A Methodology for Provably Stable Behaviour-based Intelligent Control

Christopher J. Harper, Alan F.T. Winfield
Intelligent Autonomous Systems Laboratory, University of the West of England, Bristol, UK
{cjharper@gxn.co.uk, alan.winfield@uwe.ac.uk}

Abstract

This paper presents a design methodology for a class of behaviour-based control systems, arguing its potential for application to safety critical systems. We propose a formal basis for subsumption architecture design based on two extensions to Lyapunov stability theory, the Second Order Stability Theorems, and interpretations of system safety and liveness in Lyapunov stability terms. The subsumption of the new theorems by the classical stability theorems serves as a model of dynamical subsumption, forming the basis of the design methodology. Behaviour-based control also offers the potential for using simple computational mechanisms, which will simplify the safety assurance process.

Keywords:

Intelligent control; behaviour-based control; Subsumption Architecture; Lyapunov stability; safety critical systems.

1. Introduction

Intelligent control techniques have not to date found widespread application in systems that demand high levels of dependability, such as safety critical systems. There are two reasons for this. Firstly, intelligent control systems, based upon recent techniques from artificial intelligence or mobile robotics are difficult to validate formally using conventional approaches to system analysis and test. Secondly, researchers in artificial intelligence are often content to demonstrate intelligent control in laboratory environments only; their interest is generally in pushing the boundaries of AI, rather than transferring techniques into mainstream engineering practice. In this paper we present an investigation into methods that will allow intelligent control systems to be used in the same manner as conventional control systems are used today.

1.1 Background

A system is safety critical if its failure or malfunction can cause damage, injury, or loss of life [18]. Typical examples include aircraft control systems, nuclear power station controls, or industrial process controls. Safety critical systems are required to undergo a safety certification process prior to their introduction into service. Within this process, the safety properties of the system are rigorously validated, to establish that it will be safe within a known range of environmental conditions. The process involves extensive systems analysis, design verification, and testing.

There are a number of important system properties that are relevant to its safety assurance. Some are properties of the reliability of its hardware, and the availability of redundant backup subsystems to take over if the primary subsystem were to fail. Others relate to the integrity of the system, its freedom from design error. It is the latter category that we address in this work. Two particular properties that we seek to establish in a safety critical system are liveness and safety. Liveness is the property that a system will achieve its functional purpose (satisfy its functional specification) under all external conditions and assumptions, i.e. that the system will perform its intended task. Safety, as its name suggests, is the property that a system will avoid all conditions known to be hazardous, i.e. that it will not do anything unintended. It is crucial to the understanding of safety critical systems to realise that the two properties are not complements of one another, and that the demonstration of one property does not support any claim as to the other. In most conventional systems, liveness is usually demonstrated through verification and validation methods such as static analysis or testing [37]. Safety is generally demonstrated by various failure analyses (functional and non-functional) and then by more specialised testing processes such as environmental testing, reliability testing, and fault injection testing.

It is beyond the scope of this paper to investigate the definition of 'intelligence', but it is possible to provide some characterisation of "intelligent control", and the general nature of systems of this category. One of the most succinct definitions proposed to date is due to Newell and Simon [27], who state that 'general intelligent action' as any behaviour occuring in a real situation that is appropriate to the ends of the system and adaptive to the demands of the environment, within some limits of speed and complexity.

This definition highlights several aspects of intelligent control:

(1) That intelligent control is defined not by the nature of the internal mechanisms of a system, but by the external interactions between that system and its environment.

(2) That intelligent systems adapt to their environment *and not vice versa*. Many conventional engineering systems are made reliable by designing their environments rather than the systems themselves.

(3) That intelligent control systems can adapt to their environment and still be able to achieve their goals. It should not be assumed that environments are in any way benign, or that they should remain so.

Intelligent control techniques have hitherto been largely ignored, or even actively rejected as unsuitable for safety critical systems in many industry sectors. Yet intelligent control systems offer the promise of increased levels of autonomy, adaptability or resilience to unexpected input conditions; attributes that will be increasingly required of future complex systems. The aim of this research, therefore, is to consider how intelligent control techniques may be designed so as to be able to predict or prove their operational properties in advance of their use in service. A design methodology answering these questions would form the basis for the use of intelligent control in safety critical applications.

The research described in this paper continues a strong thread of work, in the IAS laboratory, in provably stable intelligent control. The work of Jin developed stability proofs for a class of adaptive neural controller [19]. Randall extended this research to systems with closed kinematic chains [31]. In the present work we turn our attention to a class of intelligent controllers based upon the subsumption architecture, also known as behaviour-based control, first proposed by Brooks [7] and revised by Connell [9]. Behaviour-based control has become the approach of choice in mobile robotics research, yet no methods exist for assuring the dependability of such systems. Indeed, in the preamble to the reprinted version of his original paper [8], Brooks states:

> "*To my mind, there has to date been no decent mathematical analysis of the ideas presented in this paper [...] That is not to say that there should not be or can not be such an analysis. But I think it will require some very deep insights and can not rely on surface level equation generation.*"

1.2 Related Work

Current research into design methodologies for behaviour-based systems tends to fall into two general approaches, informal guidelines, and formal techniques based either on discrete logic techniques, or continuous dynamical systems techniques (hybrid control theory [32] is a mix of discrete and continuous techniques). Within the latter two approaches, research efforts can also be split into two sub-categories, those which apply formal techniques to characterise what is essentially an ad-hoc system design, and those which attempt to develop a formal theory of behaviour-based systems and then apply that theory to real problems.

The informal approach is typified by the work of Pfeiffer and Scheier [30], who have compiled an extensive set of guidelines on the construction of intelligent agents. These guidelines are useful in the construction of autonomous agents, but offer no formal or quantitative mathematical basis for their design principles. Arkin [4] takes the approach a step further towards formalisation, by providing mathematical descriptions of motor schema, a representation also used in this paper. However, the work does not go to the point of explicit proof of these specifications, and so we regard it as being an informal (albeit a well-structured) technique.

Other researchers have developed formal methodologies using discrete logic approaches. Many of these techniques have been based on the definition of a formal language, specifying the function and properties of systems as predicates acting on propositions of agent states and actions. The LIKA language of Lesperance and Levesque [21] is typical of this approach. Of particular interest is the work of Amir and Maynard-Reid [2], who develop a full theory of subsumption architecture in discrete logic, including a model of the principle of subsumption. This is believed to be the only effort other than the one described in this paper that attempts to construct a full theory for behaviour-based systems, rather than providing formal descriptions of an ad-hoc design pattern.

Propositional languages have been successful as a design technique for behaviour-based agents, but suffer from symbol-grounding issues [15] when they are implemented in physical hardware. This tends to result in additional complexity within such systems, where an additional layer converting symbols to physical actions, and sensory patterns to symbols, must be added in order to make hardware work as intended. Techniques based on dynamical systems theory have the advantage that their representations of autonomous control problems can be expressed directly in physical hardware, allowing their implementation in simpler computational mechanisms are required for symbolic techniques. For example, Beer [5] makes use of simple dynamical neural networks to provide gait control of hexapod robots and, as will be discussed later, the techniques defined in this paper can be implemented using programmable array logic. One can exploit the reduced complexity of these schemes to gain advantages in other aspects of system design. In the methodology discussed in this paper, the advantage gained is an improvement in the effectiveness of failure analysis, a key analysis requirement for certification of safety critical systems.

Much research has made use of dynamical systems techniques to define design methodologies for behaviour-based (neuro-)control technologies, often using formal techniques to prove the goal-achieving properties of systems. Particular examples include the work of Steinhage, Bergener et al. [35][36], Sequeira and Robeiro [33], and Low et al. [22]. However, as mentioned before, these approaches tend to involve the formal systematization of a system architecture, for which no theoretical basis has been defined. The research in this paper is believed to be the only dynamical systems methodology, which attempts to develop a mathematical theory of an architecture in a fully deductive manner from the ground up, rather than assuming a technological solution first and then developing a formalisation for it.

1.3 The Research

This paper describes a rational methodology for the design of systems with “Colony-style” architecture. This is a variant of Subsumption Architecture, which itself is one of many styles of behaviour-based control. The methodology [16] is intended to provide a formal basis for their design, by deduction of the transfer functions from the general dynamical model of a system and a specification of its required goals, and by deduction of the architectural organisation of a system from general principles for the minimisation of mutual interference between behaviour modules within its subsumption architecture.

This paper proceeds as follows. In Section 2 we review the class of subsumption architecture which is the focus of the work of this paper. In Section 3 we introduce two new ‘second order’ Lyapunov stability theorems, followed by a rational design procedure we refer to as Direct Lyapunov Design in Section 4. In Section 5 we develop a formal model of subsumption architecture known as Dynamical Subsumption. Having established the theoretical basis we propose in Section 6 a design methodology for intelligent control systems using the subsumption architecture. We then demonstrate the design methodology in Section 7 with an aircraft flight control simulation experiment. Finally, in Section 8 we present our conclusions and discuss some further issues of the application of the methodology to safety critical systems, especially in relation to its engineering realisation.

2. SUBSUMPTION ARCHITECTURE

The work presented in this paper is a design methodology for Colony-style Subsumption Architecture (CSA), a form of behaviour-based system architecture. This section presents the characteristic features of behaviour-based systems generally, of CSA in particular, and the reasons for adopting this specific architecture type as the preferred technology for the safety critical systems design methodology.

2.1 Behaviour-based Systems

Behaviour-based systems first emerged in the mid-1980s, most notably from work by Brooks [7] but also from Arkin [3], Payton [28] and others. Many variations of behaviour-based system architecture have been developed over the intervening years and Figure 1 illustrates three different types, (A) Subsumption Architecture [9], (B) Action Selection Networks [24] and (C) Dynamical Neural Networks [6]. While these three architectures appear to be quite diverse, in general they share some characteristic principles and features.

(1) Typically, behaviour-based systems contain components that generate global system behaviour directly, rather than some specialised process that only partially contributes to the behaviour. In most behaviour-based systems, the architecture tends to be organised into concurrent arrays of behaviour-producing subsystems, referred to in Subsumption Architecture as ‘behaviour modules’, a term which shall henceforth be used generically to apply to the functional components of any behaviour-based system.

(2) Behaviour-based systems exploit a simplifying principle, originally identified by Brooks [8], that cognition does not need an explicit process within an intelligent control system. Traditional AI theory was based on the idea that intelligent systems required specific processes that implemented cognitive reasoning. Brooks realised that for some types of behaviour (in particular those involving sensorimotor coordination), a system that simply connected sensory perceptions directly to motor actions could still be seen as exhibiting cognition, if the pattern of actions was suitably arranged. This allows a fundamental simplification to be made in the internal design of behaviour modules over their earlier function-oriented counterparts in traditional AI systems. This simplification permits non-symbolic computational mechanisms to be used in the implementation of behaviour-based systems, which has major implications for their complexity, and hence their dependability assurance (as will be discussed later).

(3) The third feature they have in common is their use of behaviour arbitration. Since the behaviour modules of a system are all active concurrently, and each generates a distinct action command for the system, the specific action to be applied to the system must be selected out of the set of commands generated at any single moment. Many different schemes have been proposed, from fixed-priority arbitration (using suppression switches) in Subsumption Architecture to behaviour vector-summation [29] or excitatory/inhibitory signals [6][24].

We have reviewed a number of behaviour-based system approaches for their suitability to application in safety critical problems. The set included CSA, Brooks’ original Subsumption Architecture [7], Arkin’s AuRA [3], Maes’ Action Selection Networks [24], Rosenblatt & Payton architecture [28] [29], the architecture of Dorigo and Colombetti’s BAT methodology [11], and Beer’s Dynamical Neural Nets [6]. The main criteria for assessing these different technologies were safety assurance issues, in particular the ease with which one could perform the causal analyses that are typical of most safety assurance exercises. Preferred approaches are those in which the architecture elements are causally reversible, allowing the specific causes of erroneous motor actions to be identified; those in which the architecture is modular, restricting the propagation of fault conditions through a system, and those which permit the exploitation of the simplifying principle mentioned in Point (2) above.

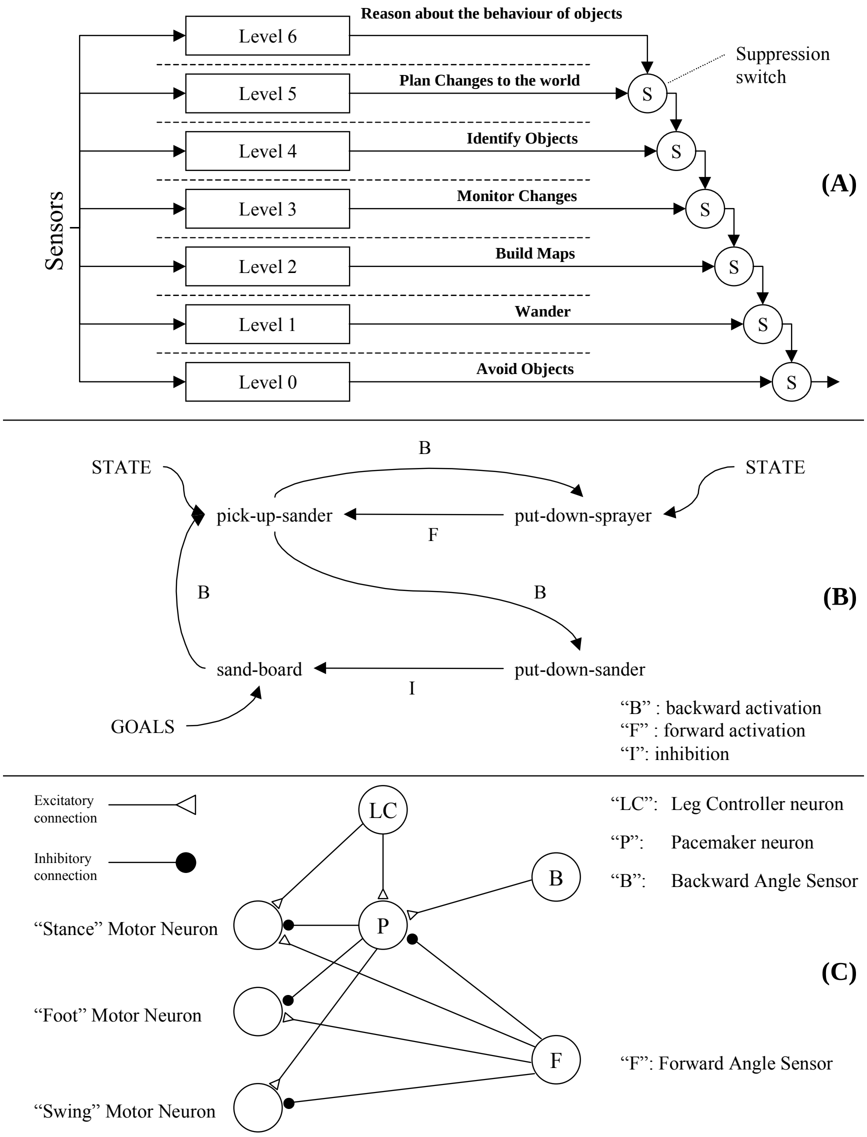


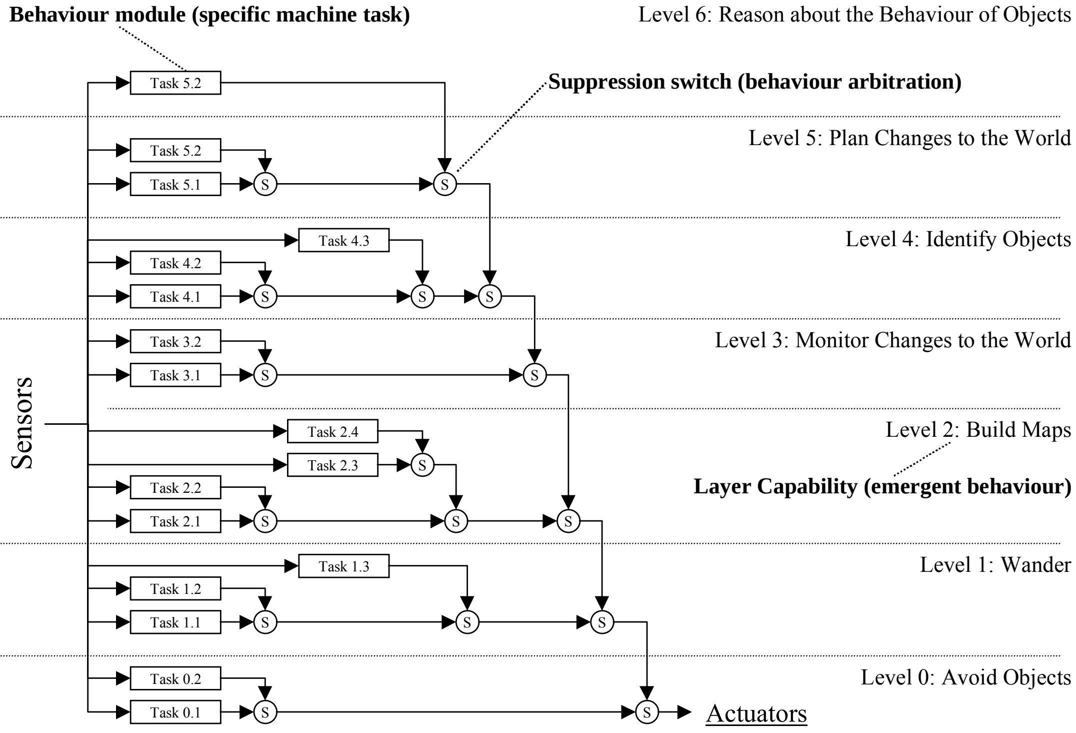
Figure 1: Behaviour-based System Architectures.

Additionally, consider inductive inference techniques such as learning classifier systems or genetic algorithms. Inductive inference is not logically sound, and the safety properties of systems whose functional components have been developed by such techniques are difficult to argue. While techniques such as machine learning and evolutionary computing are important research topics in AI, they are not absolutely necessary to intelligent control, and the research presented in this paper has concentrated on developing a design methodology that is based on more conventional 'manual' design techniques based on deductive reasoning. This avoids the potential pitfalls of attempting to build a safety argument on inductive logic.

Of the architecture types considered, the most suitable candidate architecture is the Colony-style Subsumption Architecture. It employs all the essential functional elements and design principles of the behaviour-based systems concept, while being the technology with the highest degree of the causal properties required for effective safety analysis. Furthermore, while there have been experiments involving the application of inductive learning techniques to the development of CSA systems [10], it is by no means absolutely necessary, as shown by Connell in his original development of the technology [9]. The design methodology presented in this paper has thus been applied preferentially to CSA.

2.2 Colony-style Subsumption Architecture

Colony-style Subsumption Architecture (CSA) was developed originally by Connell [9], as an improved version of the original Subsumption Architecture of Brooks [7]. The basic functional component of Colony-style Subsumption Architecture is the behaviour module, which encapsulates all the control activity required for the system to achieve a particular goal. The operational principle of the architecture is based on the notion that the transfer function of behaviour modules are partial functions, whose outputs are not defined for every possible input state. If modules do not generate output signals for all possible input states, their inactivity for some states allows other modules to control actuators if they occupy a lower-priority position within the system architecture. Figure 2, which has been adapted from Brooks' original paper [7], illustrates the concept.



Behaviour module (specific machine task)
Level 6: Reason about the Behaviour of Objects
Task 5.2
Suppression switch (behaviour arbitration)
Level 5: Plan Changes to the World
Task 5.2
Task 5.1
S
S
Level 4: Identify Objects
Task 4.3
Task 4.2
Task 4.1
S
S
S
Level 3: Monitor Changes to the World
Task 3.2
Task 3.1
S
S
Sensors
Level 2: Build Maps
Task 2.4
Task 2.3
S
Layer Capability (emergent behaviour)
Task 2.2
Task 2.1
S
S
S
Level 1: Wander
Task 1.3
Task 1.2
Task 1.1
S
S
S
Level 0: Avoid Objects
Task 0.2
Task 0.1
S
S
Actuators

Figure 2: Colony-style Subsumption Architecture - a Class of Behaviour-based Control.

A complete system comprises many behaviour modules, each satisfying a separate goal, which compete with each other for control of the system actuators via an arbitration network. The theory presented in this paper considers only priority-based arbitration, in which modules illustrated as being higher in hierarchy diagrams such as Figure 2 override the signals generated by lower-level modules, hence taking priority in their command of the machine/system. The principal type of arbitration mechanism employed in CSA (and the only one treated in this paper) is the *Suppression Switch*, for which the semantics are that the module providing the 'vertical' input overrides the module providing the 'horizontal' input, replacing the latter's output signal with its own. If the higher-priority signal is inactive, then the lower-priority signal passes through to the switch output. Since the convention is that modules on a behaviour diagram such as Figure 2 are typically drawn such that those illustrated higher up in the diagram provide the 'vertical' input, then it follows from this convention that higher-level modules (and hence higher layers) take priority over lower level ones, and that the diagram therefore represents a *hierarchy* of behaviour modules. An arbitration network can therefore be built up, to integrate all the system's behaviour modules together wherever they attempt to control the same system actuators. It may be the case, as shown in Connell's original work on the MIT Herbert robot [9], that several arbitration networks are needed, one for each actuator.

While suppression-type arbitration mechanisms are the only type considered here, the theory in this paper can readily be extended to the other types of priority-based mechanism, namely Inhibition

Switches (where the higher priority input overrides the lower priority one without replacing it) and Release Switches (where the higher-priority input enables the lower-priority input to pass to the switch output).

Within the overall hierarchy of the system, sets of behaviour modules are composed to form layers. The concept of the layer is that it describes a new externally observable capability of the machine, which is achieved collectively by the set of modules it encapsulates. The intention behind identifying layers within a subsumption architecture is that they describe the *emergent* behaviour produced by a set of modules, when superimposed on the existing lower layers. The layer concept seeks to capture the notion that, by adding new individual behaviour modules that do highly specific tasks, the machine can achieve new behaviour that cannot be achieved by any single module on its own. A formal treatment of layers requires a formal theory of behavioural emergence in systems. A full theory of how layers can be identified within a module hierarchy, or (more usefully) of how a layer description can be used to specify a set of behaviour modules (exploiting any emergence that may be achievable by such a decomposition), is the subject of ongoing research. However, the ideas presented in this paper that show how Lyapunov stability theory (and the extensions developed here) describes the mechanisms of subsumption should form an underlying basis for this work.

Wherever possible, we propose to represent the transfer function of a behaviour module as a *motor schema* - a tabular representation of the velocity field in the state space that is required for a given behaviour pattern. An example for a 'docking' behaviour for a wheeled robot is illustrated in Figure 3. This is motion in which a robot approaches a goal position from one direction. It is intended to allow the robot to connect physically with some other object such as another robot or an electrical charging station, where an approach from a different direction would not allow such a connection to be made.

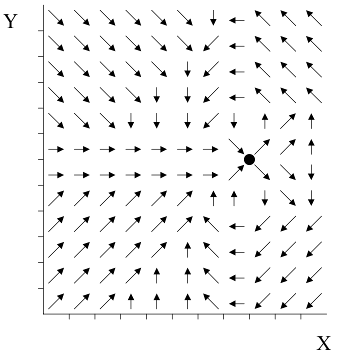


Figure 3: Example of a Motor Schema.

While recognising that the motor schema representation would not be applicable to all the behaviour modules in Figure 2, it is nevertheless our aim to represent as many of the layers as possible in this form, because the use of such a simple representation format allows the underlying computational hardware to be simplified, which brings benefits in terms of improved quality and reduced cost of safety assurance. These benefits are further elaborated in Section 8.

The methodology for designing individual behaviour modules is based on a new extension of Lyapunov stability theory, called the *Second Order Lyapunov Stability Theorems*. These theorems have two useful properties; they can prove the stability of the behaviour of a wider class of systems than is possible with the original Lyapunov Stability theorems (called the *First Order Stability Theorems*), and more importantly, stability can be achieved when only a partial control law function is defined for the system (if it satisfies the second order stability theorem). The first order stability theorems logically subsume the second order theorems whenever the system is behaving stably in the first order sense, and therefore it is not necessary for a function that is stable in the second order sense to be defined for such a condition. This logical subsumption property allows stable behaviour modules to be designed with partial functions, allowing them to be integrated into a subsumption architecture system.

3. THE SECOND ORDER STABILITY THEOREMS

3.1 The Mathematical Representation of Agent Behaviour

The central theme of our work is the development of a rigorous methodology for constructing CSA systems with provable safety and liveness properties. This requires a sound interpretation for the concept of intelligent behaviour, which allows a mathematical representation upon which rigorous methods of design and analysis can be based.

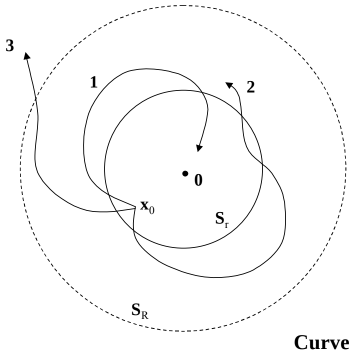
Behaviour-based AI draws extensively on ethology to provide the basic concepts of agent behaviour. Ethology generally proposes models of animal behaviour in terms of instinct patterns, such as Tinbergen's Innate Releasing Mechanisms model [38], an early attempt at a theory. Even though this model has subsequently been improved, most theories are still grounded in concepts such as instinct patterns, sensory stimuli, motor responses, inhibition, releasing, and so forth. Mathematical representations of these concepts are therefore necessary, and ethologists such as McFarland [26] have identified some key characteristics that permit mathematical or logical formalisation of descriptions of behaviour. Two key characteristics are that animal behaviour patterns are rational and purposeful. Animal behaviour is rational in that it is regular and consistent, and as such has one or more describing functions. It is purposeful in that behaviour patterns tend to be organised around goals, particular states or conditions that the animal tries to achieve, or perhaps avoid. This suggests that logically sound mathematical descriptions of behaviour are possible, if an appropriate notation can be found.

A third characteristic is that animal behaviour can be modelled using state spaces. Most ethologists tend to analyse behaviour by selecting some characteristic feature of an animal, or indicator of its condition or that of its environment. Ethological studies often concentrate on measuring the change of these indicators over time, with the pattern of change indicating which mode of behaviour is being displayed. This leads to the idea that behaviour is the change in state of the relevant measures/indicators. For complex behaviours that are characterised by several parameters, the range of possible values that can characterise the behaviour form a state space. The state of the animal is then defined as a state vector, and the change of state (i.e. behaviour pattern) modelled as a state trajectory. Properties of behaviour patterns can therefore be defined as mathematical properties of state trajectories.

These characteristics, in addition to the general characteristics of intelligent control outlined in Section 1.1, permit behaviour patterns to be specified in terms of the respective goal conditions they must achieve and the nature of the environment in which the system is situated. This is captured in a *dynamical model* of the system to be controlled. A design process for a behaviour module would therefore need to construct a motor schema over a state space representing the behavioural problem to be solved, whose structure had mathematical properties that provided some guarantee that the goals would be achieved under the stated dynamical model. One mathematical property that stands out in its applicability to this type of problem (and is in popular use [14] [26] [36]) is Lyapunov stability.

3.2 Lyapunov Stability

Lyapunov stability [23] is the property, which if proven, guarantees that the system state will always converge on some *equilibrium condition*. Once reached, the system will remain in the equilibrium condition unless disturbed. The inspiration for the theory is the physics of dissipative systems, which always converge on a fixed state if the energy of the system is finite (i.e. not replenished after the system is started). Lyapunov stability theory defines three properties of the state trajectory of a system, as illustrated in Figure 4.



Curve 1: Asymptotically Stable
Curve 2: Marginally Stable
Curve 3: Unstable

Figure 4: Types of Lyapunov Stability.

Lyapunov stability theory identifies three different categories of behaviour, based on the way in which state trajectories evolve over time. Curve 1 of Figure 4 illustrates Asymptotic Stability. If the state $\mathbf{x}_0$ of a system at time t_0 (the initial moment of time being considered by the analysis) exists sufficiently close to some equilibrium condition (defined as $\mathbf{x} = \mathbf{0}$), then the state trajectory will eventually converge on the equilibrium condition and remain there. This is formally defined by the following expression:

$$\textbf{Asymptotic stability:}\, \exists r > 0, \left\|\mathbf{x}(0)\right\| < r \quad \Rightarrow \quad t \to \infty, \left\|\mathbf{x}(t)\right\| \to 0 \tag{1}$$

The $\|\mathbf{x}\|$ term represents Euclidean Distance, i.e. $\|\mathbf{x}\| = \left(x_1^2 + x_2^2 + \ldots + x_n^2\right)$ for an n-dimensional state vector. Equation (1) asserts that if the state vector at time t = 0 is within a ball of radius r around the equilibrium condition $\mathbf{x} = \mathbf{0}$ (it is standard convention to place the equilibrium condition at the origin of the state space axes, thereby removing any constants from the distance terms), and the Euclidean distance vanishes to zero (i.e. the state converges on the equilibrium condition) as time advances, then the system is asymptotically stable.

A second type of stability is Marginal Stability, as shown by Curve 2 in Figure 4. If the state $\mathbf{x}_0$ of a system at time t_0 exists sufficiently close to the equilibrium condition, then it will always remain within some closed neighbourhood region (not necessarily the same one as the initial region) around the equilibrium condition. Marginal stability is formally defined by the following expression:

$$\textbf{Marginal stability}: \forall R > 0, \exists r > 0, \quad \left\|\mathbf{x}(0)\right\| < r \quad \Rightarrow \quad \forall t \geq 0, \left\|\mathbf{x}(t)\right\| < R \tag{2}$$

This equation states that if the Euclidean distance at time t = 0 is within a ball of radius r around the equilibrium condition, then it will remain within a ball of radius R for all future time (where $r \leq R$).

A third type of behaviour is *Instability*, as represented by Curve 3 of Figure 4. Unstable trajectories are those that do not remain within any sphere around the equilibrium condition. Unstable trajectories do not require any characterising equation; any trajectory that is not asymptotically or marginally stable is unstable by definition.

Lyapunov's theorems for the proof of marginal and asymptotic stability are based on the identification of scalar functions over a neighbourhood surrounding the equilibrium condition the state, called a Lyapunov function. If a Lyapunov function can be proven to exist for a system, then the system will have been proven to be marginally stable or asymptotically stable, depending on the properties of the Lyapunov function. For marginal or asymptotic stability, a candidate Lyapunov function V($\mathbf{x}$) must satisfy the following properties:

If, in a ball $\mathbf{B}_R$, there exists a scalar function V($\mathbf{x}$) with continuous first partial derivatives such that:

(1) $V(\mathbf{x})$ is positive definite (locally in $\mathbf{B}_R$)

(2) $\dot{V}(\mathbf{x})$ is negative semi-definite (locally in $\mathbf{B}_R$)

then the equilibrium point is marginally stable.

(3) If, in addition, the derivative $\dot{V}(\mathbf{x})$ is locally negative definite in BR then the stability is asymptotic.

It is assumed that the region of interest surrounds an equilibrium condition and that the origin point for the state variable axes is considered to be chosen so as to place the origin ($\mathbf{x} = \mathbf{0}$) at the equilibrium condition. This is usually the case in most practical applications.

The original first-order Lyapunov stability theorems have been widely used within all branches of control theory. However, there are constraints on what classes of trajectory can be proven stable with these basic theorems. The main issue is that the stability theorems are only sufficiency criteria, not necessity criteria. The theorems state that if a Lyapunov function exists for a system then it is stable. They do not say that a system is not stable if no suitable function can be found. The limitation of the Lyapunov stability theorems can be seen when one considers the trajectories of the systems whose stability they can prove. The theorems can only prove that a system is stable if its trajectories always follow paths where the rate of change of the Lyapunov function along the trajectory is zero or negative (and only negative for asymptotic stability). Hence, only systems with trajectories like those shown for the two-dimensional state space in Figure 5 can be proven stable by the theorems.

Many physical systems, and many control problems for those systems, cannot be modelled in such a way that the state trajectories are constantly convergent on an equilibrium condition. Inertia-dominated systems cannot be controlled so as to overcome their momentum instantaneously, and hence the direction of their velocity cannot be governed directly.

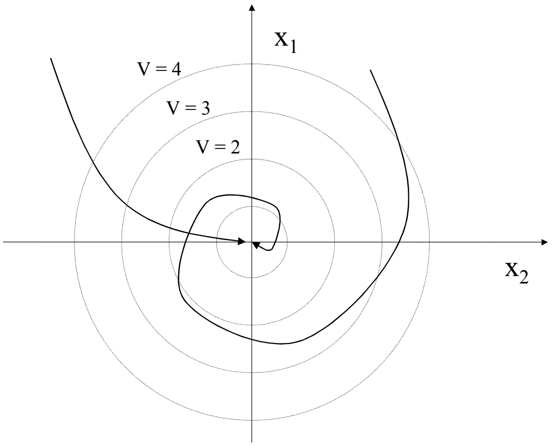


Figure 5: Trajectories Displaying First Order Asymptotic Stability.

In particular, the behaviour patterns of many intelligent mobile agents are defined as position control problems, and since physical actuators control the forces applied to a system, it is only possible to govern the acceleration (second derivative of position) directly. Hence, for Lyapunov stability techniques to be applied to such agents, we need an extension of the original theory (the Second Order Stability Theorems) in which the stability of a system's position can be proven directly in terms of the forces (second derivative) applied to it.

3.3 Second Order Lyapunov Stability

The new theorems are called Second-Order Stability Theorems because they introduce the second derivative $\ddot{V}(\mathbf{x})$ of the Lyapunov function $V(\mathbf{x})$, and allow the stability of a system to be proven within a wider set of constraints than is required for the First-Order Theorems. The concept underlying the theorems is the intuitive idea that, if $V(\mathbf{x})$ is increasing along the system trajectories, then as long as it is 'decelerating' by more than a certain amount, its value will remain bounded, and this implies that the equilibrium condition $\mathbf{x} = \mathbf{0}$ will be stable. This is represented graphically in Figure 6 where the 'deceleration' conditions, defining how a trajectory that is initially divergent from the equilibrium condition can be made to become convergent again, are shown. Note: the function $W(t)$ represents the change of value of $V(\mathbf{x})$ along the system state trajectories as the system state evolves through time.

Second order stability can be said to be inspired by physical analogy, as was the first-order stability theory. The theory is analogous to the notion of a gravitational field of a point mass, where a particle will find its way towards the centre of such a field, and the trajectory of that particle will stabilise on that central point. However, if its initial velocity within the field of a Lyapunov function is too great (being analogous to the 'escape velocity' of a rocket, for example) then the state trajectory may exceed a given boundary neighbourhood of the centre and escape (instability). Figure 6 provides an illustration of this concept.

Theorem 1 - Second Order Marginal Stability:

Let $\mathbf{x}(t)$ define the time evolution of an autonomous system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ and let $V(\mathbf{x})$ define a positive definite scalar function over a neighbourhood of $\mathbf{x} = \mathbf{0}$. Let $W(t) \equiv V(\mathbf{x}, t)$ denote the value of $V(\mathbf{x})$ *along the system state trajectories*. Hence $\dot{W}(t) \equiv \dot{V}(\mathbf{x},t)$ and $\ddot{W}(t) \equiv \ddot{V}(\mathbf{x},t)$ denote the rate of change and acceleration of the value of $V(\mathbf{x})$ along the system's trajectories. The following theorem defines *marginal stability in the second order*:

$$\dot{W}(t) \le 0 \vee \left[0 < \dot{W}(t) < \dot{W}_{max} \wedge \ddot{W}(t) < \ddot{W}_{max} < 0\right] \to [\forall R, \exists r > 0, \|\mathbf{x}(0)\| < r \to \|\mathbf{x}(t)\| < R] \tag{3}$$

Stability is possible in the presence of positive initial conditions for $\dot{W}(t)$, provided that they are bounded and that $\ddot{W}(t)$ has a negative upper bound.

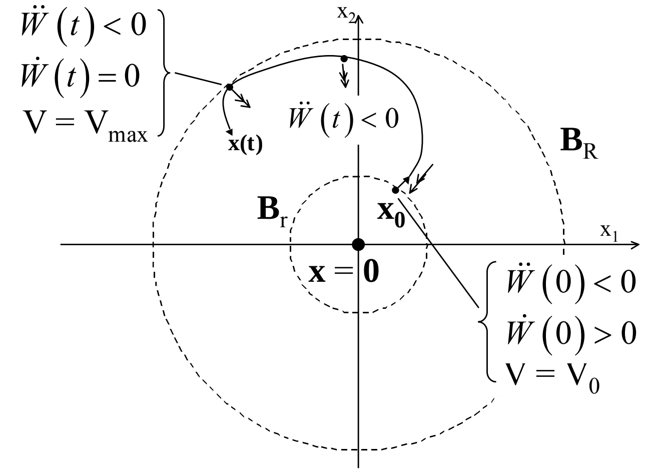


Figure 6: Graphical Representation of Second Order Stability.

The relationship between V_{max} and V_0 is defined by the equation:

$$V_{max} = V_0 + \frac{\dot{W}^2_{max}}{2|\ddot{W}_{max}|} \tag{4}$$

Theorem 2 - Second Order Asymptotic Stability:

The following theorem defines *asymptotic* stability in the second order:

$$\begin{aligned}&\dot{W}(t) < 0 \vee [0 \le \dot{W}(t) < \dot{W}_{max} \wedge \ddot{W}(t) < \ddot{W}_{max} < 0] \\ &\Rightarrow [\forall R, \exists r, 0 < r < R, \|\mathbf{x}(t)\| < r \to [\|\mathbf{x}(t)\| < R \wedge \|\mathbf{x}(t \to \infty)\| \to 0]]\end{aligned} \tag{5}$$

Note that, in this theorem, the first $\dot{W}(t)$ term is required to be negative definite, which is the criterion for asymptotic stability for the First-Order theorem.

Full proofs for these theorems are given in Appendix A and Appendix B respectively.

3.4 Expressing Safety Properties Using Lyapunov Stability Theory

Given our aim, as stated in Section 1, to develop a methodology for behaviour-based systems for safety critical applications, it is necessary to define how safety and liveness properties can be specified in terms of Lyapunov stability concepts, so that we can translate a basic specification of a system's mission, and a description of the hazards involved, into a specification of Lyapunov stability requirements that we can apply in the Direct Lyapunov Design process. We therefore require a model of how safety and liveness can be represented in these terms.

3.4.1 Liveness

Liveness is the easiest property to define in Lyapunov stability terms - it is directly equivalent to asymptotic stability. The definition of liveness given in Section 1 is that a system will achieve its functional purpose under all external conditions. Since the functional purpose of a behaviour module (and any instinctive behaviour pattern as defined in an ethological sense) is to achieve some goal condition that characterises the behaviour, the notion of liveness coincides with the notion of asymptotic stability if we define the goal condition to be achieved as the equilibrium condition of an asymptotically stable function, and the external conditions to be the initial conditions of the system and any disturbances imposed during its operation. Since all trajectories within a given region are proven to attain the equilibrium condition, it follows that under the definition of equivalence proposed here, any asymptotically stable function also possesses the property of liveness.

3.4.2 Safety

The definition of safety as a property, given in Section 1, is that a system will avoid all conditions, which have been identified as hazardous, at all times over the course of its operation. This coincides with the concept of instability in the Lyapunov sense (see trajectory 3 of Figure 4), if we accept the definition that hazards can be defined as specific states (or regions of states), and that they are defined as conditions for functions, for which an instability proof exists, as illustrated in Figure 7.

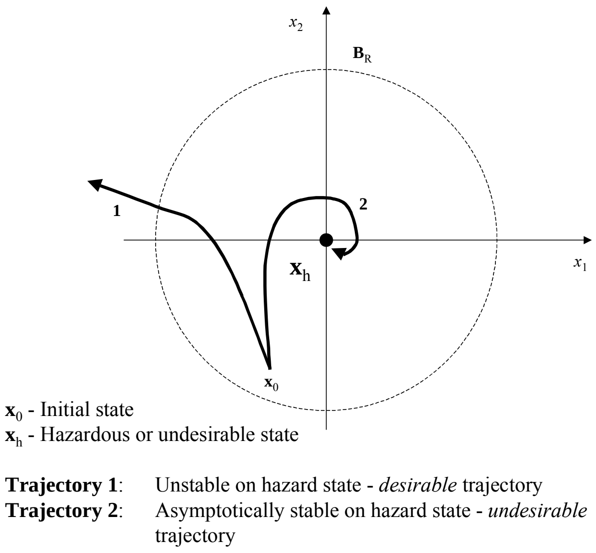

x_2
$\mathbf{B}_R$
1
2
$\mathbf{x}_h$
x_1
$\mathbf{x}_0$
$\mathbf{x}_0$ - Initial state
$\mathbf{x}_h$ - Hazardous or undesirable state
Trajectory 1: Unstable on hazard state - *desirable* trajectory
Trajectory 2: Asymptotically stable on hazard state - *undesirable* trajectory

Figure 7: The Expression of Safety Properties Using Lyapunov Stability Concepts.

The trajectories of a system proven to be unstable with respect to a particular condition will always leave a neighbourhood region of the condition for all future time, and hence it can be said that the system continuously avoids the given condition. Thus it follows that, under the equivalence definition proposed, any function that is unstable with respect to a given (hazard) condition (or set of conditions) also possesses the property of safety for that condition.

3.4.3 Dependability (Safety and Liveness Combined)

For a system (or a behaviour module) to be truly dependable, it must possess both safety and liveness properties. Its state trajectories should therefore simultaneously converge on goals, while avoiding hazards, as illustrated in Figure 8.

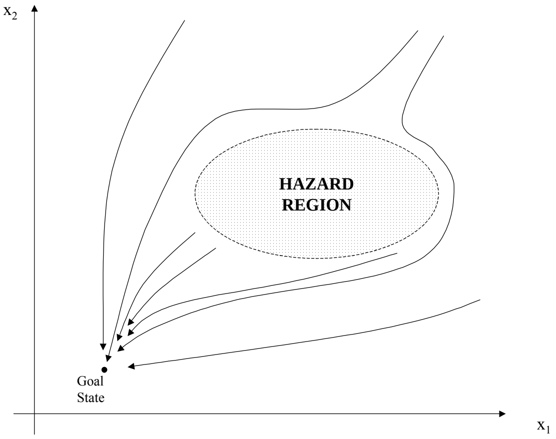


Figure 8: Trajectories Combining Safety and Liveness.

3.4.4 Discussion.

While we have described in this section the criteria for safety properties, as a relevant part of this discussion, a technique for proof-of-safety properties in motor schema will require a second order Lyapunov theorem for instability; this is work in progress. The experiment discussed in Section 7 consists only of a proof of the liveness properties of a system, not any safety properties.

The argument that Lyapunov stability properties can serve as expressions of safety and liveness properties is contingent on our assertion of the equivalences between the parameters of safety, liveness, asymptotic stability and instability that we have proposed in this section. We believe that these definitions will prove to be useful, and will allow us to demonstrate that any CSA behaviour-based systems we develop will have the properties necessary to pass a safety certification exercise. The evidence for this will lie in experimental demonstrations, the first of which we present in Section 7.

4. Direct Lyapunov Design

We use the Second Order Lyapunov Stability Theorems as the basis for a design procedure for the motor schema of a behaviour module. Typically, the (first order) Lyapunov stability theorems tend to be applied by searching for a candidate Lyapunov function, which proves the stability of a control system transfer function that is already given. Since a motor schema is a piecewise function we adopt the inverse of this procedure, searching (piece by piece) for a motor schema, which ensures that the derivatives of a pre-defined positive-definite function are negative along the system state trajectories (thereby proving stability). Since we use the Lyapunov function directly as the basis for designing a motor schema, we have named the method *Direct Lyapunov Design*. The procedure is performed by the following steps:

(1) Obtain a model of the open-loop dynamics of the system to be controlled. This can take the form of equations, explicitly defined maps, or some other model.

(2) Define the Goal State S required for the task, and the neighbourhood of S for which a task controller is required.

(3) Define a grid or mesh of points over the neighbourhood (each point is a state).

(4) For each point in the grid, select the control action that yields the most stabilising behaviour according to the second-order stability theorems, i.e. the action that yields the most negative value of $\ddot{V}(\mathbf{x})$. Make a record of $\dot{W}_{max}$ and $\ddot{W}_{max}$.

(5) Define a piecewise map function in which the grid points are the central states of each input-output pair, and their associated selected actions are the function outputs.

(6) Define the V_0 region by using Equation (4) and substituting the value of $V(\mathbf{x})$ at the outer boundary of the neighbourhood for V_{max}.

It should be noted that steps 3 and 4 constitute an inductive inference about the stability of a transfer function, because the stability of the function (a universally quantified statement) is inferred from a finite set of points (the mesh-grid of points) at which stability is calculated. Thus, stability is not strictly proven by this procedure. Therefore, as its stands, the procedure is not completely satisfactory for application to safety critical problems. However, for systems whose trajectories vary smoothly over the relevant neighbourhoods, the stability properties of the system should be preserved even where piecewise functions are defined with a finite set of input-output pairs, so for the purposes of our initial experiments, it was sufficient to develop some useful demonstrations (see Section 7).

It should be noted that this procedure is sufficient to design a motor schema that possesses liveness properties, but since there is no step for selecting control actions that are unstable with respect to any given condition labelled as hazardous, the procedure will not yield motor schema with provable safety properties.

5. Second Order Stability as a Model of Dynamical Subsumption

While Second Order Stability was originally intended as a technique to assist the design of individual behaviour modules, one of the main breakthrough developments in our research has been the realisation that the theory also models many of the basic principles of Subsumption Architecture. In this section we explain how subsumption operates within subsumption architecture systems, and show how Second Order Stability theory can be used to model those operational principles.

5.1 Subsumption in Behaviour-based Systems

While the term ‘subsumption’ actually defines a specific logical property, it has also been used in a different way to describe the operating principle of Subsumption Architecture systems. The term was first coined by Brooks [7] to describe the concurrent operation of layers within a system, and the incremental extension of a system, layer by layer, to achieve new capabilities. Each layer in a subsumption architecture (and even each module in CSA) operates entirely independently of any layer that may exist above it, As each new layer is added, it may draw upon behaviour already generated by lower layers if that behaviour should be advantageous to the goals of the new layer, or it may override the lower layers by means of suppressing their outputs through the arbitration network. Behaviour modules in the new layer only add the minimum of extra behaviour patterns needed to get the new level of capability, and no more. In this way, higher layers are said to ‘subsume’ lower ones.

Since, in a priority-based arbitration network, the presence of any output signal from a behaviour module will override any other module of lower priority, behaviour modules can only exploit the behaviour of lower-priority modules if they only generate output signals when it is necessary (for their purposes) to override the others. Hence, the transfer function of each behaviour module must be defined as a partial function which is not defined (and hence generates no signal) for the states in which the behaviour module can exploit the behaviour of the underlying system. If a behaviour module had a fully defined function, an output value would be generated for every possible input condition, the behaviour module would be constantly active, and it would completely suppress every other module of a lower priority. This would prevent the goals of the lower-priority modules from ever being achieved, and thus the ability of the system as a whole to achieve multiple goals would be destroyed.

For reasons that will be explained later, we refer to the principle of exploiting the behaviour of underlying layers as “dynamical subsumption”, in order to emphasise the behavioural aspect of the principle and to distinguish between this concept and the notion of subsumption in formal logic.

5.2 Modelling Dynamical Subsumption in CSA Systems using Second Order Stability

The development of the Second Order Stability theorems has offered a formal basis for the description of behaviour modules in Subsumption Architecture (or indeed in any priority-based arbitration scheme), and even offers a formal model of the principle of dynamical subsumption discussed in Section 5.1. We explain the argument in the following sections 5.2.1 to 5.2.3.

5.2.1 Logical Subsumption between the First and Second Order Lyapunov Stability Theorems

A partial logical subsumption relationship exists between the first order and second order Lyapunov stability theorems. The relationship becomes evident when the two asymptotic stability theorems are presented together:

First-order Asymptotic Stability: (6)

$$\dot{W}(t) < 0 \qquad \rightarrow [\forall R, \exists r, 0 < r < R, \|\mathbf{x}(0)\| < r \rightarrow [\|\mathbf{x}(t)\| < R \wedge [\|\mathbf{x}(t \rightarrow \infty)\| \rightarrow 0]]]$$

Second-order Asymptotic Stability: (7)

$$\dot{W}(t) < \dot{W}_{max} \wedge \dot{W}_{max} \geq 0 \wedge \ddot{W}(t) < \ddot{W}_{max} < 0 \qquad \rightarrow [\forall R, \exists r, 0 < r < R, \|\mathbf{x}(0)\| < r \rightarrow [\|\mathbf{x}(t)\| < R \wedge [\|\mathbf{x}(t \rightarrow \infty)\| \rightarrow 0]]]$$

Equations (6) and (7) actually express the first order and second order stability parts of Equation (5) respectively. The full theorem as written in Equation (5) hides the subsumption relationship by merging the first and second order parts into one composite formula.

By comparing the first and second order theorems in this way, it can be seen that the first-order theorem subsumes the second-order theorem whenever $\dot{W}(t)$ is negative. The two theorems have the same conclusions. But the antecedent of the first-order theorem is expressed only in terms of $\dot{W}(t)$ while the second order theorem requires an additional $\ddot{W}(t)$ term. Therefore, the first order theorem subsumes the second order one. However, the subsumption is not valid over every value of $\dot{W}(t)$, only where it is negative. Hence, the subsumption relationship is only a partial one, but it is this partial logical subsumption relationship that allows us to develop a formal model of subsumption in a Subsumption Architecture.

5.2.2 How Partial Functions Can Be Defined Using the Second Order Stability Theorem.

As we already mentioned in Section 3.3, the original reason for developing the Second Order Lyapunov Stability theorems was to provide a theory that could enable the proof of stability of behaviour module whose transfer functions generated actuator commands directly as a function of position states. Since physical actuators generate forces, which govern the second derivative of a system’s position, the effect of actuators can only govern second derivatives in the Lyapunov functions we use to construct motor schema. Therefore, the transfer function of a behaviour module need only be defined for those states where the value of $\ddot{W}(t)$ must be constrained in order to guarantee stability in the system state trajectory. For those states where the stability requires only a condition on the first derivative $\dot{W}(t)$ there is no second order term to determine, and hence *no actuator command need be defined*. Hence, only a partial function satisfying (7) is required, and a behaviour module designed using the Second Order Stability theorems can still guarantee the stability of a system even if there is no output defined where the state trajectory satisfies (6). Since we have established previously that priority-based arbitration requires the use of partial functions, it becomes clear that if a motor schema is designed using the second order stability theorems in the manner described, it can serve as a behaviour module in a subsumption architecture. The Direct Lyapunov Design procedure described in Section 4 achieves this objective.

5.2.3 How Logical Subsumption between the Stability Theorems Corresponds to Dynamical Subsumption between Subsumption Architecture Layers and Modules

In addition to enabling the construction of partial functions for behaviour modules, the Second Order Stability theorems also describe the principle of dynamical subsumption itself, the way in which higher priority modules exploit the activity of lower priority ones.

The key to this idea lies in the interpretation of the first-order stability terms, and what they represent about the behaviour of underlying layers relative to a given module. As we discussed in Section 5.2.3, a behaviour module transfer function need only be defined where the second derivative of the Lyapunov function requires constraint, as defined by (7). However, where the state trajectory satisfies (6), then no actuator function is necessary, and the transfer function need not be defined (and indeed, it should not, if the module is to be usable in a priority-based arbitration scheme).

Formula (6) identifies those conditions where the behaviour of a system satisfies the first order stability theorem for the goals of a behaviour module. For most practical control problems this describes the velocity motion of a system. The behaviour described by (6) specifies the *ungoverned* stable behaviour of the system, i.e. where the natural motion of the system is already convergent on the goals of a behaviour module, and hence requires no overriding correction. Since the action of behaviour modules is to override any lower priority layer or module, the ungoverned behaviour described by (6) must represent the behaviour produced by lower layers that nonetheless satisfies the goals of the higher priority module, which the higher priority module exploits in order to achieve its goals. This is exactly the principle of subsumption set out by Brooks in his initial proposal of subsumption architecture [7], and reviewed in Section 5.1.

This subsumption relationship relates to the elements of a Subsumption Architecture as illustrated by Figure 9.

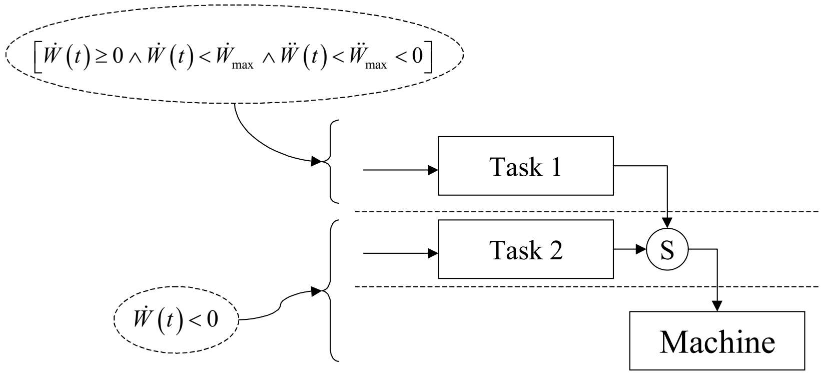

$\left[\dot{W}(t) \geq 0 \wedge \dot{W}(t) < \dot{W}_{\max} \wedge \ddot{W}(t) < \ddot{W}_{\max} < 0\right]$
Task 1
Task 2
S
$\dot{W}(t) < 0$
Machine

Figure 9: Correspondence between the logical subsumption in the Second Order Stability Theorems and dynamical subsumption in a Subsumption Architecture.

At any given layer in the architecture, the behaviour modules of that layer need not take action if the behaviour generated by the underlying layers is naturally convergent on their goals (i.e. where formula (6) is satisfied). The underlying layers may include behaviour modules performing other tasks, or may just be the machine under control. Only when the state trajectories of the underlying system stop converging on the goal state(s) of the top level, i.e. where (6) ceases to be valid and actions that ensure the validity of (7) must be defined, is there a need for the behaviour module to select a control action with which to drive the system. This is done by means of the suppression switch, as shown in Figure 9, and the action taken by the top layer must satisfy the second-order stability theorems if the stability of its goal(s) is to be maintained.

As we discussed earlier, it is standard practice in behaviour-based control to state that higher-level behaviour modules subsume the activity of lower-level ones. However, from the argument presented in this section it can be seen that, in the *logical* sense at least, it is the lower levels of a system that partially subsume the upper levels, in that they may partially achieve the goals of the higher levels without requiring any explicit control module to do so. So, from a logical perspective, lower levels partially subsume higher ones. This reversal of precedence in the concepts of logical and dynamical subsumption is the reason why the term dynamical subsumption was introduced in Section 5.1, in order to avoid confusion between the two. It is important to note that the great majority of papers and references on the subject (as typified by [7],[9],[25]and [28]) refer to dynamical subsumption, not logical subsumption.

6. THE DESIGN METHODOLOGY

The methodology for integrating behaviour modules into a wider subsumption architecture is based on the concept of managing the interference between modules. When a higher-priority module suppresses or inhibits the output of a lower-priority module it isolates the lower-priority module from the actuators of the system. Therefore, the lower module cannot take any action to control the system, and the state of the system departs from the trajectory that was being generated by the lower-priority module. When the higher-priority module ceases activity, the lower-priority module can control the system once again, but it must do so from an initial state condition that could be a large distance from the goal state it must achieve. Hence, higher-priority modules impose *disturbances* upon the behaviour of lower-priority ones.

The possibility exists that the higher-priority module could drive the system to such a departure from the goals of the lower-priority module that those goals are destabilised, and its activity will fail. Therefore, behaviour modules must be organised within a subsumption architecture to ensure that this does not happen.

The design methodology does this by two distinct measures.

(1) First, behaviour modules are organised in such a way as to minimise the mutual interference that higher-priority modules might impose on lower-priority ones. If a behaviour module is uniformly asymptotically stable, then it is totally stable, i.e. it remains at least marginally stable in the presence of bounded disturbances. Therefore, as long as the relative interference of one module upon another is minimised, then the modules can be arranged within a subsumption architecture without any further modification. The interference is minimised if the architecture obeys a constraint we have named the *Diminishing Activity Principle*:

> The higher the priority of a module within an architecture, the lower the average mark-space ratio of its output activity should be.

A theorem is proposed, and proved, in Appendix C stating that the Diminishing Activity Principle yields the architecture with the minimum relative interference between the behaviour modules of a system.

(2) The second measure for managing interference between modules is the use of *protection modules*, as illustrated in Figure 10. Since higher-priority modules can override a lower-priority module, their actions can drive the system outside the limits of the region within which the stability of the lower priority module has been designed. If higher layers of the system should do this, then when their activity ceases and the lower priority module regains control of the actuators, the system will be in a state for which the stability of the module has not been proven, and hence its behaviour cannot be assured. In order to prevent this from happening, an additional module is necessary that can override the behaviour of the main system if it should exceed the stable region of the lower priority module with which it is associated. In order to distinguish between modules that perform the intended mission tasks of the system, and those which are protecting their stability regions, we call the former category Achievement Modules, and the latter we call Protection Modules. The specification of a protection module is not to generate stable behaviour, but to ensure that the sys-

tem state remains within limits that ensure the function of its associated achievement module is valid.

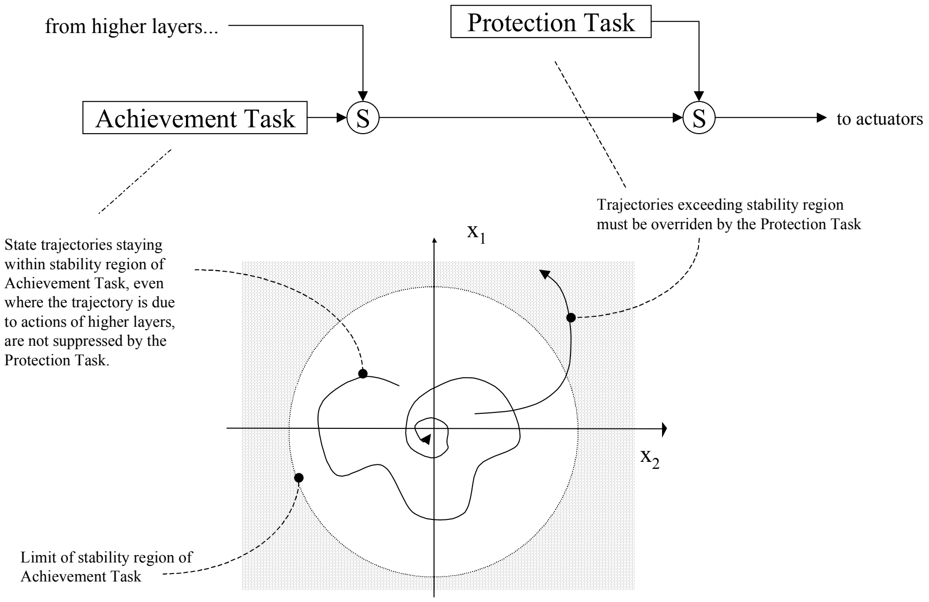

from higher layers...
Protection Task
Achievement Task
S
S
to actuators
Trajectories exceeding stability region must be overriden by the Protection Task
State trajectories staying within stability region of Achievement Task, even where the trajectory is due to actions of higher layers, are not suppressed by the Protection Task.
x_1
x_2
Limit of stability region of Achievement Task

Figure 10: Protection Modules.

The incorporation of these two measures into the structure of a subsumption architecture produces a standard model whose form and activity properties are illustrated in Figure 11:

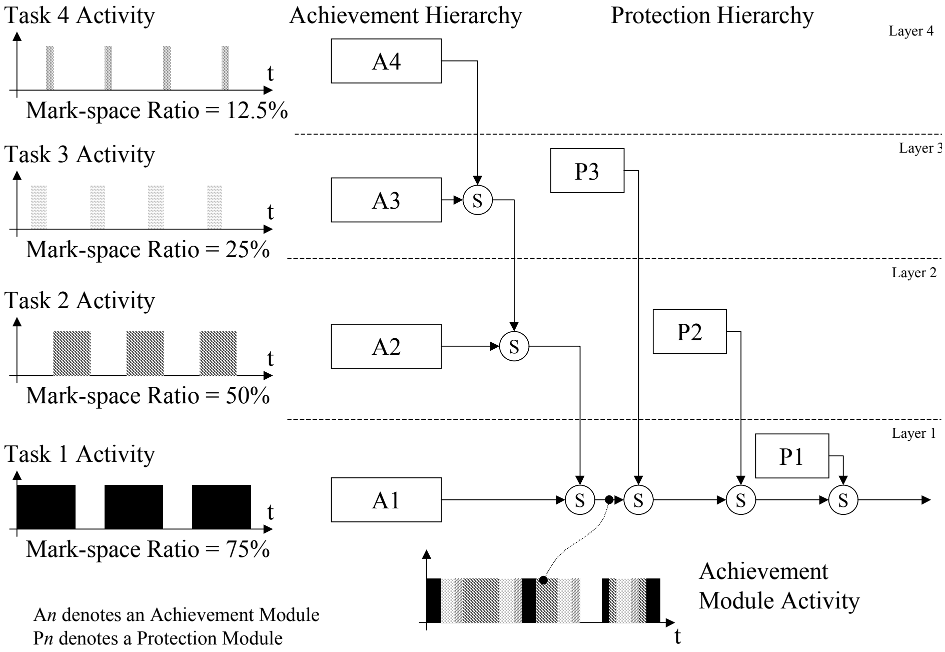


Figure 11: Standard Model for Behaviour Module Activity and Organisation within a Subsumption Architecture.

In keeping with existing Subsumption Architecture concepts [7][9] the model is layered, each layer forming a new global capability of the machine being controlled. Each layer consists of achievement modules that perform the tasks required for the new global capability, and protection modules, which ensure that the operating limits required for stable operation of the achievement modules are maintained.

By following the arbitration rules established in the earlier discussion on design methodology the architecture model forms two internal hierarchies, an *Achievement Hierarchy* and a *Protection Hierarchy*. The Achievement Hierarchy consists of all the behaviour modules (called *achievement modules*) that perform the required tasks of the system, and are organised with higher layers suppressing lower layers, according to the Diminishing Activity Principle (indicated by the activity profile graphs). The Protection Hierarchy consists of all the protection modules for the achievement modules in the other hierarchy. However, in accordance with the definition of protection modules, the arbitration network for the protection hierarchy is organised in the reverse sense to the achievement hierarchy, with lower-layer protection modules suppressing higher-layer ones. In this way, global capabilities can be added to a system without compromising the operation of lower layers through inter-module interaction.

7. Experiments and Demonstrations

As a demonstration of the design methodology, we present a simulation of an aircraft flight control system that performs a simple 'cruise control' function. The purpose of the system is to control a simulated aircraft so as to achieve a steady 'straight and level' condition at a constant airspeed. In this experiment, we attempt only to establish liveness properties for the system, leaving the full demonstration of techniques for safety properties for future work.

7.1 Simulation Model

The dynamical model of the aircraft is derived from the Body-Axes Model described by Etkin [13], making the 'flat Earth' assumption. Appendix D lists the equations of motion and the aerodynamic parameters used in the simulation.

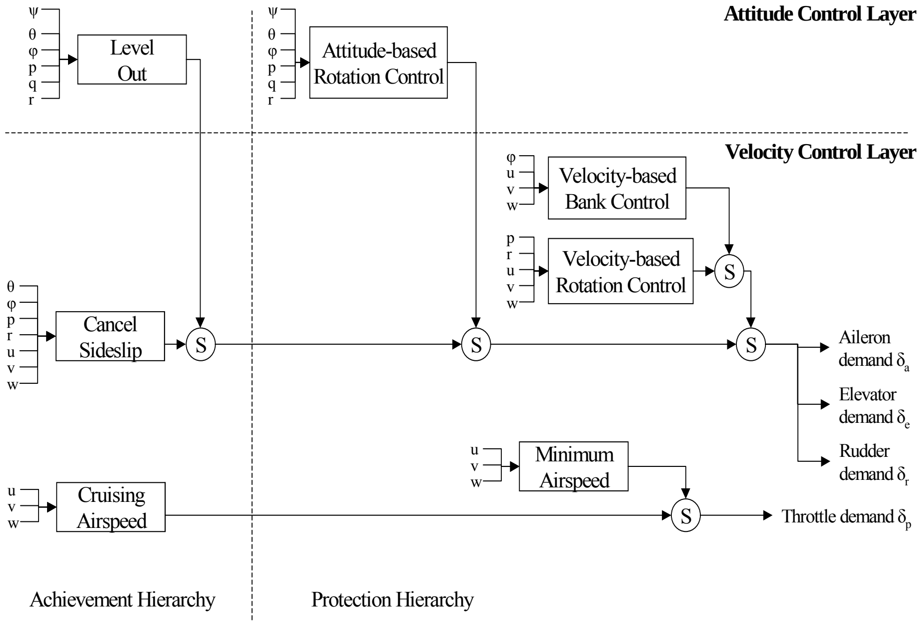


Figure 12: Experimental Subsumption Architecture for a Flight Control System Simulation.

Figure 12 contains a schematic of the actual Subsumption Architecture that was developed. It shows how the experimental architecture follows the standard model of achievement and protection hierarchies as illustrated in Figure 11.

The architecture has two layers, the bottom layer performing velocity control behaviours and the top layer performing attitude control behaviour. The reason for this is that the dynamics governing aircraft velocities (see the Force Equations in the Body Axes in Appendix D) can be stabilised in the first order, using the classical Lyapunov stability theorems. Since functions that are stable in the first order are not partial functions, their activity ratio is 100% by definition. Therefore, they are required to occupy the lowest layer. The dynamics of the aircraft's attitude (see the aircraft attitude kinematics equations in Appendix D) can only be stabilised in the second order, and hence the behaviour module is a partial function whose activity ratio is by definition less than 100%. Therefore, the attitude control layer must take the higher priority.

The structure of the architecture in Figure 12 follows the general pattern defined in Figure 11. The architecture contains an Achievement Hierarchy and a Protection Hierarchy, and the order of priority in the protection hierarchy is the reverse of that in the achievement hierarchy. The protection modules are designed to prevent the state of the system exceeding limits that prevent their respective achievement modules from achieving their stated goals. These limits are derived from the flight dynamics equations of the aircraft, as presented in Appendix D.

7.2 Results

The aircraft simulation was run for a variety of different initial conditions. The goal condition of straight and level flight is defined as zero deflection in any attitude, zero velocity in the downwards and sideslip direction, and a steady forward velocity of 67 ms^{-1}. The state trajectories of the system were plotted in two state trajectory portraits, for attitude and velocity. These portraits are shown in Figure 13 and Figure 14. The first diagram is a portrait of the three attitude variables (azimuth, elevation, and bank). The second diagram is a portrait of the velocities in the three axes of symmetry of the aircraft (forwards, downwards, and sideslip velocities).

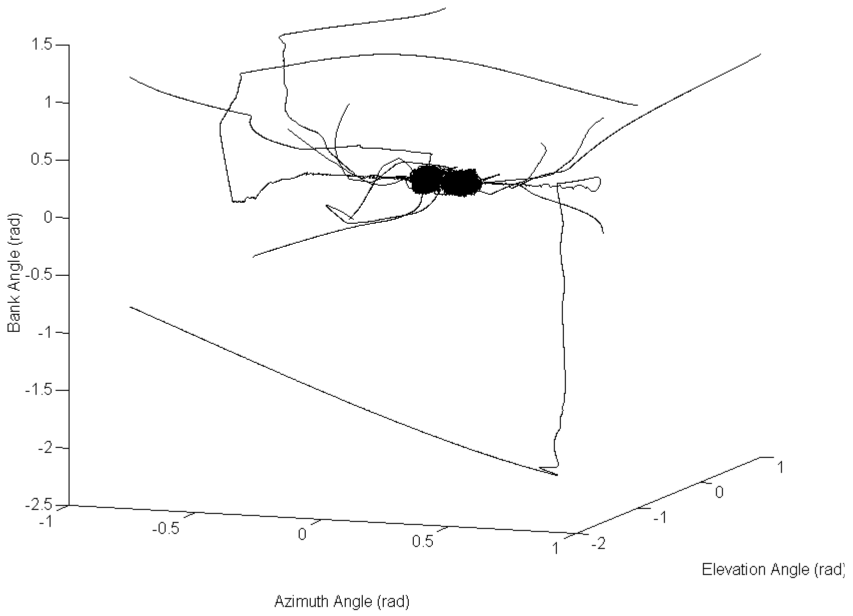


Figure 13: Airframe Attitude Trajectories.

The most noticeable aspect of Figure 13 is the fact that all the trajectories converge on a small region surrounding the intended equilibrium condition. This shows that the system as a whole can achieve marginal stability of attitude control. That the system does not achieve asymptotic stability can be explained by the fact that the mutual interference between the behaviour modules attempting to control the system tends to reduce their stability from asymptotic stability to marginal stability. This is a known property of asymptotically stable functions under disturbances known as Total Stability [34]. The fact that behaviour modules individually designed to be stable can retain that property in the presence of others, albeit reduced from asymptotic to marginal stability, is the principle that allows the system to work as a whole.

Figure 13 also has a number of trajectories that are provable only by second-order stability theorems (or higher) as opposed to classical first-order theorems. The most noticeable of these is the trajectory shown in the foreground in the lower half of Figure 13. The motion of this trajectory initially diverges from the goal state before 'turning a corner' and becoming convergent, similar in structure to the trajectory illustrated in Figure 6.

From the nature of the state trajectories on display in Figure 13, we draw the following conclusions:

(1) One can readily apply the second order stability theorems to obtain a proof of Lyapunov stability for systems. Furthermore, the second order theorems can be used to prove the stability of systems with trajectories that are initially divergent, a result that cannot be achieved using the first order theorems.

(2) The fact that the trajectories cover an extended region surrounding the specified 'straight and level' goal condition and that they all converge on a much smaller region, demonstrates the liveness of the "Level Out" module, which is the achievement module in the attitude control layer.

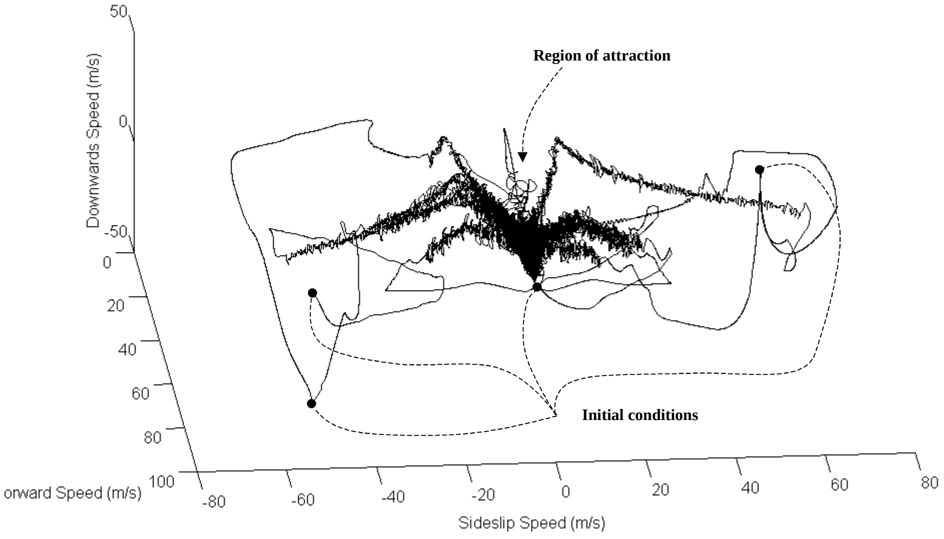


Figure 14: Airframe Velocity Trajectories.

The velocity variables whose trajectories are illustrated in Figure 14 are characterised by two features, an initial relatively smooth part, followed by the onset of a highly noisy part, with many high-frequency disturbances as the trajectory closes in on the goal condition. This is in contrast to the attitude trajectories in Figure 13, which remain relatively smooth until they reach their region of marginal stability around the equilibrium condition.

It should be noted that all the behaviour modules governing aircraft velocity in the three axes are defined by complete functions, which were actually designed by the application of first order stability theorems. It may be that this makes their operation much more sensitive to disturbances, because they are defined for every possible condition of their respective state spaces. Any disturbance at all will be met with a corresponding control action, yielding a trajectory that picks up all the interference between modules. In contrast, the attitude control modules, which were designed using the second order stability theorems, are partial functions which do not always generate control actions. Only where a disturbance produces a motion that requires action to restore the conditions of the second order theorem, as per Equation (5), will such a module generate an action. This has the effect of reducing the amount of activity generated in response to inter-module interference. In the velocity control modules, this phenomenon does not arise, and the trajectories are consequently more noisy.

7.3 Conclusions of the Experiment

We consider these results to demonstrate that the methodology described in this paper can produce viable behaviour-based systems with Colony-style Subsumption Architecture, in which the behaviour modules have been proven to be stable, which implies that liveness has been established. The results presented in Figure 13 and Figure 14 present an initial confirmation of the methodology and its underlying theories.

However, there are some limitations to what has been demonstrated. One significant limitation was the lack of smoothness in the state trajectories, which is brought about by conflicting interactions between behaviour modules, especially when the state trajectories are close to the goal conditions of the behaviour modules. Future work is aimed at improving the smoothness of behaviour generated by this type of subsumption architecture, by use of stability theorems of a higher order than the second. We are confident that the proofs for the second order theorems can be extended to the third order or higher.

Another set of limitations were the ideal assumptions made about the performance capabilities of system components. For example, it was assumed that actuators can change state instantaneously, which is not possible in real physical mechanisms. Therefore, the motor schema produced for the simulation may not have the same liveness properties if they were applied directly to a real aircraft. We believe, however, that the use of higher order stability theorems (possibly a third order stability theorem) will overcome this limitation. We aim to develop and apply this idea in future experiments (see Section 8).

Nonetheless, in spite of the limitations discussed above, we believe that the results of the experiment do validate the argument that safety and liveness properties can be specified in terms of Lyapunov stability and that these properties can be established formally within the motor schema of a CSA system.

8. Conclusions and Future Work

There are a number of conclusions to be drawn from the work presented in this paper, which have implications for future studies in this direction. Section 8.1 reviews the main conclusions to date, and sections 8.2 and 8.3 discuss some of the major directions in which we intend to pursue this research.

8.1 Conclusions of the Work to Date

The body of work presented in this paper forms the core of a design methodology aimed at the development of intelligent systems for safety critical application. The methodology includes a procedure for designing a behaviour-based system with a Colony-style Subsumption Architecture and proving the stability of the resulting design.

Since it is our intention that the methodology be suitable for developing behaviour-based systems for safety critical applications, we have developed a formal mathematical basis for their design, to provide the same level of rigour to the methodology that is achieved with conventional systems engineering technologies. We have developed a methodology from first principles, so that every concept in use had a mathematical representation derived from the basic concepts of behaviour. Some definitions, for example the definitions of how safety and liveness can be expressed in Lyapunov stability terms, have been asserted as a priori arguments. However, we believe that the experimental work done so far validates the approach.

While the methodology is primarily intended for application to systems [9], the basic theorems of stability, and their association to safety properties, makes no specific assumption at that level about the type of technology to be used. Therefore, this methodology may be adaptable to architecture types other than CSA. In particular, if the proposed models of different arbitration schemes (in Section 8.2) are successfully added, then we believe there to be significant scope for widening the applicability of the methodology.

8.2 Further Developments to the Design Methodology

While this work provides the core of a methodology, it is not complete. Further work will include:

(1) A Second Order Stability theorem for instability, which is the necessary basis for defining safety as a system property in Lyapunov stability terms. We require a theory which would prove that the existence of a Lyapunov function with the appropriate bounds on its first and second derivatives

implies that the state of a system would never come closer (in Euclidean distance terms) than some determinable value. We believe such a theorem should be derivable in a similar manner to the proofs given in Appendix A and Appendix B, but applied in the inverse sense. We aim to develop this theorem and apply it in new experiments to demonstrate the achievement of safety properties as well as liveness.

(2) A Layer Theory for CSA to account for how the behaviour of modules can combine to form an emergent capability that is referred to as the Layer in subsumption architecture terms. We believe this will require elements of environmental analysis, where we derive the specification of individual behaviour modules by studying the features of the environment in which the system is intended to be situated.

(3) An extension to the modelling of subsumption principles into non-priority based arbitration schemes. One possible mathematical basis for doing so might be based on the Passivity Theorems of Lyapunov stability theory [34]. These theorems prove that the outputs of certain classes of controller, independently proven to be stable, can be summed together and the composite system will still remain stable. We consider this to be a possible basis for proving the stability of ‘command-fusion’ styles of behaviour arbitration. If this development can be achieved, then we may be able to apply our methodology to many types of system architecture other than CSA.

8.3 Further Developments - Computational Hardware

A significant area for further exploration is how to exploit the potential benefits of the simplifying principle of behaviour-based systems (discussed in Section 2.1) for reducing the complexity of the hardware required to support such a system (compared to conventional microprocessor hardware). Since the experiment presented in Section 7 was a simulation, we have not yet investigated this potential.

The use of motor schema as the basis for implementing behaviour modules in CSA systems, as described in Section 2.2 has, we believe, an inherent advantage over conventional (symbolic) computers for the development and certification of safety critical systems. It may be possible to implement these systems using *simpler computational mechanisms* than microprocessors, with the potential for improved fault detection and tolerance over conventional microprocessors.

Motor schema are memory-less functions, and without the need for local internal memory behaviour modules can be constructed from pure combinational logic rather than synchronous circuitry. It is possible to implement these behaviour modules using Programmable Array Logic (PAL/PLA) devices, as shown in Figure 15.

Programmable Logic Arrays could directly implement motor schema if their inputs are, for instance, wired to sensors via A/D Converters and their outputs are wired to actuators in such a way that individual output wires generate individual fixed actions (the arrow symbols in the illustration). Inputs are coupled to outputs by setting programmable links, as shown by the black dots in the illustration. Since modern FPGA devices now have millions of gates, it may be possible either to implement multiple behaviour modules on the same device (occupying different memory areas), or single multivariate behaviour patterns that are of high dimensionality, which require a large number of separate points in the motor schema state space.

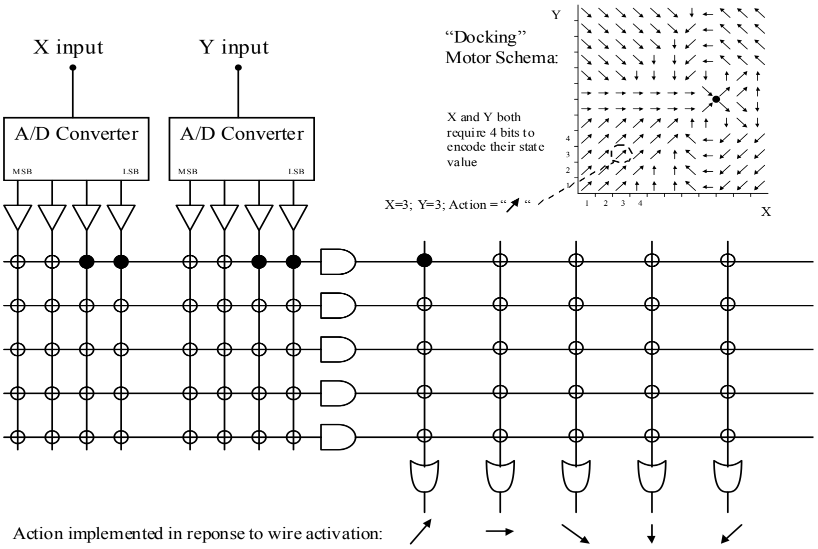


Figure 15: Implementation of Motor Schema in Programmable Array Logic.

The ability to use simpler implementations of functions leads to direct benefits in terms of the achievable diagnostic coverage and the improved possibilities for fault tolerant systems. The advantage of using relatively simple circuitry, such as programmable array logic, to implement motor schema means that the number of possible failure modes affecting the elements of the function are greatly reduced.

Further work will investigate this potential. We aim to move from simulation problems to physical machines (robots), with the subsumption architecture implemented using logic array devices, such as CPLD’s or FPGA’s. This will allow us to determine whether the arguments advanced in this paper can be exploited in the intelligent control of real physical systems.

9. REFERENCES

[1] Alcock J., *Animal Behaviour (7th Ed.)*, Sinauer Associates 2001

[2] Amir E., Maynard-Reid II P., *LiSA: A Robot Driven by Logical Subsumption* Proc. Fifth Symposium On Logical Formalizations Of Commonsense Reasoning 2001

[3] Arkin R.C., *Motor Schema Based Navigation for a Mobile Robot*, Proc. IEEE Conf. Robotics and Automation, Raleigh NC 1987, pp264-271

[4] Arkin R.C., *Behaviour Based Robotics*, MIT Press 1998

[5] Beer R.D., *A Dynamical Systems Perspective on Agent-Environment Interaction*, Artificial Intelligence Vol.72 (1995), Elsevier Science 1995, pp173-215

[6] Beer R.D., *Intelligence as Adaptive Behaviour: An Experiment in Computational Neuroethology*, Academic Press 1990

[7] Brooks R.A., *A Robust Layered Control System for a Mobile Robot*, IEEE Jrn. Robotics and Automation, Vol.RA-2 No.1, p14-23, March 1986

[8] Brooks R.A., *Cambrian Intelligence*, Chapter 1, MIT Press 1999

[9] Connell J.H., *A Colony Architecture for an Artificial Creature*, PhD Thesis, MIT Artificial Intelligence Laboratory 1989

[10] Connell J.H., Mahadevan S., *Robot Learning*, Kluwer Academic Press 1993

[11] Dorigo M. & Colombetti M., *Robot Shaping*, MIT Press 1998

[12] Esterline A.C., Rorie T., *Using the π-Calculus to Model Multiagent Systems*, FAABS 2000, LNAI 1871, pp164-179, Springer Verlag 2001

[13] Etkin B., Reid L.D., *The Dynamics of Atmospheric Flight - Stability and Control (3rd Ed.)*, Wiley 1996

[14] Gallagher J.C., Beer R.D., *A Qualitative Dynamical Analysis of Evolved Locomotion Controllers*, Proc. 2nd Intl. Conf. Simulation of Adaptive Behaviour (SAB92), pp71-80, MIT Press 1992

[15] Harnad S., *The Symbol Grounding Problem*, Physica D Vol.42, Elsevier Science Publishers 1990, pp335-346

[16] Harper C.J., *A Rational Methodology for Designing Behaviour Based Systems for Safety Related Applications*, PhD Thesis, University of the West of England 2004

[17] Huntingford F., *The Study of Animal Behaviour*, Chapman and Hall 1984

[18] Isaksen U., Bowen J.P., Nissanke N., *System and Software Safety in Critical Systems*, RUCS Technical Report RUCS/97/TR/062/A, Dept. of Comp. Sci., Univ. of Reading 1997

[19] Jin Y, Pipe AG and Winfield AFT, *Stable Neural Network Control for Manipulators*, IEEE Journal of Intelligent System Engineering (extended version), Vol 2, No 4, Winter, 1993, pp 213-222.

[20] Kermode A.C., *The Mechanics of Flight (8th Ed.)*, Pitman 1972

[21] Lesperance Y., Levesque H.J., *Indexical Knowledge and Robot Action – A Logical Account*, Artificial Intelligence Vol.73 (1995), pp69-115

[22] Low K.H., Leow W. K., Ang Jr. M.H., *A Hybrid Mobile Robot Architecture with Integrated Planning and Control*, Proc. 1st Int. Conf. AAMAS'02, Bologna, Italy, July 2002, pp219-226

[23] Lyapunov A.M., *The General Problem of the Stability of Motion*, Taylor Francis 1992 (reprint)

[24] Maes P., *Situated Agents Can Have Goals*, Robotics and Auton. Systems Vol.6 Nos.1-2, Elsevier 1990, pp49-70

[25] Mataric M.A., *Environment Learning Using a Distributed Representation*, Proc. IEEE 1990, p402-406

[26] McFarland D., Bösser T, *Intelligent Behaviour in Animals and Robots*, MIT Press 1993

[27] Newell A., Simon H.A., *Computer Science as Empirical Enquiry: Symbols and Search*, Comm. ACM Vol.19 No.3, March 1976

[28] Payton D.W., *Internalized Plans: A Representation for Action Resources*, Robotics & Auton. Sys., Vol.6 Nos.1-2, Elsevier 1990, p89-103

[29] Payton D., Rosenblatt K. *et al.*, *Do Whatever Works: A Robust Approach to Fault-Tolerant Autonomous Control*, Jrn. Applied Intelligence 2, Kluwer Academic Publishers 1992, pp225-250

[30] Pfeiffer R., Scheier C., *Understanding Intelligence*, MIT Press 1999

[31] Randall M.J., Winfield A.F.T., and Pipe A.G., *Stable on-line neural control of systems with closed kinematic chains*, IEE Proc-Control Theory Appl., Vol 147, No 6, pp 619-632, November 2000.

[32] Samad T., Balas G. eds., *Software-enabled Control*, IEEE Press 2003

[33] Sequeira J., Ribeiro M.I., *A Behaviour-based Kernel Architecture for Robot Control*, Proc. 5th IFAC Symp. Robot Ctrl. (SYROCO'97), Nantes, France, September 1997, pp833-838

[34] Slotine J-J.E., Li W., *Applied Nonlinear Control*, Prentice Hall 1991

[35] Steinhage A., *Dynamical Systems for the Generation of Navigation Behaviour*, Doctoral Thesis, Institüt fur Neuroinformatik, Ruhr-Universität Bochum, Germany, 1997

[36] Steinhage A., Bergener T., *Dynamical Systems for the Behavioural Organization of an Anthropomorphic Mobile Robot*, Proc. 5th Intl. Conf. Simulation of Adaptive Behaviour (SAB98), MIT Press 1998, pp147-152

[37] Storey N., *Safety-Critical Computer Systems*, Addison Wesley 1996

[38] Tinbergen N., *The Study of Instinct*, Oxford University Press 1951

10. APPENDICES

Appendix A. Proof of the Marginal Stability Properties of the Second Order Lyapunov Stability Theorem

As discussed in Section 3, the Second Order Marginal Stability Theorem is:

$$\begin{aligned}&\left[\dot{W}(t) \leq 0 \vee \left[0 < \dot{W}(t) < \dot{W}_{\max} \ \wedge \ \ddot{W}(t) < \ddot{W}_{\max} < 0\right]\right] \\ &\qquad \rightarrow \left[\forall R, \exists r > 0, \left\|\mathbf{x}(0)\right\| < r \ \rightarrow \ \left\|\mathbf{x}(t)\right\| < R\right]\end{aligned} \tag{8}$$

<u>Proof Sketch:</u>

The principle of the proof is that a system is Lyapunov stable in the second order if the worst-case trajectory of the system (within the constraints of the initial conditions) has an upper bound on the value of the Lyapunov Function $V(\mathbf{x})$. If the worst-case trajectory is stable, then all other trajectories will also be stable, as their initial conditions lie within the bounds established by the worst-case trajectory.

The worst-case trajectory has the highest initial value of $\dot{V}(\mathbf{x})$ and to which the least negative value of $\ddot{V}(\mathbf{x})$ is applied. By virtue of the constraints on $\dot{W}(t)$ in Equation (8), the upper limit of $\dot{W}_0$ is $\dot{W}_{\mathrm{max}}$ and the upper limit on $\ddot{W}(t)$ is $\ddot{W}_{\mathrm{max}}$. Since $\ddot{W}_{\mathrm{max}}$ is negative by definition, the upper limit of $\dot{W}(t)$ (the worst-case trajectory) is

$$\sup\left[\dot{W}(t)\right] \quad = \quad \sup\left[\int_{t_0}^{t} \ddot{W}(t)\,dt + \dot{W}_0\right] \quad = \quad \dot{W}_{\max} - \left|\ddot{W}_{\max}\right|\left(t - t_0\right) \tag{9}$$

Integrating $\dot{W}(t)$ once more yields an equation for $W(t)$. The upper limit of $W(t)$ is therefore:

$$\begin{aligned} \sup\left[W(t)\right] &= \int_{t_0}^{t} \sup\left[\dot{W}(t)\right]dt \quad + \quad \sup\left[W_0\right] \\ &= V_0 + \dot{W}_{\max}(t-t_0) - \tfrac{1}{2}\left|\ddot{W}_{\max}\right|(t-t_0)^2 \end{aligned} \tag{10}$$

This upper limit is V_{max}, the highest value of $V(\mathbf{x})$ that any system state trajectory achieves if it satisfies the constraints of Equation (8). The value of this limit can be obtained by investigating the turning points of the function, which occur when the value of Equation (9) is zero. Substituting Equation (9) into Equation (10) at the turning point yields Equation (4). Since $\ddot{W}(t)$ is always negative by definition, the turning point is a maximum, and therefore defines an upper limit on the value of $W(t)$ and as long as the constraints are satisfied, the system never leaves the neighbourhood defined by Equation (4).

<u>Full Proof:</u>

A full proof can be achieved by showing that the outer boundary V_{max} proven by Equation (10) implies that the state trajectory will remain within hyper-sphere $\|\mathbf{x}\| < R$ as defined in Equation (8).

Let the function $\mathbf{x}(t)$ define the state an autonomous system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ where $\mathbf{f}(\mathbf{0}) = \mathbf{0}$ at time t. Assume a positive-definite scalar function $V(\mathbf{x})$: $\mathfrak{R}_+^n \rightarrow \mathfrak{R}_+$ where $\mathrm{Dom}[V(\mathbf{x})] = \{\mathbf{x}\}$, where $\{\mathbf{x}\}$ is the set of all legal system states, and $\mathrm{Rng}[V(\mathbf{x})] = [0, \infty)$. Given the existence of $\mathbf{x}(t)$, there must also exist a function $W(t) \equiv V(\mathbf{x}(t))$. Let $\mathbf{x}(t)$ and $V(\mathbf{x})$ be such that $W(t)$ is continuously differentiable twice. Let the second derivative $\ddot{W}(t) \leq \ddot{W}_{max}$ be a known function, whose value has an upper bound. The first derivative $\dot{W}(t)$ is obtained by integration, and its initial condition is therefore defined as:

$$\dot{W}_0 = \dot{W}(t) - \int \ddot{W}(t)\,dt \tag{11}$$

Its upper bound is therefore:

$$\sup\left[\dot{W}_0\right] \overset{def}{\equiv} \dot{V}_{\max} = \sup\left[\dot{W}(t)\right] - \sup\left[\int \ddot{W}(t)\,dt\right] \tag{12}$$

If $\ddot{W}(t)$ is defined to be a negative definite function, i.e. $\ddot{W}_{\max} < 0$ then

$$-\sup\left[\int \ddot{W}(t)\,dt\right] = -\int \sup\left[\ddot{W}(t)\right]dt = -\ddot{W}_{\max}t \tag{13}$$

since the upper bound of the integral of $\ddot{W}(t)$ can be found by integrating its upper bound $\sup[\ddot{W}(t)]$. This is the value of $\ddot{W}(t)$ achieved by the *worst-case trajectory*, the one whose initial conditions are the maximum values allowed for stability. The upper bound of $\dot{W}(t)$ is therefore defined as:

$$\sup\left[\dot{W}(t)\right] = \ddot{W}_{\max}t + \dot{W}_{\max} \tag{14}$$

The function $W(t)$ can be determined by integration of $\dot{W}(t)$, and its upper bound can be found as follows:

$$\begin{aligned} W(t) &= \int \dot{W}(t)\,dt + V_0 \\ \Rightarrow \quad \sup\left[W(t)\right] &= \int\left(\ddot{W}_{\max}t + \dot{W}_{\max}\right)dt + V_0 \\ &= \tfrac{1}{2}\ddot{W}_{\max}t^2 + \dot{W}_{\max}t + V_0 \end{aligned} \tag{15}$$

The function $\sup[W(t)]$ is the upper limit of $W(t)$ at any time beyond $t = t_0$, which will never be exceeded. The maximum of $\sup[W(t)]$ therefore determines the upper bound of $V(\mathbf{x})$ itself, given the initial conditions.

The maximum value of $W(t)$ is found by solving Equation (14) for $\sup[\dot{W}(t)] = 0$ to find the time at which the turning point occurs:

$$t_{\mathrm{TP}} = -\frac{\dot{W}_{\max}}{\ddot{W}_{\max}} \tag{16}$$

Since the upper limit of $\ddot{W}(t)$ is and is negative by definition, the turning point must be a maximum and the upper limit of $W(t)$ at the turning point must be:

$$\sup\left[W(t)\right] = \tfrac{1}{2}\ddot{W}_{\max}\frac{\dot{W}_{\max}^{2}}{\ddot{W}_{\max}^{2}} - \frac{\ddot{W}_{\max}^{2}}{\ddot{W}_{\max}} + V_0 \tag{17}$$

The term $\sup[W(t)]$ represents the maximum value of $V(\mathbf{x})$ attained by any of the system state trajectories, and therefore is defined as V_{max}. Therefore:

$$V_{\max} = \frac{\dot{W}_{\max}^{2}}{2\left|\ddot{W}_{\max}\right|} + V_0 \qquad \text{where } \ddot{W}_{\max} < 0 \tag{18}$$

The V_0 term is the value of the Lyapunov function at the initial time t_0. Since V_{max} is the upper limit of $W(t)$, Equation (18) shows that if $V(\mathbf{x}) \le V_0$ at time t_0 then $V(\mathbf{x}) \le V_{\max}$ for all future time:

$$\begin{aligned} &\forall t \ge t_0 \quad V(\mathbf{x}_0) \le V_0 \quad \rightarrow \quad V(\mathbf{x}(t)) \le V_{\max} \\ &\text{where} \quad \mathbf{x}_0 = \mathbf{x}(t=0) \end{aligned}$$

The formal definition of Marginal Stability in the Lyapunov sense states that Equation (3) implies that any initial condition $\|\mathbf{x}_0\| \le r$ means that $\|\mathbf{x}(t)\| \le R$ for all future time, where $r < R$. Therefore, the following inference must be proven for marginal stability to be demonstrated:

$$\begin{aligned} &\forall t \ge t_0 \\ &\left[V(\mathbf{x}_0) \le V_0 \rightarrow V(\mathbf{x}(t)) \le V_{\max}\right] \rightarrow \left[\|\mathbf{x}_0\| \le r \rightarrow \|\mathbf{x}(t)\| \le R\right] \\ &\text{where} \quad 0 < r < R \end{aligned}$$

The proof is adapted from a proof of first order stability provided by Slotine and Li [34]:

Let r be the minimum Euclidean distance of any state $\mathbf{x}_i$ for which $V(\mathbf{x}_i) \ge V_0$. Since $V(\mathbf{x})$ exists and is positive definite, r exists and is strictly positive. Since r is the *minimum* distance value for a ball $\mathbf{B}_r$ about the origin $\mathbf{x} = \mathbf{0}$, any state for which the distance is less than or equal to r must imply that $V(\mathbf{x}_i) \le V_0$ and vice versa (i.e. biconditional implication). It is similarly true for R being the minimum Euclidean distance of any state $V(\mathbf{x}_i) \ge V_{\max}$.

Since

$$\begin{aligned} V(\mathbf{x}_0) \le V_0 \quad &\rightarrow \quad V(\mathbf{x}(t)) \le V_{\max} \\ V(\mathbf{x}_0) \le V_0 \quad &\leftrightarrow \quad \|\mathbf{x}_0\| < r \\ V(\mathbf{x}(t)) \le V_{\max} \quad &\leftrightarrow \quad \|\mathbf{x}(t)\| < R \end{aligned}$$

it follows by transitivity of implication that

$$\|\mathbf{x}_0\| \le r \rightarrow \|\mathbf{x}(t)\| \le R$$

Equation (18) proves that $V_{max} > V_0$ and again, therefore, it follows by definition that $r < R$. Hence, since r is strictly positive, it is proven that $0 < r < R$.

Q.E.D.

Appendix B. Proof of the Asymptotic Stability Properties of the Second Order Lyapunov Stability Theorem

The Second Order Lyapunov Stability Theorem can be proven to generate asymptotically stable behaviour. This can be done by proving that, at any time beyond the one at which the maximum value of V_{max} is obtained, all states satisfy the classical first-order theorem thereby allowing proof of asymptotic stability by appeal to Lyapunov's original stability proofs.

It has been proven in Appendix A that the maximum value V_{max} of the Lyapunov function attained by the system occurs at time t_{TP}, which is evaluated in Equation (16). This is the time at which the trajectory with the maximum initial value of $\dot{W}(t)$ is reduced to zero by the minimum magnitude of $\ddot{W}(t)$, and hence represents the limiting stable trajectory achievable by the system. This proof of stability is based on the determination of the value of $\dot{W}(t)$ at times beyond t_{TP}.

Proof:

Let time $t_1 = t_{\mathrm{TP}} + \delta t$ be some time after the achievement of the turning point. The value of $\dot{W}(t)$ for the limiting trajectory is defined by Equation (14), and at time t_1:

$$\begin{aligned} \sup\left[\dot{W}\left(t_1\right)\right] &= \ddot{V}_{\max} t_1 + \dot{V}_{\max} \\ &= \ddot{V}_{\max}\delta t \end{aligned} \qquad (19)$$

Since $\ddot{V}_{\max} < 0$ by definition, $\dot{W}(t)$ must be a negative-definite function for all times beyond t_{TP}, and therefore the Lyapunov function is asymptotically stable in the classical first-order sense beyond this time.

Q.E.D.

<u>NOTE</u>: for $\dot{W}(t)$ to be negative-*definite*, $\sup[\dot{W}(t)]$ must be negative even where t becomes zero. This requires that $\ddot{W}(t)$ must be defined as being negative at time $t_{T.P.}$, which requires that its constraint be defined for all states where $\dot{W}(t) \geq 0$. This leads to theorem stated in Equation (4) being the requirement for asymptotic stability, whereas the form of Equation (3), which permits $\ddot{W}(t)$ to remain undefined in states where $\dot{W}(t)$ equals zero, will only lead to marginal stability.

Appendix C. Proof of the Diminishing Activity Principle

As discussed in Section 6, the Diminishing Activity Principle states that the higher the priority of a module within the layers of a subsumption architecture, the lower the average mark-space ratio of its output activity should be. The Diminishing Activity Principle defines the condition of minimum mutual interference between behaviour modules in a system with a Colony-type subsumption architecture (with suppression or inhibition arbitration only).

The principle is derived from the following theorem proving that, in a process where one behaviour module suppresses another, the relative interference on the lower-priority module is minimised if the modules are arranged such that the one with the lower average activity takes the higher priority. If modules are then ranked according to diminishing activity, the resulting architecture will be the configuration, in which the total mutual (inter-module) interference is minimised.

Theorem: The relative interference of two behaviour modules is minimised if the module with the higher priority has the lower average activity mark-space ratio.

Proof:

On average, the activity of a behaviour module will have the general form illustrated in Figure 16 below:

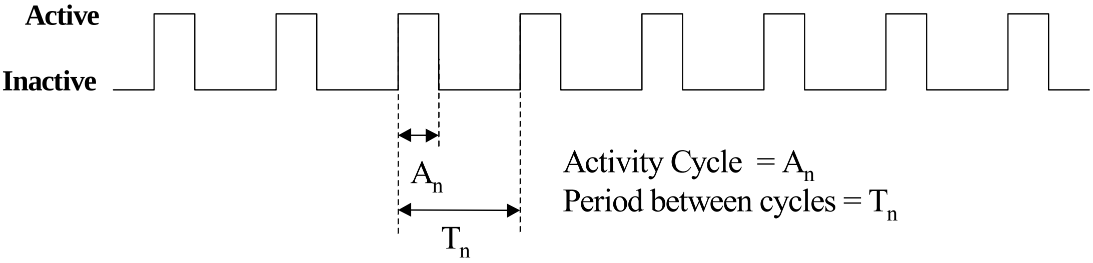


Figure 16: Activity of Behaviour Module Process *P_n*.

Each module process P_n has an average period T_n between successive bursts of activity. The average duration of activity is A_n. By definition, all A_n values are less than their respective T_n values. We define the *mark-space ratio* R_i of a process P_i as the proportion:

$$R_i = \frac{A_i}{T_i}. \tag{20}$$

Consider two Behaviour Modules P_1 and P_2 embedded within the same subsumption architecture. We define process P_2 to have a longer average period between activities than P_1. Let k be the constant of proportionality between the two: $T_2 = kT_1$. We assume the mark-space ratio R_2 of Process P_2 as being *higher than* the mark-space ratio R_1 of Process P_1.

Let the *relative interference* on a process be defined as the proportion of its activity that is lost due to suppression of its outputs by another process.

$$\text{Relative Interference} = \frac{\text{Lost Activity}}{\text{Total Activity}} \tag{21}$$

The priority arbitration scheme leading to the least relative interference to the suppressed process should be the preferred prioritisation as it will generate the minimum perturbation of the suppressed process. The relative interference between processes P_1 and P_2 is calculated as follows:

Since P_2 has the longer activity period, all calculations are referenced to the P_2 activity cycle.

If P_2 suppresses P_1:

$$\begin{aligned}\text{Total length of } P_1 \text{ activity in a cycle of } P_2 &= kA_1 = \frac{T_2}{T_1}A_1 \\ &= T_2R_1 \end{aligned} \tag{22}$$

$$\begin{aligned}\text{Total length of } P_1 \text{ activity lost due to } P_2 \text{ interference} &= kA_1 \cdot \frac{A_2}{T_2} \\ &= A_2R_1 \end{aligned} \tag{23}$$

$$\text{Relative interference to } P_1 \text{ from } P_2 = \frac{A_2R_1}{T_2R_1} = R_2 \tag{24}$$

If P_1 suppresses P_2:
Total length of P_2 activity in a cycle of $P_2 = A_2$

(by definition)

$$\text{Total length of } P_2 \text{ activity lost due to } P_1 \text{ interference} \;=\; kA_1 \cdot \frac{A_2}{T_2}$$

$$= A_2 R_1 \qquad (25)$$

$$\text{Relative interference to P from P} \;=\; \frac{A_2 R_1}{A_2} \;=\; R_1 \qquad (26)$$

Since R_2 is greater than R_1 by definition, the minimum relative interference between the two processes will be achieved if the process (P_1) with the lower average mark-space ratio of activity suppresses or inhibits the process with the higher mark-space ratio (P_2). QED.

The proof can be extended to complete subsumption architecture systems by considering the combined effect of higher layers on the modules of an arbitrary behaviour module, yielding the Diminishing Activity Property.

Appendix D. Equations of Motion and Aerodynamic Parameters of Flight Control Simulation Experiment

This Appendix presents an overview of the system model used as the basis for the experiments described in Section 7. A glossary of parameter symbols used in the equations is provided in Table 1.

The aerodynamic angles, Angle of Attack α and Sideslip Angle β, are formally defined in terms of the velocities of the aircraft along its Body Axes:

$$\alpha = \operatorname{atan}\left(\frac{w}{u}\right) \qquad\qquad \beta = \operatorname{asin}\left(\frac{v}{V}\right) \qquad (27)$$

The Body Axes system of equations is a system of twelve differential equations, organised into four groups of three. Each group is a set of equations that give the motion of the aircraft in different frames of reference, or of the motion of one frame of reference with respect to another. The equations are presented in the box on the next page.

Table 1: Glossary of Terms.

α	Angle of Attack	β	Sideslip Angle		
p	Roll Rate	q	Pitch Rate	r	Yaw Rate
I_x	Moment of Inertia in x-axis	I_y	Moment of Inertia in y-axis	I_z	Moment of Inertia in z-axis
L	Roll Moment	M	Pitch Moment	N	Yaw Moment
u	Forward airspeed (x-axis)	v	Sideslip airspeed (y-axis)	w	Downwards speed (z-axis)
X	Aerodynamic force (x-axis)	Y	Aerodynamic force (y-axis)	Z	Aerodynamic force (z-axis)
ρ	Air Density	V	Total Airspeed	S	Effective wing area
b	Wing span	$\bar{c}$	Wing mean chord length	d_p	Throttle Power Demand
d_a	Aileron Deflection Angle	d_e	Elevator Deflection Angle	d_r	Rudder Deflection Angle
ψ	Azimuth Angle	θ	Elevation Angle	ϕ	Bank Angle
x_E	Speed in Earth's x-axis	y_E	Speed in Earth's y-axis	z_E	Speed in Earth's z-axis

Moment Equations in Body Axes (assuming principal axes):

$$\dot{p} = (L + (I_y - I_z)qr)/I_x \qquad (28)$$

$$\dot{q} = (M + (I_z - I_x)pr)/I_y \qquad (29)$$

$$\dot{r} = (N + (I_x - I_y)pq)/I_z \tag{30}$$

Kinematics (aircraft attitude):

$$\dot{\psi} = (q\sin\phi + r\cos\phi)\sec\theta \tag{31}$$

$$\dot{\theta} = q\cos\phi - r\sin\phi \tag{32}$$

$$\dot{\phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \tag{33}$$

Force Equations in Body Axes:

$$\dot{u} = \frac{X}{m} - g\sin\theta - qw + rv \tag{34}$$

$$\dot{v} = \frac{Y}{m} + g\cos\theta\sin\phi - ru + pw \tag{35}$$

$$\dot{w} = \frac{Z}{m} + g\cos\theta\cos\phi - pv + qu \tag{36}$$

Kinematics (motion of aircraft in Earth-fixed axes):

$$\begin{aligned}\dot{x}_E = {} & u\cos\theta\cos\psi + v(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) \\ & + w(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)\end{aligned} \tag{37}$$

$$\begin{aligned}\dot{y}_E = {} & u\cos\theta\sin\psi + v(\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi) \\ & + w(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)\end{aligned} \tag{38}$$

$$\dot{z}_E = w\cos\phi\cos\theta + v\sin\phi\cos\theta - u\sin\theta \tag{39}$$

These equations describe translational ($\dot{u}$, $\dot{v}$ and $\dot{w}$) and the angular ($\dot{p}$, $\dot{q}$ and $\dot{r}$) accelerations in the body axes, the rates of change of azimuth, elevation, and bank angles ($\dot{\psi}$, $\dot{\theta}$ and $\dot{\phi}$) and the velocities ($\dot{x}_E$, $\dot{y}_E$ and $\dot{z}_E$) of the aircraft in the Earth Frame axes. The quantities L, M, and N are (respectively) the Roll, Pitch, and Yaw Moments generated by the control surfaces of the airframe. Control surface deflection relates to the rolling moments via a set of control derivatives, as described below:

$$L = \frac{1}{2}\rho V^2 Sb(C_{l_\beta}\beta + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r) + \frac{1}{4}\rho VSb^2(C_{l_p}p + C_{l_r}r) \tag{40}$$

$$M = \frac{1}{2}\rho V^2 S\bar{c}(C_{m_0} + C_{m_\alpha}\alpha_x + C_{m_{\delta_e}}\delta_e) + \frac{1}{4}\rho VS\bar{c}^2(C_{m_{\dot{\alpha}}}\dot{\alpha} + C_{m_q}q) \tag{41}$$

$$N = \frac{1}{2}\rho V^2 Sb\left(C_{n_\beta}\beta + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r\right) + \frac{1}{4}\rho VSb^2(C_{n_p}p + C_{n_r}r) \tag{42}$$

In these equations, δ_a, δ_e and δ_r are the deflection angles of the ailerons, elevators, and rudder respectively, ρ is air density, and S, $\bar{c}$, b are constants relating to the shape/design of the airframe, and V is the air speed (the overall magnitude of velocity, due to all its components in the x, y and z axes). The C terms are all dimensionless coefficients that characterise the performance of an airframe. It is standard practice in flight dynamics and aerodynamics to define equations in this form. Table 2 lists the parameter values that were used in the software simulations, and are in fact the parameter values for the Cessna 182 aircraft.

The value for air density ρ is the value defined for an altitude of 1500m (approximately 6000 feet) by the International Standard Atmosphere [20], a standard model for the variation of Earth's air density, pressure, and average temperature as a function of altitude. This model is in wide use in the aerospace industry.

The C_D, C_y, and C_z terms are coefficients in the equations for the aerodynamic forces generated by the aircraft in the Body Axes, which are defined by the equations below:

$$X = T(\delta_p, V) - \frac{1}{2}\rho V^2 S\left(C_{D_0} + C_{D_1}(C_{z_0} + C_{z_\alpha}\alpha_x)^2\right) \tag{43}$$

$$Y = \frac{1}{2}\rho V^2 S\left(C_{y_\beta}\beta + C_{y_{\delta_r}}\delta_r\right) \tag{44}$$

$$Z = -\frac{1}{2}\rho V^2 S\left(C_{z_0} + C_{z_\alpha}\alpha_x + C_{z_{\delta_e}}\delta_e\right) \tag{45}$$

The parameter $T(\delta_p, V)$ represents the thrust generated by the aircraft engine, while the other term in Equation (43) defines the drag generated by the aircraft.

:

Table 2: Cessna 182 Aerodynamic Parameters

b = 10.9m	S = 16.2m^2	$\bar{c}$ = 1.48m
ρ = 1.0595 kg m^{-3}	I_{xx} = 1285.0 kg m^2	I_{yy} = 1825 kg m^2
I_{zz} = 2670 kg m^2	C_{L_β} = -0.089	C_{L_p} = -0.471
C_{L_r} = 0.096 rad^{-1}	$C_{L_{\delta_a}}$ = 0.177	$C_{L_{\delta_r}}$ = 0.015
C_{M_0} = 0	C_{M_α} = -0.885	$C_{M_{\dot\alpha}}$ = -5.237
C_{M_q} = -12.434	$C_{M_{\delta_e}}$ = -1.283	C_{N_β} = 0.064
C_{N_r} = -0.096	C_{N_p} = -0.029	$C_{N_{\delta_a}}$ = -0.016
$C_{N_{\delta_r}}$ = -0.066	C_{D_0} = 0.036	C_{D_1} = 0.038
C_{z_0} = 0	C_{z_α} = 5.9	$C_{z_{\delta_e}}$ = -0.427
C_{y_β} = -0.308	$C_{y_{\delta_r}}$ = 0.187	

Biographies:

Christopher J. Harper

Chris received his BEng degreein 1989 and completed his PhD at UWE in 2004. He is a Visiting Research Fellow at the Intelligent Autonomous Systems Laboratory, where he is continuing his research into the application of behaviour-based systems to safety related applications. Dr Harper is the founder of Avian Technologies Ltd, a company which undertakes research and development into industrial applications of behaviour-based robotics, and which also performs contract/consultancy work in system development, safety assessment and certification.

Alan F.T. Winfield

In 1984, shortly after completing a PhD in Digital Communications, Alan Winfield gave up his lectureship at the University of Hull to found a company on the newly established Hull Science Park. Dr Winfield went on to establish APD Communications Ltd as one of the key UK providers of software for safety critical mobile radio systems. He left APD in 1991 to take up appointment as Associate Dean (Research) and Hewlett-Packard Professor of Electronic Engineering at UWE. Moving into the field of mobile robotics, he co-founded the Intelligent Autonomous Systems Laboratory in 1993. His work is centred on Control and Communications architectures for mobile robots. Current research has three strands: ad-hoc wireless connected robot swarms; autonomy in space robotics, and provably-stable intelligent control.