93-19). Knowledge Systems Laboratory, Department of Computer Science, Standford University.

Hayes-Roth, B. (1993b). Intelligent control. *Artificial Intelligence, 59,* 213-220.

Heckhausen, H., & Kuhl, J. (1985). From wishes to action: The dead ends and short cuts on the long way to action. In M. Frese & J. Sabini (Eds.), *Goal directed behaviour: The concept of action in psychology* (pp. 134-159). London: Lawrence Erlbaum Associates.

Horvitz, E. J. (1987). Reasoning about beliefs and actions under computational resource constraints. In L. N. Kanal, T. S. Levitt, & J. F. Lemmer (Ed.), *Proceedings of the Third AAAI Workshop on Uncertainty in Artificial Intelligence,* 8. Seattle, WA: Elsevier Science Publishers.

Horvitz, E. J., Gregory, F. C., & Heckerman, D. E. (1989). Reflection and action under scarce resources: Theoretical principles and empirical study. in *Proceedings of the Eleventh International Joint Conference on Artificial Inteliigence,* Vol. 2.

Kuhl, J. (1992). A theory of self-regulation: Action versus state orientation, self- discrimination, and some applications. *Applied Psychology: An International Review, 41*(2), 97-129.

Kuhl, J., & Kraska, K. (1989). Self-regulation and metamotivation: Computational mechanisms, development, and assessment. In R. Kanfer, P. L. Ackerman, & R. Cudek (Eds.), *Abilities, motivation, and methodology: The Minnesota Symposium on Individual Differences* (pp. 343-374). Hillsdale, NJ: Lawrence Erlbaum Associates Inc.

Lesser, V. R, Pavlin, J., & Durfee, E. H. (1989). Approximate processing in real-time problem solving. In *Blackboard Architectures and Applications.* Academic Press Inc., reprinted with permission from *AI Magazine*, vol. 9, no. 1 (Spring 1988).

Loyall, A. B., & Bates, J. (1991). *Hap – A reactive, adaptive architecture for agents* (Technical report No. CMU-CS-91-147), School of Computer Science, Carnegie Mellon University.

Mackintosh, N. (1983). *Conditioning and Associative Learning.* Oxford: Oxford University Press.

MacLean, A., Young, R. M., Bellotti, V. M. E., & Moran, T. P. (1991). Questions, options, and criteria: Elements of design space analysis. *Human-Computer Interaction,* vol. 6 (pp. 201-250).

Maes, P. (1990). Guest editorial: Designing autonomous agents. In P. Maes (eds.), *Designing autonomous agents: Theory and practice from biology to engineering and back* (pp. 1-2). Amsterdam: Elsevier Science Publishers.

Meuller, R. E. (1990). *Daydreaming in humans and machines.* New Jersey: Ablex Publishing Corporation.

Norman, T. J., & Long, D. (1994). A proposal for goal creation in motivated agents. Paper available via world-wide web – http://www.cs.ucl.ac.uk/people/ai/ – from Department of Computer Science, University College London.

Pfeifer, R. (1992). The new age of the fungus eater: Comments on artificial intelligence and emotion. Extended and revised version of an invited talk at the *AISB-91* conference in Leeds, U.K.

Pryor, L. (1994). *Opportunities and planning in an unpredictable world.* PhD thesis, Northwestern University

Pryor, L., & Collins, G. (1992). Reference features as guides to reasoning about opportunities. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society.* Bloomington, Lawrence Erlbaum Associates.

Pryor, L., & Collins, G. (1993). *Cassandra: Planning for contingencies* (Technical report No. 41). The institute for learning sciences, Northwestern University.

Rachlin, H. (1994). *Self-control: Beyond commitment.* Draft of Behavioural and Brain Sciences target article currently being circulated for open peer commentary, available via ftp from princeton.edu.

Rao, A. S., & Georgeff, M. P. (1991). *Modeling rational agents within a BDI-Architecture* (Technical Note No. 14). Australian Artificial Intelligence Institute, 1 Grattan Street, Carlton, Victoria 3053, Australia.

Rao, A. S., & Georgeff, M. P. (1992). An abstract architecture for rational agents. In *Proceedings of the Third International Conference on Knowledge Representation and Reasoning.* Boston: KR92.

Read, T. (1993). *Systemic Design: A Methodology for Investigating Emotional Phenomena.* (In the process of being updated). Unpublished document. Cognitive Science Research Centre, School of Computer Science, University of Birmingham.

Read, T., & Sloman, A. (1993). The terminological pitfalls of studying emotion. Paper presented at the *Workshop on Architectures Underlying Motivation and Emotion - WAUME 93*, Birmingham.

Reilly, W. S. (1993). *Emotions as Part of a Broad Agent Architecture*, Talk given at WAUME93, Birmingham.

Simon, H. A. (1967). Motivational and emotional controls of cognition. *Psychological Review, 74*, 29-39.

Sloman, A. (1978). *The Computer Revolution in Philosophy – Philosophy, Science and Models of Mind.* Harvester Studies in Cognitive Science, Harvester Press.

Sloman, A. (1985). Real time multiple-motive expert systems. In M. Merry (Ed.), *Expert systems 85* (pp. 1-13). Cambridge: Cambridge University Press.

Sloman, A. (1987). Motives, mechanisms and emotions. *Cognition and Emotion, 1,* 217-234.

Sloman, A. (1992a). Prolegomena to a theory of communication and affect. In A. Ortony, J. Slack, & O. Stock (Eds.), *Communication from an Artificial Intelligence Perspective: Theoretical and Applied Issues (Proceedings of the 1990 NATO Advanced Research Workshop on 'Computational theories of communication and their applications: Problems and prospects')* (pp. 229-260). Heidelberg, Germany: Springer.

Sloman, A. (1992b). The mind as a control system. In *Proceedings of the 1992 Royal Institute of Philosophy: 'Philosophy and Cognitive Science'*, Editors: C. Hookway & D. Peterson. Cambridge: Cambridge University Press.

Sloman, A. (1992c). *Towards an Information Processing Theory of Emotions.* Notes for Cognitive Science seminar, October 1992.

Sloman, A. (1993a). Prospects for AI as the general science of intelligence. In *Prospects for Artificial Intelligence – Proceedings of AISB93,* Oxford IOS Press, Editors: A. Sloman, D. Hogg, G. Humphreys & D. Partridge.

Sloman, A. (1993b). Why we need to study architectures. Notes for *Workshop on Architectures Underlying Motivation and Emotion – WAUME93,* Birmingham.

Sloman, A. (1993c). (Personal communication.)

Sloman, A. (1994a). *Management and Meta-management.* Technical note for an Emotions Meeting based on L. Beaudoin's draft thesis, Department of Computer Science, University of Birmingham.

Sloman, A. (1994b). (Personal communcation.)

Sloman, A. (1994c). *Autonomous motive-driven agents with broad architectures*, SERC research grant proposal, School of Computer Science, University of Birmingham.

Sloman, A., Beaudoin, L., & Wright, I. P. (1994). Computational modelling of motive-management processes. Cognitive Science Research Report CSRP-94-14, School of Computer Science and Cognitive Science Research Centre, University of Birmingham.

Sloman, A & the Cognition and Affect Group (1994). Explorations in design space. In *ECAI 94 11th European Conference on Artificial Intelligence*, Editor: A. Cohn. John Wiley & Sons, Ltd.

Sloman, A., & Croucher, M. (1981). Why robots will have emotions. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence,* (pp. 197-202). Vancouver.

Sloman, A., & Humphreys, G. (1992). *The Attention and Affect Project.* Appendix to JCI proposal.

Sloman, A., Shing, E., Read, T., & Beaudoin, L. (1992). Notes from an Emotions Meeting, Department of Computer Science, University of Birmingham.

Smith, B. C. (1986). Varieties of self-reference. In J. Y. Halpern (Ed.), *Proceedings* *of the First Conference on Theoretical Aspects of Reasoning About Knowledge.* Morgan Kaufman.

Wierzbicka, A. (1992). Defining emotion concepts. *Cognitive Science 16,* 539-581.

Wright, I. P. (1993). A summary of the attention and affect project. Technical report available via ftp from ftp.cs.bham.ac.uk:/pub/dist/papers/cog-affect.