Proving Newtonian Arbiters Correct, Almost Surely

by

Ian M. Mitchell

B.A.Sc., University of British Columbia, 1994

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
Master of Science
in
THE FACULTY OF GRADUATE STUDIES
(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia
October 1996
© Ian M. Mitchell, 1996
for putting up with me hogging the machine. To Brian Edmonds for writing such a slick little thesis style file. To the office staff, students, and profs of the CS department for a wonderful 26 months (of a 1 year masters degree). Very special thanks are due to David Kirkpatrick, Dinesh Pai, and Carl Seger for stepping in as second readers on a panicky last weekend.

I don't know what to say about Kaaren. Grad school's important, friends are important, but you've been the most important for the last five years. And hopefully for many more. Thank you.

Large parts of this thesis have been pulled from the paper "Proving Newtonian Arbiters Correct, almost surely" written by me and Mark, which appears in the proceedings to Designing Correct Circuits '96, held in Båstad, Sweden, Sept. 1-3, 1996.

The University of British Columbia

October 1996
Abstract

An arbiter is a circuit that grants its clients mutually exclusive access to a shared resource. The ideal arbiter cannot be built—either requests must occur synchronously, mutual exclusion is not guaranteed, or the arbiter may become hung indefinitely and fail to make a decision. This thesis explores arbiters whose failure mode is the latter; they are not live. Two specific arbiters are investigated: a “toy” arbiter, whose model was chosen for its tractability; and Seitz’s nMOS interlock, which was chosen as an example of a widely used and studied design. Both arbiters are modeled by nonlinear continuous dynamic systems, and both models are Newtonian: given an initial set of system states, the future image of that set has exactly the same dimension.

The toy is shown to be almost surely live—with probability 1, the arbiter eventually grants some pending request—while still satisfying all the other properties of an ideal arbiter. The analysis is more general than previously published results in that the inputs of the arbiter may vary while the arbiter is metastable, and conditions are given that ensure liveness even in the presence of such variations. The verification is accomplished by proof that the set of states forever trapped in the metastable region of the toy is lower dimensional than the state space of the toy as a whole. Parallels between the toy and the interlock can then be drawn.
Contents

Abstract ii

Contents iii

List of Figures vi

Acknowledgements vii

Dedication ix

1 Introduction 1
  1.1 What's so interesting? 1
  1.2 Scope and Goals 4
  1.3 Outline 5

2 Background Material 7
  2.1 Circuits and Metastability 7
    2.1.1 Mutual Exclusion and Synchronizers 7
    2.1.2 Metastability: Indecisiveness in Circuits 9
    2.1.3 The Impossibility of the Ideal Arbiter 11
  2.2 Verification and Dynamical Systems 13
5.5 Liveness, almost surely .............................................. 48
5.5.1 Liveness and metastability ....................................... 49

6 The Seitz Interlock ...................................................... 57

7 Conclusions .............................................................. 63
7.1 What has been Accomplished ....................................... 63
7.2 Suggestions for Further Research ................................. 65

Bibliography ............................................................... 68

Appendix A Details of Divergence Proof ........................... 71
List of Figures

2.1 Brockett's Ring ........................................... 18

3.1 Continuous Model ........................................ 22

3.2 Arbiter Specification ................................. 29

4.1 Ricochet Trajectory ................................... 32

4.2 α-insensitivity ......................................... 34

4.3 Phase space portraits for the Toy Arbiter .......... 38

4.4 The Toy's Output Brockett Ring ....................... 39

5.1 Regions for Liveness Proof ............................. 49

6.1 Seitz Interlock Circuit .................................. 58

6.2 Transistor Model ......................................... 59

6.3 Transistor Model Equation ............................. 59

6.4 Phase space portraits for Seitz's arbiter ............. 60
Acknowledgements

Hmmmm. Ian Mitchell writes his thesis on indecision. Who ever would have thought? Actually, the only question left is what to do next. I wonder...

Meantime, there are an awful lot of people who got me through this !@#$%&* thing (and you try cartoon swearing in $\LaTeX$). Mark Greenstreet is definitely at the head of the list. A fount of endless neat ideas, Mark took an indecisive second year masters student and found a perfectly matched topic. And since then has been providing the supervision and “Arrrgh! I’m stuck again!” answers necessary to finally finish one of these thesis thingees.

Necessary but not sufficient. I never would have gotten here, stayed here, or finished here without Mom, Dad, Eric, and the cats (the dog, well, not so much). My thanks to Fraser for reminding me that computers are more than just an arbiter with some stuff around it—they are fascinatingly capable of playing Civ II for example. To Katrina for reminding me that life is more than just a computer with some stuff around it and for always providing a perfect foil to a habitual devil’s advocate. To Kevin (and the rest of the former pit night flock) for keeping me from wasting my time at UBC studying—now I’ll be knocking on your door with a Netrunner deck, Kevin. To Piers and the gaming gang (past and present) for Sunday nights happily divorced from reality. To my office mates (Cristina, Dan, Elisa, and Sonya)
To Grandpa and Sputnik.

Wish you both could be here.
Chapter 1

Introduction

1.1 What's so interesting?

An arbiter is a circuit that grants its clients mutually exclusive access to a shared resource. In its simplest form, an arbiter has a request input and a corresponding grant output for each of its clients. When a client wants access to the resource, it signals the arbiter on its request line. The arbiter awaits request signals, and then produces a grant—only one grant may be outstanding no matter how many clients are making requests. When a client receives a grant, it may safely access the resource without conflict. Eventually, the client will finish with the resource and release its request, at which point the arbiter is free to generate a grant to any other currently requesting client.

Arbitration is such a common operation that few complex VLSI circuits are without some form of arbiter. Although construction of a failsafe arbiter in synchronous circuits (circuits with a common clock reference) is relatively easy, the push for faster and larger circuits has lead to increasing use of asynchronous (no
single clock) design techniques. And as soon as asynchronous arbiters were used, they started to fail.

As early as the 1960's, it was recognized that arbiters—and a related circuit element, synchronizers—subject to asynchronous inputs were failing occasionally. Since that time, a series of proofs have been developed to show that:

- Arbiters fail when both clients request permission to the resource at nearly the same time.
- The failure manifests itself as an inability to give digitally defined grant signals back to the clients fast enough after the requests.
- The grant signals may remain hung at an intermediate value for an unbounded interval of time. The arbiter is said to be metastable in this case.
- This time-unbounded response is a fundamental property of arbiters—any design which avoids it will fail to act like an arbiter in some other respect.

Each proof required fewer and fewer assumptions about the arbiter and/or environment; consequently, it is now clear that no ideal asynchronous arbiter may be physically constructed.

Starting with the early research attempts, arbiters were studied through mathematical models. For the simplest case of arbiter, the two client arbiter, most models take the form of bistable dynamic systems—dynamic systems which, for at least some range of parameters, have two separated asymptotically stable attractors. Associating a different client with each attractor allows a mutually exclusive mapping of the state space of the system: when the system state is near an attractor, that attractor’s client gets permission to access the resource.
Looking at phase space plots of these bistable systems shows immediately
the cause of metastability. Each attractor collects trajectories from its basin of
attraction in state space. But because stable basins of attraction must be open
sets in the state space, there must be some set of states which separates the two
basins and belongs to neither. Trajectories starting at the edge of a basin may take
arbitrarily long times to approach the attractor, thus causing metastability. And
trajectories starting in neither basin will never reach an attractor at all.

So not only is it impossible to guarantee arbitration within bounded time,
but, because of the trajectories which lie in neither basin, it is impossible to guar­
antee arbitration ever occurs. This thesis explores that possibility—the likelihood
of an arbiter never completing its task.

Intuitively, that probability is zero. A dynamic system which models a two
client arbiter can be constructed in a two dimensional space. The basins of attraction
will be two dimensional, but their separator need only be one dimensional; some
curve through the state space. For any reasonable distribution of initial states, the
probability of landing on that lower dimensional curve is then zero.

The analysis is not that simple, however. The properties of an arbiter demand
that the topology of the state space change depending on the inputs from the clients.
For example, if only one client is requesting permission to use the resource, then
there should only be one attractor in the state space (corresponding to granting
that client's request). But if the other client then also makes a request, fairness
demands that a second, corresponding attractor appears in the state space. If these
large changes to the inputs can produce such large changes in the location (or
existence) of attractors, continuity demands that small changes in the inputs must
also change the location of attractors. Moving an attractor changes the shape of
the basin of attraction, which in turn influences the shape of the curve of trapped metastable states—those states which lie on trajectories never leading to a completed arbitration.

If the clients can change the shape of the set of trapped trajectories by varying their input signals, a question then arises: could a client waiting for permission to the resource whose request signal varied slightly over time, although it definitely always remained "raised", accidentally create a set of trapped states whose dimension was equal to the dimension of the system; a set whose probability of occurring would be non-zero?

Note that a probability distribution over initial states is used in the analysis that follows, but given an initial state, the systems studied evolve deterministically. Noise is not considered in this thesis. Also, the dynamic systems used to model the arbiters are Newtonian in the sense that the dimension of subsets of the state space is preserved over time. Therefore, if the initial state space is two dimensional, the reachable state space will likewise be two dimensional.

1.2 Scope and Goals

This thesis presents a detailed examination of a simple "toy" dynamic system which behaves as an arbiter. A set of discrete properties is described which exemplifies an ideal arbiter, and the toy can be shown to satisfy all properties except liveness—it may fail to finish arbitration even after an infinite time.

The major new contribution of this thesis is to show that, for most clients, the set of trapped trajectories is guaranteed to be of lower dimension than the state space of the arbiter, and thus arbitration eventually occurs "almost surely" (i.e. with probability one). The group of clients excluded from this statement are feedback
controllers carefully designed to trap the arbiter; clients which would serve no useful purpose and thus would be highly unlikely to occur by accident in actual designs.

In addition to the concrete objective of proving the arbiter almost surely correct, the research described in this thesis has worked as a testbed for techniques whose goal is the formal verification of continuous dynamic systems models against discrete specifications. Tools such as

- Workable abstraction mappings which map states or sets of states from the continuous model into the discrete domain of the specification and back again.
- A discrete temporal logic defined over a continuous state space, so that adherence to the discrete specification can be verified.

were refined during the work with the toy, and this thesis can be taken as an example of their use.

The analysis of the toy, by itself, is a somewhat esoteric pursuit. The eventual goal of this research is the study of real VLSI arbiters, and lessons learned while examining the toy can hopefully be applied to such circuits. As an example, some parallels between the toy and a transistor level model of the Seitz interlock circuit (a real VLSI arbiter) are drawn in chapter 6.

1.3 Outline

This thesis presents the formal verification that the continuous dynamic system model of an arbiter almost surely implements a discrete specification of the properties of an ideal arbiter.

- Chapter 2 reviews related work. First, previous analysis of asynchronous arbiters is presented, with particular attention to the theorems proving the im-
possibility of constructing an ideal arbiter. Focus is then shifted to the other important body of work—the issues surrounding verification of continuous dynamic system models against discrete specifications.

- Chapter 3 provides the mathematical preliminaries necessary for the verification. Separate sections describe the types of dynamic systems studied, the formal temporal logic used for verification, the abstraction mappings between the continuous and discrete domains, and a discrete specification of an ideal arbiter.

- Chapter 4 contains the dynamic system model. The continuous implementation of the arbiter's clients in first described in very general terms, including specification of those clients excluded from consideration (the feedback controllers). The toy's differential equations are introduced, and a careful abstraction mapping is given which will allow its verification.

- Chapter 5 is the actual verification of the arbiter. The toy is shown to satisfy all properties of the ideal arbiter but liveness, and satisfy liveness almost surely.

- Chapter 6 presents the interlock circuit, a description of the methods used to turn it into a dynamic system, and an analysis of the similarities between the interlock's system and the toy's.

A chapter of conclusions and suggestions for further research completes the core of the thesis, with an appendix containing proof details following.
Chapter 2

Background Material

2.1 Circuits and Metastability

2.1.1 Mutual Exclusion and Synchronizers

Mutual exclusion, or mutex, is a fundamental concept in computer science. Granting solitary access to a resource to just one among several competing clients is a requirement of almost all complex circuits. The concept is simple—whoever asks for the resource first is granted access, and retains sole right to access for as long as is necessary. This first in, first out behaviour only needs to be followed loosely. If several requests for access arrive nearly simultaneously and no right has been given, one of the requests must be honoured (it does not matter which). Only when the chosen client releases the right of access may another client be granted access. A circuit which can enforce mutual exclusion is called an arbiter.

When clients share a common clock reference, such as in a synchronous circuit, many solutions to the mutex problem exist. In every case, advantage is taken of the fact that in a purely synchronous world, time advances in discrete steps (the
clock edges). Signal levels are only allowed to change between time steps, and so signals at each step also lie in a discrete set of values—in digital circuits the set variously called {true, false}, {high, low} or {1, 0}.

Without a common clock, in asynchronous systems, no such simple solution exists. Time must be considered a continuous variable in these systems, and so, in a theoretical sense, determining which client’s request arrived first would require infinite precision and thus a physically unrealizable circuit. Even arbitration between two separate synchronous systems will not work—the signals passing between the systems can no longer be assumed to lie in a discrete set. For example, consider a signal in one system changing value between two clock edges. Within that system, the signal will start at a discrete value on one clock edge, and finish with another discrete value at the next clock edge. But because the clocks of the two systems are unrelated, if the second system is given access to the changing signal, its clock edge may arrive at any time during the change. To the second system, the signal can take on any of a full continuous range of values.

The obvious solution is to assign, based on the continuous value of the incoming signal, some discrete value whenever the second system’s clock advances. Such a “synchronizer” element ensures that input signals have discrete values and would allow separate asynchronous circuits to submit requests to a synchronous mutual exclusion device, thus solving the asynchronous mutual exclusion problem. Unfortunately, a synchronizer must operate by determining where a continuous input value lies relative to a threshold, and assigning its output appropriately. But such determination would again require infinite precision and is thus physically unrealizable.

There is, however, a deep connection between the problems of synchronization and mutual exclusion.
2.1.2 Metastability: Indecisiveness in Circuits

As early as the 1960’s it was recognized that synchronizers and arbiters were occasionally failing due to a phenomenon known as metastability. The outputs of these systems, although designed to be stable at one of the discrete digital logic levels at each time step, could become hung at intermediate levels. This inconclusive output wasn’t stable—it would settle to a discrete level after an indeterminant period of time—and hence was called metastability. Failures occurred when the metastability lasted longer than a clock cycle, the intermediate output value escaped into the rest of the circuit, and was interpreted in contradictory ways by different parts of the circuit.

The study of asynchronous arbitration and synchronization, even in its early days, proceeded with sporadic results, and a great number of different researchers have contributed to the literature over the years.

More than two decades ago, Chaney and Molnar [CM73] were able to stimulate some common synchronization circuits repeatably enough to obtain photographic evidence of metastability on an oscilloscope. At that time it was known that synchronizers and arbiters could not be guaranteed to produce an appropriate digital output signal at any fixed time after receiving an asynchronous input signal.

Two years later, Hurtado [Hur75] tackled the case of bistable dynamical systems—systems of ordinary differential equations (ODEs) in which most trajectories are asymptotically attracted to one of two equilibrium locations in the phase space. After associating each of these “attractors” with a digital logic level, the system can be considered to output the appropriate logic level when its state is in the neighborhood of an attractor. Both synchronizers and arbiters can be modeled by bistable systems. Hurtado showed by continuity arguments on the ODEs that
there must exist a set of states which approaches neither attractor—the so-called region of indecisiveness. Furthermore, Hurtado showed that there must exist valid inputs which will take the system state from the neighbourhood of an attractor into this region.

Due to the strength of assumptions that Hurtado made about his systems, however, models of real circuits could not be expected to fulfill them. It was not until Marino [Mar81] extended Hurtado’s arguments that the theoretical metastability results could definitely be said to apply to real circuits. Unlike Hurtado, Marino does not require inputs to remain fixed. Also, Hurtado’s region of indecisiveness is a lower dimensional set in the phase space of the bistable circuit (and thus an unlikely set for the system state to get into). Marino’s important extension showed that there exist states near the region of indecisiveness which—even though they move toward an attractor over time—would fail to get close enough to either attractor to output a logically defined value in any fixed bounded time interval. The probability of metastability was thus nonzero for any bounded time. Marino argues convincingly that his assumptions are weak enough to include models of all real digital systems.

Several years later, Chapiro [Cha84] took another step in analyzing metastability. Although the bulk of his doctoral thesis was concerned with the design of high-performance asynchronous systems, Chapiro proved an important theorem related to the mutual exclusion and synchronization problems. In circuit terms, Marino’s result showed that sequential bistable machines could not avoid metastability. Chapiro applied his result to show that no machine, combinational or sequential, could give a discrete output based on the value of a continuous input in bounded time.

While the systems they studied were different—Hurtado’s and Marino’s bi-
stable dynamical systems versus Chapiro's general asynchronous digital systems—the techniques used were very similar. First, an exacting statement of the properties of the particular system was made. Second, a continuity argument was described which showed the properties to be contradictory and thus impossible to fulfill. In all, a very successful paradigm. But, although all these results were applicable to the mutex problem, none had directly addressed it. By using this two-step method on an arbiter, even more powerful results were possible.

2.1.3 The Impossibility of the Ideal Arbiter

The specific case of the arbiter, however, waited until Mendler and Stroup's relatively recent paper [MS93]. In it, they work with a fairly general example of a two client arbiter. The broader class of continuous automata instead of bistable dynamical systems is used as the model of physical devices.

The arbiter has an input and output for each of its clients—respectively called the request and grant lines. Communication follows a four-phase handshake

- In the quiescent state, both request and grant are low. When a client wants to use the resource, it raises its request (so this phase is called the request phase); however, it may not do so until its previous handshake sequence has been completed: a client may not raise its request until its grant is low.

- The arbiter is a passive element—it will not attempt any signaling on its outputs until one of its clients requests the resource. Once a request has arrived, the arbiter should issue a grant (arbitration phase): the arbiter may not raise a grant until the corresponding request is high.

- Once the grant is received, the client may use the resource (transmission
phase). However, this protocol requires persistent clients: a client may not lower its request until it has been acknowledged by a raised grant.

- Clients signal completion by lowering their request (release phase). However, mutual exclusion dictates that further grants may not be issued until the client has finished and signaled completion: the arbiter may not lower a grant until the corresponding request is low.

Note that the client must enforce the first and third rules, while the arbiter must enforce the second and fourth.

If the client follows its part of the protocol, then Mendler and Stroup identify five attributes of an ideal arbiter (stated in terms of a discrete abstraction of the continuous system):

- The arbiter must allow requests at any time—in particular, concurrent requests are allowed (called the closure property).

- The arbiter must never allow both grants to be outstanding at one time (mutual exclusion).

- When a client withdraws its request, the arbiter must withdraw the grant within a fixed time interval (reset).

- Once a grant is given, it may not be withdrawn until the corresponding request has been withdrawn. Likewise, the arbiter cannot give a grant until a corresponding request has occurred (handshake).

- If at least one request is outstanding, eventually one request will be granted (liveness).
Of particular interest is the final property, liveness. Unlike previous proofs, the “eventually” statement places no time bound on the response of the arbiter.

The authors formally define each of these properties, and then prove that no arbiter which fulfills closure, reset, handshake, and liveness can guarantee mutual exclusion. Once again, a continuity argument forms the foundation of the proof. The new contribution of the research is the proof (without any resort to dynamical systems theory) that even a time unbounded arbiter may fail to perform correctly.

Mendler and Stroup also claim, without proof, that an arbiter could be constructed which fulfills any four of the five ideal properties—a claim which is in most cases trivially obvious. It is important none-the-less; this thesis studies arbiters which satisfy closure, reset, handshake and mutual exclusion, but which fail, however improbably, to satisfy even the weak liveness of Mendler and Stroup.

2.2 Verification and Dynamical Systems

Modern circuit design suffers from the competing demands of producing faster circuits in less time. In an attempt to avoid costly mistakes in production systems, hardware verification has become a hot topic.

Verifying the operation of any system usually involves determining whether an instantiation of the system fulfills the properties of some high-level specification. In digital computer systems, such a specification is given in discrete logical terms. A discrete model of the actual system is then created, and the model is checked against the specification.

There are, however, circuit behaviours not covered by a discrete view of the world. Signal rise and fall times cannot be tracked in discrete models; consequently, those models are poorly suited for asynchronous design. And some circuit
behaviours—such as the inescapable metastabilities described above—have no discrete analogue. The first step in verifying such circuits is then determining an appropriate continuous modeling environment that supports all these additional behaviours. And because designers will continue to give specifications of digital circuits in discrete terms, the second step of verification will have to determine whether a continuous model implements a discrete specification.

2.2.1 Dynamical Systems Theory

Fortunately, nonlinear dynamical systems theory may be able to solve both problems at once. A dynamic system, as used in this paper, is a system modeled by ordinary differential equations. Instead of attempting analytic or numeric solutions to the ODEs, dynamic systems theory studies the qualitative behaviour of groups of solutions. General types of behaviour—such as shrinking to a point, growing from a point, periodic orbits or chaos—can be identified. In particular, the long term evolution of the system can be determined.

The ability of ODEs to model metastable circuit behaviour was the keystone of Hurtado's thesis [Hur75] as described above. The use of dynamic systems to verify circuits is much more recent, however. Black [Bla94] gives a good description of most of the dynamic systems theory used in the research presented here.

The dynamic systems model of a circuit consists of a current system state and a set of ODEs describing the evolution of that state. Inputs to the circuit become parameters of the ODE set. The variables of the system create a phase space, in which the current system state is a point. As time is allowed to progress, the state will travel through a set of points, forming a curve in the phase space. This curve is called a trajectory of the system. Dynamic systems theory can describe the general
behaviour of trajectories based on topological features of the phase space.

For the systems studied here, the only topological features of importance are fixed points—equilibrium positions in which the system state does not change. Based on the behaviour of nearby trajectories as time increases, fixed points can be classified into three groups: if the trajectories converge to the fixed point, it is an attractor; if trajectories diverge, it is a repellor; and if trajectories approach in some directions but recede in others, it is a saddle point. Trajectories may also loop back onto themselves, forming periodic orbits, or behave chaotically; however, neither of the systems studied here exhibit these types of behaviour.

Although certainly not the only way to study dynamic systems, both Hurtado and Black make use of the extensive literature describing the behaviour of ODE systems for fixed values of their parameters. This so-called static approach, which will be adopted for most of the work here, looks at how trajectories will evolve if the parameters—the circuit inputs—remain at constant values.

Of particular interest are the values of parameters, called critical values, where the qualitative behaviour of the system changes. For example, fixed points may move about as the parameters of the system are varied. But the number and relative locations of the fixed points remain constant unless the system passes through a critical value; here new fixed points may be created, old points may disappear, or more complicated structures may develop. This large change in the behaviour of a system at a critical value is called a bifurcation.

### 2.2.2 Relating the Continuous to the Discrete

ODEs describing the behaviour of VLSI circuits to a fair degree of accuracy can be written, as SPICE [Nag75] demonstrates. Having established those terms and
techniques from dynamical systems theory that are needed to study these ODE models, the problem of linking the continuous system with a discrete behaviour remains unresolved.

Hurtado’s answer, in keeping with his static analysis, was to associate distinct fixed point attractors with digital logic levels. The set of trajectories which approach an attractor is called its basin of attraction. If every trajectory in a circuit’s dynamic system was in the basin of attraction of an attractor, then the system state always remains bounded—a good property for any model of a physical object. And if a neighbourhood of every attractor is mapped to a discrete logic level, then every trajectory will eventually have a logic level associated with it (note that neighbourhoods must be used because no trajectory other than the fixed point itself will ever reach an attractor).

As an example of this method, consider the systems from Hurtado’s thesis. Their purpose was to model bistable circuits—circuits with two stable configurations for some constant set of inputs—such as arbiters and synchronizers. The obvious dynamic system to use was one with two separate attractors. Similar systems will be used in this paper to model arbiters.

Hurtado was able to show the impossibility of a bistable system in which the entire phase space was in one basin of attraction or the other. Some set of states must lie in neither basin, and thus approach neither attractor. Furthermore, states near this set will take extremely long times to approach their appropriate attractor. It is these states, which slowly or never approach an attractor, which are the cause of metastability.
2.2.3 Inputs that Change

For a fixed set of inputs and a mapping of attractors to logic levels, static analysis can use the basins of attraction in a dynamic system to determine which initial states will end up outputting which logic levels. However, a designer of real systems may object to having a circuit's inputs forever fixed.

Fortunately, the static analysis of dynamic systems is not as restrictive as might be expected from its name. Although attractors and their basins may move or change shape slightly, the overall behaviour of the system will remain qualitatively similar under varying inputs as long as those inputs do not approach critical values. It is then possible to choose the logically mapped neighbourhoods of attractors sufficiently large enough to include the wanderings of the attractor itself and small enough to avoid overlapping any other basins. So with inputs restricted to ranges that avoid critical values, the discrete behaviour of the system will remain the same for most trajectories even with changing inputs.

There is one remaining dilemma—system designers really do want the discrete behaviour of the system to depend on its inputs, at least when its inputs change significantly (for example, from one logic level to another). In order for the system to change its qualitative behaviour, the inputs will have to pass through critical values. But analysis of the system near these critical parameter values is difficult, because even the qualitative behaviour is no longer stable.

It could be considered fortuitous, or inevitable, that system designers have found a solution to this problem when discrete binary logic could not. In real circuit design, the logic levels are separated by an interval of voltage with an undefined logic value, and signals are expected to move through this range fairly quickly. Note that this separation is a design abstraction—real circuits will always produce outputs, but
these outputs are rarely expected to be coherent when the inputs are in the undefined range. So an obvious analogy for dynamic systems would place the critical values of parameters in the undefined input range. Consequently, for logically defined input values (which could be intervals) the system would be far from critical parameter values and thus easy to analyze. And yet the behaviour of the system would depend upon its inputs.

2.2.4 Specifying the Inputs

Dynamic systems theory can be used to model and verify circuits, so long as inputs do not remain too near critical parameter values. In real circuits, the transition of signals through the undefined range is usually controlled by monotonicity or minimum and maximum rise and fall time restrictions. Brockett [Bro89] proposed an elegant way to define such restrictions on input signals. It also serves as a method to map continuous inputs to discrete values.

For a signal $v$, Brockett's ring is actually an annulus in $v \times dv/dt$ space (see figure 2.1). For each value of $v$, it gives ranges on acceptable rates of change in $v$.

![The Annulus](image)

**Figure 2.1: Brockett's Ring**
When the signal lies in the region on the far left, denoted $B_1$, its derivative may be positive or negative; so the signal may fluctuate up and down, and remain in $B_1$ for as long as it likes. Signals leaving $B_1$ may only exit into the upper region adjacent to $B_1$ (denoted $B_2$)—exiting into the lower adjacent region ($B_4$) would require a positive derivative and signals in $B_4$ may only have negative derivative. Once in $B_2$, the signal must maintain a positive derivative and thus travel monotonically to the far right region ($B_3$), where it may once again remain for as long as it likes. Leaving $B_3$ may only be accomplished through $B_4$ which, complementary to $B_2$, requires a monotonic transition down to $B_1$.

Because the signal can remain for arbitrary periods in the left and right regions ($B_1$ and $B_3$), these are obvious choices for mapping to logic levels. The upper and lower regions ($B_2$ and $B_4$) will then represent monotonic transitions between the discrete logic levels. The outer and inner rings of the annulus in $B_2$ and $B_4$ restrict the minimum and maximum times the signal may spend in transition. By placing critical values of parameters in the intermediate range represented by $B_2$ and $B_4$, input signals will be forced to move through bifurcations with non-zero speed and remain away from the bifurcation when logically defined. Static analysis and verification of the dynamic system can then proceed with logically defined inputs.

### 2.2.5 Other ODE Approaches to Verification

The study of ODE models of circuits using static dynamic systems theory is not the only technique to use continuous models to try and verify digital circuits. SPICE models have been used for decades in trial and error verification of circuits—try a signal waveform, see if the circuit fails; repeat. That particular method, although popular, has obvious shortcomings: it cannot handle the infinite variability of signals.
possible in a continuous environment.

Kurshan and McMillan [KM91] divided the phase space of an arbiter model into boxes, and then used numerical integration to find a next state relation for each rectangle. The adherence of this relation to an arbiter specification was then checked using a traditional discrete model checker. Although the technique did take account of continuous behaviours of the ODEs that would not be exhibited in a discrete model, the analysis fell short in the case of an arbiter for two reasons: results depended heavily on the choice of rectangle size and time step used in the integration phase, and no analysis of the arbiter's metastable behaviour was attempted. A significant contribution was that the verification did not depend on the inputs remaining fixed—only that they lay in some reasonable range.

Greenstreet [Gre96] has verified a toggle element using dynamic system theory. The toggle's ODE system, unlike that of the arbiters studied below, does not have nicely separated attractors and, in fact, may contain attractive periodic orbits. It is therefore not amenable to the static analysis style described above, and other techniques, numerical in nature, have had to be used.

Thomas Henzinger and a group at Cornell have developed a tool, Hytech, for automatic analysis and verification of linear dynamic systems models in the field of hybrid control systems. Henzinger and Ho [HH95] have recently been working on ways of transforming nonlinear systems so that Hytech can be used in their verification. Those techniques are similar to the ones used below to verify that the arbiter models fulfill most of the properties of an ideal arbiter. Hytech would, however, be unable to prove that mutual exclusion is fulfilled almost surely. On the other hand, further work on Hytech should be carefully observed—its automatic verification abilities would surely be a boon should new arbiter models be investigated.
Verifying that a continuous dynamic system implements a discrete specification requires four ingredients: the dynamic system, the discrete specification, an abstraction mapping which takes states from the continuous domain to the discrete and back again, and a temporal logic defined over continuous dynamic systems which—with the abstraction mapping—allows proof of discrete assertions.

This chapter mathematically defines each of these components. Dynamic systems are only dealt with in a general sense; the specific dynamic model of the arbiter will be given in the next chapter. The same holds true of abstraction mappings, since so much depends on the model in question. Because only arbiters are dealt with in this thesis, the discrete specification will be given in full detail. Finally, a brief overview of the temporal logic that is used in the verification will be given.

3.1 Continuous Models

A continuous model is a tuple, \((F, \Omega, P)\) where \(F\) is a derivative function on some \(d\) dimensional subspace of \(\mathbb{R}^d\) (the system state space), and \((\Omega, P)\) is a probability
space. $F$ must be $C^1$ (i.e. $\partial F(x)/\partial x$ is continuous). Furthermore, $\Omega$ is assumed to be the Borel $\Sigma$-algebra over $U$ (the initial state set) with $U \subseteq \mathbb{R}^d$, and $\dim(U) = d$. A continuous model is a closed system (no inputs) modeling both the circuit of interest and its environment (see figure 3.1).

![Figure 3.1: Continuous Model—Arbiter & Environment with no inputs](image)

Autonomous (time independent), non-linear, ordinary differential equations are used to model circuits. A circuit with $m$ inputs and $n$ outputs is modeled by a function $f : \mathbb{R}^{m+n} \rightarrow \mathbb{R}^n$ that gives the time derivative of each output as a function of the current values of the inputs and outputs. In this model, circuit behaviour will only depend on its current state and inputs; consequently, autonomous ODEs are sufficient for modeling.

An environment for the circuit has $n$ inputs corresponding to the outputs of the circuit, $m$ outputs corresponding to the inputs of the circuit, and $h$ internal variables. The behaviour of the environment is given by a function (mapping $\mathbb{R}^{h+m+n}$ to $\mathbb{R}^{h+m}$) that gives the time derivatives of each output and internal variable of the environment. Let $d = h + m + n$ denote the total dimension of the circuit with its environment. A state of this system corresponds to a point in $\mathbb{R}^d$. $F$ denotes the
time derivative function for the system state, \( F : \mathbb{R}^d \mapsto \mathbb{R}^d \).

By modeling the circuit and environment together, a closed system is obtained. A trajectory \( \varphi(x_0, t) \) is the curve (parameterized by initial point and time) through \( \mathbb{R}^d \) representing the evolution of the entire system starting at state \( x_0 \) at time \( t = 0 \): \( \varphi(x_0, 0) = x_0 \). The path of a trajectory is defined by the ODE

\[
\forall \tau \in \mathbb{R}^+, \frac{\partial \varphi(x_0, \tau)}{\partial t} = F(\varphi(x_0, \tau))
\]

Because \( F \) is autonomous and has no inputs, a trajectory is completely determined by its initial state. For some fixed time \( \tau \),

\[
\varphi(\cdot, \tau) : \mathbb{R}^d \mapsto \mathbb{R}^d
\]

is a function which maps states forward \( \tau \) units in time, and which may be inverted by \( \varphi(\cdot, -\tau) \). The dual function \( \varphi(x, \cdot) \) denotes an entire trajectory (extending forwards and backwards in time) which passes through state \( x \) at time 0. Because \( F \) is \( C^1 \) in \( \mathbb{R}^d \), \( \varphi(\cdot, \tau) \) is a one-to-one mapping and is continuous in its unevaluated argument (see [HS74], chapter 8). Furthermore, if a trajectory \( \varphi(x, \cdot) \) starts in some bounded region \( W \), it will remain defined for all points in \( W \)—trajectories must either remain defined for all time or escape to infinity. As a consequence of all these conditions,

\[
\varphi(\cdot, \tau) \text{ preserves the dimensionality of subsets of } \mathbb{R}^d \tag{3.1}
\]

(see [HW41], theorem VI.7). The word "Newtonian" in the thesis title corresponds to this condition on the dynamic systems studied here.

As mentioned above, \( (\Omega, P) \) is a probability space: an initial state \( x_0 \) corresponds to an event in \( \Omega \). \( P \) must be smooth in the sense that

\[
(Q \subset \Omega) \land (\dim(Q) \leq d - 1) \implies P(Q) = 0
\]

23
This condition on $P$ rules out overly determined initial states where, for example, the value of some variable is exactly specified.

For $B \subset \Omega$ and $\tau \in \mathbb{R}^+$, $\varphi(B, -\tau) \cap U$, denotes the set of initial points whose trajectories reach a point in $B$ after exactly $\tau$ time units. By the observations in the previous paragraph,

$$P(\varphi(B, -\tau)) > 0 \implies \dim(B) > d - 1 \quad (3.2)$$

This feature of the models under discussion is central to proofs that discrete properties hold "almost surely."

3.2 Abstraction Functions

The previous section presented a framework for continuous models based on systems of non-linear differential equations. The objective of the thesis is to show that such models correctly implement discrete specifications. This verification task requires an abstraction function to map continuous trajectories to discrete sequences, which can then be checked against the specification. When the discrete state space is finite, the abstraction function can be represented by an ensemble of state predicates.

The simplest abstraction function that will be used in this thesis assigns a discrete value based on the level of a continuous variable. For example, the mapping

$$(v \geq 0) \iff v$$

states that the discrete signal $v$ is true if the continuous variable $v$ is non-negative. In absence of any further specification, $v$ can be taken to be false otherwise (i.e. when $v$ is negative).
A slightly more complex abstraction may leave the discrete variable undefined for certain values of the continuous variable

\[(v \geq 1) \implies v\]
\[(v \leq 0) \implies \neg v\]

A continuous signal level in the undefined region could be interpreted differently by different circuit elements—safe verification demands that a worst case interpretation be adopted and analyzed.

As mentioned in section 2.2.4, a more complex but very useful abstraction mapping is the Brockett ring. For a continuous signal \(v\), \(Brockett(v)\) states that it obeys a Brockett ring

\[
Brockett(v) \equiv \left( \frac{v^2}{v_{min}^2} + \frac{\dot{v}^2}{\dot{v}_{min}^2} \geq 1 \right) \land \left( \frac{v^2}{v_{max}^2} + \frac{\dot{v}^2}{\dot{v}_{max}^2} \leq 1 \right)
\]

(3.3)

The constants \(v_{min}, \dot{v}_{min}, v_{max}, \) and \(\dot{v}_{max}\) define the size of the two ellipses which bound the Brockett annulus (see figure 2.1).

The annulus can be easily divided into four regions, which can be thought of as

\[
B_1(v) = v \in B_1 = v \text{ low}
\]
\[
B_2(v) = v \in B_2 = v \text{ rising}
\]
\[
B_3(v) = v \in B_3 = v \text{ high}
\]
\[
B_4(v) = v \in B_4 = v \text{ falling}
\]

Where, for some variable \(v\), \(\dot{v} \equiv \partial v/\partial t\). When using a Brockett ring for abstraction mapping, the discrete signal is assigned a value based on the region in which the continuous variable lies. For example

\[
B_3(v) \implies v
\]

states that \(v\) is true when \(v\) is in \(B_3\), or \(v\) high.
3.3 Temporal Formulas on Probability Spaces

Given the negative results for arbiters mentioned in section 2.1, the objective is to show that an arbiter satisfies its specification, including liveness, almost surely. The specification for a discrete model is naturally written as a temporal logic formula [Pnu81, RRH93], which motivates introducing a temporal logic defined relative to a probability space.

A state predicate gives some condition about the initial state of the system—for example $B_3(x)$ is a state predicate claiming that variable $x$ is in the high region of its Brockett ring at time 0. State predicate $Q$ holds for trajectory $\varphi(x, \cdot)$ if $Q$ is satisfied by the initial state of that trajectory

$$\text{hold}(Q, \varphi(x, \cdot)) = Q \text{ is satisfied by } \varphi(x, 0)$$

A state predicate is the simplest type of temporal formula.

Let $G$ and $H$ be temporal formulas. The constructs $\neg G$, $G \lor H$, $G \land H$, and $G \implies H$ are also temporal formulas and have the obvious interpretations. The formula $\square G$ (always $G$) holds for $\varphi(x, \cdot)$ if $G$ holds for all suffixes of $\varphi(x, \cdot)$; likewise, $\Diamond G$ (eventually $G$) holds if $\square \neg G$ does not hold (recall that a trajectory is completely determined by its initial state). The formula $G \rightarrow H$ denotes a weak until:

$$\text{hold}(G \rightarrow H, \varphi(x, \cdot)) = \text{hold}(\square G, \varphi(x, \cdot))$$
$$\lor \text{hold}(G, \varphi(x, \cdot)) \implies (\exists \tau \in \mathbb{R}^+. \text{hold}(G, \varphi(x, \rho), \cdot))) \land \text{hold}(H, \varphi(x, \tau), \cdot)))$$

26
Note that \(\rightarrow\) (until) is different from the standard logical \(\implies\) (implies). Occasionally the shorthand

\[(G_1 \rightarrow G_2 \rightarrow G_3) \equiv (G_1 \rightarrow G_2) \land (G_2 \rightarrow G_3)\]

will be used.

The formula \(G \sim H\) is an abbreviation for \(G \implies \Diamond H\). For \(\eta \in \mathbb{R}^+\), the formula \(G \sim H\) holds if \(H\) holds within \(\eta\) time units of \(G\) holding:

\[\text{hold}(G \sim H, \varphi(x, \cdot)) = \text{hold}(G, \varphi(x, 0)) \implies \exists \tau \in [0, \eta). \text{hold}(H, \varphi(\varphi(x, \tau), \cdot))\]

A formula is valid if it holds for all trajectories starting in \(\Omega\), and it is valid almost surely if it holds with probability 1:

\[
\begin{align*}
\text{valid}(G) &= \forall \omega \in \Omega. \text{hold}(G, \omega) \\
\text{as}(G) &= \exists A \subseteq \Omega. (\forall \omega \in A. \text{hold}(G, \omega)) \land (P(A) = 1)
\end{align*}
\]

### 3.4 A Discrete Specification

In the discrete specification of a system, the only signals of importance are the interface between that system and its environment—in the case of an arbiter, the request and grant lines. Signals internal to the arbiter or environment can be ignored by the specification.

Only arbiters with two clients are examined in this thesis (extensions are discussed in section 7.2). The interface between the arbiter and the first client consists of two boolean valued signals: a request, \(r_1\); and a grant, \(g_1\). Likewise, signals \(r_2\) and \(g_2\) are the interface between the arbiter and the second client.

The allowed behaviour of the clients is simple to specify. The communication between the clients and the arbiter follows a four-phase handshake: a client may
only raise a request when its grant is false, and it may only lower a request when
its grant is true. This protocol is formalized by the temporal logic formula
\[ \bigwedge_{i=1}^{2} (\neg r_i \rightarrow \neg g_i) \land (r_i \rightarrow g_i) \]  
(C_0)

Mendler and Stroup [MS93] have identified five attributes that capture the
intuitive sense of an ideal arbiter:

1. **Closure**: No assumptions are made about the relative arrival times of the
   request signals. In particular, it is not allowed to specify that there is some
   minimum time separation between requests from different clients.

2. **Mutual Exclusion**: Both grants cannot be outstanding at the same time.

3. **Reset**: If a request has been withdrawn, the corresponding grant will be with­
   drawn after at most \( r \) time units.

4. **Handshake**: If a grant has been given, it will not be withdrawn until the
   corresponding request has been withdrawn. Furthermore, a grant cannot be
   given unless the corresponding request has occurred.

5. **Liveness**: If at least one request is outstanding, eventually one request will be
   granted.

As noted by Mendler and Stroup, no Newtonian arbiter can satisfy all five of these
requirements. Therefore, the fifth requirement is weakened to:

5'. **Liveness (almost surely)**: If at least one request is outstanding, eventually
   one request will be granted almost surely.

Properties 2–5 are formalized in figure 3.2. The closure property is satisfied by not
stating any assumptions about the relative arrival times of the request signals. A
verified arbiter will satisfy: valid \((G_2 \land G_3 \land G_4) \land as(G_5)\).
\begin{align*}
G_2 & \equiv \Box(\neg(g_1 \land g_2)) \quad \text{(* Mutual Exclusion *)} \\
G_3 & \equiv (\bigwedge_{i=1}^{2} \Box(\neg r_i \lor \neg g_i)) \quad \text{(* Reset *)} \\
G_4 & \equiv \bigwedge_{i=1}^{2} \Box(g_i \rightarrow \neg r_i) \land (\bigwedge_{i=1}^{2} \Box(\neg g_i \rightarrow r_i)) \quad \text{(* Handshake *)} \\
G_5 & \equiv \bigwedge_{i=1}^{2} \Box((r_1 \lor r_2) \rightarrow ((r_1 \land g_1) \lor (r_2 \land g_2))) \quad \text{(* Liveness *)}
\end{align*}

Figure 3.2: Arbiter Specification \((i \in \{1, 2\})\)
4.1 The Clients

In the discrete domain, the arbiter client need only fulfill the protocol requirement ($C_0$). Although it would be nice to keep the specification for the continuous implementation of the client that general, the analysis presented below of the arbiter's response will require a few more restrictions.

4.1.1 Brockett Constraints for the Requests

For the reasons stated in section 2.2.4, the first continuous restriction will be that the client's request lines stay within a Brockett ring. Letting $a$ and $b$ represent the request lines of the two clients, this requirement is

$$\Box (\text{Brockett}(a) \land \text{Brockett}(b)) \quad (C_1)$$

Where $\text{Brockett}(v)$ was defined in (3.3). The boundaries of the ellipses of the ring depend on which arbiter model is being used. The toy arbiter examined below,
whose signals range between ±1, will have an input ring defined by

\[ v_{\text{min}} = 0.9 \]
\[ \dot{v}_{\text{min}} = 0.9 \]
\[ v_{\text{max}} = 1.0 \]
\[ \dot{v}_{\text{max}} = 5.0 \]

A signal \( v \) restricted to a Brockett ring obeys a number of useful rules worth stating here. The first is that it remains within a fixed range of values

\[ \text{Bounded}(v) \]  \hspace{1cm} (C_{1a})

where

\[ \text{Bounded}(v) \equiv \Box(-v_{\text{max}} \leq v \leq v_{\text{max}}) \]

which for the toy is

\[ \text{Bounded}(v) \equiv \Box(-1 \leq v \leq +1) \]

The second is that it moves from region to region in the Brockett ring in a strict sequence (see figure 2.1 for definitions of the regions)

\[ \Box(B_1(v) \rightarrow B_2(v) \rightarrow B_3(v) \rightarrow B_4(v) \rightarrow B_1(v)) \]  \hspace{1cm} (C_{1b})

where \( \tau_c = \pi \) time units (the time taken to follow a trajectory along the inner edge of the ellipse through the rising or falling regions—the maximum rise/fall time).

A signal within a Brockett ring has easy to calculate bounds on its value, time derivative, and rise/fall times. To guarantee a well behaved signal, however, usually requires an additional restriction on the minimum high/low times. This restriction is to invalidate traces that spend all their time bouncing up and down, and thus spend no time in \( B_1 \) and \( B_3 \) (see figure 4.1).
Figure 4.1: Ricochet Trajectory

In order to give the arbiter sufficient time to settle after responding to a change in a client’s request status, these minimum times will be stated in terms of time elapsed after the arbiter has acknowledged the signal change.

The restriction for minimum low time is

\[(B_1(v) \land \neg g_v) \text{ holds for at least } \tau_{low} \quad (C_{2a})\]

where \(v\) is either of \(a\) or \(b\), and \(g_v\) is the corresponding grant output signal. The equivalent restriction for minimum high time is

\[(B_3(v) \land g_v) \text{ holds for at least } \tau_{high} \quad (C_{2b})\]

The combination of these two restrictions is the second requirement of the continuous client

\[C_{2a} \land C_{2b} \quad (C_2)\]

4.1.2 No Feedback Controllers

The final continuous client restriction is unrelated to those above. In order to show that the arbiter almost surely escapes metastability, it is necessary to exclude clients
which are feedback controllers that by careful design or unimaginable coincidence trap the arbiter in the metastable region. This exclusion is accomplished by requiring that the clients be relatively insensitive to variations in the grant lines when waiting for a grant. In real circuits, this insensitivity is achieved by output "metastability filters" that have very low gains when the arbiter is in its metastable region, and thus make it very unlikely that a client would accidently create a controlling feedback loop.

To give a quantitative statement of the required insensitivity, let \( a \) and \( b \) be the request lines, and \( x \) and \( y \) be the corresponding grant lines. The clients are \( \alpha \)-insensitive to the outputs of the arbiter if, given two trajectories that start from states that only differ in their \( x \) and \( y \) components, the absolute value of the difference of the \( a \) and \( b \) components of these trajectories is bounded by \( \alpha \) times the integral of the absolute value of the differences of their \( x \) and \( y \) components. This integral is used as a measure of the history of the outputs observable by the client.

A formal definition is given in figure 4.2. The predicate \( \text{waiting}(\sigma, \tau) \) holds if both clients wait for a grant throughout the first \( \tau \) time units of the trace that starts at state \( \sigma \). The predicate \( \text{ClientsSame}(\sigma_1, \sigma_2) \) holds if the only components of states \( \sigma_1 \) and \( \sigma_2 \) that differ are \( x \) and \( y \). The function \( \text{gap}_{ab}(\sigma_1, \sigma_2, \tau) \) is the difference in the \( a \) and \( b \) components at time \( \tau \) for traces starting from states \( \sigma_1 \) and \( \sigma_2 \), and measures the difference in response between the two traces. The function \( \text{gap}_{xy}(\sigma_1, \sigma_2, \tau) \) is the integral of the differences in the \( x \) and \( y \) components, and measures the observable difference in the traces' histories. The clients are \( \alpha \)-insensitive if the difference in their responses is less than \( \alpha \) times the difference in the observable histories. Clients that are feedback controllers are excluded with the following requirement:

\[
\Diamond \left( \neg (r_1 \wedge r_2 \wedge \neg g_1 \wedge \neg g_2) \lor \alpha\text{-insensitive} \right) \tag{C_3}
\]
\[
\text{waiting}(\sigma, \tau) = \forall \rho \in [0, \tau]. \quad x_1(\varphi(\sigma, \rho)) \wedge x_2(\varphi(\sigma, \rho)) \\
\wedge \neg g_1(\varphi(\sigma, \rho)) \wedge \neg g_2(\varphi(\sigma, \rho))
\]

\[
\text{ClientsSame}(\sigma_1, \sigma_2) = |\sigma_1 - \sigma_2| = |x(\sigma_1) - x(\sigma_2)| + |y(\sigma_1) - y(\sigma_2)|
\]

\[
\text{gap}_{ab}(\sigma_1, \sigma_2, \tau) = |a(\varphi(\sigma_1, \tau)) - a(\varphi(\sigma_2, \tau))| + |b(\varphi(\sigma_1, \tau)) - b(\varphi(\sigma_2, \tau))|
\]

\[
\text{gap}_{xy}(\sigma_1, \sigma_2, \tau) = \int_0^\tau |x(\varphi(\sigma_1, \rho)) - x(\varphi(\sigma_2, \rho))| + |y(\varphi(\sigma_1, \rho)) - y(\varphi(\sigma_2, \rho))| d\rho
\]

\[
\alpha-\text{insensitive}(\sigma) = \forall \sigma' \in \mathbb{R}^d. \quad \text{ClientsSame}(\sigma, \sigma') \\
\implies \forall \tau \in \mathbb{R}^+.
\]

\[
(\text{waiting}(\sigma, \tau) \wedge \text{waiting}(\sigma', \tau)) \\
\implies (\text{gap}_{ab}(\sigma, \sigma', \tau) < \alpha \cdot \text{gap}_{xy}(\sigma, \sigma', \tau))
\]

Where \( \sigma \) is a state, \( \tau \) is a time, and \( |\sigma_1 - \sigma_2| \) denotes the \( L^1 \) norm (sum of absolute values) of the component-wise difference between the two states.

Figure 4.2: \( \alpha \)-insensitivity

If both clients are waiting for a grant, (\( C_3 \)) requires that eventually a grant will be given or the clients will reach a state where they are \( \alpha \)-insensitive to the outputs of the arbiter. It is not required that the clients settle to a specific value or even that the clients settle at all. While waiting, \( a \) and \( b \) may wander within the range specified for \( r_1 \) and \( r_2 \)—only the dependence of this wandering on the observed history of \( x \) and \( y \) is restricted.

### 4.1.3 An Abstraction Mapping

As was stated before, verifying a continuous model against a discrete specification requires an abstraction mapping. This statement holds true for even the clients—many of the arbiter properties include statements about \( r_1 \) and \( r_2 \), and these discrete states must be translated into regions in the continuous domain in order to reason with the ODE arbiter models. Note that this section does not explicitly mention which continuous variable will be used in these mappings; be aware that any actual
model will have a continuous variable associated with each discrete \( r \) signal and that
the Brockett mappings given here will apply to that variable.

Intuitively, the high \((B_3)\) and low \((B_1)\) regions of the Brockett ring would
make excellent definitions of \( r \) and \(-r\) respectively. This definition, however, leads
to a signal having no discrete value when in regions \( B_2 \) and \( B_4 \). Another possible
definition would keep \( B_3 \implies r \) and define \(-r\) as everything else, but would lose
some of the symmetry which will be heavily exploited throughout the analysis below.

The abstraction mapping which was found to be most useful kept both sym­
metry and a discrete definition for a signal everywhere within the Brockett ring.

\[
\text{input rising or high} = B_2 \lor B_3 = B_{2,3} \implies r \\
\text{input falling or low} = B_1 \lor B_4 = B_{1,4} \implies -r
\]

By choosing an abstraction mapping which is defined everywhere on the input Brock­
ett ring, and having the restriction that inputs are always within their Brockett ring
\((C_1)\), it is possible to invert the mapping as well

\[
r \iff B_{2,3} \\
-r \iff B_{1,4}
\]

Note that this mapping then has the pleasant property that applying logical oper­
ations can be done equivalently in both the continuous and discrete domains. For
example

\[
r = B_{2,3} = \neg B_{1,4} = \neg(\neg r)
\]

One possible complaint about this mapping relates back to the argument in
section 2.2.4—the continuous input will have a defined discrete value as it passes
through the arbiter model’s critical parameter values (those places where the model
is unstable and therefore impossible to accurately analyze). But there is not actually
a problem. As investigation of the arbiter models will show, the output of the arbiter is always bounded, even when its parameters are near a critical value. And the Brockett ring restriction on the inputs guarantees that the signal will not remain long near the critical parameter value, and so the system will settle to a stable and easily analyzed configuration. In fact, \((C_{1b})\) combined with the abstraction mapping will show

\[
\begin{align*}
    r & \sim B_3 \\
    -r & \sim B_1
\end{align*}
\]  

Furthermore, this equation can be used to state \((C_{2a})\) in a more useful form. The arbiter’s reset property guarantees that the grant will fall within \(\Gamma\) time units of a falling request. And (4.1) bounds the time between a falling request and \(B_1\). Therefore, \((C_{2a})\) can be restated as

\[
    r_v \text{ holds for at least } \max(r_c, \Gamma) + \tau_{low}
\]  

Unfortunately, no similar rewrite is possible for \((C_{2b})\). A client’s request going high does not guarantee that the arbiter will respond with a correspondingly high grant within any bounded time. Even if the arbiter guaranteed liveness, a request by one client might not be answered for an arbitrary period of time if the other client had already received and was keeping the grant. There is no rule symmetric to reset for raised requests, and without such a rule, no revision of \((C_{2b})\) is possible.

### 4.2 The Toy Arbiter

The first arbiter model considered is a “toy” arbiter. This arbiter has simple ODEs that are amenable to analysis, and many observations from the toy arbiter can be applied to the interlock considered in chapter 6.
4.2.1 The Differential Equations

The client request lines, \( a \) and \( b \), are the inputs to the toy arbiter. The grant lines \( x \) and \( y \) are its outputs. All four variables will range between \( \pm 1 \), where values near \( +1 \) are considered high and values near \( -1 \) are considered low. The ODEs defining the dynamic system are

\[
\begin{pmatrix}
\dot{x} \\
\dot{y}
\end{pmatrix} = f(\sigma) =
\begin{pmatrix}
 a - 1 - y - ay - x - x^3 \\
b - 1 - x - bx - y - y^3
\end{pmatrix}
\] (4.2)

which were arrived at by algebraic simplification from

\[
x = (1 - y)(a - x) - (1 + y)(1 + x) + (x - x^3)
\] (4.3)

and its dual for \( \dot{y} \). The first term of (4.3) causes \( x \) to follow \( a \) when \( y \) is near \( -1 \). The second term forces \( x \) to \( -1 \) when \( y \) is near \( +1 \). The final term causes the system to favor values of \( x \) near \( +1 \) or \( -1 \)—stable fixed points will tend to form in these neighbourhoods, although an unstable fixed point at 0 may also be unavoidably introduced.

The behaviour of the toy arbiter is revealed by its phase space portraits as shown in figure 4.3. When \( a \) and \( b \) are both low, the system has a single attractor with \( x \) and \( y \) both low. This case corresponds to the discrete state where both requests and both grants are false. When \( a \) is low and \( b \) is high, the system has a single attractor with \( x \) low and \( y \) high. States near this attractor correspond to giving a grant to \( b \). The case with \( a \) high and \( b \) low is symmetric. When \( a \) and \( b \) are both high, the system has two attractors—one with \( x \) high and \( y \) low and the other \( x \) low and \( y \) high. Respectively, these attractors correspond to giving the grant to \( a \) or \( b \). There is also an unstable attractor near the origin. It is this saddle point that gives rise to metastable behaviours. Because the values of \( a \) and \( b \) are
Figure 4.3: Phase space portraits for the Toy Arbiter (fixed points marked by "+") for attractors, "o" for the saddle point)
not fixed, the exact location of this saddle point can vary. In fact, it can be located anywhere in a region of positive measure (i.e. area > 0). Even with such variations, it can be shown that, with loyal clients, it is not possible to trap a set of trajectories corresponding to a set of initial states with positive measure. This fact leads to the almost surely liveness of the arbiter.

4.2.2 The Abstraction Mapping

The development of a suitable abstraction mapping for the toy's outputs is considerably more difficult than specifying one on the inputs. Even when the inputs are confined to very symmetric and well-spaced Brockett rings by \((C_t)\), the output of the toy during transitions may be almost spastic in comparison. It is possible, however, to use a Brockett ring as an abstraction mapping for the outputs, albeit a rather distorted instance of the annulus (see figure 4.4). Unfortunately, it has not yet been possible to prove that the toy complies with the Brockett restriction on \(v\) values. Therefore, this Brockett ring can only loosely be used as the abstraction mapping—it provides bounds on the values of output variables but not their time

Figure 4.4: The Toy's Output Brockett Ring
derivatives.

Unlike the inputs, the toy's abstraction mapping for the outputs will not include discrete states for signals in the rising and falling sections of the ring ($B_2$ and $B_4$). The discrete grant signals will be defined as

\[
B_1 \implies (v < 0.5) \implies \neg g
\]
\[
B_3 \implies (v > 0.7) \implies g
\]

where $v$ is either of $x$ or $y$. The clients may interpret signals in the rising and falling regions however they wish, although a discrete signal is only allowed to change value once each time its corresponding continuous variable passes through these undefined regions (for example, as $x$ moves up from 0.5 to 0.7, $\neg g_1$ must hold until some time, after which $g_1$ must hold until the output leaves $B_3(x)$).

Despite its distortion, this output Brockett ring still makes an excellent abstraction mapping. And there is considerable evidence to indicate that a toy metastability filter could be designed which would map this distorted Brockett ring into the same ring as that used by the inputs, although such a possibility will not be further explored here.

One final note must be made about this abstraction function. It appears to allow the toy's outputs to stabilize anywhere below $v = +0.5$ or above $v = +0.7$. Since the clients are not allowed to respond to the outputs until $B_3(v)$ or $B_1(v)$, the clients appear to require infinite precision—a client must differentiate $0.4999\ldots$ from $0.5000\ldots$ even if the output stabilizes arbitrarily close to 0.5. In practice, however, the toy will never stabilize close to the edges of its high and low regions. The primarily reason for the client restriction ($C_2$) is to guarantee that the arbiter's outputs are sufficiently removed from the edges of $B_1$ and $B_3$ before the client is allowed to initiate another change in system state. Therefore, a client will never
need to make a decision when its grant is near a boundary; it can simply wait until
the output stabilizes.

4.2.3 The Continuous Model

As described in section 3.1, the continuous models investigated in this thesis consist
not only of the arbiter’s ODEs, but also of the ODEs defining the environment and
a probability space describing the initial conditions. Specifying the ODEs for the
clients, however, would remove any generality from the result, and is unnecessary
in the construction of the proof. The model’s ODE system \( F_{toy} \) merely needs

- To satisfy the client restrictions given in section 4.1 with regard to the evolu-
tion of arbiter inputs \( a \) and \( b \): \( F_{toy} \models C_0 \land C_1 \land C_2 \land C_3 \).

- To use the ODEs (4.2) to govern the evolution of arbiter outputs \( x \) and \( y \):
  \[ f \subset F_{toy} \]

- To have no inputs (which includes no explicit reference to time).

- To possibly contain some hidden variables which may influence the evolution
  of \( a \) and \( b \), but not \( x \) or \( y \).

The probability space of initial conditions will likewise not be completely
specified. The restriction placed on it will be that all possible initial conditions for
the continuous variables must map to discrete signal states in which the requests and
grants are false—the arbiter and clients must start in a quiescent state. Equivalently,
the probability space \( (\Omega, P)_{toy} \) must be drawn from the universe

\[
U_{toy} = (-1 \leq a \leq 0.9) \land (-1 \leq b \leq 0.9) \land
\quad (-1 \leq x \leq 0) \land (-1 \leq y \leq 0)
\]
Note that the outputs are restricted beyond just mapping to false discrete values. The size of $B_1$ for $x$ and $y$ is too large to permit them to start anywhere within it. They are therefore restricted to both begin with non-positive values.

The continuous model that implements the toy will consist of the ODE system and the probability space

$$\mathcal{T}OY \equiv (F, \Omega, P)_{toy}$$

This model will have a state vector

$$\sigma = (x \ y \ a \ b \ \ldots)$$

which may contain hidden environment variables as already mentioned.
In this chapter, the toy arbiter is shown to fulfill each of the five properties of an almost ideal arbiter; one section below is devoted to each property.

5.1 Closure

Closure is not a requirement for the arbiter, but a limitation on what restrictions can be applied to environment. The restrictions in section 4.1 do not constrain the relative arrival time of requests. Accordingly, this requirement is satisfied.

5.2 Mutual Exclusion

The requirement is $\Box \neg (g_1 \land g_2)$. By the definition of the abstraction function for the toy, this is equivalent to

$$\Box \neg ((x \geq 0.7) \land (y \geq 0.7))$$

which is implied by the statement proven below: $\Box (x + y \leq 0)$. The first step is to show that the arbiter’s outputs remain bounded.
Lemma 1. If the arbiter’s outputs start within the range \( \pm 1 \) and the client satisfies \( (C_1) \), then the arbiter’s outputs are also bounded in that range.

\[ \mathcal{T\Omega} \models Bounded(x) \wedge Bounded(y) \]

Proof. Because the dynamic systems under investigation are continuous, showing \( \Box(x \geq -1) \) simply requires that \( (x \geq -1) \) holds initially (as it does by section 4.2.3), and that trajectories along the line \( (x = -1) \) flow in a positive direction—that \( (\dot{x} \geq 0) \). If all trajectories along \( (x = -1) \) have increasing \( x \) values, no trajectory can cross into \( (x < -1) \). From the definition of \( \dot{x} \) (4.2)

\[
(\dot{x}(\{\sigma|x = -1\})) = a - y - ay + 1 = (a + 1)(1 - y)
\]

This construction may be read: when system state \( \sigma \) is in the set where \( (x = -1) \) holds, \( \dot{x} \) is equal to the function \( (a + 1)(1 - y) \). If \( Bounded(a) \wedge Bounded(y) \), then both terms of this equation are non-negative and so

\[
(x = -1) \wedge Bounded(a) \wedge Bounded(y) \Rightarrow (\dot{x} \geq 0) \Rightarrow \Box(x \geq -1)
\]

Similar arguments show that the system maintains \( \Box(x \leq +1) \) and \( Bounded(y) \) as well. \( \Box \)

Theorem 2. The toy satisfies mutual exclusion because one or the other of the outputs is always negative.

\[ \mathcal{T\Omega} \models \Box(x + y \leq 0) \]

Proof. This requirement is satisfied by the initial conditions. The proof then demands that no trajectory can cross from \( x + y \leq 0 \) into \( x + y > 0 \). Continuity of trajectories again makes it sufficient to show \( (x + y = 0) \Rightarrow (\dot{x} + \dot{y} \leq 0) \). Recalling the definitions of \( \dot{x} \) and \( \dot{y} \) from equation 4.2, and rewriting them by algebraic
manipulation along the line \((x + y = 0)\) gives
\[
(\dot{x} + \dot{y})(\{x| x + y = 0\}) = (2b - 2) + (a - b)(x + 1)
\]
\[
= (2a - 2) + (b - a)(y + 1)
\]

For \(a \leq b\), examine the first line.

\[
C_{1a} \implies 2b - 2 \leq 0
\]
\[
a \leq b \implies a - b \leq 0
\]
\[
Bounded(x) \implies x + 1 \geq 0
\]

A symmetric argument can be applied to the second line when \(b \leq a\). Consequently \(\dot{x} + \dot{y} \leq 0\) on the line \(x + y = 0\) as required. \(\square\)

In addition to proving that the toy satisfies mutual exclusion, these properties are frequently used below in the remaining proofs. They are therefore gathered into

\[
Bounded(x) \land Bounded(y) \land \Box(x + y \leq 0).
\]

\((A_1)\)

5.3 Reset

Only the proof for \(\Box(\neg r_1 \rightarrow \neg g_1)\) is given; the argument for \(r_2\) and \(g_2\) is equivalent. Translating the requirement through the abstraction mappings gives the continuous version.

\[
\Box(B_{1,4}(a) \rightarrow B_1(x))
\]

The proof requires two steps. The first step is to show that falling or low input leads to a low output, and the second is to show a maximum time bound on the response. The second step will be tackled separately in a lemma before the proof of the first is given in the theorem.
Lemma 3. Once a has dropped low ($B_1(a)$), $x$ must follow within bounded time ($B_1(x)$).

Proof. If $B_1(a) \equiv (a \leq -0.9)$ holds, then a negative upper bound can be placed on $\dot{x}$ in the region $\neg B_1(x) \equiv (x > 0.5)$. If all trajectories in this region have strictly negative derivatives for their $x$ components, they must all leave the region in bounded time.

\[ (\dot{x})\{\{a \leq -0.9\} \land (x > 0.5)\} = a - 1 - y - ay - x - x^3 \]
\[ = -y(1 + a) + a - (1 + x + x^3) \]
\[ \leq -(-1)(1 - 0.9) + (-0.9) - 1.625 \]
\[ \leq -2.425 \]

Since the region within $Bounded(x) \land (x > 0.5)$ is only 0.5 units wide, all trajectories will satisfy $B_1(x)$ within $0.5/2.425 < 0.21$ time units. \hfill \Box

Theorem 4. If the input a falls, the output x will follow within bounded time.

\[ \Box(B_1(a) \xRightarrow{\Gamma} B_1(x)) \]

Proof. Returning for a moment to the discrete domain, the client half of the handshake ($C_0$) guarantees $\Box(\neg r_i \rightarrow \neg g_i)$. By the definitions of $\rightarrow$ and $\sim$

\[ (p \rightarrow q) \implies (p \sim q) \]

and thus $\Box(\neg r_i \sim \neg g_i)$ is established.

All that remains is to put a bound on the time taken. It was shown (4.1) that $\neg r_1 \xRightarrow{\Gamma} B_1(a)$. Combining this fact with lemma 3 will produce a suitable time bound of $\Gamma = \tau_v + 0.21$. \hfill \Box
5.4 Handshake

Again, only the proofs of $\Box(g_1 \rightarrow \neg r_1)$ and $\Box(\neg g_1 \rightarrow r_1)$ are given. The arguments for $r_2$ and $g_2$ are symmetric. Mapping through the abstraction function, the requirements are

$$\Box(B_3(x) \rightarrow B_{1,4}(a))$$
$$\Box(B_1(x) \rightarrow B_{2,3}(a))$$

In fact, a slightly stronger statement will be proven, on the basis of

$$B_{1,4}(a) \implies (a < 0.9)$$
$$B_{2,3}(a) \implies (a > -0.9)$$

**Theorem 5.** Once high, $x$ will remain so until $a$ has fallen.

$$\mathcal{T}_0Y \models ((x \geq 0.7) \rightarrow (a < 0.9))$$

*Proof.* The same procedure as theorem 2 will be used: proof by showing that all flow along the line $(x = 0.7)$ has a positive $\dot{x}$ component for $(a \geq 0.9)$, and therefore all trajectories on the edge of the region in question flow inward until $(a < 0.9)$. Note that attention can be restricted to the region $(x + y < 0)$ because of $(A_1)$, and

$$(x + y \leq 0) \land (x \geq 0.7) \implies (y \leq -0.7)$$

$$(\dot{x})(\{\sigma|(a \geq 0.9) \land (x = 0.7)\land(y \leq -0.7)\})$$

$$= a - y(1 + a) - (1 + x + x^3)$$
$$\geq 0.9 - (-0.7)(1 + 0.9) - 2.0430$$
$$\geq 0.187$$

*□*

**Theorem 6.** Once low, $x$ will remain so until $a$ has risen.

$$\mathcal{T}_0Y \models ((x \leq 0.5) \rightarrow (a > -0.9))$$

47
**Proof.** Exactly the same method is used again. In this case the region in question is bounded above, so the objective is to show that \( \dot{x} \) along its boundary is negative.

\[
(\dot{x})(\{a \leq -0.9 \} \land (x = 0.5)) = a - y(1 + a) - (1 + x + x^3) \\
\leq -0.9 - (-1)(1 - 0.9) - 1.625 \\
\leq -2.425
\]

\[\square\]

### 5.5 Liveness, almost surely

Only the first half of the proof is given: \( \Box(\mathbf{r}_1 \leadsto ((\mathbf{r}_1 \land g_1) \lor (\mathbf{r}_2 \land g_2))) \). The argument for the second half, a request at \( \mathbf{r}_2 \), is equivalent. Note that (4.1) ensures

\[
\Box(\mathbf{r}_1 \leadsto B_3(a)) \equiv (\mathbf{r}_1 \leadsto (a > 0.9))
\]

so it is sufficient to consider the case that \( (a > 0.9) \). The proof proceeds by case analysis on \( b \).

If \( B_4(b) \), then \( b \) is falling. If \( g_1 \) occurs sometime during \( b \)'s fall, then the liveness requirement is satisfied. Otherwise, \( r_1 \) still holds when \( B_1(b) \)—the situation covered in the next case.

If \( B_1(b) \land B_3(a) \), then all trajectories head towards an attractor located somewhere in \( (x > 0.7) \) (see figure 4.3). There is a non-empty neighborhood of this attractor in which \( g_1 \) is true. If \( B_1(b) \) is maintained long enough, then the trajectory will enter this neighborhood and satisfy the liveness requirement. Otherwise, \( r_1 \) still holds when \( B_2(b) \), which leads to the next case.

If \( B_2(b) \) then \( b \) is rising. If \( g_1 \) occurs sometime during \( b \)'s rise, then the liveness requirement is satisfied. Otherwise, \( r_1 \) still holds when \( B_3(b) \), which is the final case.
If $B_3(b)$, then both clients are requesting a grant. This is the scenario that can lead to metastable behaviours.

5.5.1 Liveness and metastability

![Figure 5.1: Regions for Liveness Proof](image)

The remaining proof obligation is $\Box((B_3(a) \land B_3(b)) \leadsto ((B_{2,3}(a) \land B_3(x)) \lor (B_{2,3}(b) \land B_3(y))))$. The proof involves dividing the state space into regions. It is only necessary to analyze trajectories in the region below the $(x+y = 0)$ line because of $(A_1)$. This region is subdivided into four parts, as shown in figure 5.1. The proof below first shows that any trajectory in region 1 will satisfy $B_{2,3}(a) \land B_3(x)$ after a bounded delay. Likewise, trajectories in region 2 satisfy $B_{2,3}(b) \land B_3(y)$ with the same delay bound. Next, it is shown that any trajectory can only remain in region 3 for a bounded amount of time. Region 4—the only remaining region—contains the metastable point. As a final step, trajectories are shown to leave this region almost surely and enter region 1 or 2. In the remainder of this section, $W_1$, $W_2$, $W_3$, and $W_4$ are used to denote regions 1 through 4 respectively.

**Lemma 7.** Any trajectory in $W_1$ satisfies $B_{2,3}(a) \land B_3(x)$ after a bounded delay,
and any trajectory in $W_2$ satisfies $B_{2,3}(b) \land B_3(y)$ within the same delay bound.

Proof. Only the proof of the first statement is given; the second is symmetric for $b$ and $y$. Note also that the remaining case from the case analysis above is $B_3(a) \land B_3(b)$; combining this fact with the client’s protocol requirement ($C_0$) shows that $B_{2,3}(a) \land B_{2,3}(b)$ will hold until liveness is satisfied.

Therefore, all that needs to be shown is that $B_3(x)$ will hold within some delay bound. $W_1$ is defined by

$$(x + y \leq 0.0) \land (-0.4 \geq y - 2x) \land (y \geq -1)$$

It is straightforward to show that anywhere in this region ($x \geq -0.3$) and ($\dot{x} > 0.0176$) (the minimum for $\dot{x}$ occurs at $a = 0.9$, $x = 4/30$, $y = -4/30$). Such a positive $\dot{x}$ guarantees that $(x > 0.7)$ after at most $(0.7 - (-0.3))/0.0176 < 57$ time units (a very loose upper bound). All trajectories in $W_1$ will satisfy $B_3(x)$ within 57 time units. Likewise, all trajectories in $W_2$ will satisfy $B_3(y)$ within 57 time units.

Lemma 8. Any trajectory in $W_3$ leaves in bounded time.

Proof. Region 3 is defined by

$$(x + y \leq -0.2) \land (-0.4 \leq y - 2x) \land (-0.4 \leq x - 2y) \land (x \geq -1) \land (y \geq -1)$$

It is easy to show that anywhere in this region ($\dot{x} + \dot{y} > 0.37$) (all flow is up and to the right), which proves that all trajectories must eventually leave the region. Because $(-0.2 - (x + y) \leq 1.8)$, a trajectory must leave $W_3$ after at most $1.8/0.37 < 5$ time units.

The region that remains to be considered is $W_4$. Region 4 contains the saddle point and the location of this saddle depends on the values of $a$ and $b$, so the saddle
may move slightly even with $B_3(a) \land B_3(b)$. By $(C_3)$, either a grant is issued or the clients become $\alpha$-insensitive; it is sufficient to consider the latter case.

**Lemma 9.** Once the clients become $\alpha$-insensitive, trajectories in $W_4$ will escape almost surely.

**Proof.** Consider two states, $\sigma_1$ and $\sigma_2$ in $W_4$ such that

$$
\text{waiting}(\sigma_1) \land \text{waiting}(\sigma_2) \\
\land \text{ClientsSame}(\sigma_1, \sigma_2) \\
\land (x(\sigma_1) + y(\sigma_1)) = (x(\sigma_2) + y(\sigma_2)) \\
\land (x(\sigma_1) - y(\sigma_1)) \neq (x(\sigma_2) - y(\sigma_2))
$$

(5.1)

It is shown that trajectories $\varphi(\sigma_1, \cdot)$ and $\varphi(\sigma_2, \cdot)$ emanating from $\sigma_1$ and $\sigma_2$ must diverge; therefore, at most one of them remains in $W_4$ indefinitely. This proof will demonstrate that the dimension of the set of states whose trajectories remain trapped in $W_4$ is at most $d - 1$. Because dimension is preserved (3.1), this set of trapped states corresponds to a set of initial states of dimension at most $d - 1$, and this initial state set has probability zero by the smoothness requirement for the initial state probability function $P$ (3.2). The initial state set which escapes the metastability thus has probability one, and so escape occurs almost surely. The remainder of this section is divided into two parts:

- The proof that the trajectories $\varphi(\sigma_1, \cdot)$ and $\varphi(\sigma_2, \cdot)$ diverge—that the distance between them in the $x$-$y$ plane increases as time increases.

- An explanation of why this divergence implies the dimension of the set of trapped states is at most $d - 1$. 

51
For the rest of this analysis, it is convenient to perform a change of variables. Let

\[ u = x - y \quad \quad \quad c = a + b \]
\[ v = x + y \quad \quad \quad d = a - b \]

Rewriting (4.2) yields the toy's dynamical system in the new coordinates:

\[ \dot{u} = d + uc/2 - vd/2 - u^3/4 - 3uv^2/4 \]
\[ \dot{v} = c - 2 - v + ud/2 - vc/2 - v^3/4 - 3vu^2/4 \] (5.2)

Let \( \tau \in \mathbb{R}^+ \) such that

\[ \forall \rho \in [0, \tau], \text{waiting}(\sigma_1, \tau) \land \text{waiting}(\sigma_2, \tau) \]

Let \( u_1(\rho) \) be the \( u \) component of \( \varphi(\sigma_1, \rho) \), \( u_2 \) be the \( u \) component of \( \varphi(\sigma_2, \rho) \), and \( \Delta u = u_2 - u_1 \). Define \( v_1, v_2, \Delta v, c_1, c_2, \Delta c, d_1, d_2, \) and \( \Delta d \) in the same way.

The proof is based on a double cone in \( u-v \) space with the vertex at \( \varphi(\sigma_1, \cdot) \) and the axis lying parallel to \( v = 0 \). Note that the vertex will move as \( \varphi(\sigma_1, \cdot) \) moves. At \( \rho = 0 \), \( \varphi(\sigma_2, \cdot) \) will be inside this cone, since \( v_2(0) - v_1(0) = \Delta v(0) = 0 \).

Mathematically, the cone is defined by (where \( k_v \) is some positive constant)

\[ |\Delta v| \leq |\Delta u|/k_v \] (5.3)

Showing that \( \varphi(\sigma_1, \cdot) \) and \( \varphi(\sigma_2, \cdot) \) diverge then requires demonstrating two properties:

- All trajectories in the cone diverge from the vertex in the \( u \) direction. In this case, \( \Delta \dot{u} > k \Delta u \) for some fixed \( k > 0 \) in the cone.

- All trajectories that enter the cone stay in the cone. If trajectories are diverging in the \( u \) direction already, then \( \Delta \dot{v} < -k \Delta v \) (for some other fixed \( k > 0 \)) holding along the edges of the cone will ensure such trapping behaviour.
Without loss of generality, assume $\Delta u > 0$, and then consider the case where $\Delta v > 0$; the other case is equivalent. In order to establish the two properties stated above, a loose lower bound on $\Delta u$ in terms of $\Delta u$ and a loose upper bound on $\Delta v$ in terms of $\Delta v$ is determined. Finding these bounds requires finding a relationship between $\Delta c$, $\Delta d$ and $\Delta u$ ($\Delta v$ is taken care of by the definition of the cone). Such a relationship can be established by placing an additional restriction on $\Delta u$

$$\Delta u(\tau) \geq k_u \int_0^\tau |\Delta u(\rho)| + |\Delta v(\rho)| \, d\rho$$  \hspace{1cm} (5.4)

and then rewriting the earlier definition of $\alpha$-insensitive in the new variables

$$|\Delta c(\tau)| + |\Delta d(\tau)| < 2\alpha \int_0^\tau |\Delta u(\rho)| + |\Delta v(\rho)| \, d\rho$$

or

$$|\Delta c(\tau)| < 2\alpha \int_0^\tau |\Delta u(\rho)| + |\Delta v(\rho)| \, d\rho$$  \hspace{1cm} (5.5)

$$|\Delta d(\tau)| < 2\alpha \int_0^\tau |\Delta u(\rho)| + |\Delta v(\rho)| \, d\rho$$

Combining (5.4) and (5.5) gives the required relationship:

$$|\Delta c| \leq 2\alpha|\Delta u|/k_u$$  \hspace{1cm} (5.6)

$$|\Delta d| \leq 2\alpha|\Delta u|/k_u$$

That (5.4) is merely a tightening of an already stated restriction can be seen by differentiating it and using the definition of the cone

$$\Delta \dot{u}(\tau) \geq k_u (|\Delta u(\tau)| + |\Delta v(\tau)|)$$

$$\geq k_u (1 + 1/k_v)|\Delta u(\tau)|$$  \hspace{1cm} (5.7)

Which again requires $\Delta \dot{u}$ to be a positive function of $\Delta u$. Intuitively, this equation is providing a lower limit on the rate of exponential divergence of trajectories in the $u$ direction.
With $\Delta v$, $\Delta c$, and $\Delta d$ related to $\Delta u$, it is possible to find a lower bound in $W_4$ for $\Delta \hat{u}$ as a function of $\Delta u$, $\alpha$, $k_v$, and $k_u$ (starting from the rewritten dynamical system (5.2))

$$\Delta \hat{u} \geq (0.815625 - 2.45\alpha/k_u - 0.1175/k_v)\Delta u$$

For more details on obtaining this equation and those in the rest of the proof, see appendix A. Combining this equation with the requirement given by (5.7), an inequality on $\alpha$, $k_v$, and $k_u$ is found

$$0.815625 - 2.45\alpha/k_u - 0.1175/k_v - k_u/k_v \geq 0$$

(5.8)

An assignment to $\alpha$, $k_v$, and $k_u$ which satisfies this inequality will guarantee that points within the cone will diverge from the vertex.

In the search for an upper bound on $\Delta \hat{v}$, it is possible to replace the inequality defining the cone (5.3) with $|\Delta \hat{v}| = |\Delta u|/k_v$ since only the value of $\Delta \hat{v}$ along the edges of the cone is of interest. This replacement enables $\Delta c$ and $\Delta d$ to be directly related to $\Delta v$, which in turn allows an upper bound to be placed on $\Delta \hat{v}$

$$\Delta \hat{v} \leq (-1.9 + 0.1175k_v + 2.45\alpha k_v/k_u)\Delta v$$

Since a trapping cone requires $\Delta \hat{v}$ to be a negative function of $\Delta v$, this upper bound becomes the inequality

$$-1.9 + 0.1175k_v + 2.45\alpha k_v/k_u \leq 0$$

(5.9)

Which, for feasible $\alpha$, $k_v$, and $k_u$, guarantees that trajectories that enter the cone cannot leave.

Any assignment of $\alpha$, $k_u$, and $k_v$ which satisfies (5.8) and (5.9) will guarantee that $\varphi(\sigma_2, \cdot)$ will remain within the cone and that it will diverge from the vertex (and
thus \( \varphi(\sigma_1, \cdot). \) Because \( W_4 \) has bounded extent, at least one of the diverging \( \varphi(\sigma_1, \cdot) \) and \( \varphi(\sigma_2, \cdot) \) must escape \( W_4 \)—completing the proof of divergence. An example of a feasible assignment is

\[
\alpha = 0.05 \quad k_u = 0.35 \quad k_v = 4.06
\]

The remaining task is to show that the divergence of \( \varphi(\sigma_1, \cdot) \) and \( \varphi(\sigma_2, \cdot) \) implies that the set of trapped states is of maximum dimension \( d - 1 \). The system state is represented by the vector

\[
\sigma = (x \ y \ a \ b \ \ldots) = (u \ v \ c \ d \ \ldots)
\]

which has \( \text{dim}(\sigma) = d \). Now consider the subvector

\[
\sigma_{-u} = (v \ c \ d \ \ldots)
\]

which contains all state variables except \( u \). For any set of assignments to \( \sigma_{-u} \), there may be a choice of \( u_1 \) such that the trajectory \( \varphi(\sigma_1, \cdot) \) starting at \( \sigma_1 = (u_1 \ \sigma_{-u}) \) becomes trapped.

But then any trajectory \( \varphi(\sigma_2, \cdot) \) starting at a state \( \sigma_2 = (u_2 \ \sigma_{-u}) \), which differs from \( \sigma_1 \) only in the \( u \) value, will escape. This fact is proven by the divergence argument above. From (5.1), the states \( \sigma_1 \) and \( \sigma_2 \) chosen for the divergence argument differed only in their \( u \) values: \( \text{ClientsSame}(\sigma_1, \sigma_2) \) states that the \( c, d \) and hidden variables are the same for both states; \( (x(\sigma_1) + y(\sigma_1)) = (x(\sigma_2) + y(\sigma_2)) \) can be rewritten as \( (v(\sigma_1) = v(\sigma_2)) \); and \( (x(\sigma_1) - y(\sigma_1)) \neq (x(\sigma_2) - y(\sigma_2)) \) states that the \( u \) values differ. The trajectories starting at these two states were shown to diverge, so if \( \varphi(\sigma_1, \cdot) \) is trapped then \( \varphi(\sigma_2, \cdot) \) must escape.

Therefore, at most one value of \( u \) for any choice of \( \sigma_{-u} \) lies on a trapped trajectory. This value of \( u \) can be written as a function of the other variables.
\[ u_{\text{trap}} = f(\sigma_u) \]—the function will not be defined for values of \( \sigma_u \) where no state can become trapped. The manifold in \( \mathbb{R}^d \) defined by \( f(\sigma_u) \) is of maximum dimension \( d - 1 \), and if trajectories can only become trapped if they start from the manifold, then the set of trapped trajectories is likewise of maximum dimension \( d - 1 \).

As stated at the beginning of the proof, preservation of dimension (3.1) shows that the initial state set leading to the trapped trajectories must be of the same maximum dimension, and the smoothness of the initial probability distribution (3.2) implies that escape occurs almost surely.

**Theorem 10.** The toy satisfies liveness almost surely.

**Proof.** The case analysis showed the toy would satisfy liveness eventually unless both requests ended up high. In the case where both requests were high, the \( x-y \) state space was broken into regions. Equation (A₁) requires that no trajectory enter the region \( x + y > 0 \). Lemma 8 showed that trajectories in \( W_3 \) would enter \( W_1, W_2, \) or \( W_4 \) in bounded time. By lemma 9, trajectories which enter \( W_4 \) will escape almost surely (if suitably \( \alpha \)-insensitive clients are used), and since they cannot reenter \( W_3 \), they must go into \( W_1 \) or \( W_2 \). And lemma 7 showed that all trajectories in \( W_1 \) or \( W_2 \) will satisfy liveness within bounded time.

\[ \Box \]
Chapter 6

The Seitz Interlock

From an analysis point of view, the toy arbiter's simple equations make it an ideal development platform. It is, however, far from an operating VLSI circuit. Once techniques have been tried on the toy, it is hoped they can be applied to a real arbiter circuit. In this thesis, Seitz's interlock circuit [Sei79] will serve as the second arbiter model. It is a widely used and well studied VLSI arbiter design.

As mentioned in section 2.2.5, Kurshan and McMillan [KM91] have published a verification of this arbiter. Their verification was based on model checking using a coarsely quantized representation of the phase space and a next state relation obtained by numerical integration. Their analysis could not, however, provide any guarantees that the arbiter would eventually leave the metastable region. Instead, they avoided the issue by adding an assertion that the arbiter escaped the metastability. Since this assumption does not hold for any real arbiter, its inclusion in their model is unsatisfying.

Figure 6.1 shows Seitz's arbiter. Transistors q4, q5, q9 and q10 are depletion mode transistors (threshold voltage $V_t < 0V$), and act as pull-up resistors in this
configuration. The remaining transistors are enhancement mode ($V_t > 0V$). The six transistors to the left of line $kk$ form a cross-coupled pair of NOR gates. These transistors are the heart of the interlock—the internal arbiter—and like all arbiters they are subject to metastable states. The remaining four transistors to the right of $kk$ are the metastability filter.

To turn the circuit into an ODE system, a standard first-order transistor model is used (the same model as Kurshan and McMillan). The transistors are considered to be perfect voltage controlled current sources with constant capacitances from gate to source (see figures 6.2 and 6.3).

This model ignores non-linear capacitances, subthreshold conduction and short channel effects such as channel-length modulation and drain-induced threshold lowering. The transistor parameters $V_t$, $K$, and $S$ are also assumed to be fixed and equal for all transistors of similar type.
Figure 6.2: Transistor Model (from [KM91])

\[
I_{ds} = \begin{cases} 
0 & \text{if } V_{gs} \leq V_t \text{ (cutoff)} \\
KS V_{ds} (2(V_{gs} - V_t) - V_{ds}) & \text{if } V_{gs} > V_t \text{ and } V_{ds} \leq V_{gs} - V_t \text{ (linear)} \\
KS (V_{gs} - V_t)^2 & \text{if } V_{gs} > V_t \text{ and } V_{ds} > V_{gs} - V_t \text{ (saturation)} 
\end{cases}
\]

Where the transistor parameters are threshold voltage \((V_t)\), transconductance \((K)\), and shape factor \((S = W/L)\)

Figure 6.3: Transistor Model Equation

Unlike the toy, the interlock uses a combination of active high and active low signals. Both the inputs and outputs are active low—external requests and grants occur when the signal itself is below transistor threshold (typically 0.6V [GD85]). The internal signals \(x(1)\) and \(x(2)\) are active high, although the metastability filter operates more on the difference between the two rather than based on any absolute value.

It is the internal arbiter that generates mutually exclusive grant signals, and so the metastability filter will be ignored in this analysis—\(x(1)\) and \(x(2)\) will be used as the outputs. Choosing process parameters appropriate to a 2 \(\mu\)m nMOS process [GD85, p. 453] produces the phase portraits shown in figure 6.4. Like the toy, there is only one attractor in the system unless both inputs are asserted. In this final case there are two attractors and a single, central saddle point.
Figure 6.4: Phase space portraits for Seitz's arbiter (fixed points marked by "+" for attractors, "o" for the saddle point)
A comparison of figures 4.3 and 6.4 shows that the interlock's dynamic system and the toy's are qualitatively the same. It is therefore not surprising that proofs of closure, mutual exclusion, reset, and handshake equivalent to those in chapter 5 could be constructed. Because the model is more complex, numerical optimization would be required when verifying bounds on the flow.

The proof of liveness was actually simpler for the interlock than for the toy. This is due to the first order transistor model that ignores subthreshold conduction. When an input is asserted, it is below the enhancement device threshold voltage and the currents into nodes \( x(1) \) and \( x(2) \) are independent of variations in the inputs. The location of the saddle-point is therefore fixed, and its stable manifold is of dimension \( d - 1 \). Increasing the sophistication of the circuit model to include subthreshold conduction would result in movement of the saddle-point with varying inputs. However, this movement would be much smaller in relative terms than for the toy. Therefore, an argument similar to the one in section 5.5 should apply.

A more accurate transistor model has not been adopted in the hope that further research may make it unnecessary. By analyzing the linear behaviour of the arbiter systems near the metastable point, it is hoped that a quantitative estimate of the effects of model inaccuracies can be developed. If it is known that further refinement of the transistor model will produce some maximum perturbation of the current model that is less than that needed to invalidate the proofs, then the additional refinement is unnecessary. The never-ending list of terms that can be added to a transistor model for more accuracy is thus truncated at a reasonable length. Such a "proof despite inexact models" has not yet been formalized however, and the idea is presented here merely to explain the lack of more accurate transistor modeling in this chapter.
Although the internal arbiter is the most interesting component of the circuit from the viewpoint of arbitration, it would be negligent to leave out at least an informal description of the right-hand portion of the circuit. The metastability filter operates on a simple principle—if the internal signals do not differ by more than a transistor threshold, there has certainly not been a successful arbitration, and therefore both outputs should remain high (remember, a grant is output low). Only when one internal signal is significantly above the other will the first signal's filter transistor turn on and lower its output level. In a circuit model that neglects subthreshold conduction, this filter makes the outputs of the arbiter, \( x(3) \) and \( x(4) \), independent of the internal nodes \( x(1) \) and \( x(2) \) when the arbiter is near the metastable point. Even if subthreshold conduction is taken into account, the filter will not allow a grant until any metastability has been resolved, and has gain very close to zero near the metastable point. In addition to hiding metastability, the filter "cleans up" the output signal. It is conjectured that such filters can re-map the reachable state space of the outputs of an arbiter—for example, re-map the ugly output Brockett ring of the toy into a ring identical to the much nicer input Brockett ring—but no attempt at such a proof has yet been made.
Chapter 7

Conclusions

7.1 What has been Accomplished

In an asynchronous environment, a perfect arbiter cannot be built. The permanent indecision of metastability is an inescapable component of the mutual exclusion problem. But for a large class of clients, the probability of the arbiter never making a decision is exactly zero.

This thesis studied the problem of verifying the correct operation of arbiters. Traditionally, hardware verification techniques have needed to build discrete models of circuits before verifying the model against a discrete specification. Because metastability is a decidedly continuous behaviour, a discrete model cannot do it justice. A continuous modeling environment is needed.

Dynamic systems provide just such an environment. Two potential arbiter systems were described. The toy is a collection of very simple ODEs, making it possible to use many analytic and algebraic techniques in its examination. The Seitz interlock is a well studied and commonly used VLSI design, and its dynamic
system model was constructed from a popular and reasonably realistic first order
ODE model for MOS transistors.

Verification of a continuous model against a discrete specification requires,
in addition to the model, three more ingredients:

• The discrete specification, in this case a precise description of the properties
  of an ideal arbiter.

• An abstraction mapping which maps from the continuous to the discrete and
  back.

• A temporal logic which allows proof of discrete formulas using the abstraction
  mapping and dynamic system model.

Each of these items was drawn from verification literature in either the metastability
or dynamic systems field. The specification, adopted from Mendler and
Stroup [MS93], gives a precise yet general definition of a two client arbiter. The
primary abstraction mapping is the Brockett ring [Bro89] which, along with its
other favourable properties, allows a signal’s continuous value to vary even when its
discrete value is fixed. The temporal logic is being developed by Greenstreet, and
the arbiters represent its first full-scale use in verification.

Combining these elements made it possible to prove that the toy obeys all
properties of an ideal arbiter save liveness. For most clients, the toy arbiter would
eventually make a decision and satisfy liveness with probability one, a proof ac-
complished without requiring fixed inputs. The only clients excluded are high-gain
feedback controllers that trap the arbiter at the metastable point—a highly unlikely
design for a useful client. The proofs of the first four ideal arbiter properties for the
Seitz interlock are similar and were omitted, while an analogous proof of liveness
for the interlock was made trivial by the simplicity of the transistor model.

In conclusion, this research has shown the effectiveness of the temporal logic and abstraction mappings in formal proof that a continuous dynamic system fulfills a discrete specification. In addition, it has provided a firm foundation for further exploration of arbiters using this paradigm.

### 7.2 Suggestions for Further Research

Study of these arbiter models is not yet finished. As mentioned in the body of the thesis, the models investigated are designed to be used for at least two further explorations.

The first is to show that the toy arbiter, and possibly thereafter the Seitz interlock, can act as clients for themselves. This objective would involve designing a dynamic system to act as a metastability filter for the toy. This filter would map the toy's outputs such that their reachable state space would fall within the ring used for the toy's inputs. By showing that the outputs obey the restrictions on the inputs, it becomes possible to build multi-client arbiters through chains or trees of two-client arbiters.

But before more work can be done on the interlock, the deficiencies of the model must be addressed. Instead of simply adopting a more accurate transistor model, which—no matter how many more terms are added—will never be perfect, it should be feasible to develop a quantitative estimate of how much variation in the model would render the current results invalid. It is possible to estimate the maximum magnitude of the effects of refinements in the transistor model. If that magnitude is less than the allowable model variation, then no further refinement is necessary, since the results will remain valid regardless.
Three additional avenues of research could use this material as a starting point. Unlike the two ideas above, those below have not yet been investigated to any length.

- The interlock is the only VLSI arbiter analyzed using this technique; however, arbiters are such common circuit elements that many other designs exist which could be verified. These designs may have different protocols or objectives than the interlock, and so both continuous models and discrete specifications would need to be developed.

- Although ignored here, noise is an extremely important and well studied effect in real circuits; consequently, it would be nice to investigate its effects on the arbiters. The argument about diverging traces from section 5.5 suggests an exponential "stretching" process that is relatively insensitive to small variations in the signals. This insensitivity hints that noise may be inconsequential in arbiters from a probabilistic perspective—a number of papers have claimed similar statements about synchronizers [Vee80, HLS84]. That liveness is fulfilled almost surely has required the introduction of probability into this research. Hopefully the stretch to including noise by stochastic differential equation techniques would not be too great.

- One of the largest emphases in the verification community is the push towards automation. In the actual verification of real systems, theory has always outstripped practice because the amount of training needed to apply the theory made its commercial use impossible. The same is true of the methods used above, but automation may be possible. Hytech [HH95] is already using techniques similar to those used in the continuous analysis of the arbiters. It
may, in time, be possible to construct a theorem checker containing the same
temporal logic rules used in this research.
Bibliography


Appendix A

Details of Divergence Proof

In addition to the variables defined earlier, define $u_{avg} = (u_1 + u_2)/2$ and $v_{avg}, c_{avg}, d_{avg}$ similarly. With these definitions, it is possible to write the derivative of the divergence of the two trajectories (starting from (5.2)):

\[
\Delta \dot{u} = \Delta d + 1/2(c_{avg}\Delta u + u_{avg}\Delta c) - 1/2(d_{avg}\Delta v + v_{avg}\Delta d) - 1/4\Delta u(u_1^2 + u_2^2 + u_{avg}^2) - 3/4(1/2)(\Delta u(v_1^2 + v_2^2) + 4u_{avg}v_{avg}\Delta v)
\]

\[
\Delta \dot{v} = \Delta c - \Delta v + 1/2(d_{avg}\Delta u + u_{avg}\Delta d) - 1/2(c_{avg}\Delta v + v_{avg}\Delta c) - 1/4\Delta v(v_1^2 + v_2^2 + v_{avg}^2) - 3/4(1/2)(\Delta v(u_1^2 + u_2^2) + 4u_{avg}v_{avg}\Delta u)
\]

Since the states under investigation are in $W_4$ and both are waiting, (loose) bounds can be placed on all basic variables involved:

\[-0.3 \leq u \leq 0.3\]
\[-0.15 \leq v \leq 0\]
\[1.8 \leq c \leq 2.0\]
\[-0.1 \leq d \leq 0.1\]

The definition of the cone gave a relation (5.3) between $\Delta v$ and $\Delta u$

\[|\Delta v| \leq |\Delta u|/k_v\]

71
while (5.6) related $\Delta c$ and $\Delta d$ to $\Delta u$.

\[
|\Delta c| \leq 2\alpha |\Delta u|/k_u \\
|\Delta d| \leq 2\alpha |\Delta u|/k_u
\]

A minimum for $\Delta u$ in terms of $\Delta u$ can now be found. Note that most terms in this expansion can be chosen positive or negative because of the absolute value signs in the bounds above. In these cases, the sign is chosen so that the term as a whole has the worst (most negative) value. The only terms which cannot change sign are $c$ (always positive), $\nu$ (always negative), and squares. The latter two values have zero in their range, so that if the term cannot be made negative, the term is zeroed out of the equation.

The (very loose) bound calculated in this way is:

\[
\Delta u \geq \Delta d_{\text{min}} + 1/2(c_{\text{min}} \Delta u + u_{\text{min}} \Delta c_{\text{max}}) - 1/2(d_{\text{max}} \Delta v_{\text{max}} + v_{\text{min}} \Delta d_{\text{min}}) \\
-1/4\Delta u (3u_{\text{max}}^2) + 3/4(\Delta u) v_{\text{min}}^2 + 2u_{\text{max}} v_{\text{min}} \Delta v_{\text{min}} \\
\geq -2\alpha \Delta u/k_u + 1/2((1.8)\Delta u + (-0.3)2\alpha \Delta u/k_u) \\
-1/2((0.1)\Delta u/k_v + (-0.15)(-2\alpha \Delta u/k_u)) \\
-3/4\Delta u (0.3)^2 - 3/4(\Delta u(-0.15)^2 + 2(0.3)(-0.15)(-\Delta u/k_v)) \\
\geq (0.815625 - 2.45\alpha/k_u - 0.1175/k_v) \Delta u
\]

The differentiated requirement (5.7) that needed to be satisfied by $\Delta u$ is

\[
\Delta u \geq k_u (|\Delta u| + |\Delta v|) \\
\geq k_u (1 + 1/k_v) \Delta u
\]

Combining these two inequalities yields the final restriction (5.8) placed on $\alpha$, $k_u$, and $k_v$ by $\Delta u$

\[
0.815625 - 2.45\alpha/k_u - 0.1175/k_v \geq k_u (1 + 1/k_v) \\
0.815625 - 2.45\alpha/k_u - 0.1175/k_v - k_u - k_u/k_v \geq 0
\]

72
Replacing the definition of the cone by $|\Delta v| = |\Delta u|/k_u$, a loose upper bound on $\Delta \hat{v}$ can be found through similar substitutions. For upper bound calculation, terms are chosen most positive (or zeroed if no positive value is possible)

$$\Delta \hat{v} \leq \Delta c_{max} - \Delta v + 1/2(d_{max}\Delta u_{max} + u_{max}\Delta d_{max})$$

$$-1/2(c_{min}\Delta v + v_{min}\Delta c_{max})$$

$$-1/4\Delta v(3v_{min}^2) - 3/4(\Delta vu_{min}^2 + 2u_{max}v_{min}\Delta v)$$

$$\leq +2\alpha \Delta v/k_u - \Delta v + 1/2((0.1)k_v\Delta v + (0.3)2\alpha k_v\Delta v/k_u)$$

$$-1/2((1.8)\Delta v + (-0.15)(2\alpha k_v\Delta u/k_u))$$

$$-0 - 3/4(0 + 2(0.3)(-0.15)(k_v\Delta v))$$

$$\leq (-1.9 + 0.1175k_v + 2.45\alpha k_v/k_u)\Delta v$$

Since $\Delta \hat{v}$ need only be a negative function of $\Delta v$ (we didn’t need any more restrictions), the last line is the required inequality (5.9)

$$-1.9 + 0.1175k_v + 2.45\alpha k_v/k_u \leq 0$$