Self Managed Systems - A Control Theory Perspective

Richard Taylor, Chris Tofts
Trusted Systems Laboratory
HP Laboratories Bristol
HPL-2004-49
March 25, 2004*

E-mail: {richard.taylor, chris.tofts}@hp.com

self-management, control, distributed

Self-managed systems are essentially closed loop control systems. When designing such systems we should ensure that they do not allow fundamentally bad properties, too slow convergence, oscillation, chaotic behaviour, stuck modes; just as for any control system. How can we perform these checks in the presence of arbitrary (Turing complete) control functions? We argue that the space of control functions and compositions should be restricted to those with known 'good' properties and demonstrate such a space within cellular automata.

* Internal Accession Date Only
Approved for External Publication

Self Managed Systems - A Control Theory Perspective

Richard Taylor and Chris Tofts
Model Based Analysis Group
HP Laboratories
{richard.taylor, chris.tofts}@hp.com

Abstract

Self-managed systems are essentially closed loop control systems. When designing such systems we should ensure that they do not allow fundamentally bad properties, too slow convergence, oscillation, chaotic behaviour, stuck modes; just as for any control system. How can we perform these checks in the presence of arbitrary (Turing complete) control functions? We argue that the space of control functions and compositions should be restricted to those with known 'good' properties and demonstrate such a space within cellular automata.

1 Introduction

As systems are growing more complex their control and configuration requirements frequently exceeds the comprehension and ability of their operators. Indeed, there are circumstances, the classic being modern fighter aircraft, where the system is designed *ab initio* to be uncontrollable by a human operator. The desire to move control from the human to the system is rational. Humans have a limited response rate, often make mistakes, are expensive to maintain, and difficult to train. In fact given all of these limitations it is unclear why we have retained humans is so many control loops for so long. The main reason for maintaining humans in management (or control) situations is their flexibility. Any fixed control system will only respond in the way in which it is designed, it cannot be aware of wider goals which can often be in conflict with its current control response. In almost all complex systems there are unforeseen situations to which a closed control system will not have a designed response.

It is important to remember that self-managed systems are actually closed loop control systems. They are by definition, systems that make observations about their current state[1], have the ability to control some (rarely all) aspects of their behaviour with the aim of reaching some 'good' (or even optimal state) and then repeat. The previous statement is simply the description of closed loop control. However, with such control systems there is a clear requirement to demonstrate that the control achieves the designers intent. This is of paramount importance when the intention is to leave the system running without human intervention for any extended period[2]. The rest of this paper will present a quick review of the observations from control theory of a 'good' control system, discuss how this might be demonstrated in a complex self-managed system and conclude with practical examples based on the exploitation of known cellular automata behaviour.

[1] This obviously can include environmental inputs

[2] An extended period in this setting can be very short, for a fighter aircraft about 2 seconds

2 Control Theory Perspective

There are a well known list of properties that are essential for a control system to be considered 'good'. These are the results of centuries of engineering practice and have generally been incorporated as the result of a disaster. A non- exhaustive list will certainly include:

1. convergence on the required solution over the whole input space;
2. adequate responsiveness;
3. stability of solution on stable inputs, no unnecessary oscillations;
4. no snap over on small changes of the inputs, 'chaos'.

For small numbers of inputs and response controls this is a well understood problem, all-be it when the control system is describable as a set of coupled differential equations. However, the question arises as to what should replace this approach to deriving control system properties in a complex setting, in particular where the control system uses coupling that cannot be easily analysed using differential equations or indeed computations that cannot be described in this setting. There is a fundamental question as to whether we should employ control systems that cannot be analysed and should we restrict ourselves to mathematically analysable settings? There are further problems of potential higher order neglected factors in original model and of the consequences of approximation. For instance the Lotka-Volterra equations of population dynamics are not stable as a discrete probabilistic system and require the introduction of refuges.

There are two basic solution approaches to this problem. The first is to verify that any particular control system has the required properties by employing the appropriate mathematics. In the case of control systems whose nodes employ Turing complete levels of computation this is well known to be a difficult problem. The second is to limit the set of measure-response functions in the control system so that they are known to have the required properties in any setting. In the next section we outline how this can be achieved for cellular automata. As an example of control on a cellular automata, consider the problem of maintaining a pattern within the game of Life, given an arbitrary starting position and the ability either to set a limited number of cells, or only to set cells occasionally.

3 Example Solution

Cellular automata make fascinating abstract, as well as practical example of systems that might be considered to have 'emergent' properties. Cellular automata (CA) are discrete dynamic systems (space, time and state values), with determinism and with local interaction. A CA is a finite dimensional lattice of sites whose values are restricted to a finite set of integers. The value of any site at any time step is determined as a function of the values of neighbouring sites at the previous time step and itself.

In one simple example of Life, space is represented by a uniform mesh of 'processing' cells. Each cell within that space is connected to its eight nearest (surrounding) neighbours. Every cell executes a program that exchanges a token (dead or alive) with these eight neighbours, and then alters its own token based upon the number of 'live' neighbours that surround it. Too many neighbours and the cell 'dies' (over crowding), too few and the cell also dies (loneliness), a critical number and a cell is 'born' (i.e. change token from dead to alive).

Such a simple system, replicated across a large continuous surface gives rise to apparently complex behaviour from simple seeds, as groups of cells are born, expand and die. Regular patterns of creation and dispersal

appear to occur spontaneously, and in many systems, stable recurring patterns become established. It is the observation that recurrent, stable 'self organised' structures have emerged from apparently trivial 'rules' of computation and communication that have made CA of great interest as one model of parallel and distributed computation.

Stability through self organisation on a massive scale has excited many researchers with an interest in scalable, robust parallel systems. Indeed practical and robust implementations of load balancing and fault kernels for operating systems, as well as telecommunications switching systems have been based on simple, well understood automata designs. Observations on the rate of dispersion of system perturbations (through load or fault conditions), combined with the required behaviour of system boundaries, for example, make it possible to construct reliable statements about the upper and lower bounds of system performance and degradation.

Unfortunately, life is not as rosy as one might think. If conventional computing and communication systems are used to construct CA and their resultant behaviour examined, a simple qualitative distinction can be made between them (sometimes referred within the research community as the *class* of the automata)

1. evolve to an homogenous state
2. evolve to simple, separated periodic structures
3. generate 'chaotic' aperiodic patterns
4. generate complex patterns of localised structures

While the forward problem - the description of properties of a given CA such as reversibility, invariants and criticality are well worn with observation based analysis, the reverse problem, the development of a set of techniques that that will generate a rule or a set of rules that can reproduce quantitative observations is primitive in all but the simplest subset of systems (mainly, but not exclusively a subset of class 1 and 2 automata). The bulk of 'successful' reverse analyses make use of a combination of probabilistic search strategies with forward experimental checking. For large systems, especially those of types 3 and 4, the limitations are obvious. Validation of design through simulation and/or construction is not practicable for large (and in the real world potentially expensive in both construction costs and the financial and social implications of error) systems. Successful commercial systems that do rely on CA stability typically live in the class 1 and 2 domains, where stability may often (but not always) be predicted analytically.

4 Conclusions

Whilst it is tempting, given the 'emulate anything theorem', the presence of Turing complete systems in a closed loop control system is extremely dangerous. It is very difficult to demonstrate the absence of unsatisfactory control behaviour in systems that employ such measure-response functions. Restricted models of computation can give 'good' solutions by construction, even with distributed solutions as the example from cellular automata demonstrated. Given the cost of validating control systems that are constructed *ad hoc* from an arbitrary underlying computation framework, and the risk that such a validation may well be incomplete, should this be a domain of control? Given the need for large numbers of control systems, and the speed with which the demands on the designer vary, it is clearly imperative that we derive as wide a class as possible of compositions and control functions that are inherently 'good'; instead of relying on the skills of a human designer and the observation of successful behaviour on a limited number of simulation exercises.