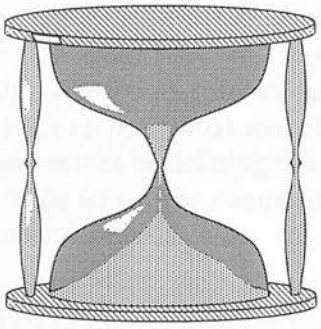
Continuous Automata:

Bridging the Gap Between Discrete and Continuous Time System Models

Martin D. Westhead

Ph.D.
University of Edinburgh
1997



If you can look into the seeds of time,
And say which grain will grow and which will not,
Speak then to me, who neither beg nor fear
Your favours nor your hate.
— Banquo, from Macbeth Act 1, Scene 3

Abstract

The principled use of models in design and maintenance of a system is fundamental to the engineering methodology. As the complexity and sophistication of systems increase so do the demands on the system models required to design them. In particular the design of *agent systems* situated in the real world, such as robots, will require design models capable of expressing discrete and continuous changes of system parameters. Such systems are referred to as *mode-switching* or *hybrid* systems.

This thesis investigates ways in which time is represented in automata system models with discretely and continuously changing parameters. Existing automaton approaches to hybrid modelling rely on describing continuous change at a sequence of points in time. In such approaches the time that elapses between each point is chosen non-deterministically in order to ensure that the model does not over-step a discrete change. In contrast, the new approach this thesis proposes describes continuous change by a continuum of points which can naturally and deterministically capture such change. As well as defining the semantics of individual models the nature of the temporal representation is particularly important in defining the composition of modular components. This new approach leads to a clear compositional semantics based on the synchronization of input and output values.

The main contribution of this work is the derivation of a limiting process which provides a theoretical foundation for this new approach. It not only provides a link between discrete and continuous time representations, but also provides a basis for deciding which continuous time representations are theoretically sound. The resulting formalism, the Continuous I/O machine, is demonstrated to be comparable to Hybrid Automata in expressibility, but its representation of time gives it a much stronger compositional semantics based on the discrete synchronous machines from which it is derived.

The conclusion of this work is that it is possible to define an automaton model that describes a continuum of events and that this can be effectively used to model complete mode-switching physical systems in a modular fashion.

Acknowledgements

The whole PhD experience is a personal journey. My journey brings to mind the Wizard of Oz, a long and tortuous adventure towards a goal which is ultimately just completing the journey.

As I approach the end of my journey as Dorothy, I would like to give thanks to the many and varied travelling companions as well as those people I met along the way.

First thanks must go to my supervisor John Hallam, in the role of the Good Witch of the South, who always popped up in the corridor to ask the right questions and catch my sloppy mathematics. It was one of John's first suggestions that I look at Hybrid Systems. It took me over a year before I agreed with him. In the role of good witch of the North was my second supervisor, Chris Malcolm, who I must thank for some very engaging philosophical discussions. The last of which was concluded at around 4am after three bottles of wine.

Next I wanted to thank those overly generous individuals who were kind enough to proof read my thesis. Many thanks to those respected citizen's of Emerald City who were Ashley Walker, Jacek Malec, Simin Nadjm Tehrani, Neil MacDonald and Vineet Gupta.

There were many cameo roles and guest appearances along the along this yellow brick road. A number of people have given me time, help and encouragement, in no particular order, I would like to thank Rod Brooks, Gerald Bérry, Amir Pnueli, Paul Caspi, Albert Beinvenist, Nicholas Halbwachs, Ahmed Bouajjaniand and Anders Ravn. Thanks also to the friends I met during my visit to Linköping in Sweden: Jan-Eric Strömberg, Åke Wernersson, Eric Sandewall, and of course Jacek and Simin. A special thank you to Stuart Anderson who gave me a great deal of time and who originally pointed me towards synchronous systems.

Of my companions along the way I wanted to mention the Mrobots group at Edinburgh, in the role of the Munchkins, a great crowd of people who have been an ever present source of distractions. I regret not having had more time to spend with them.

Thanks to my old officemate, Jeremy Wyatt, in the role of the Scarecrow, who was always smarter than he realized and with whom I spent many an afternoon turning Behaviour based robotics on its head, and many an evening quaffing pints of heavy in the Doctors.

I am deeply grateful to my parents (back in Kansas) Leo and Lizzie who have supported me throughout and have sacrificed a great deal to get me where I am. I hope I can support them too in the times ahead.

I must also thank my brother Keith, rock star and stockbroker, in the role of the Tin Woodman with whom I have spend a lot of time particularly in the last year discussing matters of the head and heart. He has been a very close friend and companion for many many years.

My final thanks go to Ashley, the Lion who always wondered if she'd have the courage to make it to the end, who has helped and supported me greatly over the last four

years as we have shared the joys and pains (mainly pains) of our respective PhDs and built a life together in the gaps in between.

As I approach the submission of this thesis there is no doubt that on this journey I have learnt a great deal and there have been times of great joy and satisfaction but right now I really feel like *there's no place like home...*

Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

Martin D. Westhead
Edinburgh
June 14, 1998

Contents

Abstract ii

Acknowledgements iv

Declaration v

List of Figures xiii

Preface xiv

I Philosophy 1

1 Modelling interaction 2

1.1 The study of interaction 2

1.1.1 Situated agents 3

1.2 Existing models 5

1.2.1 Discrete valued models 5

1.2.2 Continuous models 7

1.2.3 Mode-switching systems 7

1.2.4 Relationship to this work 11

1.3 Summary 12

2 Agent Control 13

2.1 Reactive systems 13

2.2 Automata 14
2.3 Determinism 15
2.4 Input and output 16
2.5 Parallelism 18
2.6 Verification 21
2.7 Summary 21

3 Time **23**

3.1 What is time? 23
3.1.1 Time and change 24
3.1.2 The unreality of time 25
3.2 Classifying models of time 27
3.2.1 Temporal structure 27
3.2.2 Synchrony labels and temporal metrics 28
3.2.3 Parameter assignment 29
3.2.4 Impetus 31
3.3 Summary 32

4 Discrete time **33**

4.1 Discrete time models 34
4.1.1 Limitations of discrete time 34
4.1.2 Discrete unlabeled nonmetric time 35
4.1.3 Metric labeled models 37
4.1.4 Nondeterministic increments 39
4.2 Summary 41

5 Continuous Time **42**

5.1 Density and continuity 43
5.2 Continuous time models 44
5.3 Summary 46

II Theory 47

6 Flows 48

6.1 What is a flow? 48

6.2 Equivalence of flows 51

6.3 Flow structure 52

6.4 Valued flows 55

6.5 Summary 57

7 Discrete automata 58

7.1 Events 59

7.1.1 Valuations 59

7.2 Automata Components 61

7.2.1 Signal sets 61

7.2.2 Value sets 61

7.2.3 Characteristic relation 62

7.3 Flow machines 63

7.3.1 The synchronous product 64

7.4 Delta flow machine 65

7.5 Summary 66

8 The search for the uncountable 67

8.1 Strings 69

8.2 Combs 70

8.2.1 Limit flow structure 75

8.2.2 Limit flow values 76

8.3 Continuous I/O machines 78

8.4 Summary 79

9 The continuous clock 80

9.1 The clock definition 81

9.2 Validating the definition 81

9.2.1 Finding combs 82
9.2.2 Limit equivalence 83
9.3 Summary 85

10 Generating machines **86**
10.1 Trajectories 86
10.1.1 Continuity of trajectories 88
10.2 Dynamical systems and integrators 88
10.3 Summary 90

11 Filling in the holes **91**
11.1 Divider points 92
11.1.1 Trajectories 93
11.1.2 Dynamical systems 94
11.2 Limit flows 94
11.3 Filling in the holes 96
11.3.1 Left continuity 97
11.3.2 Right continuity 98
11.4 Extracting sequences 99
11.5 Summary 101

12 Interactive machines **102**
12.1 Limits over assignment flows 102
12.2 Synchronous product 104
12.3 Summary 106

13 Examples **107**
13.1 Thermostat 107
13.2 Hour glass 108
13.3 Twin clocks 109
13.4 Blues Brothers 111
13.4.1 The bridge 112

13.4.2 The car . . . 113
13.4.3 The geometry . . . 115
13.4.4 Results . . . 116
13.5 Summary . . . 119

14 Related work **120**
14.1 Branicky, Borkar and Mitter's model . . . 120
14.2 Relationship to hybrid automata . . . 123
14.3 Continuous I/O machines . . . 125
14.3.1 Compositionality . . . 125
14.3.2 World view . . . 127
14.3.3 Discrete change . . . 127
14.4 Summary . . . 128

15 Conclusion **129**
15.1 Contributions . . . 132
15.2 Future work . . . 133
15.2.1 Verification . . . 134
15.2.2 Computational properties . . . 135
15.2.3 Practical application . . . 135
15.3 Final note . . . 136

Bibliography **137**

A Proofs from chapter 7 **146**

B Proofs from chapter 8 **148**

C Proofs from chapter 9 **151**

D Proofs from chapter 10 **157**

E Proofs from chapter 11 **163**

F Proofs for chapter 12 **165**

G Proofs for chapter 14 168

Notation 172

Index 174

List of Figures

1.1 A situated agent system 4
1.2 The discrete states of a water tank 8

2.1 A finite state machine 15
2.2 Removing nondeterminism with oracle input signals 17
2.3 A synchronous observer 21

3.1 System models 29
3.2 Another system model 30
3.3 A system model with variable step size 31
3.4 Reactive and proactive time machines 31

4.1 The water tank revisited 35
4.2 A refined water tank model 36
4.3 Variable granularity 37
4.4 Adding an integer metric 38
4.5 Tank with a real metric 38

6.1 A flow of valuations 49

7.1 A discrete automata 59
7.2 Modelling a stopwatch 65

8.1 A piece of a comb c_n over the flow sequence X_n 70
8.2 The calculation of spread 74

9.1 A delta clock 81

10.1 A trajectory machine 87

11.1 A machine with a discontinuity 92
11.2 A left-continuous state transition 97
11.3 A right-continuous state transition 98
11.4 The use of mirror signals 99
11.5 Discrete shadow signals 100
11.6 A bistable sampling clock 101
11.7 Sampled discrete shadow signals 101

13.1 A thermostat 108
13.2 Two clocks in different relative frames of reference. 110
13.3 A car jump 112
13.4 The model of the bridge 113
13.5 The car model 114
13.6 The critical points and coordinate systems 115
13.7 The geometry of the world 116
13.8 The results of the car jump 117

B.1 The proof of continuity 149

C.1 The existence of a general comb 152
C.2 The proof of uniqueness of values 154
C.3 The proof that there is a descent to any r_0 155

D.1 A trajectory machine 157

G.1 A delay transition 169
G.2 Proof of equivalence to BBM model 170

Preface

A long time ago
In a galaxy far far away...

— *George Lucas*

The story so far

This thesis began in autumn of 1993 in a stuffy lecture theatre in the University of Edinburgh. I was in the final year of my undergraduate degree in Artificial Intelligence (AI) and Computer Science (CS) and for the first time we were being presented with the work of Rodney Brooks in the form of his paper "A Robot that Walks; Emergent Behaviors from a Carefully Evolved Network" [Brooks 89]. I was excited by the ideas. Instead of concerning himself with the knowledge representation issues that so much of AI seemed to be hung up on Brooks was focusing on the problems to do with an agent's interaction with the environment. His robots were controlled by a system of interacting finite state machines. Each machine was responsible for controlling a behaviour of the robot, and they were arranged into layers of increasing sophistication. His walking robot, Genghis, was a compelling demonstration of these ideas, but it seemed to me, that his ad-hoc construction could be significantly improved on. I decided to abandon my plans to do a PhD in bio-computing and instead consider how formal CS methods could be used to improve design methodology in this new area of robotics.

I began by investigating various formalisms that could be applied to behaviour based systems. After considering basic finite state machine theory [Hartmanis & Stearns 66], and Milner's CCS [Milner 89], Stuart Anderson, in the CS department at Edinburgh, pointed me at synchronous languages. In many ways the synchronous languages, and the associated field of reactive systems, represented a movement in computer science parallel to the behaviour based movement in AI which Brooks championed. Like behaviour based systems, reactive systems are focussed on real world interaction, and modularity is based on behavioural sub-units. Unlike behaviour based systems, they use formally defined languages and clever compilers [Berry & Gonthier 92, Halbwachs *et al.* 91a] that can reduce parallel descriptions to efficient serial code and verification techniques [Halbwachs *et al.* 93] capable of automatically proving properties of real code. One of the drawbacks in Brooks' approach is the unpredictable nature of the distributed control system. Decompositional aspects of formalisms like the synchronous languages are well understood [Abadi & Lamport 90, Abadi & Lamport 91], and so a controller constructed in this way would have a well defined behaviour.

Armed with these powerful tools from computer science, I set about looking at the problem of designing a robot control system. Many workers in the field are looking at problems to do with controller architecture and organization: how do you arrange the interaction of your behaviours? These tools went a long way to solving or removing these problems so it seemed that all that was needed was a convincing demonstration. Unfortunately two significant problems remained.

Both problems became apparent to me during my time at the department of Computer and Information science in Linköping University, Sweden. The first and most damning was pointed out to me by Åke Wernersson. I had been struggling for some time to think of a suitable example to investigate the use of these design tools. The problem was that all the examples I could think of were either too trivial, or led to a number of low level sensory-actuator problems that would need to be solved. I reached the conclusion that the main reason for this was that, in fact, the most significant problem that currently faces a robot designer is the uncertainty in sensing the real world. In other words, the biggest problem in robotics is that of perception.

It is felt, particularly in behaviour based robotics, that sensors are inherently noisy and

unpredictable, and the solution to this lies in controller architecture. In my opinion, this position is fallacious. The solution to this problem is to properly understand and use the physics of the transducers themselves. Even in cases where behaviour based architectures have seemed to help [Westhead 93] closer inspection reveals that the improvements in performance are really due to increased understanding of the robots interaction with the world. A good example is provided by the use of Polaroid SONAR transducers which provide a notoriously noisy range reading. The reason for this is that a very crude controller is used to drive the transducer to do things to which it is not well suited. Very sophisticated perception is possible with these transducers, if their properties are properly used and understood [Walker *et al.* 97, Kuc & Siegel 87]. In Linköping, Åke Wernersson was carrying out similarly detailed investigations of other sensor capabilities [Bergqvist *et al.* 95]. This seems to me the over-riding issue in intelligent robotics.[1]

The second problem to keep me from designing robot systems is a modelling problem which I discussed at length with Jan-Eric Strömberg and Simin Nadjm-Tehrani. In order to design interactions in which state is maintained in the environment, it is necessary to build models which represent relevant parts of the environment. It is possible to build discrete models of the environment; in many real-world systems there are points at which natural switches in the systems dynamics occur, and between these points continuous changes can be approximated by a discrete system. However, there comes a point beyond which the discretization of a model inherently leads to inaccuracies, and a new approach is required.

It is this problem of modelling mode-switching systems which this thesis addresses. I felt dissatisfied with existing approaches to hybrid systems which displayed poor modularity and would not interface neatly with discrete formalisms that might describe controllers. Furthermore the presence of technical problems such as Zeno time (discussed in Chapter 4), led me to feel that the existing approaches to the modelling of time were clumsy. Anders Ravn pointed out to me that it is a feature of the field of hybrid systems that everyone must invent their own hybrid modelling system. In

[1] By *perception* I do not mean a passive sensing process. On the contrary the word is chosen to imply a process of extracting information from the environment which is likely to involve moving around and interacting with it.

many ways that is what I set out to do. It appeared to me that many of the problems with hybrid systems resulted from trying to squeeze a description of continuous change into a sequence of events. I became fascinated with the idea that it might be possible to define a limiting process over a finite state machine so that both continuous and discrete changes could be described as a *continuum* of events.

My first attempts failed, and I am deeply grateful to Jacek Malec for taking the trouble to spot the flaws and explain them to me. Eventually, I came up with a limiting process which has the necessary properties.

The resulting model is fundamentally different from any that I am aware of in computer science or engineering. It is more elegant in its representation of time than existing hybrid representations. This elegance leads to a sound and well defined semantics based on the synchronous product for composing hybrid models. Unlike Hybrid automata [Alur *et al.* 95] or Hybrid transition systems [Nadjm-Tehrani 94] which introduce labels for synchronization, Continuous I/O machines synchronize on input and output signals. This not only allows the composition of continuous machines with other continuous machines, but also the composition of continuous machines with discrete controllers.

Furthermore is possible to extract from the continuum a sequence of events that avoids all the Zeno time issues. The approach has the potential for practical advantages too. The use of a limiting process means that, for any continuous automata that you might build, there is a discrete machine which will approximate it to arbitrary accuracy; furthermore the relationship to the continuous machine is a well defined. This discrete machine can be executed to provide a simulation of the model and could perhaps be used to formally prove properties of the system that could not be proved in the continuous case.

However, establishing the usefulness of Continuous automata is beyond the scope of this thesis. As Milner points out in his preface [Milner 89], the establishment of a modelling theory depends on factors which are very hard to quantify such as how well it captures the intuitions of the designers. The main contribution of this thesis is not the modelling system itself, but the limiting process which makes such a system possible

by bridging the theoretical gap between discrete and continuous time representations of the world.

Thesis structure

The thesis is structured in two parts. The first part takes a philosophical perspective on the modelling issues involved in representing situated agent systems which contain discrete and continuous components. In this section the problems which the thesis addresses are described intuitively and several alternative approaches are explored. The issues surrounding the modelling of change through time are discussed in detail.

The second part of the thesis builds on this analysis of the modelling problems to develop a theoretically sound description of a novel approach based on applying a limiting process to a discrete automaton. After presenting some examples which demonstrate the compositionality of the approach, the modelling system is compared to existing hybrid system representations.

Conventions

This thesis draws on work from a number of different disciplines including mathematics, computer science, artificial intelligence, engineering and philosophy. Since the ideas contained in it may be of interest to workers from any of these areas, I have tried to make as few assumptions as possible about the reader's backgrounds. A summary of notation can be found at the end of the thesis (pp 172) along with an index of technical terms.

The term *continuous* is used in two distinct ways in this thesis. This ambiguity is unfortunate but follows from two distinct ways in which it is used in different bodies of literature. Its first meaning is the classical mathematical description of a function that describes an unbroken curve. The second use is an adjective derived from the noun *continuum* which refers to the topological properties of a connected, dense set such as the real numbers. It is used in this second way to refer to the following:

- *continuous time* referring to the topology of events describing the evolution of a

system as a continuum,

- *continuous automata* meaning an automata model whose output is a continuum,
- *continuous temporal metric* meaning a metric over time which is a continuum,

In modern scientific writing it can be considered bad form to use the first person singular presumably because it can lead to an apparent lack of objectivity. This leads to text that over uses the passive voice or the authorial 'we'. I intend to follow the well established tradition in philosophical writing and use the first person where I feel it is appropriate, particularly to emphasize when a statement is a personal opinion. I will use the second person to refer to you, the reader, and use first person plural to refer to both of us.

Part I

Philosophy

1

Modelling interaction

A theory has only the alternative of being right or wrong. A model has a third possibility: it may be right, but irrelevant.

— *Manfred Eigen*[1]

The difference between *craft* and *engineering* is the principled use of models. Modelling in this context refers to the activity of deriving mathematical relations over variables representing physical quantities in a system. This chapter attempts to characterize the types of systems which are of interest here and to justify the modelling approach that is adopted.

1.1 The study of interaction

In a special double issue of *Artificial Intelligence* (1995) Phil Agre characterized what he saw as a new emerging area of Artificial Intelligence which he summarized as follows:

Using principled characterizations of interactions between agents and their environments to guide explanation and design. [Agre 95]

For a long time Artificial Intelligence was focused on the problems of describing an intelligent system in almost complete isolation from an environment. Increasingly many

[1] Jagdish Mehra (ed.) The Physicist's Conception of Nature, 1973.

researchers, particularly those involved in robotics, have been trying to address the problems, and take advantage of the opportunities, that arise when an agent interacts with a real environment.

When you begin to look at the dynamics of the interaction of a physical agent with its physical environment, the importance of modelling complete behavioural loops becomes clear. Take for example an aircraft autopilot. The task of the autopilot is to adjust the aircraft's controls in order to maintain a particular speed, altitude and heading.

Suppose you are presented with a design for new autopilot software. How could you determine whether or not it will work? It is impossible to answer this question given just the program, you would need to understand the complete system including the aerodynamics, system sensors *etc.* Of course there are questions that can be answered about the software in isolation but the most interesting and useful behaviour is a product of the interaction between the various system components and the environment and cannot be attributed to any individual piece.

The importance of representing the complete system is not just limited to tightly coupled dynamical systems either. Studies such as [Kirsh 95] suggest that we make continual use of our interaction with the world, in order to simplify everyday tasks. This is not surprising when you bear in mind that the world around us has a great deal of inherent structure. We face in a single direction at a time, we can only directly manipulate objects within reach, we can only sense the world in the immediate vicinity. These limitations simplify the reasoning that we, as agents, have to perform to carry out tasks that involve interacting with the world around us.

The problem that is then to be addressed in this part of the work is how to go about modelling the interaction of an agent with its environment. There is a very wide class of system that can be usefully modelled in this way. Such systems we will call *situated agent systems.*

1.1.1 Situated agents

In order to provide a general vocabulary for talking about components of the systems that we will discuss in this thesis, the concept of a *situated agent* is intro-

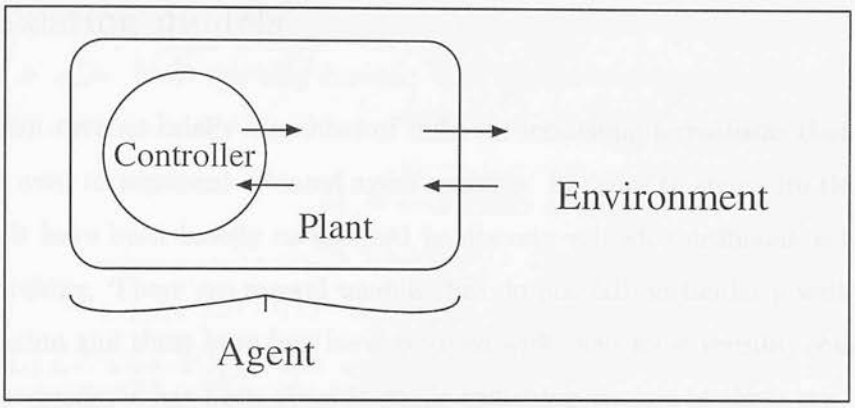


Figure 1.1: A situated agent system

duced [Rosenshein & Kaelbling 86, Sandewall 94]. A *situated agent system* is composed of three components: a *controller*, a *plant* and an *environment* (see Figure 1.1). The *plant* represents the physical system that is being controlled by the controller. The environment represents a source of uncontrolled and unpredictable disturbance.

Consider, for example, a lift (or elevator). The lift *controller* will typically be some piece of programmable logic which remembers which buttons have been pressed and can direct the lift to go to the appropriate floor. The *plant* is the lift itself including the lifting mechanism and all the doors, door sensors and buttons inside and outside the lift. The *environment* consists of the users of the lift who provide input by calling the lift, selecting a floor, getting trapped in the doors, *etc.*

The boundaries between these components are not always very clear. For example a lift will have a continuous control mechanism that allows it to smoothly approach its destination. This continuous control system might be considered part of the controller or part of the plant depending on the intentions of the model's designer. In closed systems (*i.e.* ones without inputs or outputs) there may be no environment, and of course, an environment may contain any number of agents.

The purpose of this system description is to define terms, not to suggest an architecture. Most embedded systems – regardless of their control architecture – will have parts that can be identified with these components.

1.2 Existing models

This section reviews briefly a number of different modelling formalisms that have, or could, be used to represent situated agent systems. In order to structure this section the models have been loosely categorized as discrete valued, continuous valued[2] and mode switching. There are several models that do not fall particularly well into this categorization and these have just been grouped with their most sensible companions. Particular emphasis has been given to mode switching models as these are the focus of this work.

1.2.1 Discrete valued models

Discrete valued models are models in which the parameters can only take values from a discrete set. In some, but not all cases, this set of values will be finite.

Rosenschein and Kaelbling [Rosenshein & Kaelbling 86, Rosenschein & Kaelbling 95] developed a logical representation based around situated automata. Their logical formalism can be used to precisely specify individual elements in the world and their relations to each other from the perspective of an agent. The descriptions can be used to compile efficient implementations of agent controllers that preserve the logical relations. The biggest drawback is the limited expressibility of the logic which does not seem suitable for representing dynamic properties of the world. Important related work was carried out by Eric Sandewall whose book *Features and fluents* [Sandewall 94] develops a broad categorization of logics for representing the interactions of agents with their environments. It is argued that different logics should be applied to different classes of world depending on the nature of the features that an agent might encounter. An important feature for example is *inertia*. A world is inert if properties in it are preserved unless changed by the agent. Inert worlds are easier to reason about. Sandewall's classification extends to worlds which are represented with continuous values and in continuous time. However this part of the work has not yet been developed very far.

Lyons *et al.* [Lyons & Hendriks 95, Lyons & Arbib 89] approached this modelling problem with a process algebra. The algebra is designed specifically for the purpose, and

[2] Not to be confused with discrete and continuous *time* models, see Chapters 4 and 5

the agent's described within this algebra can have behaviours added to their repertoire online by a higher level planning process. The system is so designed that even with incremental additions certain properties of the systems can be guaranteed.

In many ways the use of process algebras would seem appropriate for this problem. The interaction of agent and environment is certainly a communication. It is clear from the first chapter of his book that Milner [Milner 89] intended to keep the concepts of agent and of communication in CCS sufficiently abstract that they could apply to modelling of systems such as these. Indeed some interesting theoretical work has been carried out using CCS and its variants [Toffs 89b, Moller & Toffs 89, Moller & Toffs 91] to model the behaviour of ants and their group interactions [Toffs 89a, Toffs 91].

Amongst the modified versions of CCS that were used in these studies is SCCS, which replaces the asynchronous communication of CCS with a synchronous model[3]. Synchronous communication is at the heart of a family of languages called the *synchronous languages* which have been specifically designed for the construction of discrete embedded control systems. Synchronous languages [Halbwachs 93] come in several types, ESTEREL [Berry & Gonthier 92, Berry 92] is a language based on a process algebra of the same name. LUSTRE [Halbwachs *et al.* 91a] and SIGNAL [Benveniste *et al.* 93a] are both functional, data flow languages [Benveniste *et al.* 93b, Halbwachs *et al.* 91b] (SIGNAL has already been used to represent hybrid systems [Benveniste *et al.* 93b]). Argos [Maraninchi 91] and Statecharts [Harel 86, Harel 87] are graphical languages based on hierarchical automata representations. Synchronous languages are very important to this thesis. They are used to describe the discrete sequences of machines over which limits are taken.

Underlying all the process algebras and synchronous system representations are automata [Hartmanis & Stearns 66]. Automata do not represent complex systems very elegantly. It becomes difficult for a human to understand what an automata is doing once it gets beyond a small number states. However they are important in that they provide the underlying semantics for all these models.

[3] The tradeoffs between synchronous and asynchronous communication will be discussed in section 2.5

1.2.2 Continuous models

Continuous dynamical systems are used extensively in control engineering. They are particularly effective at representing real world dynamics such as chemical plants, aeroplane flight and robot motion. They are however very limited in their ability to deal with discrete changes in system parameters such as the end points of the motion of a robot arm. Most of the work of control engineering focuses on the continuous control problem that occurs between such end points.

It has been argued by anti-representationalists in the Artificial Intelligence community such as [Smithers 92, Smithers 94] and [vanGelder 94] that agent environment interactions are most appropriately modelled using continuous dynamical systems. It is suggested that conventional models of computation are inappropriate because they focus on calculation and representation. Their alternative is continuous dynamical systems which focus on change and interaction.

In fact reactive systems (discussed in section 2.1) are a more appropriate alternative that would answer most of their objections [Westhead 95]. They too focus on change and interaction but are discrete. Nonetheless some interesting work has been done using this approach, notably that of Beer [Beer 95] and Steels [Steels 87].

1.2.3 Mode-switching systems

At the risk of oversimplifying, the previous two sections can be summarized as follows. Discrete models from computer science are very well understood and can be manipulated to design very complex systems. However, they are limited in their ability to express continuous change. Continuous models from engineering, on the other hand, are very expressive but can only be usefully manipulated in limited simple circumstances (essentially linear systems). *Mode-switching* or *hybrid* systems bring together continuous representations with discrete switches between modes of operation. If this is done successfully it will combine the advantages of discrete and continuous models.

It might be argued that, in fact, no real systems display perfect switching behaviour; all changes in the real world occur continuously. The question is however: what is the most effective model? The systems that interest us here are continuous dynamical

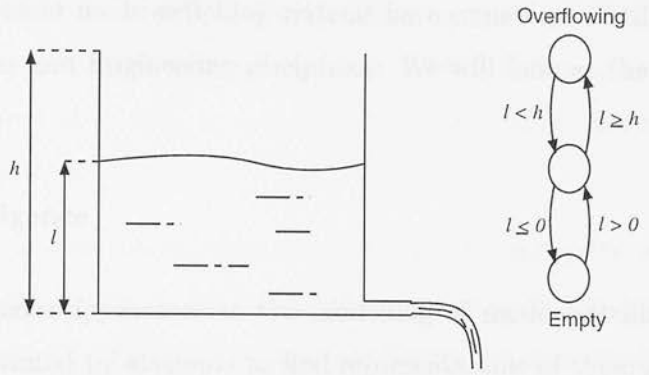


Figure 1.2: The discrete states of a water tank

systems which exhibit occasional, catastrophic dynamical changes, at which points the dynamics change orders of magnitude faster than at other times. This behaviour is often most usefully modelled as a discrete switch.

Switching behaviour is ubiquitous in engineering systems, it occurs naturally in many forms. Consider the water tank shown in Figure 1.2. The mode-switches in this system are inherent. When the water level is between h and 0 the tank's behaviour conforms to a simple dynamic equation. When the level reaches either of these end points, the system's behaviour changes abruptly to be either overflowing or empty. There are of course many other examples of naturally occurring switches including end points of motion of say a robot arm, or the electro-chemical firing of a neuron.

In addition to naturally occurring switching, devices are often designed to exhibit near perfect switching behaviours; examples include relays, transistors, clutches, free-wheeling devices, hydraulic valves *etc.* One reason for introducing switching behaviour into a system is economy. For example, the near perfect switching capabilities of MOS (Metal Oxide Silicon) transistor technology allows for the construction of smaller, cheaper and more powerful power supplies. A second reason is that the use of discrete switches also makes it easier to design systems that maintain complex logical properties. It is hard to imagine that something as complex as a modern microprocessor, for example, could be designed without a model based on discrete switches[4].

[4] Of course the brain *evolved* without recourse to modelling, but can still be seen to use switching technology

Attempts to represent mode-switching systems have come from Artificial Intelligence, Computer Science and Engineering disciplines. We will look at the contributions of each field in turn.

Artificial intelligence

Artificial Intelligence approaches to the modelling of mode-switching systems have been largely motivated by attempts to find representations of these systems that can be manipulated by a computer. *Qualitative Physics* is an approach to this area that attempts to characterize physical interactions using only a qualitative representation of their parameters. Producing a qualitative representation involves carving up the continuous space into suitable discrete pieces. The states in the water tank 1.2 are good examples. The qualitative abstraction of this model might include three distinct states: empty, overflowing and in-between. Qualitative reasoning might allow you to say that if the model was in the in-between state and the level l was decreasing then at some point in the future the model would enter the empty state.

Examples of this work include the component based approach [deKleer & Brown 84, deKleer 84], the compositional approach [Falkenhainer & Forbus 91] and the graphs-of-models approach [Addanki *et al.* 91].

Other AI approaches have centred around the use of logics to represent mode-switching properties, examples include [Sandewall 89].

Computer science

Hybrid systems have been receiving increasing interest from computer scientists over the last 20 years. There are two main approaches taken, logical representations and automata representations. Logical representations provide high level, concise and expressive description languages. Automata representations are more concrete and suitable for modelling system dynamics; they allow automatic verification techniques although these are currently quite limited in scope. For a theoretical comparison between logic and automata approaches see [Bouajjani *et al.* 95, Bouajjani & Lakhnech 96].

There are a number of logical representations that have been applied to this area, they

include: timed temporal logics TPTL [Alur & Henzinger 89] and MITL [Alur *et al.* 91], the Duration temporal logic DTL [Bouajjani *et al.* 93], the Calculus of Durations (extended in [Chaochen *et al.* 92]), temporal logic of actions TLA [Abadi & Lamport 91] and TLA$^+$ [Lamport 93].

The closest work to that presented in this thesis are the automata approaches. Each discrete state of the automata describes a dynamic mode of the system. The automata transitions are discrete switches that take the system from one mode to another. Examples of these systems include Hybrid Transition Systems [Nadjm-Tehrani 94] the Nerode-Kohn model [Nerode & Kohn 93] and Hybrid Automata [Alur *et al.* 95] (this latter work is returned to in more detail in Chapter 14). Simpler timed automata can also be used [Maler *et al.* 92, Nicollin *et al.* 92].

All the computer science approaches betray their computational origins by working with a sequential, *i.e.* discrete, perspective of time, despite the continuous nature of the systems being modelled. This allows analysis of the systems using the same mathematical tools used for sequential machines, but leads to some inelegant consequences which will be discussed at the end of Chapter 4.

Engineering

The engineers also have two subfields that address mode-switching systems: *physical modelling* and *control.*

Physical modelling is a branch of engineering that focuses on systematic approaches to the extraction, construction and manipulation of models of physical systems, of particular interest are modular techniques for representing large, complex systems. An important formalism in this subfield is that of *Bond graphs* which represent systems in terms of the energy transfered between components. This high level of abstraction allows Bond graphs to be applied to electrical, hydraulic and mechanical systems. It also abstracts away a number of causality issues that are at odds with the principles of modularity, allowing the causality to be decided after a model has been composed (see [Strömberg 94] for more details). There have been several attempts to introduce switches to bond graph representations including [Gebben 81, Karnopp 83,

Karnopp 85, Dauphin-Tanguy *et al.* 89, Rinderle & Subramaniam 91, Ducreux *et al.* 93, Broenick & Wijbrands 93, W. Borutzky & Wijbrands 93, Strömberg 94, Söderman 95].

Control engineers have also approached the problem of mode-switching systems, but from a more pragmatic perspective, typically with the intention of analyzing or designing a specific class of system. These approaches have grown up by introducing mode transitions to continuous dynamical representations, examples include [Tavernini 87, Back *et al.* 93, Antsaklis *et al.* 93, Brockett 83, Branicky *et al.* 94].

Unlike the computer science models these formalisms use continuous representations of time.

1.2.4 Relationship to this work

This thesis is concerned with the development of a new approach to the modelling of mode-switching systems which involves building a discrete model that approximates the continuous behaviour of the system in time steps parameterized by a single bounded variable Δ. The idea is then to make the model precise by allowing the bounds on Δ to tend to zero. The resulting system falls somewhere between the computer science and engineering models. On the one hand rather than describing sequence it describes a continuum similar to the engineering models. On the other it is constructed from an abstraction of the synchronous languages and so shares with them important aspects of compositionality and verification methodology.

The focus of the work has been on solving the mathematical problems associated with describing a suitably robust and general limiting process, rather than on trying to develop a design tool. Nevertheless the approach has a number of practical advantages:

- every continuous model is guaranteed to have a discrete approximation of arbitrary accuracy,
- furthermore, that discrete approximation can be implemented using existing compilers to simulate the system's behaviour,
- both continuous model and discrete approximation can be composed with discrete controller models (or indeed implementations if they have been implemented in

a synchronous language) to provide complete system descriptions,

- because of the strong compositional semantics, properties of the composed systems can be verified using the technique of synchronous observers [Halbwachs *et al.* 93]. In the case of simple (discrete) properties, this verification process can be carried out automatically with existing tools [Westhead & Nadjm-Tehrani 96].

From a modelling point of view, this approach differs from many others in that the importance of a modular composition of models in which real controller implementations (in an appropriate synchronous language) can be included.

More importantly the approach breaks new theoretical ground in taking the limit of a sequence of automata. It brings closer together the understanding of discrete and continuous processes and opens up the possibilities of extending discrete sequential techniques to the study of continuous change.

1.3 Summary

In this chapter the modelling objective was identified as the representation of situated agent systems. A large number of possible representation schemes were discussed. These fell broadly into three areas:

- discrete models,
- continuous models,
- and mode-switching models.

The third of these was identified as the most promising for complete agent systems, and an automata based representation was favoured because of the focus on dynamics and the possibility of automatic verification.

The next chapter looks specifically at issues in finding automata representations for situated agent controllers.

2

Agent Control

Intelligent control exerts influence without appearing to do so. Unintelligent control tries to influence by making a show of force.

— *Lao Tzu*[1]

This chapter looks at issues involved in modelling the controller of a situated agent system. The controller model is used as starting point for this work because in order to model the whole system the interface between the controller and the rest of the system is very important. From the previous chapter it is assumed that the controller will be a discrete automaton and it turns out that several of the criteria for choosing it as a controller also make it suitable as a starting point for a limit machine.

2.1 Reactive systems

What sort of model is appropriate to describe the controller of a situated agent? In the last decade several workers both in intelligent robotics [Malcolm 91, Smithers 92, vanGelder 94] and computer science [Harel & Pnueli 85] have recognized the short comings of classical models of computation in real time applications. Harel and Pnueli [Harel & Pnueli 85] propose a distinction between what they call *transformational systems* and *reactive systems* as follows:

[1] from *Tao Teh King* c. 600 A.D.

- A *transformational* system involves information processing. In modelling such systems attention is focused on the capability of a system to return with the answer within a reasonable time, described as a function of input size. Classical models of computation, such as Turing Machines, tend to be transformational.

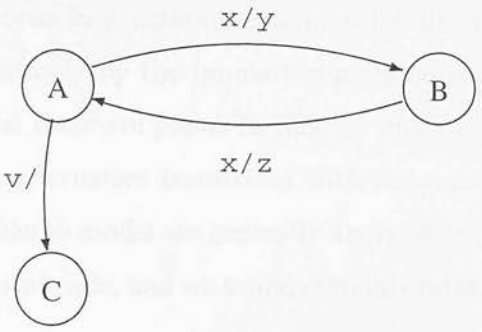
- The operation of a *reactive* system, in contrast, is primarily about its interaction with the environment. Such systems include digital watches, microwave ovens, video recorders, computer games, word processors, operating systems, microprocessors, communications systems, *etc.* Reactive systems would ideally respond instantaneously to their environment.

This is not to suggest that transformational models don't interact, or that reactive systems don't process data, but this distinction provides different perspectives that can be taken on a system that reflect the designer's intention and should influence the way it is modelled.

Controllers of situated agents, by their interactive nature, are going to be most effectively modelled using modelling descriptions that are reactive in nature. There is a range of formalisms which meet this requirement, including process algebras such as CCS and the synchronous languages discussed in Section 1.2.1. At the heart of all these modelling descriptions is the automata model.

2.2 Automata

An *automaton* or *machine* is a representation of a dynamical system. The simplest automaton is a finite state machine. A finite state machine has a fixed number of states, a transition function describing how it moves from one state to another with inputs and outputs associated with transitions. Figure 2.1 illustrates a standard way of depicting machines as graphs where states are represented by nodes and the transitions between them by labelled directed edges. The labels on the edges show the inputs and outputs associated with an edge separated by a '/' sign. For example, suppose this machine is in state A. If it receives the input signal `x` it will emit the output signal `y` and move to state B. The occurrence of an input, and the output it triggers will be referred to as an *event*. The transition from state A to state C illustrates that



the

Figure 2.1: A finite state machine

transitions can occur with no output.

Finite state machines can be extended to describe more general systems including plant and environment by adding values to the signals, and allowing simple mathematical operations and functions to be carried out over them. When valued signals are added the machines "state" strictly becomes not just the node of the graph it is in but also the values of those signals. However in this thesis *state* is always used to refer to the system's discrete state and *current event* is used to refer to the signal values.

It would appear that the operation of such systems is inherently sequential; inputs and outputs will occur in sequence so each event (apart from the first) will have a unique predecessor. The aim of this thesis is to demonstrate how the automata model can be extended to represent continua of events, and thus, how it can be used to model not only the controller, but also continuous changes in the plant and environment.

The rest of this chapter outlines some of the issues that need to be addressed in choosing an appropriate automata model.

2.3 Determinism

Determinism refers to whether the relation describing output and next state of the machine is function respect to the machine's state and current input or not. If it is a function then the machine is deterministic and the outcome for a given input and state is fixed, if not then machine is nondeterministic and one of many possible outcomes

may occur. In other words in a deterministic machine the state and output of the model are determined entirely by the inputs the machine receives and the time. In a nondeterministic model there are points in time at which a nondeterministic choice must be made between alternative transitions with the same inputs. The physical systems that we would like to model are generally above the quantum level and so can be considered to be deterministic, and we would certainly intend to build deterministic controllers.

Nondeterminism can be used to model uncertainty in the plant or environment. It is considered by some [Nadjm-Tehrani 94] to be a very important tool in modelling. Because there is a duality between nondeterminism and input, nondeterministic choices can be made deterministic by adding extra *oracle* input signals which choose between the alternatives. An oracle is a theoretical machine which can accurately predict the outcome of nondeterministic events. So in the presence of an oracle the machine is deterministic. We cannot, of course build an oracle but since inputs are by their nature undetermined (in advance) this model could be thought of as having the same range of behaviour as the nondeterministic one.

For example consider the machine in Figure 2.2 (a). This is a very simple machine which on input of the signal `x` will go nondeterministically to either state B or C. Figure 2.2 (b) shows the same machine but with oracle input signal `a` added. This new model is strictly deterministic, however since we can never know when this special input will occur its behaviour is effectively the same as before. Moreover from the point of view of verification the two machines are largely considered to have the same properties.

This is all that is required to model uncertainty in the world. For this reason nondeterministic models are not considered explicitly here and this simplifies the theory.

2.4 Input and output

Many modelling systems (such as CCS [Milner 89]) ignore the distinction between input and output, treating a communications event as a synchronization in which it is unimportant who transmitted and who received. However, when specifying the behaviour

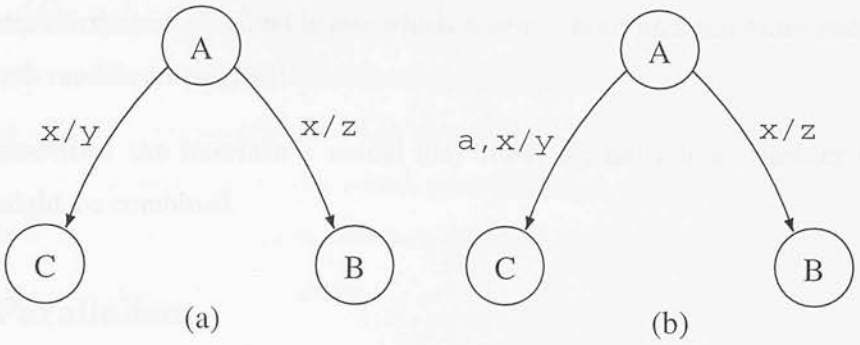


Figure 2.2: Removing nondeterminism with oracle input signals

of a situated agent it is desirable to be able to make assumptions about the behaviour of the environment. For example, if we wanted to prove that an agent displayed a property P under environmental assumptions A, we would attempt to prove "$A \Rightarrow P$" let us call this Φ. So A is *not* being used as a specification of the environment's behaviour, if it were then the property Φ would fail if the environment failed to satisfy A, rather, it is a prerequisite for testing Φ. If the environment fails to satisfy A, then Φ is immediately true. Thus the agent's behaviour is only tested against P if the environment conforms to the assumptions A. In order to write specifications which include environmental assumptions it is necessary to be able to distinguish whether a signal was generated by agent or environment [Abadi & Lamport 90, Halbwachs *et al.* 93], so it is necessary to describe the interfaces in terms of inputs and outputs.

An important topic in automata theory is the question of language acceptance. In this study the automaton is taken to be a state machine with inputs but no outputs, and one distinguished accepting state. The question is, when started from an initial state, and given a sequence of inputs, will the automata end up in the accepting state or not. The set of input sequences that lead to the automata ending in the accepting state is called the *language* that is accepted by the machine. Let us call models which have no outputs *accepting.*

A second class of model is one which has no inputs and simply generates an output, examples of these can be seen in [Ramadge & Wonham 87, Ramadge & Wonham 89]. Let us call such models *generating.* If a generating machine is deterministic then it describes a single output sequence.

Finally our third class of model is one which accepts input and generates output. Let us call such models *interactive*[2].

Having described the interface a model may have, we must now consider how two models might be combined.

2.5 Parallelism

The composition of models in parallel is extremely important since it allows modular construction. Modularity in a formalism is highly desirable. It allows models to be built up from different reusable components, making them easier to construct and analyze. Of course modularity does not require parallelism, but, in any reasonably system the ability to describe concurrently interacting components is desirable [Westhead 92]. In particular in an agent system it is clearly useful to be able to exchange discrete controllers in the same hybrid model, or to introduce an agent into a new environment without having to rebuild the entire model each time. This requires a strong, well understood composition semantics.

Parallelism is poorly understood by some research communities. It has been suggested implicitly in the robot literature that the use of parallelism in the implementation of a robot controller can make it capable of behaviour that a serial implementation could not manage. In information processing terms the only advantage that a parallel implementation can lend is speed. A multi-processor computer has the potential to work faster than a single equivalent processor. However, whilst speed is significant in the construction of reactive systems none of the architectures proposed for parallel design of behaviour based robot controllers address efficient use of their parallel resource [Brooks 89, Malcolm *et al.* 89]. Without care, communications overheads can lead to a parallel implementation running slower than a serial one. Parallelism can also introduce unforeseen and unintended behaviour.

The real advantage of parallelism, in this area, is in modelling, particularly in modelling the controller. A typical reactive system may have to execute several tasks in parallel.

[2] The term *interactive* is chosen over an alternative such as *transducer* to emphasize the reactive nature of the system

It is perfectly useful for a designer to be able to consider each concurrent task in isolation rather than having to consider the global behaviour at each stage. It is not important whether the parallel model is then run on multiple processors, on a single machine using threaded execution, or even passed through a compiler that turns it into a serial piece of code. This descriptive parallelism is fundamentally different from the execution parallelism discussed above.

In computer science, models of parallelism come in one of two flavours depending whether the communications are synchronous or asynchronous. Just as there are misunderstandings about parallelism itself, so there are also misunderstandings about these terms. In an asynchronous model, concurrent communication events are modelled as taking place one after the other but the order in which they occur is nondeterministic. So if we have two signals `a` and `b` emitted in parallel by an agent in an asynchronous model, this means that the agent will either emit `a` followed by `b` *or* `b` followed by `a` but before the event actually happens we cannot know which will occur. The assumption is that events will never synchronize, but two events in parallel are so close together that it is undetermined which will come first.

In contrast, in a synchronous model, concurrent events are modelled as really happening at exactly the same instant. A common misconception is that synchronous systems presuppose a synchronizing clock. This is not the case. The synchronization that takes place is the synchronization of output with input.

At first glance, you might suppose that the synchronous model seems more credible as a model of parallelism since it allows two events to occur at the same instant. However it has the apparent drawback that computation is modelled as occurring instantaneously *i.e.* the output is modelled as being synchronized with the input that triggered it. Infinitely fast computation is difficult for anyone to swallow.

It has to be remembered that both these approaches are modelling approaches which make abstractions from the real world. It turns out that a system designed with synchronous parallelism can work perfectly well in an extremely asynchronous environment. Gerard Berry's digital watch program [Berry *et al.* 91], for example, was decomposed and successfully run across the Internet on different sides of the Atlantic.

The synchrony assumption allows the compiler to determine a causal preordering on events that is preserved even if the events do not occur simultaneously in practice. Likewise a system designed using an asynchronous parallel model need not preclude events in an implementation really synchronizing.

There are other differences between the two models. Combining two asynchronous models leads to a rapid explosion in the number of states and transitions to be considered because all of the nondeterministic possible orderings of events must be considered. In a synchronous model this problem is significantly simplified and, in tightly coupled systems, combining machines in parallel can actually lead to a reduction in the number of reachable states [Westhead & Nadjm-Tehrani 96].

On the other hand, modelling computation as occurring instantaneously can lead to causality problems. These arise in the situation where a conflicting sets of events are triggered or signals could be responsible for triggering themselves. This is a much studied issue and details can be found in [Halbwachs *et al.* 93, Berry & Gonthier 92, Berry 95].

The work in this thesis is based on a synchronous description of parallelism. The original motivation for this decision came from the theoretical advantages of smaller combined machines, and the powerful synchronous tools explicitly designed with reactive control in mind [Westhead 95]. However, in retrospect, a synchronous view of parallelism is necessary in order for the limiting process to work. One of the consequences of an asynchronous view is that time is represented by an incomplete (partial) ordering of events. The limiting process makes use of the temporal structure, and requires a total ordering on the events, which is of course provided by synchronous parallelism.

It may be possible to extend the limiting process to work for partially ordered events. On the other hand, it may be easier to investigate simulating asynchronous parallelism using the existing model. Milner [Milner 83] showed that synchronous systems could simulate asynchronous ones, but not the other way around. By assuming synchrony, just as by assuming determinism, the temporal structure of the models is simplified making it easier to define a limiting process. Adding these extensions lies beyond the scope of this thesis.

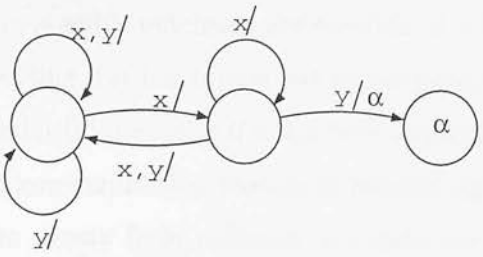


Figure 2.3: A synchronous observer

2.6 Verification

Adopting a synchronous approach to parallelism allows us to use *synchronous observers* [Halbwachs *et al.* 93] to express and verify properties. The synchronous observer is a special kind of accepting machine which, instead of having an accepting state, has an alarm state. Any transitions to this alarm state result in the emission of an alarm signal 'α'. This machine can be used to express safety properties of a system. For example consider the observer in Figure 2.3. Observers watch the input-output sets of systems. In this simple case the input-output set is $\{\mathtt{x}, \mathtt{y}\}$. This observer allows any sequence of these except the event $\{\mathtt{x}\}$ followed by the event $\{\mathtt{y}\}$ at which it raises the alarm.

When our observer is combined in parallel with the system under observation, the result is a single machine which expresses their combined operation. If, in this machine, there is no reachable transition with an α signal on it then the system will not violate the property. If such a transition exists then not only does it demonstrate that a violation can occur, but all the paths that lead to that transition illustrate the sequences of events and states that will lead to that violation.

2.7 Summary

This chapter discussed a number of general features that might be possessed by the automaton used to control a situated agent.

Determinism refers to whether the output and next state of the machine are defined as a function or a relation with respect to the machines state and current input. If it

is a relation one of many possible outcomes are possible, if a function the outcome is fixed. A machine is accepting if it has inputs but no outputs, it is generating if it has outputs but no inputs and it is interactive if it has both inputs and outputs. Synchrony and asynchrony refer to communication models of parallel agents. If communication is synchronous then two events from different machines are modelled as occurring at the same instant (even if one caused the other), in contrast if communication is asynchronous then concurrent events are modelled as occurring one after the other but in a nondeterministic order.

Throughout this thesis a number of different automata models are developed. All of them will be deterministic synchronous machines.

The focus of this thesis is the way in which time is represented in modelling situated agent systems. The next three chapters look at different aspects of the representation of time, before we move on to look at the mathematics.

3

Time

Alice sighed wearily. "I think you might do something better with the time," she said, "than waste it asking riddles with no answers."

"If you knew Time as well as I do," said the Hatter "you wouldn't talk about wasting **it**. *It's* **him**."

"I don't know what you mean," said Alice.

"Of course you don't!" the Hatter said tossing his head contemptuously. "I dare say you have never even spoken to Time!"

"Perhaps not," Alice cautiously replied: "but I know I have to beat time when I learn music."

"Ah! that accounts for it," said the Hatter. "He won't stand beating..."

— *Lewis Carrol*[1]

3.1 What is time?

The concept of time presents us with a strange enigma. On the one hand, we use time constantly in everyday thought and communication. It is very deeply ingrained in the way we think about our lives and the world about us. Children easily develop a usable notion of time. Indeed there are few abstract concepts that are as well used or implicitly

[1] Alice in Wonderland

understood as time, and yet time continues to defy philosophical analysis. Many books have been written on the subject from a philosophical [Grunbaum 63, Landsberg 82, Seddon 87, White 92, Gale 68] and an Artificial Intelligence perspective [Hajnicz 96]. Yet still it raises controversy and paradox at every turn. St. Augustine writes:

> *For what is time? Who can easily and briefly explain it? Who can even comprehend it in thought or put the answer into words? Yet is it not true that in conversation we refer to nothing more familiarly or knowingly than time? And surely we understand it when we speak of it; we understand it also when we hear another speak of it.*
>
> *What then, is time? If no one asks me, I know what it is. If I wish to explain it to him who asks me, I do not know.*
>
> — *Augustine* [2]

The first serious attempt to analyze the concept of time occurs in Aristotle's "Physics" [Aristotle 68]. He took time to be made up of a continuum of indivisible, preset now-moments. His analysis is still studied today, although many of the questions it raised remain unanswered.

3.1.1 Time and change

One of the issues that Aristotle tried and failed to explain was the relationship between time and change expressed best in the famous arrow paradox attributed to Zeno of Elea: The arrow in flight must be moving at every instant in time. But at every instant it must be *somewhere* in space. However if the arrow must always be in some one place, it cannot at every instant also be in transit, for to be in transit is to be *nowhere*.

It took the mathematics of Weierstrass to finally lay this paradox to rest. Weierstrass provided the $\epsilon - \delta$ tools that allow a rigorous understanding of the infinitesimal, and in so doing showed that the moving arrow is really always at rest. Bertrand Russell commented:

[2] (354-430) Tr 1948 ch XIV[Augustine 68]

> *Weierstrass, by strictly banishing from mathematics the use of infinitesimal, has at last shown that we live in an unchanging world, and that the arrow in its flight is truly at rest. Zeno's only error lay in inferring (if he did infer) that, because there is no such thing as a state of change, the world is in the same state at any one time as at any other.* [Russell 81]

This idea that time can be captured as a set of static moments is used later in describing our system models.

3.1.2 The unreality of time

Since the time of Aristotle many others have questioned the nature of time and modern paradoxes associated with the concept also haunt us. Perhaps the most famous and controversial paper on the philosophy of time this century was that by McTaggart [McTaggart 93] in which he argues that "nothing that exists can be temporal and therefore time is unreal." This surprising conclusion is generally felt to be flawed. Moore pointed out that "if time is false then there are no temporal facts: nothing is earlier or later than anything else. But plainly it is false that there are no temporal facts, for it is a fact that I am *presently* inscribing this sentence and that my breakfast yesterday *preceded* my lunch." Unfortunately it is still disputed where the error in McTaggart's argument lies.

McTaggart suggested that we essentially have two perspectives on time. The first is the notion of an instantaneous present that divides the past from the future, from this perspective we can speak of the relative position of an event in time with respect to our present, that is we can refer to tomorrow, last month, next Tuesday *etc.* This could be seen as a *relative* view of time. The second perspective on time he presents is in terms of fixed events which are temporally ordered thus we refer to the 12th of January 1714, which is before the 25th March 1999. This is an *absolute* view of time. The former he calls A series time, the latter B series (these concepts will be used later):

> *I shall give the name A series to that series of positions which runs from the far past through the near past to the present moment, and then from the present through the near future to the far future, or conversely. The*

> *series of positions which runs from earlier to later, or conversely, I shall call the B series.* [McTaggart 93]

His argument then proceeds by attempting to establish that B series time was dependent on A series time in the sense that without A series time, B series time can not exist. He then goes on to give an argument regarding the logical implications of tense, inherent in an A series perspective, which leads to a vicious circle when viewed as a B series. The argument is based around the following contradiction: a moment can only be in one of the past, the present or the future, and yet every moment in time will at some stage begin in the future, become present and then eventually past. Thus every moment has to be past, present and future, contradicting our first premise. The answer to this contradiction is, of course, that no moment is in the past, present and future simultaneously; rather there are three separate and non-contradictory statements that can be made for a current moment M, say, it was in the future, is in the present, and will be in the past. However McTaggart then responds that all this is saying is that M is in the future at a moment of past time, is past at a moment of future time and present at a moment of present time. This too, he argues, is inconsistent. This is because the answer sets up a second order time for which there are inconsistencies similar to his original objection. The answer to this second order situation sets up a third order time and this is where the infinite regress enters. For a more complete account of this argument, I refer the reader to the paper itself [McTaggart 93] and an insightful reconstruction of the argument by Mellor [Mellor 93].

The paradox arises from juxtaposing the two distinct perspectives on time. In the observer's perspective of moving through time given in the A series, events such as the death of Queen Anne are at one point in the future, they momentarily occur in the present and then move into the past. This idea is called temporal becoming and is strongly linked with time as being changing and dynamic. This is in contrast to the static, god-like view of time taken from the B series perspective in which time is static and changeless. Events are fixed and unmoving: Queen Anne died in 1714.

There are three approaches to answering the paradox. The first is to claim that time can be explain entirely by the B series, and to reject the idea of temporal becoming (Bertrand Russell was considered the father of this approach). The second approach

is conversely to reject the idea of B series time. One of the strands of this counter argument, presented by C.D. Broad, is that there are important ontological differences between the past and the future, "the future is open, a realm of possibilities where as the past is closed, a realm of actualities" [Broad 68]. The third answer to McTaggart is to accept both time series, but insist they are never confused. This position is taken by J. N. Findlay and J. J. C. Smart (both in [Gale 68]). Essentially, they propose that each explanation in its own right is satisfactory and the contradiction only arises when the two are used together.

3.2 Classifying models of time

This section considers *temporal systems*, system models that evolve through time, and attempts to tease apart and classify different features of these models that are often bundled together.

3.2.1 Temporal structure

Having presented some of the ideas as to the nature of time let us turn our attention to the problem of modelling a system whose behaviour evolves through time. What we are interested in is modelling the parameters that characterize the state of the system and their change through time. At an instant in time, which we will call an *event*, these parameters are assumed to have unique (static) values. Continuous change can be described, as Russell and Weierstrass suggest, by a continuum of such instants.

The behaviour of the system over time is thus described by an ordered set of events called a *flow*. The *structure* of a flow describes its topology. A topological property is defined as any property which is preserved under all continuous one-one transformations of that structure. So, for example, a circle is topologically equivalent to an ellipse or a square or a triangle, but topologically different from a bounded line. A circle can be mapped to the other shapes by stretching it; mapping it to a line would involve cutting it, thus introducing a discontinuity.

A very important feature of a flow is the completeness of its ordering. This thesis deals only with totally ordered models of time but partial orders are possible and could have

interesting theoretical properties. Essentially partial ordering implies asynchronous parallelism in a model.

Another topological property, *density*, determines how tightly packed together the instantaneous events in the flow are. There are two structures that we will consider. Chapter 4 looks at *discrete* time in which the points are arranged in a sequence, such that each one (apart from the first) has a unique predecessor. In Chapter 5 we will look at flows of time in which the events are dense, so that between any two events there is a third, and consequently no points have unique successors or predecessors.

A *continuum* is a flow which is dense and *connected*. A connected flow is one in which not only are the points tightly packed together, but there are no gaps. This property is described in more detail in Chapter 5.

The final structural property that is of interest is whether a flow has end points. In other words in the ordering there is a *start* point which is less than any other point, or an *end* point which is greater than any other point. A flow with a start point is said to be *closed at the start* and a flow with an end point is said to be *closed at the end*. Most of the flows that we will consider will be closed at the start but may be open or closed at the end because although we will usually be modelling systems from some fixed starting point, the end point may or may not be fixed.

3.2.2 Synchrony labels and temporal metrics

Closely associated with the structure of a flow are its *synchrony labelling* and *metric*. A flow must have a structure, but these properties are optional refinements which increase the descriptive power of the flow.

A synchrony labelling relates events in one flow to events in another so that the flows can be combined. It allows models to be combined in parallel by providing a mechanism for identifying simultaneous events. When two models are being composed, a one-one order preserving (isotonic) function is needed to match up simultaneous events from each model. In a synchronous language, for example, synchrony labellings are provided implicitly by the signal names.

A metric relates events within a flow by providing a measure of distance in time between

any two events. It provides a yardstick against which rate of change can be measured. Typically the metric is defined by identifying the flow with a monotonic set of values. Such values can also be used as a synchrony labelling and are then referred to as a *temporal metric.*

Synchronous languages do not possess metrics. Metrics can be added to a model; indeed more than one can be used (referred to as *multiform time*). However in a synchronous language time is represented as just an ordered sequence of changes.

Separating out structure, synchrony labels, and the metric can lead to confusing ideas of time. For example, what is meant by time in which the ordering of the metric does not correspond to the ordering of the flow's structure? Synchronous languages are examples of models without metrics. Special relativity theory provides examples of models (see Chapter 13) in which the synchrony labelling of model components differs from the metric they use.

The temporal metric in most cases is effectively a time stamp on an event. In this thesis these time stamps are simply considered as special parameters of events. (The parameters of the models described in the second part of this thesis are given as signal values and the temporal metric is given a special signal t.)

3.2.3 Parameter assignment

Modelling a temporal system involves describing the values of parameters at each event. The same descriptions can be given in different ways depending on the nature of the model.

Consider Figures 3.1 and 3.2 which shows three representations of systems changing over time, Figure 3.1 (a) describes x as a function of whole number time t, (b) describes x as a difference equation and (c) is a finite state input/output machine.

(a)	x	$=$	$2^t,\ t \in \mathbb{N}$
(b)	x_i	$=$	$2x_{i-1}$

Figure 3.1: System models

Models (a) and (b) could describe the same system, but the natures of the descriptions

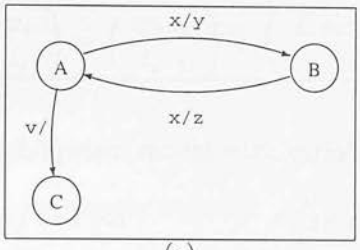


Figure 3.2: Another system model

are different. Model (a) gives a static view of time which I suggest corresponds to McTaggart's B series (absolute) time. The only way in which we see change is by comparing two static events. The value of x at time t is denoted by a function $f(t)$ in this case t^2, but it could just as well represent the health of Queen Anne in which case t in 1715, would have a different value when t is 1700. Model (b) on the other hand describes the system in terms of its change, from one value to the next, corresponding closely to McTaggart's A series (relative) time. Associated with the evolution of the difference equation is a natural notion of past, present and future. Inherent in the representation is the idea of *current* state, which is a reflection of some or all of the *past* states that it has been through and determines the *future* states. It should also be noted that Model (b) is under-specified. In order to describe Model (b) in the form of a function of time (like Model (a)) initial conditions must be specified.

Model (c), shown in Figure 3.2 is an interactive automaton, which is included to demonstrate a second notion of past, present and future in models, the ontological difference suggested by C. D. Broad and alluded to earlier in the chapter. Much like the difference equation, the interactive automata describes a dynamical system possessing an internal state, and consequently expresses the same idea of past present and future. However unlike the difference equation, the future of the automaton is unknowable in that it depends on the inputs that it *will* receive. So in considering the behaviour of the automaton there is an ontological difference between what has been and what is to come.

Let us refer to the stateless function of time exemplified by model (a) as being *static*, and the dynamical system (model (b)) as *dynamic*.

$$\text{(d)} \quad \begin{pmatrix} x_i \\ t_i \end{pmatrix} = \begin{pmatrix} x_{i-1} \\ t_{i-1} \end{pmatrix} + \begin{pmatrix} \Delta \sin(t_{i-1}) \\ \Delta \end{pmatrix}$$

Figure 3.3: A system model with variable step size

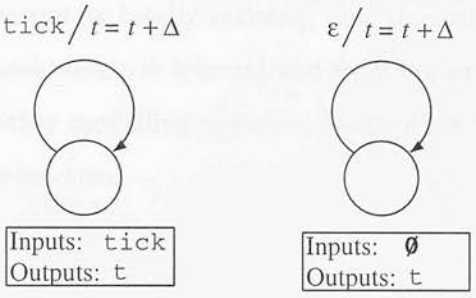


Figure 3.4: Reactive and proactive time machines

As we noted in the last section the temporal metric, if it exists, will be defined as a special parameter of the model. An important distinction can be drawn between models in which the temporal metric is *internal*, *i.e.* described with respect to a change in internal parameters, or *external*, where the values are imported.

Consider model (d) in Figure 3.3. This model has a single parameter Δ which can be varied to alter the temporal size of each step the model takes. In a model with an internal metric, the value of Δ is determined within the model. In a model with an external metric, Δ is an input to the model which cannot be determined in advance.

3.2.4 Impetus

Just as the metric can be internal or external to a model, so can the impetus for change to occur. Consider the two machines in Figure 3.4. The first shows changes occurring *reactively* in response to an input signal `tick` from the outside world. The name of the input is not important. In the standard reactive model change is triggered by any input. The second model shows change occurring on a transition labelled with an ε. This is a symbol used to label transitions that can occur spontaneously. In contrast to the term reactive, I call such machines *proactive*.

Control systems can be entirely reactive, responding to changes that occur in the real world. On the other hand, plants and environments may be capable of initiating action

and so are modelled better by proactive models.

As a final note I should point out where the continuous automata developed in the second part of this thesis falls in this classification. I claim that they are continuous (obviously), their output is totally ordered, and they use dynamic descriptions of change. Their temporal metric is internal and they are proactive. These features distinguish them from other modelling systems. Most other continuous models have external metrics and are reactive.

3.3 Summary

This chapter introduced a number of ideas about the nature of time and offered a classification of temporal models. Various aspects of structure were touched on: the difference between continuous and discrete time; total and partial order; and periods of time with closed ends. Two different purposes for a temporal metric were explored first as a measure of temporal distance, and second as a synchrony labelling.

Different ways of describing parameters assigned to events were considered: static, where time provides an index over events; and dynamic, in which events are described in terms of changes from one to another. The temporal metric was also considered a parameter and the terms internal and external metric were presented to convey whether the rate of change of a machine was an internal feature or the result of an input. Finally impetus was discussed and reactive and proactive change were distinguished depending on whether the machine could initiate changes for itself or whether it depended on external input to trigger change. These features were used to classify the model that will be developed in Part II.

The next chapter considers, in more detail, discrete models of temporal systems. Discrete models represent the starting point for this work, and it is assumed, as we saw in Chapter 2 that in system design discrete descriptions would be used to model an agent's controller.

4

Discrete time

Time is nature's way of making sure that everything doesn't happen at once.

— *Unknown*

Modern scientists and philosophers generally take time to be continuous. Some early thinkers, notably Aristotle, also believed time to be continuous although most Helenic models considered time to be discrete.

The idea that time may be quantized is not really so strange when you consider that most people are ready to accept the idea that matter is atomic, so, for example, the seemingly solid and *continuous* page in front of you is composed of a finite number of very small, but indivisible packets of matter separated by large amounts of free space.

The quantization of matter has led to a modern argument for discrete time. It has been suggested that there is a smallest subatomic particle, whose diameter is one *hodon*. The fastest velocity that can be attained is that of light. So we can calculate the time it would take for a pulse of light to travel a hodon, that is the length of time it would take to cross the smallest distance at the fastest speed. This quantity of time is called a *chronon*.

It has been argued that the chronon represents the smallest meaningful unit of time. Therefore, since it has a natural quantization, time must be discrete. However as

Newton-Smith points out [Newton-Smith 80], this construction presupposes that the pulse of light travels gradually across the distance of the hodon; it does not jump suddenly from one side to another. So, despite the fact that this unit of time may turn out to be important in physics, the argument does not lead us to the conclusion that time is discrete.

Perhaps, the most compelling argument against a discrete view of time is that our strongest physical theories of the universe assume its continuity. So, if time is best represented as continuous, then why bother with discrete models?

4.1 Discrete time models

There are ways in which discrete time models can be very useful. For example, the operation of a digital computer is often described as discrete. What we mean by this is that there is a useful level of abstraction at which a computer's state can be modelled as being constant for an interval of time and then switching discretely and instantaneously to another. In fact the state of the machine is not constant over this interval, but it is guaranteed by the design to be stable and uniquely defined before the next switching occurs, and so the abstraction is valid and it allows us to focus on a sequence of global states of the machine.

This abstraction to sequence is a very powerful modelling assumption that has lead to the development and application of an armory of mathematical, software engineering tools and programming languages. Modern discrete finite state methods can automatically verify, decompose, synthesize and efficiently implement in hardware, systems of many hundreds of thousand of states. Modular use of such tools have enabled human beings to design and build artifacts as complex as a microprocessor and so it is reasonable to conclude that agent controllers will frequently be best designed and modelled as a discrete system.

4.1.1 Limitations of discrete time

So let us turn our initial question around: why do we need to consider continuous change at all if discrete time provides such a useful simplification? There are two

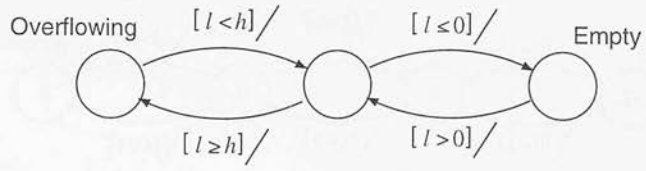


Figure 4.1: The water tank revisited

answers to that question. The first is that some of the properties of the system that we would like to investigate are most effectively modelled using continuous techniques. The stability of a system is such a property. There are circumstances in which oscillations can occur that do not resolve themselves to a single steady state. Even when a steady state is guaranteed, there remains the question: How long will it take to reach it? The second problem with discrete models of continuous systems is that of finding the right discrete abstraction. In discretizing time, what we are really doing is discretizing change. Therefore, for every parameter of our model that we might think of as changing continuously, we need to characterize that parameter by a series of discrete changes. The question is: which ones?

4.1.2 Discrete unlabeled nonmetric time

We saw, in Chapter 1, an example of a water tank (Figure 1.2). In this system there is a natural discretization of the world into three states: the tank is empty, overflowing, or at some height in between. This allows us to build the simple discrete model we saw earlier and which is shown here in Figure 4.1.

A problem arises when these three states are insufficiently expressive to meet our needs. Suppose the water tank was feeding a hydraulic system, and there was a requirement for a critical pressure. In order to achieve this critical pressure the tank had to be at least half full. Our three state model of the world would be insufficient to determine whether the tank could provide that pressure.

You might argue that the modelling approach itself is not flawed, it is just that this model is too limited. If we were to split the center state into two (see Figure 4.2) we could model this property. If the model stays in states A and B then the property

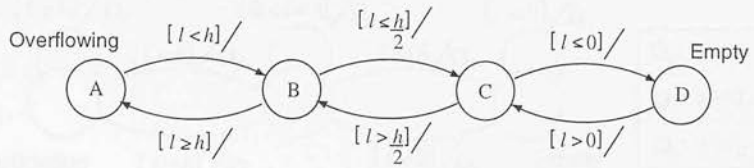


Figure 4.2: A refined water tank model

succeeds, otherwise it fails. However this still leaves three problems with the model:

- For each such property we could be forced to make a new split and we do not want to have to construct a new model to investigate each property.
- There is no way of measuring or talking about the rate of change of the water level.
- The model tells us nothing about how to integrate the changes in state of the water tank with the behaviour of any other components of the system, unless they are causally related to those changes in the tank. Suppose we had another water tank in the system, we should be able to combine the two and have a model which expresses which will empty first, but there is no way to relate the emptying of one with the other.

The first problem is related to the granularity of the decomposition, the second is due to the lack of a metric in the model, and the third problem is due to the absence of a synchrony labelling. As it stands the model is an example of the simplest form of time: discrete, nonmetric, unlabeled time. It amounts to no more than a sequence of changes in the world.

Instead of refining the model by adding more discrete states, we can introduce discrete dynamics to the states themselves. This is illustrated in Figure 4.3. This is obviously a more elegant refinement despite the oversimplification of the dynamics, since it allows us to easily vary the granularity of the model by changing the value of c.

You might suggest that this model also allows us to talk about rate of change: *the water level will change by an amount c every iteration.* However, to do so is to use the

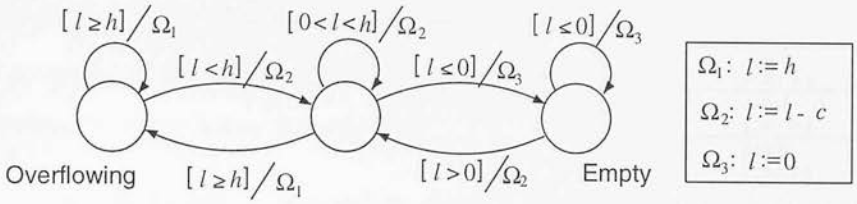


Figure 4.3: Variable granularity

iterations as an implicit metric. The statement is really: *the water level will change by an amount c* **in the time taken** *for every iteration.* This is not necessarily a useful thing to do. There is no guarantee that the period of time taken for every iteration is fixed since there is no measure of that in the model. So the time may vary in an unknown way, in which case the "time taken for every iteration" will simply reflect the time that it took for the water level to change by an amount c at that iteration. In other words the statement we started with becomes *the water level will change by an amount c in the time taken for the water to change by an amount c*; which is obviously true, but not useful.

4.1.3 Metric labeled models

Now consider the model in Figure 4.4. In this model a temporal metric has been introduced in the form of the clock signal `t` . This provides both a measure of distance in time, and a synchrony labelling. In this case the integers are being used as a metric. With the value of `t` establishing the temporal distance between changes, it now makes sense to talk about the rate of change of the water level being c. If the same temporal metric is used by other parts of the system then these parts can now be meaningfully combined since we understand the way in which their respective changes will be synchronized.

So we have laid to rest two of the three problems that we faced. The problem of granularity is still with us though. Indeed we have just made it worse. Choosing the integers as a metric has the effect of fixing the granularity of the model. The only way to change the granularity now is to rescale time in the model, and of course this would mean rescaling time in all the other models we might combine with this model, so it

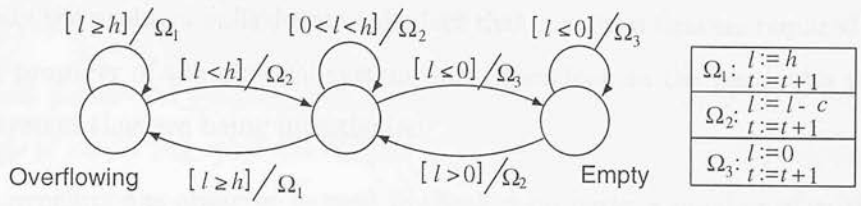


Figure 4.4: Adding an integer metric

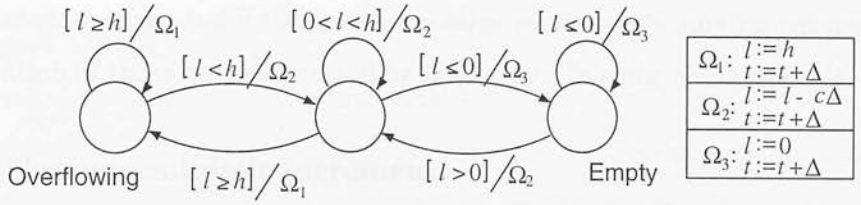


Figure 4.5: Tank with a real metric

is not an ideal solution.

A much better solution would be to change the metric from integers to reals. This will return us to the situation that we were in with Figure 4.3 in which the granularity of the model could be made arbitrarily small. This is shown in Figure 4.5 where the step size is fixed by the parameter Δ. Unfortunately, this is not an end to the story. This model still leaves us with the daunting task of having to choose the right value for Δ. The choice is very important. Making Δ smaller reduces the inaccuracies in the model but at the same time it increases the number of iterations that are required to describe the parameter changes over any interval of time. This in turn makes it harder to simulate or analyze the model.

Furthermore, although the size of the errors can be made arbitrarily small by reducing the step size there is no guarantee that they can be eliminated altogether. It is possible to construct models for which no fixed step size will accurately hit the parameter values at all the state transitions.

Of course there is no reason why the step size should be fixed, but allowing it to vary complicates the problem further because now there are many values to choose rather than one, and it is difficult to predict in advance how the step size should be changed.

Essentially the problem boils down to the fact that the discretization required is not an inherent property of the physical system, but dependent on the particular properties of that system that are being investigated.

When a synchronous observer is used to check a property a number of critical state transitions are added to the system, which lead to the emission of an α alarm signal if the property is violated. So switches can be placed anywhere in the system's parameter-state space and if the system contains continuously varying parameters any discretization of those parameters will be inadequate in some circumstances.

4.1.4 Nondeterministic increments

Must we then resign ourselves to the fact that discrete models of time are inherently approximate? There is at least one more alternative, which is used by most automata based hybrid systems models (including [Alur *et al.* 95, Caspi & Halbwachs 86, Henzinger & Ho 95, Nadjm-Tehrani 94]) and that is to use a nondeterministic step size so that the transition parameters are hit precisely.

As before, the output of the machine is a sequence of events advancing in time. However an output sequence is only considered to be valid if the state transitions occur at the earliest possible opportunity. In other words any sequences containing events which over-step a state transition are rejected. For example consider a sequence of outputs from the model in Figure 4.5, in which the model begins in the middle state, with $0 < l < h$ and l increasing. Ideally the model would change state just as $l = h$. However depending on the values of Δ and c some runs of the model may overshoot h before they can change state. Nondeterministic time steps allow you to reject such runs as invalid and so the critical transitions will always be hit. However this piece of mathematical magic does not come problem free.

The first problems that it introduces is that of Zeno time (after Zeno of Elea). Since we have now a very loose description of the sequences that our model is producing care must be taken to avoid degenerative Zeno time sequences in which Δ is always positive but time is ultimately bounded. An example is the sequence $\Delta = 0, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ which will never advance t beyond 1. Zeno sequences can be avoided by insisting that valid time

sequences must be unbounded.

A less well considered problem with nondeterministic sequences of time is that there is no way of predicting what the output of the model will be on any particular run, aside from the fact that it will hit the parameter values on transitions. Indeed, since an infinite number of possible sequences can be generated from a run of finite length, the probability of the model generating the same output twice is strictly zero.

Why should this be a problem? After all, if the model is guaranteed to have the correct value on state transitions; isn't that enough? In many ways it is, but it leads to a strange situation when trying to decide whether two models are equivalent. There are various ways in which the equivalence of two models can be defined. An important concept in systems design is that of behavioural equivalence, where two models are equivalent when, regardless of their internal structure, they produce the same output in response to the same input events [1].

By this definition, an automaton which selects nondeterministic increments of time is not even equivalent to itself. Now, we might insist that, in assessing the equivalence of such machines, they not only receive the same input, but also use the same time sequence. Then a machine would be equivalent to itself. However, this leaves us with a new problem: for each machine there is a set of valid time sequences, and two machines with different internal structures will have different sets of valid time sequences even if the systems they describe are essentially equivalent. So, the two machines can only be compared with respect to the intersection of their valid time sequences. Such an intersection is guaranteed to exist, but this remains an inelegant solution.

It has already been emphasized that the compositionality of models is a strong theme in this work. Synchronous languages demonstrate how a synchrony labelling can be defined directly in terms of machine events to provide a very natural basis for composition. Because their events are nondeterministically chosen many existing automata based hybrid models (e.g. [Alur *et al.* 95]) have to introduce a new labelling systems for composition. Defining continuous change in terms of a continuum of events on the other hand allows us to continue to use events as the basis for synchronization,

[1] The implied relation is an extension of trace equivalence to input/output systems.

although a metric is needed to define a unique mapping.

4.2 Summary

This chapter considered the benefits and drawbacks of discrete time representations. We considered a model of the water tank system beginning with the simplest possible view of time: discrete unlabelled nonmetric. This model was incrementally enhanced to consider a number of different discrete views of time, ending with a labelled, metric, nondeterministic time which would give a precise model of our system. This final model still retained certain undesirable features. In the next chapter we address these by looking at continuous representations of time.

5

Continuous Time

> *While time can coherently be supposed to be discrete we have no good reasons for taking seriously the hypothesis that it is so. For no one has been able to produce viable physical theories that treat time as discrete. Indeed, all mainline physical theories represent time by a parameter ranging over the real numbers and in so doing treat time as continuous. Interestingly, ... we can construct equally viable alternatives to these physical theories in which time is treated as merely dense and not continuous.*
>
> — *Newton-Smith*[1]

It was Aristotle who first spoke of time as being continuous when he wrote:

> *...what is moved, is moved from something to something and all magnitude is continuous. Therefore the movement goes with the magnitude. Because the magnitude is continuous, the movement too must be continuous, and if the movement, then the time;* [Aristotle 68]

It is a moot point exactly what Aristotle meant by continuous. Bereft as he was of the mathematics of Cantor, it is unclear if he was using the word in the same way that we do here. It is clear that one property of time that he considered continuity to imply

[1] "The Structure of Time" [Newton-Smith 80]

was infinite divisibility: "it is clear that everything that is continuous is divisible to what is itself always divisible".

5.1 Density and continuity

By itself, however, infinite divisibility leads only to a space that is merely *dense* such as the rational numbers $\mathbb{Q}$. A continuous space such as the real numbers is both dense and *connected*. In practice it is difficult to tell the two apart. As he suggests in the quotation (see chapter head), Newton-Smith shows that an approximation to continuous physics can be constructed which only uses the rationals. For every real there is a rational arbitrarily close, and since every experiment we might propose to test the theory is subject to some level of inaccuracy in the instrumentation, this approximation is indistinguishable from the real thing by any empirical investigation.

Nevertheless, there are important mathematical differences between these spaces. A dense set has the property of infinite divisibility, that is, between any two points in the set there is always a third point. However, it differs from a continuous space in that is unconnected, *i.e.* it has gaps in it. For example consider the square root of two, this is a number which is a member of the reals, but not of the rationals and its absence leaves a hole albeit not a very noticeable one.

More precisely we can define the difference by considering the idea of a cut. A cut divides an ordered set S into two sets S', S'' such that (i) every member of S is in S' or S'', (ii) no member of S is in both and (iii) every member of S' comes before every member of S''. The set S is then continuous if and only if for every cut of S into S', S'' there exists either a unique least member of S'' or a unique greatest member of S'. For example the reals are continuous so supposing we make a cut at 2, and take S'' to be all numbers greater than or equal to 2, and S' to be all numbers strictly less than 2, then S'' has no unique greatest member, since the numbers in S'' come arbitrarily close to 2, and for any number you might choose, there is one closer. However S' has a unique least member, 2 itself.

Now we can see that the rationals cannot be continuous. Let us make a cut at $\sqrt{2}$ such that every member of S' is a rational whose square is less than 2 and S'' is a

rational whose square is greater than 2. Then there is no unique greatest member in the former, and no unique least member in the latter.

Another important difference between the reals and the rationals is their *cardinality*. Any infinite set from which there is a one-one mapping to the natural numbers is said to be *countable*. There is such a mapping for the rationals, but there is not for the reals. Thus the reals are said to be *uncountable*. This feature will be revisited when we look at limiting processes in Chapter 8.

So, in principle continuous time models are possible, and in practice such models can be indistinguishable from their continuous counterparts despite the missing values. However dense models are more cumbersome by nature and do not tend to be used. The possibility was introduced here to introduce some important concepts that will be revisited in the theoretical investigation of the limiting process (Chapter 8). Let us now move on to look at continuous time models.

5.2 Continuous time models

Most physical models including the dynamical systems used by control engineers use a continuous representation of time. The hybrid models that have grown out of this tradition also use continuous time. As we saw in Chapter 1 these include [Tavernini 87, Back *et al.* 93, Antsaklis *et al.* 93, Brockett 83, Branicky *et al.* 94]. In principal we could build dense physical models of time however the mathematics of such models is much less appealing and they have no advantages.

This work however is the first attempt (to my knowledge) to have an automata model describe a continuous output. The rest of this chapter will provide an overview of what a continuous automaton will be like, and intuitively how we will get there.

Recall the final model of the water tank, Figure 4.5 used in the previous chapter. The size of each time step is parameterized by the single value Δ. As we saw in the previous chapter, the usual way to produce a precise model of the system is to allow Δ to vary nondeterministically. The alternative to this is to allow the value of Δ to tend to zero in such a way that the machine describes a continuum rather than a sequence, and thus

models every point of the continuously changing parameters rather than an arbitrary choice of points.

There are a number of restrictions required in order to make the limiting process valid. Perhaps the most important of these is that the machine is restricted on the frequency of state changes. In an automaton, let us refer to a transition that goes from a state q to a different state q', as a *state transition*, and let us call a transition that goes from state q back to state q a *looping* transition. Then a continuous machine is restricted to having, at most, a finite number of state transitions or discontinuities in its parameters in any closed, bounded interval of time. The machine can, of course, make an infinite number of looping transitions in such an interval[2].

The continuous automaton provides us with a mechanism to express precisely the continuous and discontinuous features of the systems that we want. It avoids any Zeno time problems because it describes every point instead of a nondeterministically chosen sequence and the output of a deterministic machine is deterministic so there are no equivalence complications either.

This simplification of the output means that it becomes straightforward to define a synchronous product over continuous machines giving a strong compositional semantics.

Because of these strong semantics safety properties of a continuous automaton can be expressed using the technique of synchronous observers [Halbwachs *et al.* 93] (providing introducing the observer does not bring about infinitely many state changes in any closed bounded interval of time.) The observer would have to make uncountably many transitions, but there is no reason why this should be a problem for an observer.

Carrying out uncountably many transitions is a problem for a discrete controller. Assuming that we want such a controller to be implementable it is important that it sees only a *sequence* of inputs. It is also important that successive inputs are not infinitely close together, *i.e.* they are separated by some interval of time. In order to achieve this, when combining a sequential machine with a continuous one, the sequential machine is

[2] The distinction here between state and looping transitions is unfortunate, since it distinguishes between bisimilar machines i.e. it is possible to have two bisimilar machines A and B such that A has a limit and B doesn't. However if they exist the limit of bisimilar machines will be bisimilar. If any machines in a bisimilar class have limits, the unique minimal machine will also have a limit.

restricted to seeing signals only at state transitions of the continuous machine. Since these have already been restricted to occurring at most finitely often in suitable intervals of time their occurrences must be countable in total and separated by intervals of time of nonzero length (see Chapter 11).

It may be that state transitions are not frequent enough in a model or don't occur in the right places for a particular controller. However, the model can be combined in parallel with machines that provide additional discontinuities for the purpose of making the parameter values visible at particular moments, or at regular intervals.

We have now come full circle. Having sketched a continuous model of time, we have described how it can be sampled to produce a discrete sequence which includes every mode change of the system. Interestingly, this will include all the points that the nondeterministic model is designed to describe, but in a deterministic way.

5.3 Summary

In this chapter we made a distinction between sets that are just dense and sets that are dense and connected and hence continuous. We then looked at some of the issues involved in defining a continuous automaton. This chapter concludes the first part of the thesis. In the second part the process of defining a continuous automaton as the limit of a discrete approximation is explored mathematically.

Part II

Theory

6

Flows

"certainly we feel that time flows"

— *J. J. C. Smart*[1]

The idea of a flow is to introduce a mathematical object which can express all the different notions of time that we have considered so far. In particular, the flow has to be able to express a *sequence*, as well as a *continuum* of events. Its definition must be independent of a metric, and we will need to define properties of a flow that will allow us to distinguish sequence from continuum. Clearly the common factor in all of this is that a flow should be defined as an ordered set of points.

In this second part of the thesis there will be great deal of new notation and concepts introduced to help the reader keep track of this there is a summary and index at the back of the thesis.

6.1 What is a flow?

A flow is a totally ordered set of points (see Figure 6.1), each of which can be associated by means of a function θ with an item of data, from a set $\mathcal{D}$. Typically this data item will be a valuation over a signal set. Total ordering of the set means that the ordering relation is defined to apply to any two pairs in the set. Partially ordered flows are

[1] Taken from [Gale 68]

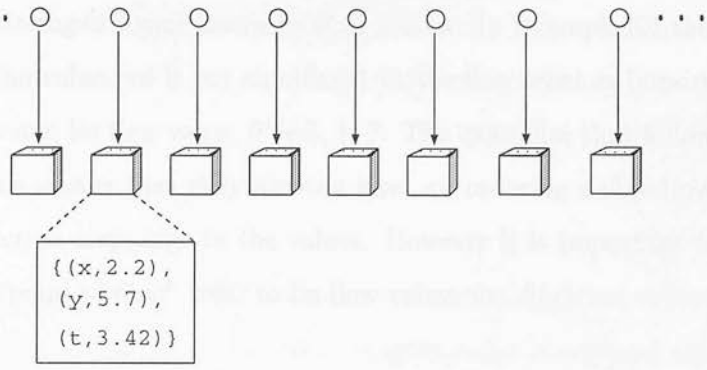


Figure 6.1: A flow of valuations

possible but will not be considered here as they complicate the limiting process. The flow is formally defined as follows.

Definition

A **flow** X is a tuple $(F, \preceq, \theta)$ where F is a set of elements with a total order $\preceq$ (and hence: $\succeq, \prec, \succ$ and $\approx$) defined over them. Each point $\mathbf{x} \in F$ has a value $\theta(\mathbf{x})$, where $\theta : F \to \mathcal{D}$ is a function from F to a dataset $\mathcal{D}$.

Let us also define some flow notation for a flow $X = (F, \preceq, \theta)$. In order to simplify the notation in places, X will occasionally be written in place of the set F, so $\mathbf{x} \in X$ will be taken mean $\mathbf{x} \in F$ and $X \setminus F'$ for a set of points $F' \subseteq F$ to mean the flow $(F \setminus F', \preceq', \theta')$, where $\preceq'$ and θ' are just $\preceq$ and θ with suitably restricted domains. (Notice that a total ordering is preserved under the removal of points.) Finally, the cardinality of the flow is taken as $|X| \stackrel{def}{=} |F|$.

Here is a simple example of a flow:

Example 6.1 *Consider the points:* $F \stackrel{def}{=} \{red, orange, yellow, green, blue, indigo, violet\}$. *Let* $\preceq$ *be an ordering defined by the order in which* F *is listed above, and let* θ *be a number which maps them to the numbers* $7, \ldots, 1$ *such that* $\theta(red) \stackrel{def}{=} 7, \theta(orange) \stackrel{def}{=} 6, \ldots, \theta(violet) \stackrel{def}{=} 1$. *Then* $X = (F, \preceq, \theta)$ *is a flow.*

The next few examples are useful in that they illustrate infinite flows with different structures. However, they may also be a little confusing. A crucial feature of the concept of a flow is that the value of a point $\mathbf{x}$ itself is not important, it may not have

one, only the ordering of $\mathbf{x}$ and the value $\theta(\mathbf{x})$ matter. In Example 6.1 the fact that the first point has the value *red* is not significant to the flow what is important is that it is the first point and its flow value, $\theta(red)$, is 7. The examples that follow use numeric flow points. This means that they already have an ordering defined over them, and it is easy to describe mappings to the values. However it is important to realize that references to a "point's value" refer to its flow value, *i.e.* $\theta(\mathbf{x})$.

Example 6.2 *The tuple* $(\mathbb{N}, \leq, \lambda\mathbf{x}.\mathbf{x})$ *is a flow. The values of the points are the natural numbers, in order. This will be referred to as the natural number flow, or* $\|\mathbb{N}\|$.

Notice that flow points are notated using a boldfaced text: $\mathbf{x} \in F$.

Example 6.3 *Similarly, the flow,* $(\mathbb{R}, \leq, \lambda\mathbf{x}.\mathbf{x})$ *is a continuous flow whose values are the real numbers, again in order. This will be referred to as the real number flow, or* $\|\mathbb{R}\|$.

The ordering of the points in a flow is, of course, independent of their values. So we can define flows such as those in examples 6.4 and 6.5 in which the values descend and oscillate along the respective flows.

Example 6.4 *This flow,* $X = (\mathbb{N}, \leq, \lambda\mathbf{x}.\frac{1}{\mathbf{x}})$, *has points with values* $\frac{1}{n}, \forall n \in \mathbb{N}$. *Notice that these points will get smaller along the flow.*

Example 6.5 *This flow,* $X = (\mathbb{N}, \leq, \lambda\mathbf{x}.\sin(\mathbf{x}))$, *has points with values* $\sin(n), \forall n \in \mathbb{N}$. *Notice that these points will oscillate along the flow.*

Many different types of flow are possible depending on the type of the values. The term 'dataset' is used here to suggest the possibility of complex structured objects, such as functions, sets, trees *etc.* The most common dataset associated with a flow will be *valuations* which relate signal names to values. Valuations will be defined formally in Chapter 7.

The concept of a flow is closely related to the idea of a fluent, a concept used in the AI literature but thought to have originated with Newton. A fluent is a function from a

time domain to a data object. The difference here is that the intention is to explicitly throw away all information about the time domain except its underlying structure.

6.2 Equivalence of flows

One of the most important aspects of defining a new mathematical object is to provide a definition of what it means for two such objects to be considered equivalent. The equivalence property that is required for flows is one that compares the flow values point by point, but ignores the points themselves. For example the following flow is equivalent to the flow we met in Example 6.1:

Example 6.6 *Let us define a flow* $X = (F, \preceq, \theta)$ *where* $F \stackrel{def}{=} \{a, b, c, d, e, f, g\}$, $\preceq$ *is defined by their position in the alphabet and* θ *is defined as* $\theta(a) = 7, \theta(b) = 6, \ldots, \theta(g) = 1$.

So we need a way to maintain the order of points; let us define a mapping as *isotonic* if it preserves order between flows.

Definition

A function $g : F_X \to F_Y$ from the points in a flow $X = (F_X, \preceq_X, \theta_X)$ to the points in a flow $Y = (F_Y, \preceq_Y, \theta_Y)$ is **order preserving** (or **isotonic**) if $\forall \mathbf{x}, \mathbf{x}' \in F_X$,

$$\mathbf{x} \preceq_X \mathbf{x}' \Rightarrow g(\mathbf{x}) \preceq_Y g(\mathbf{x}')$$

On the other hand, it has already been pointed out that a flow should be independent of any values or metric defined over the flow points. So we come to the following definition of equivalence.

Definition

Two flows X, Y are **equivalent** $X \equiv Y$ if there is a one-one order preserving function $g : F_Y \to F_X$ such that $\forall \mathbf{y} \in Y$,

$$\theta_X(g(\mathbf{y})) = \theta_Y(\mathbf{y})$$

It is not difficult to see that this relation will have the properties of reflexivity, symmetry and transitivity that are required of an equivalence relation.

Pairwise equivalence can be extended to global equivalence over a set as follows. (This definition will be important in defining the limit of a sequence of flows.)

Definition

A set of flows R, is said to be **globally equivalent** if

$$\forall X \in R, \forall Y \in R, \quad X \equiv Y$$

6.3 Flow structure

In this section we define the structural (topological) properties of flows. In particular it will be useful to define properties of a flow which is like a sequence, and properties of a flow which is like a continuum. So far the only tools that we have to examine flows are equivalence, and flow ordering. This is enough to get us started. The simplest structural property that we can look for is the presence of a start or an end to the flow.

Definition

The **end** $end(X)$ of a flow $X = (F, \preceq, \theta)$, if it exists, is a point $\mathbf{x} \in X$ such that $\forall \mathbf{y} \in F, \mathbf{y} \preceq \mathbf{x}$. Similarly the **start** $start(X)$ is a point $\mathbf{x} \in X$ such that $\forall \mathbf{y} \in F, \mathbf{y} \succeq \mathbf{x}$. If $start(X)$ exists then X is said to be **closed at the start**, if $end(X)$ exists it is said to be **closed at the end**.

It has already been noted that we are especially interested in flows which are closed at the start, but open or closed at the end. We will refer to these as *forward* flows. Forward flows will be used to represent the output of our automata models. They are closed at the start because automata are conventionally assumed to have a starting point or initial state.

Another property that can be immediately defined is density of a flow. A flow is dense if there is a third point, between any two points you care to choose.

Definition

A flow, X, is said to be **dense**, if for any points $\mathbf{x}, \mathbf{x'} \in X$, with $\mathbf{x} \prec \mathbf{x'}$, there exists a third point, $\mathbf{y} \in X$, such that $\mathbf{x} \prec \mathbf{y} \prec \mathbf{x'}$.

In order to take our investigation further we will need to be able to chop flows up into smaller pieces in order to examine their structure. This is done with a subflow.

Definition

The flow Y is a **subflow** of a flow $X = (F_X, \preceq_X, \theta_X)$, written $Y \sqsubseteq X$ if Y is equivalent to a flow $(F', \preceq_X, \theta_X)$ such that $F' \subseteq F_X$

As you would expect, for two flows X and Y if $X \sqsubseteq Y$ and $Y \sqsubseteq X$ then $X \equiv Y$. Indeed, it might have been more natural to introduce the concept of subflow before that of equivalence. However, equivalence was presented first because it formalized the irrelevance of the values of members of F.

In particular, a subflow Y of a flow X is *contiguous* if the points in Y are contiguous in X.

Definition

A subflow Y of a flow X is **contiguous** if for all isotones $g : Y \to X$, $\forall \mathbf{x} \in X$ if $\exists \mathbf{y}', \mathbf{y}'' \in Y$ such that $g(\mathbf{y}') \prec \mathbf{x} \prec g(\mathbf{y}'')$ then $\exists \mathbf{y} \in Y$ such that $\mathbf{x} = g(\mathbf{y})$.

The operator $Sub^{op\mathbf{x}}(X)$ is used to define contiguous subflows, where op is one of $\prec, \preceq, \succ$ or $\succeq$. For a flow $X = (F, \preceq, \theta)$ and a point $\mathbf{x} \in X$ it is taken to mean any flow equivalent to $(F', \preceq, \theta)$ where $F' \stackrel{def}{=} \{\mathbf{y} \in X | \mathbf{y}\ op\ \mathbf{x}\}$.

With this operator we can define intervals of a flow, so an interval $(\mathbf{x}, \mathbf{y})$ of a flow X is taken to mean $Sub^{\prec\mathbf{y}}(Sub^{\succ\mathbf{x}}(X))$ and $[\mathbf{x}, \mathbf{y}]$ is taken to mean $Sub^{\preceq\mathbf{y}}(Sub^{\succeq\mathbf{x}}(X))$ and so on.

Now that we can divide flows into intervals, it is possible to investigate how tightly packed a flow is. We already have a definition of *density*, with this definition we can approach the definition of a sequence. In previous chapters a sequence was described as an ordered set in which every point (except the first, if it exists) has a unique predecessor. We will call flows with this type of structure *regular*. In order to formally define regular flows we must first define what it is for a point to be the predecessor of another.

Definition

For point $\mathbf{x}$ in a flow $X = (F, \preceq, \theta)$ the predecessor point, written $\overleftarrow{\mathbf{x}}$, if it exists, is the end of the subflow upto $\mathbf{x}$. i.e. $end(Sub^{\prec \mathbf{x}}(X))$. Likewise the start of the subflow from a point $\mathbf{x}$ $start(Sub^{\succ \mathbf{x}}(X))$ is the successor to $\mathbf{x}$ written $\overrightarrow{\mathbf{x}}$.

Definition

A flow $X = (F, \preceq, \theta)$ is **regular** if $\forall \mathbf{x} \in X \setminus \{start(X)\}$, the predecessor point $\overleftarrow{\mathbf{x}} \in X$ is well defined.

Example 6.7 $\|\mathbb{N}\|$ *is a regular flow.*

Having already defined density we can now show that points in dense flows never have predecessors, and therefore dense flows cannot be regular.

Lemma 6.1 *For a dense flow* X, $\forall \mathbf{x} \in X, \nexists \overleftarrow{\mathbf{x}}$.

Proof (adapted from classical mathematics) Take some $\mathbf{x} \in X$ ($\mathbf{x} \succ start(X)$, if $start(X)$ exists). Now assume the lemma is false, and there does exist $\overleftarrow{\mathbf{x}} \stackrel{def}{=} end(Sub^{\prec x}(X))$. But density of X implies that $\exists \mathbf{y} \in X, \overleftarrow{\mathbf{x}} \prec \mathbf{y} \prec \mathbf{x}$ and this contradicts the definition of $\overleftarrow{\mathbf{x}}$ so there can be no $\overleftarrow{\mathbf{x}}$. □

It can also be shown that regular flows cannot be dense. Although there are flows which have regular regions and dense regions, such flows are neither regular nor dense.

A third important structural property is that of connectedness. An intuitive definition of connectedness was given in Chapter 5. This definition is formalised here.

Definition

A **cut** (A, B) is the partitioning of a flow X into two continguous subflows A, and B, such that $\forall \mathbf{a} \in A, \mathbf{b} \in B, \mathbf{a} \prec \mathbf{b}$. The flow X is **connected** if for any cut (A, B), A is closed at the end or B is closed at the beginning.

These two definitions lead to an obvious example:

Example 6.8 *The flow* $(\mathbb{Q}, \leq, \lambda \mathbf{x}.\mathbf{x})$ *is a dense but unconnected flow.*

With the definition of connected we can now define a continuous flow which we'll refer to as a continuum.

Definition

A **continuous flow**, or **continuum** is a flow which is dense and connected.

That concludes our investigation of flow structure, we move on now to look at mathmathical properties of flows with values.

6.4 Valued flows

In this section some properties of flows with metric values are considered. These properties include monotonicity and continuity of values in the flow.

The concept of a continuous function is extremely important in mathematical analysis. Intuitively, it represents a function whose graph is represented by an unbroken line, that is, a line that can be drawn without lifting your pen from the paper. The notion is very useful to us here too. A θ-continuous flow is a dense flow such that for any point **a**, all the points in the immediate neighbourhood have values as close as you like to the value of **a**. The size of the neighbourhood will depend on how close you want the values to be but no matter how close you choose there is a suitable neighbourhood.

It is important to note the distinction between a *continuous* flow, which is defined above as a dense connected flow, and a θ-continuous flow which is a flow with a 'smooth' value set.

This definition uses the term *metric dataset*. This refers to a dataset with a distance operator d (such as absolute difference *etc.*) which conforms to the standard requirements of a metric (for more details see the proof that $\ominus$ is a metric in Appendix A.)

Definition

A flow with a metric dataset, T is **θ-continuous at a point a** if for every $\epsilon > 0$ there exist points $\mathbf{k}$ and $\mathbf{k}'$ (with $\mathbf{k} \prec \mathbf{a} \prec \mathbf{k}'$) such that $d(\theta(\mathbf{x}), \theta(\mathbf{a})) < \epsilon$ for all $\mathbf{x}$ such that $\mathbf{k} \prec \mathbf{x} \prec \mathbf{k}'$.
A flow T is **θ-continuous** if it is θ-continuous at all points $\mathbf{a} \in T$.
For a flow to be **θ-continuous to the left** of a point (or **θ-continuous upto** a point) $\mathbf{a}$ requires only the existence of $\mathbf{k}$ above. Similarly for a flow to be **θ-continuous to the right** (or **θ-continuous from** a point) $\mathbf{a}$ requires only the existence of $\mathbf{k}'$.

Another important feature of a flow is **monotonicity**. A flow is monotonic if successive values always either increase or decrease, respectively.

Definition

A flow F, with an ordered dataset, is said to be **monotonic** if

- either $\theta(\mathbf{x}) \geq \theta(\mathbf{x}')$ for all $\mathbf{x} \prec \mathbf{x}'$
- or $\theta(\mathbf{x}) \leq \theta(\mathbf{x}')$ for all $\mathbf{x} \prec \mathbf{x}'$.

The term **strictly monotonic** is used when the inequalities are strict.

In addition to talking about the end points of flows, it is useful to be able to talk about the value at the limit of a flow, even if the end point doesn't exist. This point is the flow's *supremum*. The supremum is essentially the upper closure point of the flow, if the flow is closed at the end then the supremum of X is simply $end(X)$. Otherwise it is defined as a point greater than any in X, with a value defined as the limit along the flow. The point is guaranteed to exist because a point can be added and the ordering relation can be explicitly changed to ensure that it satisfies the properties. The value on the other hand may not exist if the flow values do not tend to a limit.

Definition

For a flow $X \stackrel{def}{=} (F, \preceq, \theta)$ which is closed at the end the **supremum** $\sup(X)$ is $end(X)$. Otherwise construct a flow $X' \stackrel{def}{=} (F^+, \preceq^+, \theta^+)$ where $F^+ \stackrel{def}{=} F \cup \{\mathbf{p}\}, \mathbf{p} \notin F$. The relation $\preceq^+$ is the same as $\preceq$ except that $\forall \mathbf{x} \in X, \mathbf{p} \succ^+ \mathbf{x}$. The function $\theta^+ \stackrel{def}{=} \theta \cup \{(\mathbf{p}, v)\}$, and finally $\mathcal{D}^+ = \mathcal{D} \cup \{v\}$.
Where $\mathbf{p}$, is taken to have the value v, if it exists such that $\forall \epsilon > 0 \ \exists \mathbf{k} \in X$, s.t.$d(v, \theta(\mathbf{y})) < \epsilon \ \forall \mathbf{y} \succ \mathbf{k}$. If this value does not exist then $v \stackrel{def}{=} \perp$. Symmetrically an infimum can be defined.

Our last definition defines how flows can be composed. It is assumed that there exists a suitable union operator over the flow values, and that a suitable one-one isotonic function can be defined between the points in the respective flows. The new flow is then just a pointwise union of the flow values.

Definition

Two flows $X = (F_X, \preceq_X, \theta_X), Y = (F_Y, \preceq_Y, \theta_Y)$ can be **zipped** together, written $X \amalg_f Y = (F_X, \preceq_X, \theta_{X \amalg_f Y})$ with respect to a one-one order preserving function $f : F_X \to F_Y$ where $\theta_{X \amalg_f Y}(\mathbf{x}) = \theta_X(\mathbf{x}) \cup \theta_Y(f(\mathbf{x}))$, providing $\theta_X(\mathbf{x}) \cup \theta_Y(f(\mathbf{x}))$ exists for all $\mathbf{x}$. For regular forward assignment flows we write $X \amalg Y$ since there is only one suitable f.

6.5 Summary

In this chapter the concept of flows was introduced. Properties regarding the structure and value set of the flow were defined. In particular a continuum was defined to be a dense connected flow.

The flow was presented as a description of the evolution of events through time. This is the beginning of the theory the next stage is to define models which can describe regular flows.

7

Discrete automata

On two occasions I have been asked [by members of Parliament], 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.

— *Charles Babbage (1792-1871)*

Automata are dynamic relations describing flows of events. In this chapter we will define exactly how these dynamics are specified in the case where the flows are discrete. Schematically the relation is illustrated in Figure 7.1. This shows a flow of input events I and a flow of output events O related by a machine M with internal state and a characteristic function χ. The machine is shown to take input from the current input event, the previous input event and the previous output event in order to calculate its output. All the automata types that will be presented will be capable of defining sensible generating machines, that is machines which produce output flows but do not accept inputs.

In a modular modelling framework the way in which models are combined is critically important to their overall semantics. The flow machines described in this chapter will be composed using a synchronous product based on the one used in ESTEREL and LUSTRE *etc.* This provides a strong compositional semantics which is inherited by the

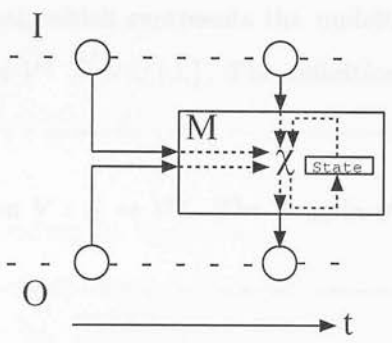


Figure 7.1: A discrete automata relating input and output flows through time.

limit machines.

Automata are presented as descriptions of event flows where a complete event is the union of corresponding input and output events. This chapter begins by looking at how events themselves are defined.

7.1 Events

You may recall that an event is intended to be a snapshot of the state of the world. The world that we are observing in this snapshot is represented by a set of parameters which are typically real valued, although they could take values from any set you please. These parameters will also provide the machines with the means with which to communicate. For this reason they are called *signals.*

Signals are used both as value stores and communications mechanisms, and in this sense they behave very much like a shared memory. Output signal values persist so, at each step of a machine's operation, unless a signal value is explicitly modified, its value remains the same as it was on the previous step.

7.1.1 Valuations

The signals in our model are drawn from a set S. An event is defined by a *valuation* which is a function $V : S \to \mathcal{V}$ from the signal set to some value set $\mathcal{V}$. In fact, the value

set must include the value $\perp$ which represents the undefined value which all signals take initially, so let us write $\mathcal{V}^+ \stackrel{def}{=} \mathcal{V} \cup \{\perp\}$. The definition of a valuation is given by:

Definition

A **valuation** is a function $V : S \to \mathcal{V}^+$. The domain of V, written $D(V)$ is the set S.

Valuations are typically represented as sets of pairs, so, for example, a valuation over a signal set $S = \{\mathtt{x}, \mathtt{y}\}$ which sets $\mathtt{x}$ to 3 and $\mathtt{y}$ to 4, would look like: $V \stackrel{def}{=} \{(\mathtt{x}, 3), (\mathtt{y}, 4)\}$, and of course $D(V) = \{\mathtt{x}, \mathtt{y}\}$.

The set of all valuations is given the symbol $\mathcal{E}$ and the set of valuations V_S over a signal set S is $\mathcal{E}_S$. The union of two valuations $V_S \cup V_R$ is defined simply as their set union unless there is a signal $\mathtt{s} \in S \cap R$ such that $V_R(\mathtt{s}) \neq V_S(\mathtt{s})$, in which case the union is undefined.

The notion of θ-continuity (introduced in Chapter 6), which expresses continuity of parameter change, can be extended to apply to valuations if a distance metric is defined over them. This can be defined as follows (in Appendix A a proof is given that this is indeed a metric):

Definition

The distance between two valuations V_S, V'_S over the signal set S is defined as:

$$|V_S \ominus V'_S| = \Sigma_{s \in S} |V_S(s) - V'_S(s)|$$

We say that the difference between any non-numeric values x and y is 1 unless they are equal in which case it is 0. e.g.

$$\perp - \perp \stackrel{def}{=} 0$$
$$(\forall v \in \mathcal{V}^+ \setminus \{\perp\})\ \perp - v = v - \perp \stackrel{def}{=} 1$$

In another version of this work [Westhead & Hallam 96a], the value semantics distinguished between signal presence and signal value, just as the semantics of ESTEREL [Berry & Gonthier 92] does. However, this led to an unnecessarily complicated model which did not help illustrate the limiting process, and so it fell by the wayside. In this presentation of the model the presence or absence of a signal is irrelevant, transitions

are predicated entirely on signal values.

7.2 Automata Components

This section describes the automata themselves. The automata description is given in three parts: the signal set of input, output and state signals; the value set; and the characteristic relation. We will look at each in turn before presenting a formal definition of a machine.

7.2.1 Signal sets

The signal set of a machine M is notated S_M it is made up by a union of three disjoint signal sets : the input signals I_M the output signals O_M and the state signal ξ_M. The state signal is used to store the machine's state, it is an internal signal which is not available to the machine's environment. Occasionally the E_M may be used to refer to the event set $I_M \cup O_M$ which are the externally available signals.

There are a couple of notation conventions regarding signals that need to be described. Let us say that the current event of a machine is described at a point $\mathbf{x}$ in the event flow $(F, \preceq, \theta)$. Now we will occasionally refer to the current event as $e \stackrel{def}{=} \theta(\mathbf{x})$ and the previous event as $\overleftarrow{e} \stackrel{def}{=} \theta(\overleftarrow{\mathbf{x}})$. For a signal $\mathbf{s} \in S$ from the set S of signals in the machine, its value will be represented by $\overleftarrow{s}$. Finally the notation $\overleftarrow{S}$ will be applied to the signal set S when the signal values being referenced come from the previous valuation. So, in a sense $\overleftarrow{S} \stackrel{def}{=} D(\theta(\overleftarrow{\mathbf{x}}))$ and a for valuation $V \in \mathcal{E}_{\overleftarrow{S}}$, $V(\mathbf{s}) = \overleftarrow{s}$.

7.2.2 Value sets

The machine's value set V_M is the union of two sets of values: $\mathcal{V}$ which define the values over the signals in E_M and Q_M which defines the set of values the state signal can take. In most of the examples in this thesis $\mathcal{V}$ will be $\mathbb{R}$ although any metric value set could be used. The state signals usually range over a set of finite values (this will be assumed from here on) although infinite value sets could perfectly well be used.

7.2.3 Characteristic relation

The characteristic relation χ_M is the meat of the machines description. It is a relation between valuations:

$$\chi_M : \mathcal{E}_{I_M \cup \overleftarrow{S}_M} \times \mathcal{E}_{S_M}$$

So χ_M relates the values of the previous signals $\overleftarrow{S}_M$ and the current input signals I_M to all the current signals S_M.

If the characteristic relation is a function, then the machine is deterministic. If for all combinations of previous event and input, there is an output defined then the machine is proactive. It has already been observed that this formulation of flow machines removes the distinction between signal presence and signal value which exists in many of the synchronous languages. This leads to a simplified semantics, but removes the distinction between reactive and proactive machines. For a reactive machine to produce an output, at least one input signal must be present. A proactive machine can produce an output even when there are no input signals present. Signal presence is not defined here and transitions occur based only on signal value.

Definition

A machine M with characteristic relation χ_M can be said to be:

- **Deterministic:** if $\forall x \in \mathcal{E}_{I_M \cup \overleftarrow{S}_M}$ there exists exactly one $(x, y) \in \chi_M$.
- **Proactive** $\forall x \in \mathcal{E}_{I_M \cup \overleftarrow{S}_M}$, s. t. $\exists (x, y) \in \chi_M$

It is going to be important to us to be able to compose characteristic relations, so let us define a union operation over valuation relations. This union reflects the classical composition of signal values in the synchronous product. First we need another piece of notation.

For a set of pairs $(a, b) \in S, a \in A, b \in B$, and a set $A' \subseteq A$ let $S \downarrow A' \stackrel{def}{=} \{(a, b) \in S | a \in A'\}$.

Definition

The **union** of two valuation relations $R_1 : \mathcal{E}_{d_1} \times \mathcal{E}_{r_1}$ and $R_2 : \mathcal{E}_{d_2} \times \mathcal{E}_{r_2}$ signal domains d_1, d_2 and ranges r_1, r_2 respectively is defined as follows:

$$(x, y) \in R_1 \sqcup R_2 \Leftrightarrow \begin{array}{l} (((x \cup y) \downarrow d_1), (y \downarrow r_1)) \in R_1 \text{ and} \\ (((x \cup y) \downarrow d_2), (y \downarrow r_2)) \in R_2. \end{array}$$

In order to facilitate the limiting process it is necessary to "unfold" the characteristic relation along a flow. The resulting flow is called the realization of the relation.

The characteristic relation in a deterministic machine defines current output value at a particular point in time in terms of the current and previous input, and the previous output (see Figure 7.1). The realization expresses this global function as a flow of functions each of which is tied to its predecessor. The resulting realization flow is a flow of functions each of which defines a valuation in terms of all the inputs to that point previous inputs and the initial conditions of the machine. In a nondeterministic machine these functions are relations. Formally the realization is defined as follows.

Definition

For a characteristic function χ with initial valuation V_0, the **realization** of χ is a forward regular flow $(F, \preceq, \theta)$ in which θ is defined recursively, as follows:$\forall \mathbf{x} \in F$

$$\begin{array}{rcl} \theta(\mathbf{x}_0) & \stackrel{def}{=} & \{(\{\}, V_0)\} \\ \theta(\mathbf{x}) & \stackrel{def}{=} & \{(x, (y \downarrow S_M)) | (x, y) \in \chi \sqcup \theta(\overleftarrow{\mathbf{x}})\} \end{array}$$

In effect this definition recursively composes the characteristic function along the flow. At each point however the output side of the relation is restricted to the current signal set. Otherwise the relations at each point would output all the previous signal valuations up to that point.

7.3 Flow machines

We are now in a position to define a *flow machine.*

Definition

An **I/O flow machine** is a triple $(S_M, V_M, R(\chi_M))$ where:

- $S_M \stackrel{def}{=} I_M \cup O_M \cup \{\xi\}$ is a signal set formed by the disjoint union of the input and output signals and the state signal.
- $V_M : \mathcal{V}_M^+ \cup Q$ the set of values the signals can take composed of external signal values and a set of state values including an initial value $q0 \in Q$.
- $\chi_M : \mathcal{E}_{\stackrel{\leftarrow}{S_M \cup I_M}} \times \mathcal{E}_{S_M}$, is the characteristic function for the machine. It is restricted such that $\forall (x, y) \in \chi_M, (x \downarrow I_M) = (y \downarrow I_M)$.

Notice that the characteristic function is restricted to ensure that the input signals are unchanged on both sides of the relation. The set of state values Q is assumed to be finite.

7.3.1 The synchronous product

Two flow machines can be composed using the synchronous product this is defined by the following.

Definition

The **synchronous product** of two flow machines $M_1 = (S_{M_1}, V_{M_1}, R(\chi_{M_1})$ and $M_2 = (S_{M_2}, V_{M_2}, R(\chi_{M_2})$ for which $O_{M_1} \cap O_{M_2} = \emptyset$[1] is defined to be

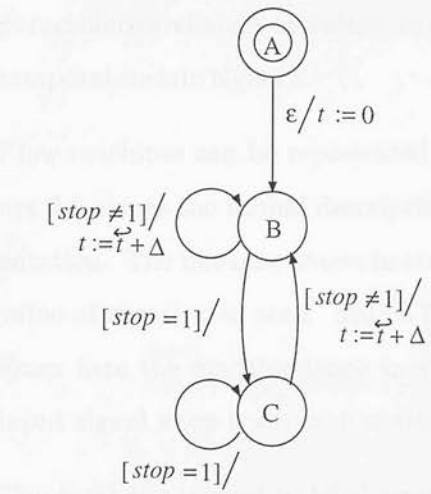
$$M_1 \| M_2 \stackrel{def}{=} (S_M, V_M, R(\chi_{M_1}) \amalg R(\chi_{M_2})$$

where:

- $S_M : I_M \stackrel{def}{=} (I_{M_1} \setminus O_{M_2}) \cup (I_{M_2} \setminus O_{M_1})$, $O_M \stackrel{def}{=} O_{M_1} \cup O_{M_2}$, $\xi_M \stackrel{def}{=} \xi_{M_1}$ or ξ_{M_2},
- $V_M : \mathcal{V}_M \stackrel{def}{=} \mathcal{V}_{M_1} \cup \mathcal{V}_{M_2}$, $Q_M \stackrel{def}{=} Q_{M_1} \times Q_{M_2}$,

There are two obvious ways in which the synchronous product could be defined these are either to zip the two realization flows together (as we have done here) or to take the realization of the union of the two characteristic relations. However the two definitions are equivalent, and a proof of this is given in Appendix A.

[1] This restriction over the machine outputs is not essential and is avoided in some synchronous languages (*e.g.* ESTEREL). However it simplifies the semantics.



$$
\begin{array}{ll}
S_M & \left\{ \begin{array}{l} I_M = \{\texttt{stop}\} \cup \\ O_M = \{\texttt{t}\} \cup \\ \{\xi\} \cup \{\Delta\} \end{array} \right. \\
V_M & \left\{ \begin{array}{l} \mathcal{V} = \mathbb{R} \cup \{T, F\} \cup \\ Q_M = \{A, B, C\} \end{array} \right. \\
\chi_M & \left\{ \begin{array}{l} \{(\{(\xi, A), (stop, T\}, \{(\xi, B)(t, 0)\}), \\ \ (\{(\xi, A), (stop, F\}, \{(\xi, B)(t, 0)\}), \\ \ (\{(\xi, B), (stop, F)\}, \{(\xi, B)(t, \overleftarrow{t} + \Delta)\}), \\ \ (\{(\xi, B), (stop, T)\}, \{(\xi, C)\}), \\ \ (\{(\xi, C), (stop, F)\}, \{(\xi, B)(t, \overleftarrow{t} + \Delta)\}), \\ \ (\{(\xi, C), (stop, T)\}, \{(\xi, C)\})\} \end{array} \right.
\end{array}
$$

Figure 7.2: Modelling a stopwatch

A major drawback with this definition of the synchronous product is that it can lead to causality problems. The problem is that this product operation does not preserve determinism or proactivity. In other words the product of two machines with uniquely defined outputs, can have no outputs, or many possible outputs. This is a well studied problem, the theoretical solution is to define the synchronous product as the result of a fixed point operation. In practice compile time checks on synchronous languages pick out these causality problems. For more details see [Halbwachs *et al.* 93, Halbwachs & Maraninchi 95, Berry & Gonthier 92, Berry 95]

7.4 Delta flow machine

This final definition describes the delta flow machine as a flow machine with an additional internal signal Δ which is subject to certain restrictions.

Definition

A **delta I/O flow machine** M is a flow machine with an extra internal signal $\Delta \in S_M, \Delta \notin I_M, O_M$, it takes real values so $\mathbb{R} \subseteq V_M$ and χ is constrained such that

$$0 < \min(\Delta)_M \leq \Delta \leq \max(\Delta)_M$$

for constants $\min(\Delta)_M, \max(\Delta)_M$.

The signal Δ is added to provide a variable time step. It is conventionally used to

parameterize change in values in particular it defines the step size of the standard temporal metric signal t.

Flow machines can be represented as a graph of states connected by transitions. Figure 7.2 shows the formal description of a simple machine next to the graphical representation. The machine starts in state A and immediately moves to state B setting the value of signal `t` to zero. Notice how an italic font is used to represent signal value. From here the machine loops in state B incrementing the value of signal `t` until the input signal `stop` is set to 1 at which point the incrementation of `t` ceases.

The symbol ε is used to label empty conditions, effectively it represents the predicate T which is always true.

7.5 Summary

Automata are considered here as defining relations between valuation flows. This chapter began by presenting an overview of flow machines. Valuations were formally defined followed by flow machines themselves. An example machine was present and we then went on to define a composition operator, the synchronous product. The chapter ended with a definition of the delta flow machine an extension of the flow machine with an additional internal signal.

In the next chapter we begin to look at the mathematics behind the limiting process itself.

8

The search for the uncountable

The infinite! No other question has ever moved so profoundly the spirit of man.

— *David Hilbert (1862-1943)* [1]

To infinity and beyond!

— *Buzz Lightyear*[2]

In this chapter we face the grizzly problem of turning a discrete sequence of sharp points into an infinitely smooth continuum. Flows were originally conceived with the idea that they could describe both sequences and continua. It is reasonably straightforward to see how to construct a discrete machine, which can describe an approximation to continuous change, to an arbitrary degree of accuracy. Let us suppose that we can take a sequence of such machines which steadily increase in their accuracy. The problem is; how to define a robust formal limiting process that makes the jump to the continuous machine? For the purposes of this discussion a continuous machine can be seen as one which is capable of producing a flow which contains the values of all the real numbers, in order.

A naïve approach might be to look at the sequence of flows of these machines as a sequence of sets of points, which indeed they are, and then take the limit of these sets.

[1] in J. R. Newman (ed.) The World of Mathematics, 1956.

[2] from *Toy Story* Walt Disney 1995

Every step in the sequence adds new values to the flow. Intuitively one might expect this to lead to the limit that we want. However it turns out that there will be lots of points missing. It is reasonably instructive to understand why, because it motivates the more complex limiting process that has been defined instead.

The problem lies in the definition of the limit of a sequence of sets.

The limit of a sequence of sets is defined as the union over an infinite sequence of sets. A consequence of this definition is that any value that occurs in the limit must have been introduced at some finite point in that sequence. Unfortunately this approach cannot introduce all the points that we need. One way to see this is that the number of sets in the sequence is countable, as is the number of points in each set. The union of countably many countable sets can only lead to the construction of a countable set and unfortunately the reals are uncountable.

Another way to see the problem is to consider an example. Suppose that we are taking the limit of the simplest of all delta machines, the delta clock, to whose study Chapter 9 is devoted. The delta clock is a very simple beast with a single signal `t` initialized to zero and incremented as $t := \overleftarrow{t} + \Delta$.

Suppose that for each machine the value of Δ is taken to be a constant such that at each machine in the sequence Δ is divided by two. So the first flow will have values $(0, \Delta_0, 2\Delta_0, 3\Delta_0, \ldots)$, the second flow will be $(0, \frac{\Delta_0}{2}, \Delta_0, \frac{3\Delta_0}{2}, \ldots)$ and so on. Then this sequence will only produce limit values of the form $\frac{n\Delta_0}{k}$ where $n \in \mathbb{N}$ and k is a power of two.

Using this sort of construction it is possible to construct a limit set containing all the rationals. If we start with $\Delta = 1$ and, at level n, divide Δ by n, then we get $(0, 1, 2, 3 \ldots)$ followed by $(0, \frac{1}{2}, 1, \frac{3}{2}, \ldots)$ and then $(0, \frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \ldots)$. To show that the limit contains all the rationals you simply have choose one at random and show where it was introduced. So for an arbitrary rational $\frac{u}{v}$ if you go far enough along the v^{th} flow in the sequence, this number is guaranteed to be there. On the other hand, $\sqrt{2}$ will not be there at all.

This construction will *only* model a system with rational numbers. It was argued in Chapter 5 that in practice only elegance is lost in restricting a model to the rationals;

so would this limiting process suffice? Unfortunately it suffers other deficiencies. Most notably it is not very robust. Small changes in the values of Δ can significantly change the ultimate result. For example if you set the initial value of Δ to be $\sqrt{2}$ instead of 1 but keep the same descent sequence, then there will be no rationals in the limit values. Instead the limit will be composed of numbers of the form $\frac{\sqrt{2}u}{v}$ for $u, v \in \mathbb{Z}$.

The limiting process that was eventually arrived at is far more robust than this. It does not even require that Δ be constant along a flow, simply that the bounds over Δ decrease down the sequence. It also supplies an uncountable limit, and as we shall see in the next chapter, the limit of a sequence of delta clocks expresses the whole of the positive real number line.

The trick to achieving this limit is to define branching paths, collectively called a *comb*, which connect individual points in the sequence of flows. If there are enough branches in the paths, then following these paths to infinity can lead to enough points. Before we see how there needs to be a short digression to introduce some new notation. We will use *strings* to provide an indexing over the comb and ultimately give the ordering of the limit flow.

8.1 Strings

A *string* is simply a concatenation of characters taken from an alphabet. For example $\sigma = rlr$ is a string over the alphabet $\{rl\}$. Strings can be concatenated simply by juxtaposition so if $\sigma' = ll$ then $\sigma\sigma' = rlrll$ and $\sigma lrl = rlrlrl$. The notation σ_i refers to the i^{th} character in σ and $\sigma[i]$ refers to the string of the first i characters of σ. $|\sigma|$ is the length of a string.

If a string is composed of characters which have an ordering defined over them this can be extended to form a lexicographical ordering over the strings. For example consider alphabet $A = \{0, 1, 2, 3, 4\}$ we can say that for two strings σ, σ' over A,

$$\sigma \leq \sigma' \text{ iff } \sigma_n \leq \sigma'_n$$

where n is the first place at which σ and σ' differ, i.e. $\forall i < n, \sigma_i = \sigma'_i$ and $\sigma_n \neq \sigma'_n$. If

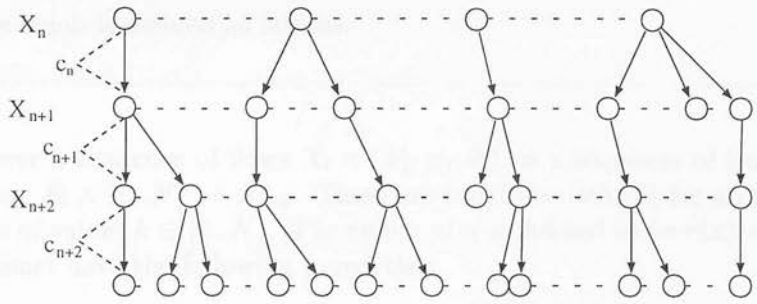


Figure 8.1: A piece of a comb c_n over the flow sequence X_n

such an n does not exist then

$$\sigma \leq \sigma' \text{ iff } |\sigma| \leq |\sigma'|$$

8.2 Combs

In this section the concept of a *comb* is introduced. The comb provides a structure over a sequence of flows X_n. The idea is depicted in Figure 8.1. The flows X_n, X_{n+1} and X_{n+2} are depected as horizontal sequences of circles. The comb is a sequence of functions c_n represented by arrows which map all the points of X_n to points in X_{n+1}. These links form paths called *descents* which can be considered as individual limiting sequences.

The comb is constructed as a sequence of child functions c_n that map the parents - members of flow X_n, to the children - members of flow X_{n+1}. This is illustrated in Figure 8.1. Child functions have three properties they must be :

- **isotonic** — that is the mapping preserves the ordering of the two flows
- **exclusive** — that is each child has at most one parent, though parents may have multiple children.
- **complete** — every point in every flow in the sequence after the first has a parent.

So the comb defines a function from points in a flow to points in the next flow. The child function $c(\mathbf{x}, i), i \in \{0, \ldots, k\}$ maps from a single point $\mathbf{x} \in X_n$ to a set of $k+1$ children in X_{n+1}. The value k is the *reach* of the point written $r(\mathbf{x})$.

Formally, the comb is defined as follows.

Definition

A **comb** over a sequence of flows $X_i = (F_i, \preceq_i, \theta_i)$, is a sequence of isotonic child functions $c_i : F_i \times \{0..N\} \to F_{i+1}$. These are functions defined for all $\mathbf{x} \in X_i$, for some range of values $k \in \{0..N\}$. The **reach** of $\mathbf{x}$ is defined to be $r(\mathbf{x}) = N$. Child functions must have the following properties:

isotonicity $c(\mathbf{x}, k) \preceq c(\mathbf{x}', k') \Leftrightarrow \mathbf{x} \preceq \mathbf{x}'$

exclusivity $c(\mathbf{x}, k) = c(\mathbf{x}', k') \Leftrightarrow \mathbf{x} \approx \mathbf{x}', k = k'$

completeness $\mathbf{x} \in X_i, i > 0 \Rightarrow \exists \mathbf{x}' \in X_{i-1}$ s.t. $\mathbf{x} = c(\mathbf{x}', k)$ for some k.

Points in a comb can be uniquely identified by strings, called *comb identifiers*. These uniquely identify a point by describing a path through the child functions. The first character in the string is an index to the flow X_0 (also known as the *root flow*). From then on each successive character of the string describes the number of the child chosen at that branch. For example the great grand child of the i^{th} point in X_0 found by taking its j^{th} child, the k^{th} child of that and then the l^{th} child of that can be described by the string $ijkl$ formed from the concatenation of the four integers. This point is notated as $\langle ijkl \rangle$.

If the sequence of flows is infinite the strings can also be infinite. An infinite string which describes a path from a point in the flow is called a descent of the comb. Associated with a descent is a *descent point*. The descent points identified by the infinite string σ is notated as $\langle \sigma \rangle$. If the flows have values then the descent also defines a value sequence. If this sequence converges to a limit, then the value associated with the descent point is the limit of this value sequence.

In particular let us identify two special descents, an infinite string of 0's which represents a descent down the leftmost path from a point, and the rightmost descent which corresponds to always choosing the $c(\mathbf{x}, r(\mathbf{x}))$ child at each point. Taking these descents from points in the comb will be very useful in proving properties about combs. The rightmost and leftmost descents from the point $\langle \sigma \rangle$ are notated $\langle \sigma^L \rangle$ and $\langle \sigma^R \rangle$.

The set of infinite descents can be thought of as analogous to the decimal expansion of the reals, except that there are always ten choices of character for each place in that

expansion where as in a comb the number of children at each point in a descent can vary arbitrarily.

Definition

A **comb identifier** of a comb C_n defined over a sequence of flows X_i, is a finite string σ which can be defined recursively as follows: σ is a comb identifier if $\forall i > 0$

$$\begin{aligned}&\mathbf{x}_0 \in X_0,\\ &\langle\sigma_0\rangle = \sigma_0,\\ &\sigma_i \in \{0,\ldots,(r(\langle\sigma[i-1]\rangle))\}, \text{and } \langle\sigma_i\rangle = c_{i-1}(\langle\sigma[i-1]\rangle)\end{aligned}$$

Definition

A string σ is a **descent** if it has unbounded length (i.e. there is no N such that $N > |\sigma|$) and $\forall n, \sigma[n]$ is a comb identifier.
Associated with a descent is a **descent point** $\langle\boldsymbol{\sigma}\rangle$. The value assigned to this point is called the **descent value**, $\theta(\langle\boldsymbol{\sigma}\rangle)$, and is defined as the limit of the sequence of values $\theta_1(\langle\sigma[1]\rangle), \theta_2(\langle\sigma[2]\rangle), \ldots$, if it exists.
The **rightmost** descent from any point $\langle\sigma\rangle$ is an infinite string σ^R where $\sigma^R_m = r(\langle\sigma[m-1]\rangle)$, for all $m > n$. Likewise the **leftmost** descent is a string σ^L where $\sigma^L_m = 0$ for all $m > n$.

Having defined the infinite paths through the comb we are now ready to define the limit of a comb over a sequence of flows. This is a flow made up of the descent points, with a total ordering defined over them by their descents and a value, if it exists, defined to be the corresponding descent value. It should be clear that this satisfies the definition of a flow.

Definition

The **limit X_{lim} of a comb** c_i over a sequence of flows X_i, is a flow $X_{\text{lim}} \stackrel{def}{=} (F_{\text{lim}}, \preceq_{\text{lim}}, \theta_{\text{lim}})$ consisting of root descent points

$$F_{\text{lim}} = \{\langle\boldsymbol{\sigma}\rangle \mid \forall n \geq 0, \sigma_n \in \{0,\ldots,(r(\langle\sigma[n-1]\rangle))\} \text{ and } \sigma \neq \sigma[n-1]^R\}$$

with an ordering defined as

$$\langle\boldsymbol{\sigma}\rangle \preceq_{\text{lim}} \langle\boldsymbol{\rho}\rangle \Leftrightarrow \sigma \leq \rho$$

The function θ_{lim} is defined, where it exists, as mapping the descent points $\langle\boldsymbol{\sigma}\rangle$ to their respective descent values. And $\mathcal{D}_{\text{lim}}$ is the set of all these values.

Notice that at every stage the rightmost descents are removed. This is because those

descents lead to a duplication of values in a well behaved comb, just as $0.999\ldots$ is a decimal duplication of the value 1. The removal of all rightmost descents ensures that every value in a strictly monotonic limiting flow is uniquely defined by a descent (see proof of uniqueness of descents of the continuous clock flow — Lemma 9.2.)

A useful refinement of this definition is to look at the way in which values approach the limit across the comb. If the limits of values are approached at very different rates the comb can converge unevenly. For example, in the worst case, all the points in the delta clock can converge to a value of zero.

This is very similar to the situation that arises in the analysis of limits of functions. The solution there is to define a special kind of convergence: *uniform convergence.* If the sequence converges uniformly to the limit, then all the points approach the limit at a similar rate and some of the collapsing points are avoided.

Definition

Let X_n be a convergent sequence of valuation flows, and Z_{lim} be any contiguous subflow of X_{lim} which is closed at both ends. Then a comb c_i is said to converge **uniformly** if, for any $\epsilon > 0$, there exists an N such that $\forall \langle \boldsymbol{\sigma} \rangle \in Z_{\text{lim}}, |\theta_n(\langle \sigma[n] \rangle) \ominus \theta_{\text{lim}}(\langle \boldsymbol{\sigma} \rangle)| < \epsilon$, $\forall n > N$. Uniform convergence is also defined over real valued flows, in which case $|\theta_n(\langle \sigma[n] \rangle) - \theta_{\text{lim}}(\langle \boldsymbol{\sigma} \rangle)| < \epsilon, \forall n > N$.

When a limit is being taken of a Delta machine there is a second related problem that can cause adverse effects on the values of final limit flow. In this case, there are two limiting processes going on simultaneously: the limiting of the flow points with the comb, and the limiting of the values of the bounds over Δ. With no way of relating these processes to one another it is easy to end up with combs that collapse in places because they don't branch fast enough with respect to the decreasing size of Δ. Such combs are difficult to analyze, so we define a class of combs that avoids the problem.

The critical property is the *spread* of each point which intuitively measures the sum of the Δ values at each ancestor of that point in the limit. In the delta clock this corresponds to the difference between the rightmost ancestor's value and the leftmost ancestor's value.

Figure 8.2 illustrates the idea. Spread is defined using the concept of *descendants.* The

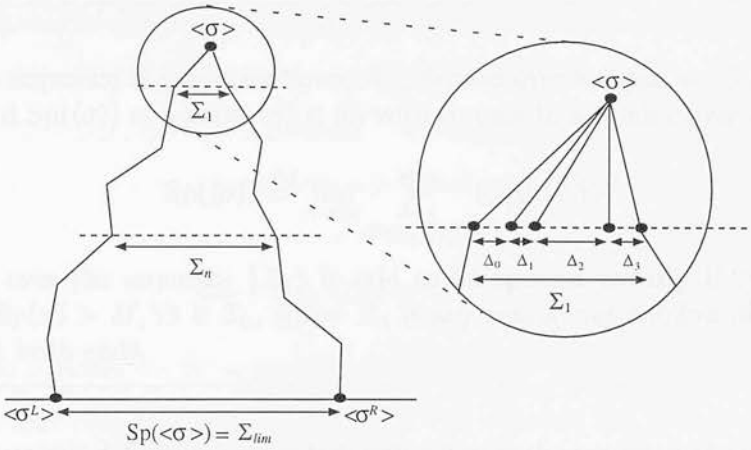


Figure 8.2: The calculation of spread $\mathrm{Sp}(\langle\sigma\rangle)$ for a point $\langle\sigma\rangle$

i^{th} descendants of a point are the set of its i^{th} generation ancestors. The exploded area in Figure 8.2 shows the first set of descendants of the point $\langle\sigma\rangle$. The sum of the values of Δ at each of the descendants at each level can be found (labelled as $\Sigma_1, \ldots, \Sigma_n$ in Figure 8.2). The spread of the point, $\mathrm{Sp}(\langle\sigma\rangle)$, is defined as the limit of these sums as the generations tend to infinity.

Spread is said to be *even* if for some flow X_i in the sequence, for any subflow of X_i there is a minimum bound M such that every point in the subflow has a spread larger than M.

Definition

The **descendants** $e_j(\langle\sigma\rangle)$ **at level** j of a point $\langle\sigma\rangle$ in a comb over a sequence of flows X_n where $\langle\sigma\rangle \in X_n$ so $|\sigma| = n$, can be defined as follows. Let I be the set of all comb identifiers with prefix σ (i.e. $\psi \in I \Leftrightarrow \psi[n] = \sigma$. Then $e_j(\langle\sigma\rangle)$ is the set $\{\psi \in I \,|\, |\psi| = n + j\}$. The sequence $e_0(\langle\sigma\rangle), \ldots, e_i(\langle\sigma\rangle), \ldots$ is the descendant sequence.

Definition

Consider a sequence of valuation flows, X_n, defined over a signal set S s.t. $\Delta \in S$. The **spread** $\mathrm{Sp}(\langle\sigma\rangle)$ of a point $\langle\sigma\rangle \in X_j$ with respect to a comb c over X_n is given by

$$\mathrm{Sp}(\langle\sigma\rangle) \stackrel{def}{=} \lim_{n\to\infty} \sum_{\mathbf{y}\in e_n(\langle\sigma\rangle)} \theta_n(\mathbf{y})(\Delta)$$

The comb over the sequence $\{X_j\}$ is said to be **spread evenly** if $\forall n, \exists M > 0$ such that $\mathrm{Sp}(\mathbf{x}) > M, \forall x \in Z_n$, where Z_n is any contiguous subflow of X_n which is closed at both ends.

One final important definition associated with a flow is the sequence of children in each descent. Such a sequence is called a branching sequence, and some of the results we obtain will require properties of the branching sequence.

Definition

The **branching sequence** $\{b_n\}$ for a descent σ from a point $\langle\rho\rangle$ is an integer sequence of the number of possible branches at each point in the descent, i.e. $b_n = (r(\langle\sigma\rho[n]\rangle))$.

Having defined combs and described various features of them, I will now go on to present some general results about the structure of the limit flow and about its values. Then I will use these concepts to arrive at a definition for continuous I/O machines at the end of this chapter.

8.2.1 Limit flow structure

The first result is to show that providing all the descents in comb have enough children, the number of points in the limiting sequence will be uncountable. In fact not many children are needed, but every branching sequence must have an infinite subsequence in which every point has at least two children.

Theorem 8.1 *In convergent comb c_n over a sequence of flows X_n, if in any branching sequence $\{b_n\}$, there is an infinite subsequence $\{b_i\}$ such that for all i $b_i \geq 2$, then the limit flow $X_{\lim}$ is uncountable.*

A complete proof of this result is given in Appendix B, with other results from this chapter. The idea of the proof is fairly straightforward. It can be seen that each descent point is identified uniquely by its descent. This is clear from the definition of ordering of the limit points, since two points are only equal in the order if they share a descent. It is then shown that these strings are uncountable, by the process of diagonalization, the same technique used to show that the real numbers are uncountable.

The next two lemmas are consequences of the isotonic property of child functions in combs. The first result demonstrates that if two descents are distinct at a point in the flow sequence, then their limit points will be in the same order that each of their respective ancestors were.

Lemma 8.2 *(Monotonicity of limit flow points) For a convergent comb c_i over a sequence of flows X_n,*

$$\exists n_0, \langle \sigma[n_0] \rangle \prec \langle \sigma'[n_0] \rangle \Leftrightarrow \langle \boldsymbol{\sigma} \rangle \prec \langle \boldsymbol{\sigma'} \rangle.$$

Proof Because the child functions are exclusive and isotonic then $\langle \sigma[n_0] \rangle \prec \langle \sigma'[n_0] \rangle \Leftrightarrow \forall n > n_0, \langle \sigma[n] \rangle \prec \langle \sigma'[n] \rangle$. Then see that the ordering of the points $\langle \boldsymbol{\sigma} \rangle, \langle \boldsymbol{\sigma'} \rangle \in X_{\lim}$ is based on string ordering, as it is in any of the flows in X_n. This ordering is based on the first difference in the strings, which must occur at some finite n if it is to occur thus:

$$\forall n, \sigma[n] < \sigma'[n] \Leftrightarrow \sigma < \sigma'$$

□

8.2.2 Limit flow values

The next lemma is related to the previous one but reflects orderings on values rather than on points. Notice that it is a slightly weaker result since two branches that strictly differ at each level can still tend to the same value. To simplify the notation a shorthand is used in the next lemma: the value of a signal $\mathtt{s}$ at a point $\langle \sigma \rangle$ in a valuation flow is normally written $\theta(\langle \sigma \rangle)(\mathtt{s})$, but this will be shortened to $\langle \sigma \rangle_{\mathtt{s}}$ (and $\langle \boldsymbol{\sigma} \rangle_{\mathtt{s}}$ for points in the limit).

Lemma 8.3 *(Monotonicity of limit values) For a convergent comb c_i over a sequence of valuation flows $X_i, \forall n \geq 1$*

$$\forall n, \langle \sigma[n] \rangle_{\mathbf{S}} < \langle \sigma'[n] \rangle_{\mathbf{S}} \Rightarrow \langle \boldsymbol{\sigma} \rangle_{\mathbf{S}} \leq \langle \boldsymbol{\sigma'} \rangle_{\mathbf{S}}.$$

Proof The statement is established by contradiction. Assume that $\forall n$, $\langle \sigma[n] \rangle_{\mathbf{S}} < \langle \sigma'[n] \rangle_{\mathbf{S}}$ but $\langle \boldsymbol{\sigma} \rangle_{\mathbf{S}} > \langle \boldsymbol{\sigma'} \rangle_{\mathbf{S}}$. Let $\langle \boldsymbol{\sigma} \rangle_{\mathbf{S}} - \langle \boldsymbol{\sigma'} \rangle_{\mathbf{S}} = \delta$. By convergence of the two sequences, it must be possible to find an N such that $\forall n > N$

$$\langle \sigma[n] \rangle_{\mathbf{S}} > \langle \boldsymbol{\sigma} \rangle_{\mathbf{S}} - \frac{\delta}{2}$$

and

$$\frac{\delta}{2} + \langle \boldsymbol{\sigma'} \rangle_{\mathbf{S}} > \langle \sigma'[n] \rangle_{\mathbf{S}}$$

but $\langle \boldsymbol{\sigma'} \rangle_{\mathbf{S}} + \frac{\delta}{2} = \langle \boldsymbol{\sigma} \rangle_{\mathbf{S}} - \frac{\delta}{2}$ so

$$\langle \sigma[n] \rangle_{\mathbf{S}} > \langle \boldsymbol{\sigma} \rangle_{\mathbf{S}} - \frac{\delta}{2} > \langle \sigma'[n] \rangle_{\mathbf{S}}$$

giving a contradiction. □

The last result of this chapter provides conditions under which the limit flow of a comb will be θ-continuous at a point. You may recall from Chapter 6 that a θ-continuous flow was a flow in which the values along the flow change smoothly. Intuitively, the theorem says that for a point $\langle \sigma \rangle$, providing the comb converges uniformly and the distance between the values of $\langle \sigma[n] \rangle$, and the values of its successor and predecessor continue to decrease with n, then the limit flow is θ-continuous at the point $\langle \boldsymbol{\sigma} \rangle$.

Theorem 8.4 *The limit $X_{\lim}$, if it exists, of a sequence of valuation flows X_n with respect to a comb c_n is θ-continuous at a point $\langle \boldsymbol{\sigma} \rangle \in X_{\lim}$ if:*

- *c_i converges uniformly*
- *for any $\epsilon > 0$ there exists N such that for all $n > N$.*

$$\left| \theta(\langle \sigma[n] \rangle) \ominus \theta(\langle \overleftarrow{\sigma[n]} \rangle) \right| < \epsilon \text{ and } \left| \theta(\langle \sigma[n] \rangle) \ominus \theta(\langle \overrightarrow{\sigma[n]} \rangle) \right| < \epsilon$$

A full proof of this result can be found in Appendix B. To prove continuity it is necessary to find points on either side of the point $\langle\sigma\rangle$ which are arbitrarily close in value. This is done by considering the sequences of the point's neighbours which are sufficiently far down to lead to suitably close points.

That ends our investigation of general properties of combs. The chapter is concluded by providing a definition of a continuous machine which makes use of these limits.

8.3 Continuous I/O machines

We are now ready to define the limit of a sequence of flows, and following from this, the limit of a sequence of delta flow machines. This definition is significant since most of the rest of the thesis is devoted to demonstrating its validity

---

Definition

For a sequence of flows X_n, let the set of combs c which converge uniformly with even spread over X_n be called C. Then the limit of X_n exists if the set of all flows defined by the limits of $c \in C$ is globally equivalent. In this case it is taken to be any one of these limit flows from that set.

An important aspect to of this definition is its generality. The limit of a sequence of flows is defined to be a limit over *all* combs which converge uniformly with even spread. The following questions must be asked:

- Do any such combs exist?
- And, if they do, do they produce the same limit?

If the answer to either is no then the limit is not well defined. From this definition the limit of a sequence of delta flow machines is simply:

Definition

The limit, if it exists, M_{lim} of a sequence of delta flow machines $M_\Delta^{(n)} = (S_M, V_{M_\Delta}, R(\chi))$ is a **continuous I/O machine** $M_{\text{lim}} = (S_{M_\Delta} \setminus \{\Delta\}, V_{M_\Delta}, R(\chi)_{\text{lim}})$. Where $R(\chi)_{\text{lim}}$ is the limit of $R(\chi)$, if it exists.

Proving that this definition for continuous machines is valid for a wide range of delta

machines is the task of the next two chapters. In the first, we look in detail at the limiting process as it is defined for the simplest non-trivial delta machine, the delta clock. These results for the clock are then built on in the subsequent chapters to show that the definition applies to more sophisticated machines.

8.4 Summary

This chapter began the investigation of a limit over Delta flow machines. The chapter began by introducing combs and a number of important concepts associated with them. Then their structural properties were investigated, and under fairly loose constraints a class of combs was found which had uncountable limits. Properties of flow value were investigated next, in particular a class of combs in which the parameters changed continuously (θ-continuous) was described. Finally the definition of a Continuous I/O machine was presented.

9

The continuous clock

When your watch gets out of order you have a choice of two things to do: throw it in the fire or take it to the watch-tinker. The former is the quickest.

— *Mark Twain*[1]

This chapter investigates the definition of a continuous I/O machine by looking at the simplest example, the continuous clock. All the results in this chapter lead up to the final result, Theorem 9.5, which states that the continuous clock is well defined. This involves first proving that at least one comb exists for a suitably general class of sequences of delta clocks, and then demonstrating that all combs over these sequences give equivalent limiting flows.

The existence of the continuous clock is important for the development of this theory. In principle continuous machines could be defined to be reactive, taking the positive real number flow $\|\mathbb{R}^+\|$ as a input and producing trajectories as output. The existence of the continuous clock removes the need for an input to define the topology of the output flow and means that continuous machines can, as originally intended, spontaneously produce output.

[1] Following the Equator, Pudd'nhead Wilson's New Calendar

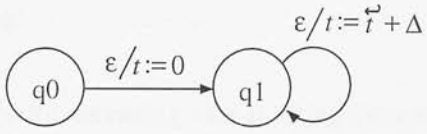


Figure 9.1: A delta clock

9.1 The clock definition

The continuous clock is defined to be the limit of a delta clock which, as we have already seen, has a single initialization transition which sets t to 0, and a single looping transition which increments the value of t by Δ each iteration. This is shown in Figure 9.1, formally it is defined as follows.

Definition

The continuous clock is a machine C_{lim} defined as the limit of the sequence of delta flow machines $C_n = (S_{C_\Delta}, V_{C_\Delta}, T_n)$ called delta clocks, where $S_{C_\Delta} = \{t, \xi, \Delta V_{C_\Delta} = \mathbb{R} \cup \{q0, q1\}$, and $T_n = R(\chi)$ and

$$\chi = \{(\{(\overleftarrow{\xi}, q0)\}, \{(\mathtt{t}, 0), (\xi, q1)\}),$$
$$(\{(\overleftarrow{\xi}, q1)\}, \{(\mathtt{t}, \overleftarrow{t} + \Delta), (\xi, q1)\})\}$$

Notice that the signal values 't' are presented in italics, whereas the signal names '`t`' are in Courier font.

9.2 Validating the definition

Having formally defined the continuous clock we must now begin the process of proving that the limit is well defined for some non-empty set of sequences of delta clocks. There are two stages to this process. The first is an existence proof which involves finding a non-empty set of sequences of delta machines for which at least one comb can be found which meets the criteria for a limit sequence. The second stage is to show that all sequences that meet these criteria lead to equivalent, well defined, limit flows.

9.2.1 Finding combs

A very important measure for assessing any limiting process is its generality. In this case: how big is the set of sequences of delta clocks over which the limit is well defined? Consider, for example, the limiting process that defines a Riemann Integral. This takes an approximation of the area under a curve as the sum of a set of rectangles that are drawn underneath it, called the Riemann sum. As the width of the rectangles decreases to zero the difference between their area and the area under the curve tends to zero, so that the approximation becomes exact. Part of the reason that this technique is so well accepted as a theory of integration is its generality. The set of sequences of Riemann sums over which the limit is well defined is very large. When integrating a region of a curve, any set of rectangles can be used at each stage so long as they fit side by side in the region under consideration and the top of each one intersects with the curve at some point. If the maximum width of the rectangles at each stage tends to zero, their sum will tend to the area.

The point is that Riemann did not insist that the rectangles were of equal widths or that they tended to zero at the same rate. Either of these conditions would have made the final result less compelling. One of the strongest results of this thesis is the generality of the limiting process, which is comparable to that of Riemann's limiting process.

Let us define a broad restriction on sequences of delta machines. For any machine, the values of Δ are bounded; let us insist that these bounds converge down the sequence. Machines which satisfy these restrictions are said to have *convergent delta bounds*

Definition

A sequence of delta flow machines M_n is said to have **convergent delta bounds.** if for all $\epsilon > 0, \exists N$ such that $\max(\Delta)_n < \epsilon, \forall n > N$.

So the first step in validating the definition is to show that with this general restriction on sequences, a comb can always be found which meets the required criteria. In fact, the result states that for any sequence of clocks with convergent delta bounds there is a *subsequence* of them over which a clock can be defined. This is a weaker result

that stating that all clocks with convergent delta bounds have limits but the proof requires tighter constraints on the changes in Δ, in particular we need to ensure that the maximum value of Δ at each flow is half the minimum value of Δ for its predecessor.

Theorem 9.1 *For all sequences of delta clocks C_k which have convergent delta bounds, there exists a subsequence of clocks C_n which have a well defined limit: the continuous clock $C_{\lim}$.*

The proof of this result is involved, but not very instructive. It involves defining, for the general case, a subsequence and corresponding comb, and the proving that these have the required properties. It can be found in Appendix C.

9.2.2 Limit equivalence

Having shown that suitable combs exist the next step is to show that the set of limits over all of them is globally equivalent. This process is simplified because we already know that the flow we are expecting is a valuation flow intuitively equivalent to the positive real number flow $\|\mathbb{R}^+\|$. To prove that this is indeed the result of the limiting process we begin with two lemmas. The first shows that each descent value is unique and the second proves that these values are unbounded.

Lemma 9.2 *Any two distinct points in the continuous clock have different values.*

The proof relies on the fact that if two limit points are different then either they are from different root points or their descents differ. An argument is presented that demonstrates, that in either case, because of the even spread of the flow, the limit points must be separated by a finite value, and so must differ. The proof is shown in full in Appendix C.

Lemma 9.3 *The values of points in the flow $T_{\lim}$ are unbounded.*

The proof of this result uses the fact that the root flow is unbounded and the comb is evenly spread. The full proof is in Appendix C.

These lemmas help us to prove the following theorem, which states that the limit contains all the positive real numbers.

Theorem 9.4 *For any $r_0 \in \mathbb{R}^+ \cup \{0\}$, there is a point $\langle\sigma\rangle \in T_{\lim}$ in the continuous clock such that $\theta_{\lim}(\langle\sigma\rangle)(\mathtt{t}) = r_0$.*

This proof involves choosing an arbitrary real number r_0 and demonstrating that there is a descent whose values become arbitrarily close to it. The descent value is the limit of this sequence, so must be r_0. The details can be found in Appendix C.

Finally the main result of the chapter which ties together all the results so far, and shows, as promised, that the limit of any convergent sequence of delta clocks is uniquely defined as the continuous clock.

Theorem 9.5 *For any sequence of delta flow clocks with convergent delta bounds, the limiting continuous clock is well defined.*

Proof First at least one example of a clock exists (from Theorem 9.1).

Secondly it is necessary to show that any two limiting combs c_n, d_n over a set of delta clock flows will produce equivalent limits. This is the case because any flow in the limit of a sequence of delta clocks is equivalent to the flow $X = ([0, \infty), \leq, \lambda\mathbf{x}.\{(\mathtt{t}, \mathbf{x})\})$. Since for $\mathbf{x} \in X, \mathbf{y} \in T_{\lim}$ the mapping

$$g(\mathbf{x}) \stackrel{def}{=} \mathbf{y} \quad \Leftrightarrow \quad f(\mathbf{x}) = \theta_{\lim}(\mathbf{y})(\mathtt{t})$$

is:

- well defined (since values are unique Lemma (9.2)
- covers $\mathbb{R}^+$ (Theorem 9.4)
- is isotonic by monotonicity of values (lemma 8.3).

Therefore, up to flow equivalence, there is only one flow defined. □

As a corollary to this result it can now be proved that the output of the continuous clock is a continuum[2].

Corollary 9.6 *The history of a continuous clock is a continuum.*

Proof Clearly the flow $X = ([0,\infty), \leq, f(\mathtt{x}) \stackrel{def}{=} \{(\mathtt{t},\mathtt{x})\}, [0,\infty))$ is dense and connected, and therefore a continuum. The proof above shows the history of the continuous clock to be equivalent to this. □

9.3 Summary

In this chapter the continuous clock was defined formally, and the definition validated. The validation involved demonstrating the existence of a general set of delta clocks for which a suitable comb was guaranteed to exist, and then showing that all such combs lead to an equivalent limit.

The next chapter continues the investigation into continuous I/O machines by building on this result, to define machines which will express continuous functions of time, integrations, dynamical systems, and ultimately allow for the construction of interactive machines.

[2] It would be nice to be able to prove this result over limit flows of some more general class of combs. Unfortunately the proof of completeness (which is the missing piece) over reals requires the axiom of completeness. Since we have no such axiom, and cannot reasonably add one to this invented structure, this result could not be established earlier.

10

Generating machines

I have always hated machinery, and the only machine I ever understood was a wheelbarrow, and that but imperfectly.

— *Eric Bell (1883-1960)*[1]

So far, it has only been shown that the definition of the limit of a delta machine can be applied to a sequence of delta clocks. This chapter builds on that result to demonstrate how the definition of continuous machines can be applied to machines with output flows which can be described as trajectories (functions of time), dynamical systems and integrator functions (functions involving integration of parameters). Each of these results builds on the final result of the last chapter by showing that the limits of these machines are well defined for any sequence of delta bounds and corresponding comb for which the continuous clock is well defined.

10.1 Trajectories

In this section we look at machines with output flows defined as trajectories, that is functions of the signal `t` (see figure 10.1).

$$u := f(t).$$

[1] In H. Eves Mathematical Circles Adieu, Boston: Prindle, Weber and Schmidt, 1977.

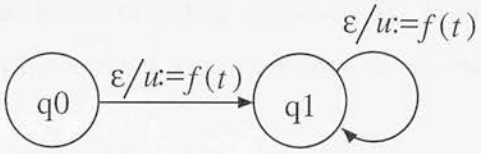


Figure 10.1: A trajectory machine (the assignments to t are assumed)

The aim of this section is to show that, with certain restrictions on f, the limits of machines containing such assignments are well defined. The continuity properties of the outputs of these machines are also investigated and it is demonstrated for example that a flow defined in this way is θ-continuous if f is a continuous function.

The flows that we are about to encounter involve increasing numbers of signals. In order to help deal with this let us introduce a piece of flow notation, the *flow component* $X(\mathtt{s})$, which refers to a flow of the values (note: not valuations) of a single signal '$\mathtt{s}$'.

Definition

For a valuation flow $X \stackrel{def}{=} (F, \preceq, \theta)$, a **flow component** $X(\mathtt{s})$ is defined to be $X(\mathtt{s}) \stackrel{def}{=} (F, \preceq, \lambda \mathbf{x}.\theta(\mathbf{x})(\mathtt{s}))$.

Associated with this definition is a small lemma which shows that if all the components of a valuation flow converge, then the whole flow converges. The proof is straightforward, and given in Appendix D.

Lemma 10.1 *For a sequence of valuation flows X_n over a signal set S, if $\forall \mathtt{s} \in S, X_n(\mathtt{s})$ converges then so does X_n.*

The next result shows that a delta machine that describes a simple trajectory of a continuous function has a well defined limit for any comb and delta bounded sequence over which a delta clock would converge. The theorem refers to a sequence of delta flow machines $M_n \stackrel{def}{=} (q0, V, S, X_n)$, like that shown in Figure 10.1, defined over the signal set $S \stackrel{def}{=} \{\mathtt{u}, \mathtt{t}, \Delta\}$.

Theorem 10.2 *For the sequence of machines M_n if f is continuous then for any comb over which $X_n(\mathtt{t})$ converges, X_n also converges to a limit $X_{\lim}$ in which the values of the signal u are defined as the trajectory $f(t)$.*

The proof of this can be found in full in Appendix D. It involves showing that the valuation sequences which contain t and x converge as the value of t converges.

10.1.1 Continuity of trajectories

Before moving on to look at dynamical systems there are a number of corollaries regarding the continuity of the limit flow $X_{\lim}$ that lead on from the last result. The relationship between continuity of assignment functions and θ-continuity of the flows is very important since it is not until Chapter 11 that we will look at how to introduce discontinuities into the flows.

These results refer to a second machine $M_n' \stackrel{def}{=} (S', V, X_n')$, which is very similar M_n except that there are additional signals v and w so $S' \stackrel{def}{=} S \cup \{\mathtt{v}, \mathtt{w}\}$. As before the value of signal u is given by $u := f(t)$. Here the value of signal w is described as a function of the signal v given by $w := h(v)$.

Corollary 10.3

1. *The flow $X'_{\lim}(\mathtt{u})$ is θ-continuous (except at start$(X'_{\lim})$) iff f is continuous.*
2. *If $X'_{\lim}(\mathtt{v})$ is θ-continuous then it can be described as a continuous trajectory where $v = g(t)$ for a continuous function g.*
3. *The flow $X'_{\lim}(\mathtt{w})$ is θ-continuous (except at start$(X'_{\lim})$) if h is continuous and v is θ-continuous.*

The proofs for these, unsurprising, results are in Appendix D along with the proof of the next lemma which extends the continuity result to valuations.

Lemma 10.4 *In a dense value flow $X_{\lim}$ over a signal set S, if $\forall \mathtt{s} \in S$, $X_{\lim}(\mathtt{s})$ is a θ-continuous flow then $X_{\lim}$ is a continuous flow.*

10.2 Dynamical systems and integrators

In this section we extend the limiting process to machines which contain dynamical systems. Dynamical systems are extremely important in modelling all kinds of physical

systems. Control engineering depends heavily on the mathematics of these equations.

In general a dynamical system can be modelled by the limit of a machine with signals defined by expressions of the form:

$$u := \overleftarrow{u} + \Delta f(\overleftarrow{e}). \tag{10.1}$$

where $\overleftarrow{e}$ is a valuation of all previous signal values including $\overleftarrow{u}$.

The following result establishes criteria under which the limit of some of these machines is well defined. It refers to a sequence of delta flow machines M_n with output flow X_n and signal set S with $\{\mathtt{t}, \mathtt{u}\} \subseteq S$ where, the value of $\mathtt{u}$ is assigned by the expression:

$$u := \overleftarrow{u} + \Delta f(\overleftarrow{t}, \overleftarrow{u})$$

Theorem 10.5 *The limit flow $X_{\lim}$ exists for any comb which converges over the delta clock if f has a bounded partial derivative with respect to its second variable and if the solution of the differential equation:*

$$\frac{du}{dt} = f(t, u) \tag{10.2}$$

has a bounded second derivative. Furthermore the flow it describes satisfies equation 10.2.

The proof makes use of a standard result in numerical analysis. In effect the delta machine is calculating an Euler's method approximation to the differential equation. Under the conditions in the theorem, the error in this approximation is bounded by a constant multiple of $\max(\Delta)$ and so the result will tend to the required limit. More details are given in Appendix D.

It is reasonably straightforward to extend this result to the more general case in equation (10.1). Applying Corollary 10.3 and Lemma 10.4 any continuous valuation can be rewritten as a trajectory. Once expressed as a trajectory, Theorem 10.5 can be applied.

Let us introduce here a piece of notation. A dynamical system taken to be the limit of an expression like

$$u := \overleftarrow{u} + \Delta f(\overleftarrow{e})$$

which, at the limit will conform to the equation:

$$\frac{du}{dt} = f(e)$$

Since the delta machine is describing an assignment to u, and that assignment is an integration, this limit will usually be written:

$$u := \int f(e).$$

Integrator functions are a special case of dynamical systems where signals have assignments of the form

$$u := \overleftarrow{u} + \Delta f(\overleftarrow{a})$$

in which $\overleftarrow{a}$ is a valuation which does not containing u or any signals whose values depend on u. The above result applies to integrators without modification. However a separate proof of the limit of integrator assignments is offered as Theorem D.1 which can be found in Appendix D. This proof uses the concept of *divider points* which is not introduced until Chapter 11.

As a final note to conclude this section, it is worth pointing out that the continuous clock is a simple example of an integrator system, in which $f() \stackrel{def}{=} 1$ thus $t := \int 1$. In other words it is an integration over a constant unitary rate of change, and so makes the natural choice for a global temporal metric.

10.3 Summary

In this chapter the definition of a continuous machine was extended from clocks to apply to functions of time, integrals and dynamical systems. In the next chapter the limit is taken one stage further and applied to machines which not only output values but also accept input.

11

Filling in the holes

Time sneaks up on you like a windshield on a bug.

— *Jon Lithgow*

In this chapter we consider how to handle discontinuities in flow values. So far we have considered generator machines which deal exclusively with continuously changing parameters. Since the state of a machine is represented by a set of discrete valued signals and discrete valued signals cannot change continuously this has actually meant that, excluding the initial state, these machines have been restricted to a single state.

A discontinuity is a point in the flow at which the values jump, so the values immediately before it and the values immediately after it do not meet. The discontinuities that we will consider will be isolated so that the points on either side will be θ-continuous and so present no problem to the existing limiting process. The trouble is that some combs will find these points to have values meeting those on the right hand side (right-continuous) and some will have them meeting the values on the left hand side (left-continuous). The limit flow only exists if the set of all limits is globally equivalent and if the combs do not agree on the values of these points of discontinuity, the limit cannot be defined.

To begin with it will be demonstrated that the limits of the machines that we have seen so far are only affected at the points of discontinuity themselves. Next we will consider how the limit can be redefined to exclude the values at these points. The last

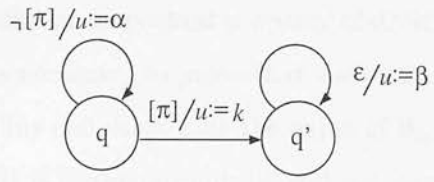


Figure 11.1: A machine with a discontinuity (the assignments to t are assumed)

part of the chapter will then present how to fill in the missing values.

We will only consider discontinuities caused by state changes explicitly. However most of the results will be sufficiently general that they would apply to discontinuous trajectories or dynamical systems without modification. Furthermore any sort of discontinuity can be effectively modelled with a state discontinuity.

11.1 Divider points

Let us begin by defining the sequence of points at which the discrete change occurs that leads to the discontinuity. We'll call these points *divider points.*

The definition refers to the general state change depicted in Figure 11.1. This shows a machine fragment $M_n = (V, S, X_n)$ with a signal set S with $\{\mathtt{t}, \mathtt{u}\} \subseteq S$, in whose output flow X_n, the values of the signal u are defined with respect to an assignment $u := \alpha$ in state q and as $u := \beta$ in state q' after the predicate π becomes true. The value of u on the transition between the two will be undefined at the limit since this is the point of discontinuity.

Notice that all state changes will look like this because by assumption there can only be at most finitely many discontinuities in any closed bounded interval of time. So the machine must spend some non-zero period of time in a state and therefore, each state will have to have a loop in it which is executed for some nonzero interval of t.

Definition

In each flow X_n there is a point at which π becomes true and the assignment to u switches from α to β. Let us call this point $\mathbf{k}_n^{\alpha|\beta} \in X_n$ a **divider point** for u. We insist that state signals are right-continuous (see section 11.3) so let $\mathbf{k}_{\lim}^{\alpha|\beta} \in X_{\lim}$ be the point at which π becomes true in $X_{\lim}$.

The next definition identifies an important property of divider point sequences, that of being *constrained*. This is necessary to prove that each discontinuity only disrupts the limit at a single point. This definition uses the value of $\theta_{\text{lim}}(\mathbf{k}_{\text{lim}}^{\alpha|\beta})(\mathtt{t})$. Now the point $\mathbf{k}_{\text{lim}}^{\alpha|\beta}$ will exist but since it is a discontinuity it will not have a well defined valuation there. However the clock variable $\mathtt{t}$ contains no discontinuities and were we to constrain the flow of the machine to this signal, it would therefore have a well defined limit. So $\theta_{\text{lim}}(\mathbf{k}_{\text{lim}}^{\alpha|\beta})(\mathtt{t})$ is, in fact, a well defined value.

Definition

In a sequence of flows X_n successive divider points for a signal $\mathtt{u}$ can form a sequence $\mathbf{k}_n^{\alpha|\beta}$. Such a sequence is said to be **constrained** if there exists an M such that

$$\left|\theta_n(\mathbf{k}_n^{\alpha|\beta})(\mathtt{t}) - \theta_{\text{lim}}(\mathbf{k}_{\text{lim}}^{\alpha|\beta})(\mathtt{t})\right| < \max(\Delta)_n M$$

11.1.1 Trajectories

The divider point thus marks a discontinuity in the state valuation of the machine. This may be a discontinuity in the signal $\mathtt{u}$. It is possible that the assignment α and the assignment β meet at the value k, but in that case the flow is continuous and there is nothing further to do so let us assume the worst. The next result shows that even if $\mathbf{k}_{\text{lim}}$ is a discontinuity, then if α is a trajectory the limit flow is convergent up to $\mathbf{k}_{\text{lim}}$. Similarly if β is a trajectory then the flow is convergent from $\mathbf{k}_{\text{lim}}$.

Lemma 11.1 *For the general transition depicted in Figure 11.1. Where the sequence of divider points* $\mathbf{k}_n^{\alpha|\beta}$ *for* $\mathtt{u}$ *is constrained.*

- *If* α *is a continuous trajectory* $f(t)$ *then for any comb* c *which converges over* $X_n(t)$*, the limit* $X_{\text{lim}}(\mathtt{u})$ *is well defined up to (but not necessarily including)* $\mathbf{k}_{\text{lim}}^{\alpha|\beta} \in X_{\text{lim}}$.

- *If* β *is a continuous trajectory* $g(t)$ *then for any comb* c *which converges over* $X_n(t)$*, the limit* $X_{\text{lim}}(\mathtt{u})$ *is well defined from (but not necessarily including)* $\mathbf{k}_{\text{lim}}^{\alpha|\beta} \in X_{\text{lim}}$.

The proof of this result (given in detail in Appendix E) shows that the points either side of a change in the mode of a machine can be well defined at the limit.

11.1.2 Dynamical systems

Discontinuous dynamical systems provide less of a problem. Because of the way in which the limit is described for a dynamical system, it is clear that the flow *up to* a constrained divider point will be well defined. It simply remains to show that the flow *from* a constrained divider point will be unaffected. This situation is dealt with in the extension to Theorem 10.5 in Appendix D. In essence the addition of a divider point adds an extra error term, but because it is a constrained divider the term tends to zero neatly, and uniform convergence is unaffected.

11.2 Limit flows

So the limit of flows *up to* a discontinuity, $Sub^{\prec \mathbf{k}}(X_{\lim})$, and *from* a discontinuity, $Sub^{\succ \mathbf{k}}(X_{\lim})$, are well defined. However in order to complete the description of continuous generating machines, it is necessary to provide a definition of the limit that can cope with a finite number of discontinuities in any closed bounded interval of time.

The definition is in two parts. The first part provides the structure of the flow, and defines the point values where they are θ-continuous. This is analogous to what we have already seen, except that the points of contention, the discontinuities, are left undefined. The second part fills in these missing values.

To achieve the first part it is necessary to define a new type of equivalence. Two uncountable flows are *equivalent almost everywhere* if they differ by, at most a countable sequence of points.

Definition

Two flows X, Y, each with uncountably many points are said to be **equivalent almost everywhere** written $X \cong Y$, if $\exists S_{XY} \subset X, S_{YX} \subset Y$ such that

$$X \setminus S_{XY} \equiv Y \setminus S_{YX}$$

and the subflows: $(S_{XY}, \preceq, \theta)$ and $(S_{YX}, \preceq, \theta)$ are regular. The smallest such set S_{XY} is called the set of **nonequivalent points** (of X to Y) in which case flow $X \setminus S_{XY}$ is called the **equivalence flow**.

Notice that the equivalence flow is uncountable, since the set of nonequivalent points is regular and therefore countable. Removing a countable number of points (a null set), from an uncountable one leaves an uncountable set.

From this definition we can construct a corresponding definition of global equivalence over a set of flows.

Definition

A set of flows R, each with uncountably many points are said to be **globally equivalent almost everywhere** , if for any flow $X \in R$

$$X \cong Y, \forall Y \in R$$

such that $P_X \stackrel{def}{=} \bigcup_{Y \in R} S_{XY}$ and the subflow $(P_X, \preceq, \theta)$ is regular. The set P_X is called the set of **global nonequivalent points** (of X over R). The flow $X \setminus P_X$ is called the **global equivalence flow** of R.

Before we use this new definition of global equivalence to fix the definition of flow limits, it is necessary to prove that the *global equivalence flow* is a well defined object *i.e.* for a set of flows R that are globally equivalent almost everywhere, there is a single global equivalence flow defined above. The following result gives this for piecewise θ-continuous valuation flows, that is, valuation flows which are θ-continuous at all but a finite number of points in any closed bounded interval, and either left or right-continuous at those.

Lemma 11.2 *For a set of uncountable piecewise-θ-continuous valuation flows R which are equivalent almost everywhere, the global equivalence flow is well defined.*

The proof of this is given in Appendix E. It involves establishing that for any two flows

taken from R, the definition of the global equivalence flow gives an equivalent result.

There is a potential theoretical problem in removing the set of global nonequivalent points. Even if each comb has a null set of isolated discontinuities, there are almost certainly uncountably many suitably convergent combs, and the union of uncountably many null sets may be non-null. This could result in a contiguous stretch of the flow being undefined, rather than the expected isolated points. Of course such a set of combs would fail to satisfy the definition of being globally equivalent almost everywhere, because the global nonequivalence points would be non-regular.

It is important to notice that this situation can only arise for machines which do not have sensibly defined limit flows. The reason is that uniformly convergent combs with even spread cannot introduce discontinuities into the limit flow. This is a corollary of Lemma 11.1. Therefore the only discontinuities in the limit flow are present because the delta machine is not waiting a nonzero length of time in each state or because of the use of discontinuous functions.

Having established that this definition is sound, let us refine the definition of the limit of a sequence of flows.

Definition

For a sequence of flows X_n, let the set of combs c which converge uniformly, with even spread over X_n be called C. Then the limit of X_n, exists if the set of all flows defined by the limits of $c \in C$ is globally equivalent almost everywhere. In which case it is taken to be any one of these limit flows from that set and redefining the values of the global nonequivalent points, to '$\perp$'.

This definition leaves undefined values in the flow. In the next section we see how to provide definitions for these values.

11.3 Filling in the holes

In this section we discuss the conventions used to define the values of the nonequivalent points in a limit flow. All points are instantaneous in time so from one point of view the value of isolated individual points is not very significant for a physical model. On the other hand these discontinuities are the points at which a discrete controller can

read the machine's parameters (as we decided in Chapter 4), and so from that point of view, the values are critical.

Fortunately, there is a very natural way to decide what a point's value should be. There are two principal candidates: either the flow should be left-continuous and meet up with the continuous values to the left or it should be right-continuous and meet up with the values to the right.

For most signals it is up to the designer of the machine as to whether its value should be left or right-continuous at a point. However, there is a special case: the value of the state signals at a discontinuity should always be *right-continuous* if the definition of a continuous machine is to retain the conventions of its discrete delta machine parents.

The value of a point at a state discontinuity is reflected by its value on the state transition between different states. In order for this to make sense in the discrete case the value of the state signals on the transition itself will be those that define the new state, the previous values will be those that define the old state. If this is to be retained in the limit the state signals will always be right-continuous.

11.3.1 Left continuity

Left continuity is the most straightforward to demonstrate in a continuous machine, as shown in Figure 11.2. This shows a looping transition in state $q0$ predicated by $[x < 1]$ which when the predicate on the state transition becomes true ($x = 1$) the transition to $q1$ is taken, whereupon the value of x is immediately set to zero.

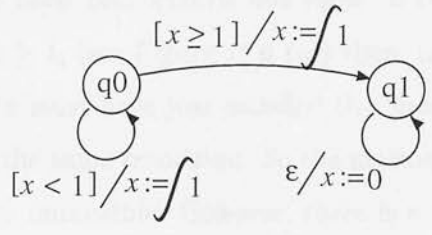


Figure 11.2: A left-continuous state transition

Left-continuous discontinuity occurs whenever the assignment to a signal on a state transition is the same as the assignment on the looping transition from the state that it

left. In the case of ambiguity, a left continuous transition can be indicated by notating the assignment on the state transition with a superscript (l), as in: $x^{(l)} := \int 1$

Values at left-continuous discontinuities are defined as the supremum of the flow up to the point of discontinuity.

11.3.2 Right continuity

Right continuity, in the simplest case, is just as straight forward. Consider, for example Figure 11.3. Here the machine loops in state $q0$, until $y = 1$, when it moves to state $q1$. In the first state x is set to 1, on the transition the value is set to 0, and there it stays in $q1$.

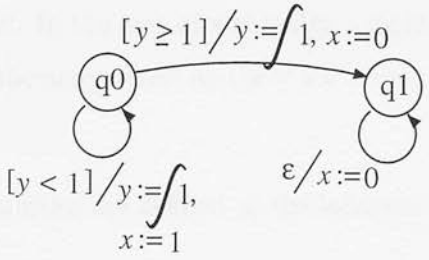


Figure 11.3: A right-continuous state transition

In this example a different signal, y, was used in the predicate to trigger the exiting of the loop. The reason that this extra signal was introduced relates to the causality conditions of the synchronous machine. You may recall that a signal on a transition in a synchronous machine must have exactly one value. If the transition is predicated on x to occur when say $x \geq 1$, (see Figure 11.4 (a)) then, in order for that transition to be taken, the value of x must have just satisfied this predicate. It is impossible to set the value of x to 0 on the same transition. So the machine fragment in Figure 11.4 (a) for example is causally impossible. However, there is a trick that helps solve this problem, called a mirror signal.

A *mirror signal* for x, written $\overline{\mathrm{x}}$, follows the value of the signal through a loop and then copies its dynamic assignment on the exit transition. This is illustrated in Figure 11.4 (b). In order to simplify the labelling this situation can be summarized by simply using

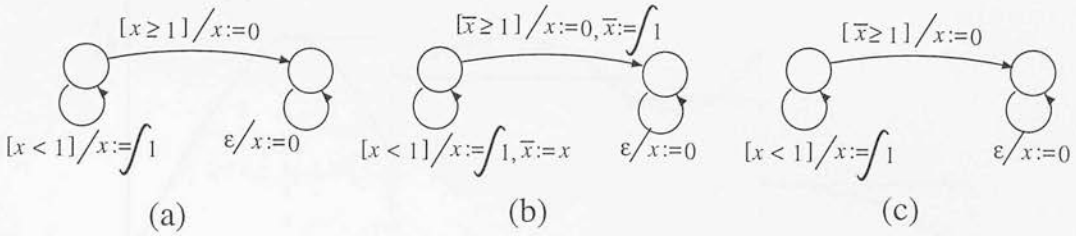


Figure 11.4: The use of mirror signals: (a) A causality error (b) The solution (c) Same as (b) with syntactic sugar

the mirror signal in the test Figure 11.4 (c).

So a right-continuous discontinuity occurs whenever the assignment to a signal on a state transition differs from the assignment on the looping transition from the state from which it has departed. In the case of ambiguity, a right continuous transition can be indicated by notating the assignment on the state transition with a superscript (r), as in: $x^{(r)} := 0$

Right-continuous discontinuities are defined as the infimum of the flow from the point of discontinuity.

11.4 Extracting sequences

It was pointed out in Part I of the thesis that one of the problems with taking a continuous approach to time for plants and environments is that it makes it difficult to combine them with discrete controllers. In order to make this combination possible we have to find suitable sequences that can be extracted from the continuum of values.

The points of discontinuity in the model are ideal for this because not only are they guaranteed to form a sequence, but they also represent the most interesting points in the model where changes of mode take place. To extract these points we define another set of signals called *discrete shadowing signals* as follows:

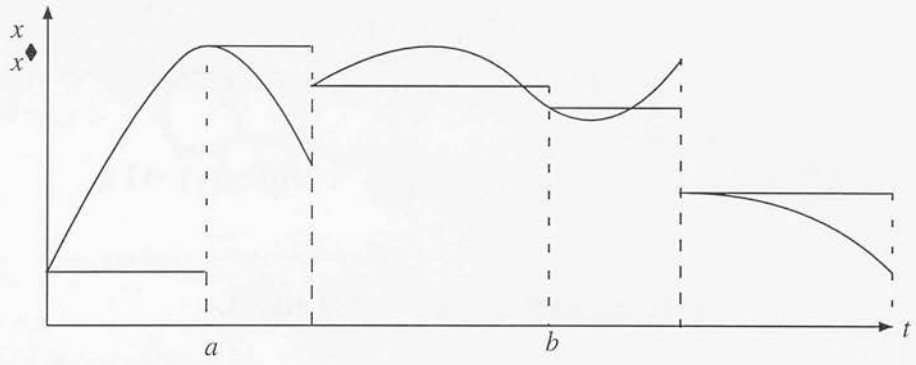


Figure 11.5: The relationship between a continuously varying signal and its discrete shadow, the points a and b represent places where discontinuities occurred in other values.

Definition

For a continuous machine M with a signal set S the set of **discrete shadowing signals** $S^{\blacklozenge}$ is a set of signals

$$\mathtt{x}^{\blacklozenge} \in S^{\blacklozenge} \Leftrightarrow \mathtt{x} \in S$$

with values defined at every transition in the machine as

$$x^{\blacklozenge} := \begin{cases} x & \text{if } x \neq \overleftarrow{x} \\ \overleftarrow{x^{\blacklozenge}} & \text{otherwise} \end{cases}$$

At the limit the value $\overleftarrow{x}$ is taken to be $\sup(Sub^{x\prec}(X_{\lim}))$.

Figure 11.5 shows how the shadow signal might relate to a signal of continuously changing value. Notice that the value of the shadow signal will be set on every discontinuity of the machine.

It may be the case that we are interested not only in the values of signals at the existing discontinuities, but also at points between them. This problem is solved by adding a sampling clock (Figure 11.6) which periodically brings about a discontinuity. This results in a shadow trace like the one in Figure 11.7.

Finally different sample clocks with different rates can be mixed together and turned on and off at different points in the operation of a machine, to allow fine grained sampling of fast changes, and coarse sampling of slower.

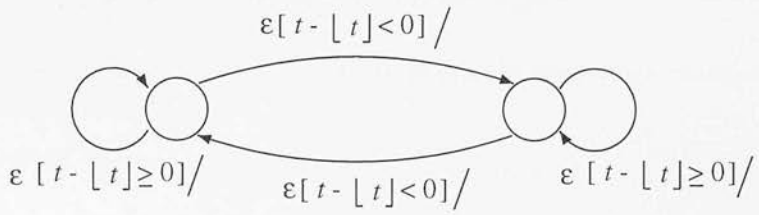


Figure 11.6: A bistable sampling clock

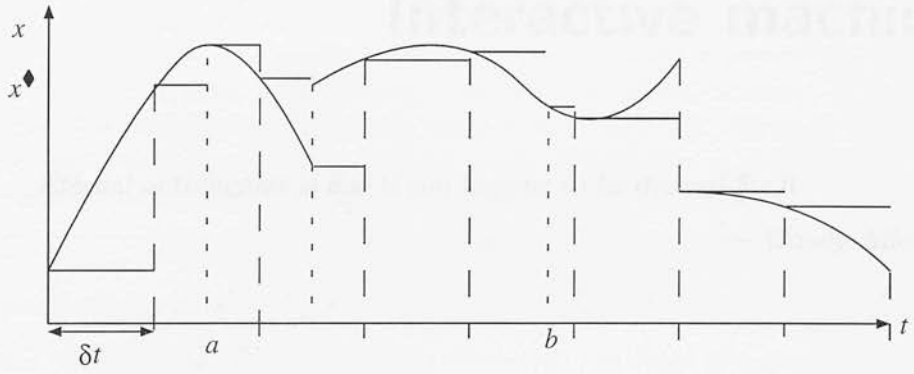


Figure 11.7: A continuously varying signal and its sampled discrete shadow, once again the points a and b represent places where discontinuities occurred in other values.

11.5 Summary

This chapter addressed the problem of discontinuities in the limiting flow. It began by demonstrating that the limiting processes that have already been developed continue to define flows on either side of a discontinuity even though the value at the discontinuity itself may be lost. Then, a new definition of the limiting process was presented which allowed us to define limits of flows which contain discontinuities. We moved on to look at how to define the values of these discontinuities at the limit. Finally *discrete shadowing signals* to provide a piecewise constant signal that could be read by a discrete controller.

12

Interactive machines

Eternal nothingness is fine if you happen to be dressed for it.

— *Woody Allen*

In this chapter inputs are added to the continuous machine. So far we have concentrated on generating machines and all the flows that we have been dealing with have been valuation flows. This chapter extends these results to define continuous *interactive* machines. The main result of this chapter, and of the thesis, is that the synchronous product of two limit machines is equivalent to the limit of the synchronous product of their respective delta machine parents.

Most of the hard work has already been done in showing that valuation flows converge. All that remains is to show that, under appropriate circumstances, assignment flows also converge as expected.

12.1 Limits over assignment flows

The first step is to define convergence over assignment flows. This is done in the same way that convergence is defined for functions. The following definition uses a new shorthand $\chi[V] \stackrel{def}{=} \chi \sqcup \{(\{\}, V)\}$.

Definition

A sequence χ_n of characteristic functions is said to **converge** at a point V to a limit $\chi_{\lim}$ if $\forall \epsilon > 0, \exists N$ s.t.

$$|\chi_n[V] \ominus \chi_m[V]| < \epsilon, \forall n, m > N$$

A sequence χ_n of characteristic functions is said to **converge uniformly** to a limit $\chi_{\lim}$ if $\exists N, \forall V \in \mathcal{E}_{d(\chi_n)}, \forall \epsilon > 0$, s.t.

$$|\chi_n[V] \ominus \chi_m[V]| < \epsilon, \forall n, m > N$$

A characteristic function is said to be **convergent almost everywhere** if it fails to converge at at most finitely many points in any closed bounded interval, in this case the limit is defined only at the points at which it converges. It is said to be **uniformly convergent almost everywhere** if it is convergent almost everywhere and converges uniformly over those points at which it does converge.

The first result shows that convergence of valuations and convergence of characteristic functions are mutually consistent in the sense that the limit of a sequence $\chi_n[V_n]$ is equal to $\chi_{\lim}[V_{\lim}]$.

Lemma 12.1 *Consider a characteristic function* $\chi_n \to \chi_{\lim}$ *and a valuation sequence* $V_n \to V_{\lim}$ *with* $V_n \in \mathcal{E}_{d(\chi_n)}$, *and the sequence of valuations* $\chi_n[V_n]$. *Then providing* $\chi_{\lim}$ *is continuous at* $V_{\lim}$,

$$\lim_{n\to\infty} \chi_n[V_n] \equiv \chi_{\lim}[V_{\lim}].$$

The proof is straightforward and is found by applying the limit first to the function, and then to the valuation, and using the triangle inequality for '$\ominus$' to combine them (see Appendix F for details).

Let us restrict the type of assignments that we will consider to those that we have already found limits for as generating machines. We call such machines *standard.*

Definition

A characteristic function is said to be **standard** if it consists only of assignments of the form:

$$u := \overleftarrow{u} + \Delta f(\overleftarrow{e}) \tag{12.1}$$

or

$$u := f(e). \tag{12.2}$$

A standard characteristic function can be partitioned into two pieces the **delta partition** χ_n^{Δ} consisting of assignments of the form (12.1), and the **fixed partition** χ_n^f the set consisting of assignments of the form (12.2).

A *standard delta flow machine* is just one in which the characteristic function is standard.

Next it is proved that for two standard characteristic functions, the limit of their union is the same as the union of their respective limits. This result is critical in enabling us to reach a definition of the synchronous product for continuous machines; the synchronous product relies on the process of zipping the assignment flows together, which in turn involves taking the union of the assignments at each point. It is important, therefore, to check that the union is still well defined at the limit.

Lemma 12.2 *Consider two standard characteristic function sequences* χ_n *and* ψ_n. *Let* χ_n, ψ_n *be uniformly convergent almost everywhere, such that* $\chi_n \to \chi_{\lim}, \psi_n \to \psi_{\lim}$. *Let* $\chi_n \cup \psi_n$ *be well defined for all* n. *Then* $\chi_n \cup \psi_n$ *is guaranteed to converge to* $\chi_{\lim} \cup \psi_{\lim}$ *at all points* V_S *where* $\chi_{\lim} \cup \psi_{\lim}[V_S] \downarrow D(\chi^f_{\lim} \cup \psi^f_{\lim})$ *is continuous.*

The proof of this lemma can be found in Appendix F. It proceeds by dealing with the delta partition and the fixed partition separately, since one is changing, and the other is not.

12.2 Synchronous product

We are now nearly ready to define the synchronous product for continuous machines. Before doing so, we need to define a way for the flows of the respective machines to synchronize. There is only one way to map two forward regular flows together, but

the uncountable number of points in continuum flows could be aligned in an infinity of ways.

The answer is that we need to decide on a temporal metric for the machines. The obvious way to do this is to use $\mathtt{t}$. If both machines share the signal $\mathtt{t}$ then this provides a unique mapping for them to synchronize with. Proving this happens automatically is the next result.

Lemma 12.3 *Consider the flows X_1, X_2 of two timed continuous I/O machines M_1, M_2 which share only the clock signal $\mathtt{t}$. There is at most one order preserving 1-1 mapping $f : X_1 \to X_2$ such that $X_1 \amalg_f X_2$ is well defined.*

Proof

$$(X_1 \amalg_f X_2) \text{ is well defined} \Leftrightarrow$$
$$\forall \mathbf{x} \in X_1,\ (X_1(\mathbf{x}) \cup X_2(f(\mathbf{x}))) \text{ is well defined} \Rightarrow$$
$$\forall \mathbf{x} \in X_1, X_1(\mathbf{x})(\mathtt{t}) = X_2(f(\mathbf{x}))(\mathtt{t})$$

The right hand-side defines a unique 1-1 order preserving mapping between the points, hence the result. □

It should be pointed out that insisting that both machines define $\mathtt{t}$ does not imply that only one such metric can be defined for each machine. In Chapter 13 an example of a model of relativistic clocks is given to illustrate this point. In this case the synchronizing clock $\mathtt{t}$ is chosen for a particular frame of reference.

We can now define the synchronous product of two Continuous I/O machines.

Definition

Let M_1 and M_2 be two Continuous I/O machines with $O_{M_1} \cap O_{M_2} = \{\mathtt{t}\}$. The **synchronous product** of M_1 and M_2, denoted by $M_1 \parallel M_2$, is the machine M where

- $q0_M \stackrel{def}{=} q0_{M_1} \cup q0_{M_2}$
- $V_M \stackrel{def}{=} V_{M_1} \cup V_{M_2}$
- $\Xi_M \stackrel{def}{=} \Xi_{M_1} \cup \Xi_{M_2}$
- $I_M = (I_{M_1} \setminus O_{M_2}) \cup (I_{M_2} \setminus O_{M_1})$, $O_M = O_{M_1} \cup O_{M_2}$
- $F_M = F_{M_1} \amalg F_{M_2}$

Finally, the main result of the thesis is to show that the synchronous product of two continuous machines is equivalent to the limit of the synchronous product of their delta flow parents.

Theorem 12.4 *For any two delta-flow machine M_Δ, M'_Δ the synchronous product of their limits is equivalent to the limit of their synchronous product. i.e.*

$$M_{\text{lim}} \| M'_{\text{lim}} \equiv \lim_{\Delta \to 0} M_\Delta \| M'_\Delta$$

The proof of this theorem involves showing that the flows of the respective machines are equivalent. This is done by choosing an arbitrary point in one of the flows, finding the corresponding point in the other with a matching value of t, and showing that the other signal values also match.

12.3 Summary

This concludes our investigation of the limiting process. It has been demonstrated to be a well defined limit over a very wide variety of machines, limiting sequences and combs. It was shown that the limiting process can also be applied to an assignment flow which describes an interactive machine. The result was approached by first investigating the limits of characteristic functions, and then applying these limits to entire flows. To conclude the chapter the main result established the validity of a synchronous product over Continuous I/O machines.

13

Examples

...the source of all great mathematics is the special case, the concrete example.

— *Paul R. Halmos* [1]

Elwood: Its 106 miles to Chicago. We've a full tank of gas, half a pack of cigarettes, its dark and we're wearing sunglasses

Jake: Hit it!

— *Dan Ackroyd and John Belushi* [2]

In this chapter we look at a number of simple examples which illustrate continuous automata as modelling systems. The final example shows how delta machine simulation can be used for verification.

13.1 Thermostat

The model in figure 13.1 shows a model of a thermostat heating a room. The room has a thermal capacity C, and leaks heat at a rate L, so the dynamical system describing

[1] I Want to be a Mathematician, Washington:MAA Spectrum, 1985.

[2] in *The Blues Brothers* Warner Brothers 1980

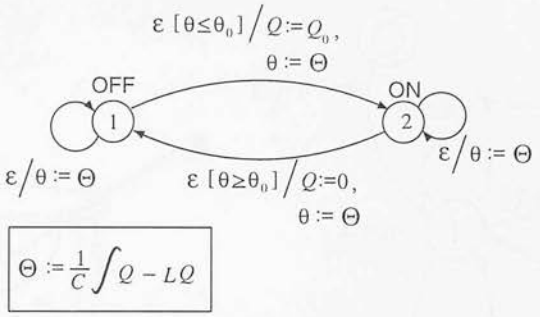


Figure 13.1: A thermostat

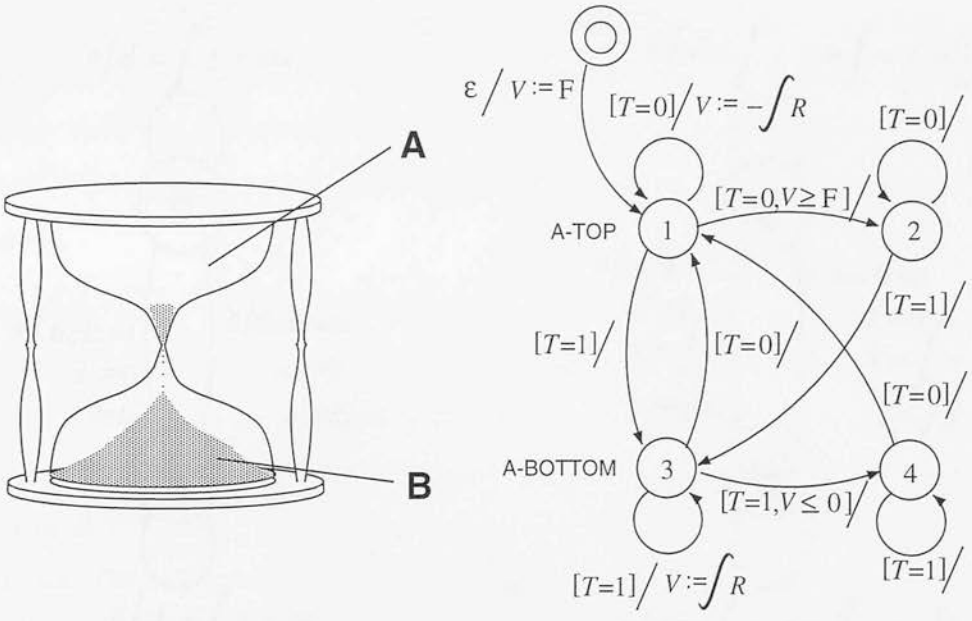
the room's temperature is:

$$Q = L\theta + C\frac{d\theta}{dt}$$

where Q is the quantity of heat input into the system, and θ is the temperature of the room. A thermostat controlled heater introduces two states. When the room exceeds a temperature, let us call it θ_0, the thermostat turns the heater off and Q is set to zero. Once the temperature has fallen back below θ_0 the thermostat switches the heater back on and it outputs a fixed quantity of heat Q_0 until the room is once again hot enough.

13.2 Hour glass

Consider the hour glass and model depicted in figure 13.2. The hour glass itself is labelled as having two ends, A and B. The model has five states in total, an initial state, and four others. States 1 and 2 represent the A end of the hour glass being uppermost, states 3 and 4 represent the B end of the hour glass being uppermost. In states 1 and 3 there is sand in the upper bulb draining down, in states 2 and 4 the draining is complete, the upper bulb is empty and the system is static. The model has one continuous parameter V which represents the amount of sand in the A bulb, and two constants: F, the total volume of sand, and R, the rate at which it can flow from one bulb to the other. Finally there is a single input `T` which controls the turning of the glass.



13.3 Twin clocks

This example is a very simple illustration of the modelling of a problem in Einstein's Special Theory of Relativity (STR) [Einstein 61]. It is included here to demonstrate how real physical models can consider synchronization of components to be independent of the local temporal metric, and that we can build two continuous I/O machine models with local and sensibly compose them using the synchronous product.

The underlying premise in STR is that observations of physical systems must take place from a *frame of reference* that is located in space. Suppose, for example, I am observing a distant event such as a lightning flash, the event only occurs in my frame of reference when information about the event has reached me. The speed at which information can travel is limited by the velocity of light, and so the lightning does not strike in my perspective until I see the lightning flash from my frame of reference. Einstein thus rejects the idea of global synchronization of events since there is no way to know if two clocks that have been separated are still running at the same rate. He proposes instead that synchronization is relative to the observer for whom an event has only occurred when he or she has seen it. Consequently the velocity of an object relative to the frame of reference of the observer can affect the observer's perception of the time it takes that object to change.

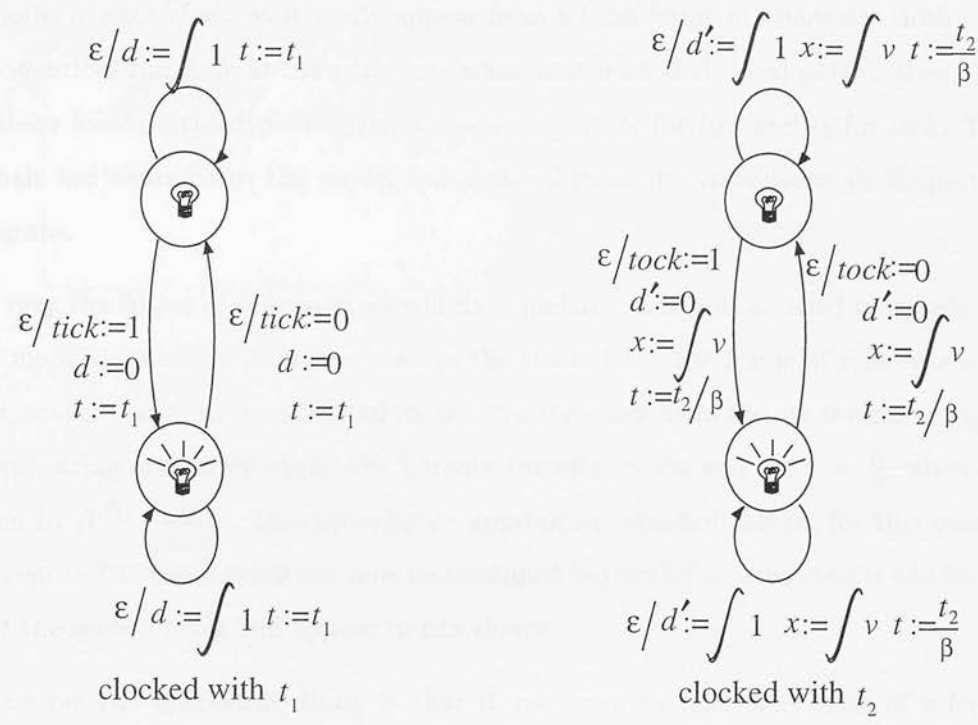


Figure 13.2: Two clocks in different relative frames of reference.

The idea is illustrated here with a simple example of two clocks modelled here in Figure 13.2. Let us suppose that each clock turns a light on for a second and then off for a second. In our model this is represented by the signals `tick` and `tock` in the respective clocks that are switched from one to zero and back.

Let us suppose that the clock on the left (*tick*) is stationary, on earth say, and the clock on the right (*tock*) is moving at a velocity v close to the speed of light. In order to avoid any effects of acceleration, let us also suppose that the path of *tock* brought it momentarily within the same frame of reference as *tick* and at that instant the clocks were synchronized.

Now as *tock* pulls further away the light that it emits has to travel further to reach earth, and so the appearance is, from the perspective of the observer on earth with the first clock, that it is getting slower.

The relationship between the stationary, earth bound frame of reference, and that of the moving clock is given by Lorenz's equation. To model the effect we can first build

a model of each clock, as it would appear from a local frame of reference. Both clocks are identical and flash at the same rate when seen from their local perspectives. Time in these local perspectives is given a special signal, t_1 for *tick* and t_2 for *tock*. These signals are assumed in the model but omitted from the transitions to simplify the diagram.

To vary the frame of reference we will fix a global t that will be used to synchronize the models. Since we decide to observe the clocks from the frame of reference of the first, we set $t := t_1$ in the left hand model. On the other hand the other clock is rapidly disappearing and so we apply the Lorentz transformation and set $t := \frac{t_2}{\beta}$ where β is given by $\beta \stackrel{def}{=} \frac{1}{\sqrt{1-\frac{v^2}{c^2}}}$. This provides an appropriate synchronization for this frame of reference. The two models can now be combined in parallel as usual and it can be seen that the second clock will appear to run slower.

Of course the interesting thing is that if you were to take the frame of reference of the second clock and watch the flashes of light from the earth bound clock the converse effect would occur and the earth bound clock would appear to run slower. Therein lies the essence of the twin's paradox, an excellent treatment of which is given in [Newton-Smith 80].

13.4 Blues Brothers

In the opening scene of *The Blues Brothers*, Elwood (Dan Ackroyd) collects his elder brother Jake (John Belushi) from jail in a second-hand ex-police car. In order to prove to Jake the performance of the vehicle, he later calls his "lady of blessed acceleration", he performs the following stunt (see Figure 13.3). They are waiting in a line of traffic for a bridge to open. Pulling out of the queue Elwood hits the throttle and sends the car powering towards the rising bridge. He continues to accelerate up the slope until the car reaches the end and hurtles through the air in a graceful arc towards the other side. Will the car make it across the gap?

This example is used to illustrate a number of aspects of this approach to the modelling of mode-switching systems.

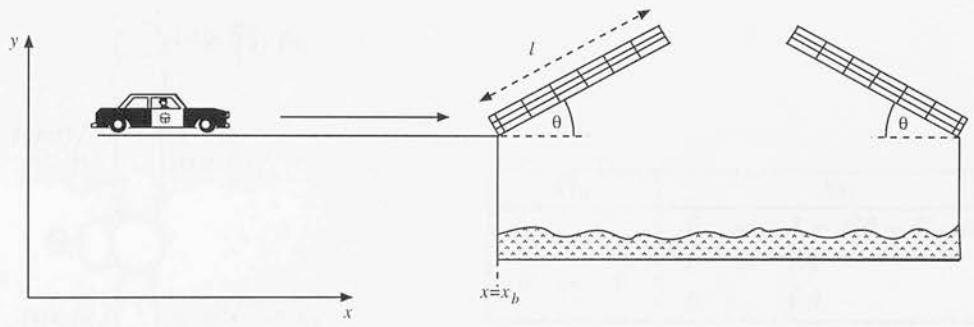


Figure 13.3: A car jump

- It illustrates the use of modularity in modelling mode switching systems.
- It provides a discretization over a geometric space.
- It illustrates the use of simulation for verification purposes.

I should emphases that the system model used is very simplistic, and the example is only intended to illustrate the use of the formalism.

13.4.1 The bridge

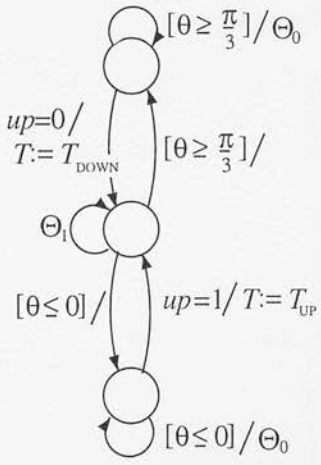
Let us begin by modelling the bridge. It consists of two symmetric spans driven by motors that exert a constant torque T. In addition each span is under the influence of gravity which pulls downwards with magnitude mg through its centre of gravity, at $l/2$ units along its length. This produces a torque of magnitude $\frac{mgl}{2}\cos\theta$. Finally opposing the direction of travel is a force composed of friction and back emf $\mu = k\dot{\theta}$. Equating with rotational inertia gives us the following:

$$\frac{ml^2}{3}\ddot{\theta} = T - \frac{mgl}{2}\cos\theta - k\dot{\theta}$$

which can be reparameterized to give:

$$\ddot{\theta} = A - B\dot{\theta} - C\cos\theta$$

The bridge can travel between $\theta = 0$ to $\theta = \frac{\pi}{3}$. It is triggered to raise by setting the value of up to 1.



Θ_0	Θ_1
$\ddot{\theta} := 0$ $\dot{\theta} := 0$	$\ddot{\theta} := A - B\dot{\theta} - C\cos\theta$ $\dot{\theta} := \int \ddot{\theta}$ $\theta := \int \dot{\theta}$

Figure 13.4: The model of the bridge

The model can be seen in Figure 13.4 in which the bottom state represents the lowered bridge, and the top state the bridge raised to its maximum. For simplicity the velocity and acceleration of the bridge in these states is assumed to be zero. The state in between shows the bridge moving under the applied force T_{UP} or T_{DOWN} depending on its direction.

13.4.2 The car

The car is modelled by a single point at a position (β, γ). In order to maintain independence of the car from the rest of the model the geometry component of the model switches the coordinate system of the car at a couple of points in its journey. To enable these switches to take place the car also has two input signals (β', γ') which can be used to set (β, γ) on state transitions between dynamics. The car also takes as input θ, the angle of the bridge.

The car has a motor which drives its mass M with force F against air resistance r so that on the flat road it has an basic equation:

$$\ddot{\beta} = \frac{F - r\dot{\beta}}{M} \tag{13.1}$$

On the bridge the coordinates system will switch so that the horizontal axis will lie

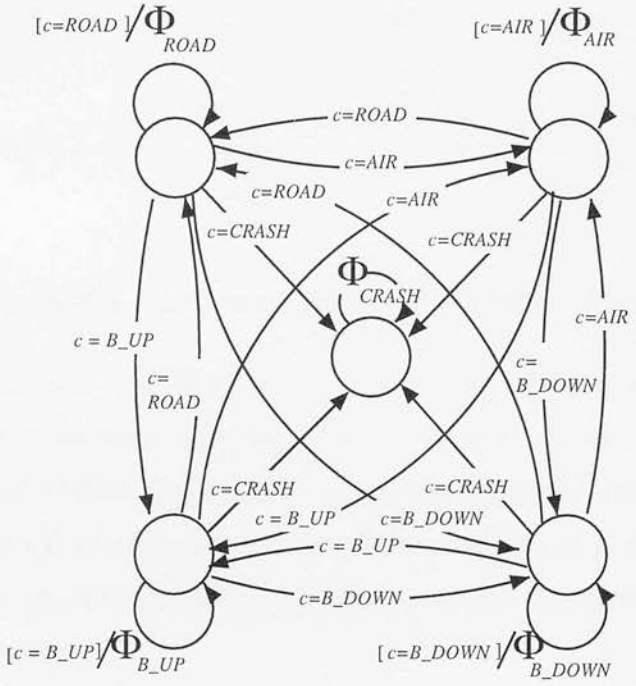


Figure 13.5: The car model

along the bridge so this equation becomes on the upward slope:

$$\ddot{\beta} = \frac{F - r\dot{\beta}}{M} - g\sin(\theta) \tag{13.2}$$

and on the downward slope:

$$\ddot{\beta} = \frac{F - r\dot{\beta}}{M} + g\sin(\theta). \tag{13.3}$$

These simplified dynamics do not take into account the force applied by the bridge to the car, or the force applied by the car to the bridge. It is assumed that the mass of the car is sufficiently small that it makes no difference to the raising of the bridge, and it is also assumed that the speed of the raising of the bridge is sufficiently slow that it has little effect on the forward motion of the car.

Finally the car in flight is assumed to be in free fall:

$$\begin{aligned} \ddot{\beta} &= -\frac{r\dot{\beta}}{M} \\ \ddot{\gamma} &= -g + \frac{r\dot{\gamma}}{M} \end{aligned} \tag{13.4}$$

The car model is shown in Figure 13.5. It takes as input information from the environment to cause it to switch between the different dynamical models. The symbols

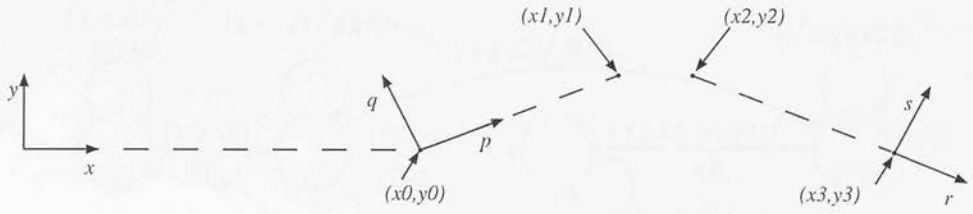


Figure 13.6: The critical points and coordinate systems

$\Phi_{ROAD}, \Phi_{B\_UP}, \Phi_{B\_DOWN}$, and Φ_{AIR} represent equations systems derived from equations (13.1),(13.2), (13.3) and (13.4) respectively. The state the car is in is determined by the input signal c which can be set to one of five constant values *ROAD, B_UP, B_DOWN, AIR* or *CRASH*. The car model is fully connected, so it can move from any state to any other based on changes to the input c, except for the *CRASH* state from which it cannot leave.

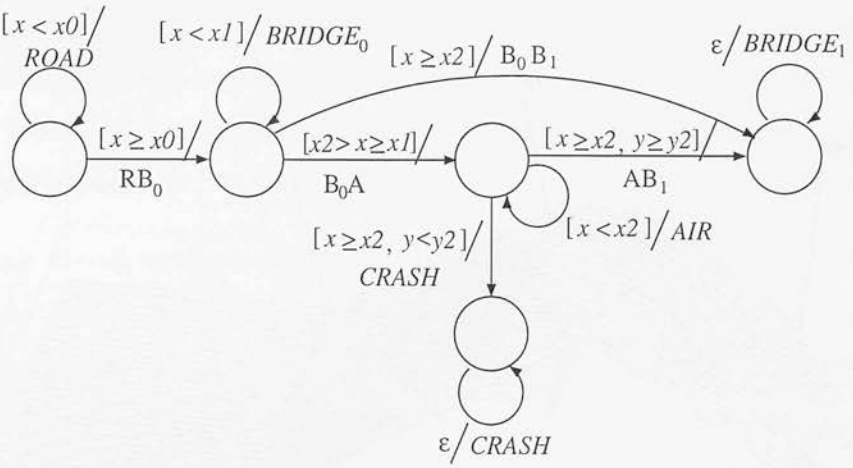
13.4.3 The geometry

There are a number of parameters in this model, which describe critical points in the scene (see Figure 13.6):

$(x0, y0)$	Foot of the first span (constant)
$(x1, y1)$	End of the first span (dependent on θ)
$(x2, y2)$	End of the second span (dependent on θ)
$(x3, y3)$	Foot of the second span (constant)

Figure 13.6 also shows a number of coordinate systems that the geometry uses to keep the car ignorant of the geometry of the world, and hence to keep the components modular. The coordinates (x, y) represent the global coordinates system. Then (p, q) is a coordinates systems with the origin at the foot of the first span of the bridge, which rotates with the bridge, and (r, s) is coordinates system with origin at the foot of the second span which rotates with that part of the bridge. Let us define projections Π_{pq} and Π_{rs} which map from (p, q) to (x, y) and from (r, s) to (x, y) respectively.

The geometry of the scene is modelled in Figure 13.7. The world is modelled in five states. One for the road, one for the first span of the bridge, one for the second span of bridge, one in between where the car is in flight, and a fifth below this to represent the crash state. Each looping transition makes the appropriate conversion from the car's



ROAD:	c:=ROAD	$\binom{x}{y} = \binom{\beta}{\gamma}$	RB_0:	c:=B_UP	$\binom{\beta'}{\gamma'} = \binom{0}{0}$
BRIDGE_0:	c:=B_UP	$\binom{x}{y} = \Pi_{pq}\binom{\beta}{\gamma}$	B_0A :	c:=AIR	$\binom{\beta'}{\gamma'} = \binom{x}{y}$
BRIDGE_1:	c:=B_DOWN	$\binom{x}{y} = \Pi_{rs}\binom{\beta}{\gamma}$	B_0B_1:	c:=B_DOWN	$\binom{\beta'}{\gamma'} = \Pi_{rs}^{-}1\binom{x}{y}$
AIR:	c:=AIR	$\binom{x}{y} = \binom{\beta}{\gamma}$	AB_1:	c:=B_DOWN	$\binom{\beta'}{\gamma'} = \Pi_{rs}^{-}1\binom{x}{y}$
CRASH:	c:=CRASH				

Figure 13.7: The geometry of the world

coordinate system (β, γ) to the world coordinates (x, y). Each state transition signals the car to switch to the appropriate dynamics, by changing c, and reinitializes the car's coordinates scheme appropriately. Notice that the geometry of the scene could be significantly changed without having to change the model of the car.

13.4.4 Results

This system was investigated by writing a delta machine for this model, and executing it as a simulation. The models were first written in ESTEREL which performed the synchronous product, then the resulting finite state machine simulation was hand coded in Mat-lab, to produce the simulation figures. In principle by writing floating point extensions for ESTEREL (a straight forward task) the model could have been simulated entirely in ESTEREL .

The results are not very meaningful in absolute terms because I have very little idea as to the actual values for any number of the physical constants used. (What is the air resistance of a 1969 Dodge sedan? What is the mass of a road bridge?) However I fixed the values of the car so that it had a top speed of around 200kmh^{-1} which it

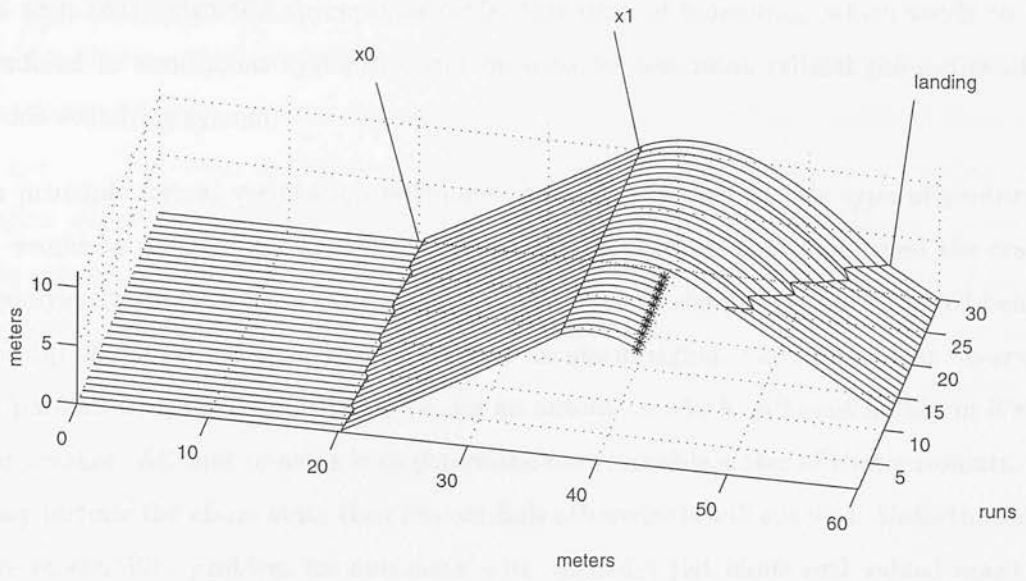


Figure 13.8: A trace of the car's (x, y) positions over 30 runs, crashes are shown with a $*$.

could reach in around 20 seconds, and the bridge took around 340 seconds to rise from horizontal to fully raised ($\frac{\Pi}{3}$), which it did with a short initial acceleration.

I combined the models together, allowing the bridge to begin raising for 50 seconds before advancing the car. The car had a 20 metre stretch of road before it hit the first bridge span, and each bridge span was 20m long.

The car was then run over the bridge around thirty times, increasing the torque delivered by the engine at each stage. The results are plotted on Figure 13.8, which shows the trace of the car's (x, y) position over the 30 runs. The runs that ended in the car crashing are marked with a $*$. Lines have been added to the graphs to show where the car met the foot of the bridge, where it left the first span into the air, and for those successful runs where it landed on the bridge at the other side[3]. The jagged appearance of the lines is due to the time step of the simulation.

[3] Any readers who have not seen the film will be happy to know that Jake and Elwood's car did indeed have the required torque to take it safely to the other side of the bridge in the words of Jake "The car's got pickup".

The example illustrates the simulation of a reasonably complex scenario, and it can be seen that, with the appropriate tools, this type of modelling, which tends to be confined to continuous systems, could be used to determine critical properties of a mode switching system.

In principle formal verification techniques could be applied to this type of problem. It would be possible, for example, to construct an observer that monitored the crash condition, such that if the car had left the first span of the bridge, but dropped below the tip of the second span then it emitted an alarm signal. Combining this observer in parallel with the model then provides an automata which will emit an alarm if the car crashes. All that remains is to determine the reachable states of that automata. If they include the alarm state then the car fails otherwise it will succeed. Unfortunately the reachability problem for automata with variables (let alone real valued ones) is hard and still a big research question. Models such as this are way to complicated for the tools currently available.

On the other hand there may be potential in combining formal techniques with simulation methods, for which Continuous I/O machines and their Delta-flow I/O machine partners would be ideally suited. The problem with simulation as a verification technique is covering the operating space of the model. On the one hand the simulation skips over points and so may skip over a property violation. On the other simulation time is expensive and an exhaustive search of real valued parameter spaces could be totally impractical.

It might be possible to solve the first problem if certain properties of the model can be proved. For example if it can be proved that the system parameters cannot change faster than some bound G. Then time steps can be chosen to ensure that important property violations will not be overlooked.

To address the second problem it might be possible to use observer-like techniques to limit the areas of simulation to those in which property violations could occur.

13.5 Summary

In this chapter we have looked at a number of examples of mode switching models constructed from continuous automata. Other examples have been published including a gate in [Westhead & Hallam 96b] and the landing gear system of a Swedish J39 fighter defense aircraft [Westhead & Nadjm-Tehrani 96]. This last example focuses on the use of observers to carry out formal verification of the system, rather than on its mode switching nature. The version of Continuous I/O machines that was used differed somewhat from that used here, but the same techniques can be applied to these models.

14

Related work

A Mathematician is a machine for turning coffee into theorems.

— *Paul Erdös*

This chapter is intended to relate continuous I/O machines to other work in the field of hybrid systems. Two other modelling systems have been chosen for this comparison Branicky, Borkar and Mitter's unified framework of hybrid control [Branicky *et al.* 94] which comes from the control engineering and Hybrid Automata [Alur *et al.* 95] from computer science. Both alternative models are presented and in each case a result is given relating the expressiveness of the formalisms.

Because of the different ways in which the outputs are expressed in each case rigorous proofs would be very tedious. In both cases constructive proofs are sketched (in appendix G), and although they only relate the machines in one direction they are intended to illustrate that the formalisms are comparably expressive.

The final section in this chapter focuses on the Continuous I/O machine and emphasizes how it differs from these existing representations.

14.1 Branicky, Borkar and Mitter's model

For brevity let us refer to this model as the BBM model. It was built to provide a descriptive formalism that encompassed a number of different formalisms from the

engineering side of Hybrid systems, these include Tavernini's model [Tavernini 87], the Back-Guckenheimer-Myers model, the Nerode-Kohn [Nerode & Kohn 93] model and Brockett's [Brockett 83] model.

Branicky, Borkar and Mitter's focus on hybrid model's of the form:

$$\dot{x}(t) = \xi(t), t \geq 0 \tag{14.1}$$

where $x(t)$ is the *continuous component* of the state taking values in some subset of Euclidean space. $\xi(t)$ is a *controller vector field* that generally depends on $x(t)$, the *continuous component* $u(t)$ of the control policy and the aforementioned discrete phenomenon.

They distinguish four types of discrete change:

Autonomous Switch — This is a discontinuity in $\xi(\cdot)$ which is brought about by $x(\cdot)$ hitting some boundary. An example might be a control system with hysteresis.

Autonomous Jump — This is a discontinuity in $x(\cdot)$ brought about by hitting a boundary. Collisions are a good example, such as ball hitting a wall which will result in a discontinuous change in the balls velocity.

Controlled Switch — This is a discontinuity in $\xi(\cdot)$ in response to a control command. A good example of this would be the discontinuities brought about in the manual transmission of a car [Brockett 83].

Controlled Jump — This is a discontinuity in $x(\cdot)$ in response to a control command. The inventory management system given in [Bensoussan & Lions 84] provides examples of such control.

Instead of the explicit discrete states found in automata models this approach carves up the world states into discrete subsets. For $x(\cdot)$ the state space is $S \subseteq \mathbb{R}^d$, which is split into subspaces S_i such that $S = \bigcup_i S_i$ and $S_i \subseteq \mathbb{R}^{d_i}$, $d_i \leq d \in \mathbb{N}$. Then each S_i is further divided into $A_i, C_i, D_i \subset S_i$. These are the *autonomous jump* sets, *controlled jump* sets and *destination* sets respectively. The sets A, C, D will refer to $\bigcup_i A$, $\bigcup_i C$, $\bigcup_i D$. The sets U, V will be used to represent the sets of continuous and discrete control respectively. Then the following mappings are assumed:

- *vector fields* $f_i : S_i \times S_i \times U \to \mathbb{R}^{d_i}, i \in \mathbb{N}$.
- *transition map* $G : A \times V \to D$.
- *transition delay* $\Delta_1 : A \times V \to \mathbb{R}^+$.
- *impulse delay* $\Delta_2 : \bigcup(C_i \times D_i) \to \mathbb{R}^+$.

The dynamics of the system consists of a sequence of jumps. There is a sequence of *prejump times* $\{\tau_i\}$ and a sequence of *post jump times* $\{\Gamma_i\}$ which satisfy $0 = \Gamma_0 \leq \tau_1 < \Gamma_1 \leq \tau_2 < \Gamma_2 < ... \leq \infty$. Over each interval $[\Gamma_{j-1}, \tau_{j-1})$ with nonempty interior, $x(\cdot)$ evolves according to equation 14.1 in some S_i. At the next jump time τ_j it jumps to some $D_k \in S_k$ in one of two ways:

- $x(\tau_j) \in A_i$ in which case it must jump to $x(\Gamma_j) = G(x(\tau_j), v_j) \in D$ at time $\Gamma_j = \tau_j + \Delta_1(x(\tau_j), v_j)$, $v_j \in V$ being a control input. This is an autonomous jump.
- $x(\tau_j) \in C_i$ and the controller chooses to move the trajectory discontinuously to $x(\Gamma_j) \in D$ at time $\Gamma_j = \tau_j + \Delta_2(X(\tau_j), x(\tau_j))$. This is an impulsive jump.

During a jump, *i.e.* in the period $[\tau_j, \Gamma_j)$ the system is frozen. For $t \in [0, \infty)$, let $[t] = \max_j\{\Gamma_j | \Gamma_j \leq t\}$. The vector field of equation (14.1) is given by:

$$\xi(t) = f_i(x(t), x[t], u(t)),$$

where i is such that $x(i), x[i] \in S_i$ and $u(\cdot)$ is a U-valued control process. For clarity the short hand $G(x, v; i) = (x'; j)$ will be used to denote the transition from $x \in A_i \subset S_i$ to $x'D_j \subset S_j$.

It is not difficult to see that this model could be seen as describing the flow $(\mathbb{R}^+, \leq, \theta(\mathbf{x}) \stackrel{def}{=} x(t), \mathbb{R}^d)$. This leads us to the following theorem

Theorem 14.1 *For any system H described as a BBM model, there exists a Continuous I/O machine M such that the output flow M is equivalent to the flow described by H.*

The proof involves the construction of a machine to simulate the hybrid system. It is sketched in Appendix G.

An immediate corollary of this result is that Continuous I/O machines can also express any model described by either Tavernini's model [Tavernini 87], the Back-Guckenheimer-Myers model, the Nerode-Kohn model or Brockett's model.

14.2 Relationship to hybrid automata

The most prominent automata based modelling approach to hybrid systems from computer science is Hybrid Automata. The hybrid automaton [Alur *et al.* 95] is probably the most established alternative formalism for representing hybrid systems. This section will briefly introduce the formalism, and presents a result that a Continuous I/O machine is at least as expressive as a hybrid automaton (a sketch of the proof is given in Appendix G. The converse result should also hold because as the proof illustrates the machines are quite conceptually close, but a proof of this is not offered.

In order to establish how continuous automata fit into the hybrid systems literature, a proof is offered that demonstrates that for any time deterministic hybrid automaton H there is a Continuous I/O machine which describes a flow describing all the points in any run of the H. The converse result is that for any Continuous I/O machine there is a hybrid automata H that will produce only points described in the flow of M.

A **hybrid automaton** $H = (Loc,\ Var,\ Lab,\ Edg,\ Act,\ Inv)$ is a six tuple:

- *Loc* a finite set of vertices called *locations*
- *Var* a finite set of real valued variables. A valuation ν is a function that assigns a real value $\nu(x) \in \mathbb{R}$ to each variable $x \in Var$. The set of valuations is V. A state $\sigma \in \Sigma$ is a pair (l, ν)
- *Lab* a set of synchronization labels including $\tau \in Lab$
- *Edg* a finite set of edges, called *transitions*. Each edge $e = (l, a, \mu, l')$, goes from l to l' with synchronization label a, and a *transition relation* $\mu \subseteq V^2$. Each location must have a stutter transition $(l, \tau, \{(\nu, \nu)\}, l)$. This is an identity

transition which simplifies the definition of the parallel product and ensures that the transition system is reflexive.

- *Act* a labelling function that assigns to each location, a set of *activities*. An activity is a function from the non-negative reals to V. The activities of each location are *time-invariant*: $\forall l \in Loc, x \in Var, t \in \mathbb{R}^{\geq 0}, (f+t) \in Act(l)$ where $(f+t)(t') = f(t+t') \forall t' \in \mathbb{R}^{\geq 0}$.

  For all locations l, activities f and variables x we write f^x the function $\mathbb{R}^{\geq 0} \mapsto \mathbb{R}$ such that $f^x(t) = f(t)(x)$.

- *Inv* a labelling function that assigns each location l an *invariant* $Inv(l) \subseteq V$.

The state of a hybrid automaton $\sigma = (l, \nu)$ can change in two ways: either by a discrete transition that changes the location and the variables according to the transition relation, or by a *time delay* that changes only the value of the variables according to the activities of the current location.

The automaton stays at a location iff the location invariant is true. Associated with the automaton H is a labelled transition system $\mathcal{T}_H = (\sigma, Lab \cup \mathbb{R}^{\geq 0}, \rightarrow)$, where the *step relation* $\rightarrow$ is the union of the *transition-step relations* $\rightarrow^a$, for $a \in Lab$,

$$\frac{(l, a, \mu, l') \in Edg \quad (\nu, \nu') \in \mu \quad \nu, \nu' \in Inv(l)}{(l, \nu) \longrightarrow^a (l', \nu')}$$

and the *time-step relations* $\rightarrow^t$ for $t \in \mathbb{R}^{\geq 0}$,

$$\frac{f \in Act(l) \quad f(0) = \nu \quad \forall 0 \leq t' \leq t. f(t') \in Inv(l)}{(l, \nu) \longrightarrow^t (l, f(t))}$$

The following theorem relates Continuous I/O machines and Hybrid Automata. A sketch of the proof is given in Appendix G. An earlier proof of this result was offered in [Westhead & Hallam 96a], it was constructed when the ideas were in an early stage and contains constructions that have been since shown to be invalid.

Theorem 14.2 *For any hybrid automaton H in which the activities can be described as trajectories or dynamical systems, there exists an equivalent Continuous I/O Machine M.*

A proof of this result is sketched in Appendix G. It shows how to construct a Continuous I/O machine that has behaviour equivalent to an arbitrary hybrid automaton. Nondeterminism in the Hybrid Automata is modelled using inputs to the Continuous I/O machine. The converse result should also follow.

This equivalence result has a number of immediate corollaries that can be drawn from work that has been done on Hybrid Automata.

- *Any model expressible by timed automata [Alur & Dill 94] can be described by a Continuous I/O machine.* Timed automata are essentially automata with clocks which are subsumed by Hybrid Automata.

- *The reachability problem for Continuous I/O machines is undecidable* (since it is undecidable for Hybrid Automata [Alur *et al.* 95].) The reachability problem is the following. Given a machine M, a state q and a valuation V_{O_M}[1] over the output signals the question is can M ever be in state q and output V_M? This result is very significant because reachability is central to verification. In fact the situation is worse than this, the result actually holds for very much simplified machines.

14.3 Continuous I/O machines

Continuous I/O machines take a radically different approach from either of the preceding models. As we have seen the resulting formalism is similar in expressibility to these approaches however there are a number of important differences. In this section we explore those differences and argue that the Continuous I/O machine provides a stronger model of hybrid systems than either of these approaches.

14.3.1 Compositionality

Compositionality of models is very important. Modular modelling can support much more complex models and more consistent experimentation and comparison between

[1] In general the question can actually be reduced to will M ever reach q because the entry conditions to q could be made to include the valuation.

models. This is particularly true of situated agent representations where we would like to build separate controller plant and environment models and compose them for validation. In these terms important compositional features of models include the following:

- the distinction between input and output to a model component which is essential for analyzing the properties of interactions between an environment, plant and controller [Abadi & Lamport 90].
- a precise understanding of when models can and cannot be composed,
- the ability to compose discrete controllers with complex continuous systems.
- the ability to compose multi-agent systems.

The BBM and hybrid automata models do not satisfy all these properties. The BBM model is based around a system of equations so it is relatively straightforward to compose models, however it is difficult to determine if the composed model has a solution. It does make a distinction between input and output in the sense that controller and plant are described separately. However the intention is to represent a single controller and plant so it is not an ideal formalism to represent multiple agent systems in an environment.

Composition in hybrid automata is more complicated. It is not described here but involves a product operation which composes machines by synchronizing their discrete changes with respect to a special label set. Composition was something of an afterthought in this approach. It does not distinguish between input and output and composition with a discrete controller is not addressed.

The most important distinction between the Continuous I/O machine and the other models is the strength of its compositional semantics. The limiting process allows us to extend the synchronous product to continuous event systems. The causality issues that arise from this operation also exist and the analysis that has already been done in this area can be applied directly, giving a precise understanding of composition failure. There is a clearly defined notion of input and output signals corresponding to that suggested in [Abadi & Lamport 90]. It has already been demonstrated how discrete

controllers can be composed with continuous plant models and it has even been shown how separate model components can have independent local temporal metrics.

14.3.2 World view

One of the consequences of the limiting processes is that the machine's output is a continuum. This is an important difference when compared to Hybrid automata. Their approach of choosing nondeterministic time steps is intended to produce a sequence of points that can be analyzed using existing computer science approaches. However it also leads to the introduction of Zeno time and equivalence complications.

The continuous output produced by continuous automata are amenable to generalizations of sequential analysis techniques. We have already discussed, for example, how synchronous observers can be applied. Furthermore, this new approach can yield a deterministic sequence of points capturing all the important behaviour of the nondeterministic hybrid automata sequence without raising any of the associated issues.

The BBM model like the Continuous I/O machine describes a continuous output. However it is a passive description that has to be fed a continuous time series in order to produce its output. The Continuous I/O machine, on the other hand, is self contained; it can be generative and provide a complete description of system change through time without external input. This generative property would not be possible without the limiting process.

14.3.3 Discrete change

The BBM model is capable of describing complex discrete changes. However the automata approach provides a stronger framework for understanding and analyzing those changes bringing with it, as it does, the opportunity for the application of existing computer science approaches.

This places Hybrid automata and Continuous I/O machines in a stronger position to make sense of the discrete changes in the system. However the flip side to this is that the BBM model could be argued to provide a stronger representation of the continuous dynamics.

14.4 Summary

This chapter described two general formalisms for modelling hybrid systems one from engineering, one from computer science. Results were stated for the equivalence of these representations to Continuous I/O machines, and proofs of these results are outlined in Appendix G. Corollaries of these results lead to Continuous I/O machines being capable of describing models represented in one of six other formalisms, and having an undecidable reachability problem. Overall this demonstrates the generality and expressibility of this approach.

15

Conclusion

The machines that are first invented to perform any particular movement are always the most complex, and succeeding artists generally discover that, with fewer wheels the same effects may be more easily produced.

— *Adam Smith*

The aim of this thesis was to investigate the use of time in the modelling of systems which involved the interaction of an agent with its environment [Agre 95]. We called such systems *situated agent systems* [Rosenshein & Kaelbling 86, Sandewall 94]. Three different approaches were proposed

- *discrete valued*, which includes logics [Rosenshein & Kaelbling 86, Sandewall 94] as well as process algebra approaches such as [Lyons & Hendriks 95, Milner 89], the various synchronous languages [Berry & Gonthier 92, Halbwachs *et al.* 91a, Benveniste *et al.* 93a, Harel 86, Maraninchi 91] and finite state automata [Hartmanis & Stearns 66].

- *continuous valued* involving descriptions of continuous dynamical systems, advocated by [Smithers 92, Smithers 94, vanGelder 94] exemplified by [Beer 95, Steels 87].

- *mode-switching* or *hybrid systems* which describe both continuous change and discrete switching. Formalisms which can be used to model mode switching

systems include

- the *Qualitative Physics* approaches which include [deKleer & Brown 84, deKleer 84, Falkenhainer & Forbus 91, Addanki *et al.* 91],
- the various *logic* based formalisms which have been applied including [Alur & Henzinger 89, Alur *et al.* 91, Bouajjani *et al.* 93, Chaochen *et al.* 92, Abadi & Lamport 91, Lamport 93] ,
- the *automata* models [Nadjm-Tehrani 94, Alur *et al.* 95, Maler *et al.* 92, Nicollin *et al.* 92, Nerode & Kohn 93],
- the various switched extensions to *bond graphs* which include [Gebben 81, Karnopp 83, Karnopp 85, Dauphin-Tanguy *et al.* 89, Rinderle & Subramaniam 91, Ducreux *et al.* 93, Broenick & Wijbrands 93, W. Borutzky & Wijbrands 93, Strömberg 94, Söderman 95],
- and finally *piecewise continuous dynamics* [Tavernini 87, Back *et al.* 93, Antsaklis *et al.* 93, Brockett 83, Branicky *et al.* 94].

The conclusion was that discrete models could be powerfully manipulated but were insufficiently expressive, continuous models could only be manipulated to a limited extent but were highly expressive, and that a mode switching approach might be able to take advantage of the best aspects of both the other alternatives.

The exploration of modelling issues began by focusing on the discrete controller to begin with. It was decided that it would be deterministic, have inputs and outputs and communicate synchronously. Progress through time would be represented as an ordered set of events that we called a *flow*. Discrete time is represented by a flow in which events occur in sequence. Continuous time is represented by a continuum of events.

In Chapter 4 the consequences of adopting a discrete representation of time were explored. It was demonstrated that any sequence of fixed time steps would lead to inaccuracies in the model because threshold values would be overshot. The only way to ensure that a sequence of points hit every threshold was to make the step size non-deterministic. This can lead to problems in defining behavioural equivalence, and Zeno

time problems in which time sequences could occur in which time would never advance beyond a point.

In Chapter 5 continuous modelling of time was investigated. Continuous time models avoid these equivalence and Zeno time concerns, and it was proposed that a continuous automata model could be constructed by letting the time interval in a discrete machine tend to zero.

The theoretical part of the thesis began with Chapter 6 which formalized the concept of flows and defined flow properties such as continuous, regular (sequential), θ-continuous (or value continuous). Chapter 7 then formally defined the discrete valued machines called Delta-flow I/O machines which were used to define regular flows. Chapter 8 defined the limiting process over a sequence of machines which involved taking limits down tree structures called combs which threaded their way through the flows described by the sequence of machines.

The next four chapters were then involved in proving that this definition was sound for increasingly complex classes of machine. Chapter 9 demonstrated that the limit existed under very broad conditions, for sequences of simple incrementing machines called Delta clocks. Chapter 10 built on this result to prove that for machines which described trajectories (functions of time) and dynamical systems that the limit also existed. Chapter 11 extended the definition of the limit to incorporate discontinuities in flow values. Chapter 12 showed that the definition could apply to machines whose output was dependent on inputs, and proved the primary result of the thesis, that the synchronous product over Continuous I/O machines was well defined.

Chapter 13 provided some simple examples of the use of Continuous I/O machines to model mode-switching systems. Finally Chapter 14 formally described two alternative hybrid formalisms Branicky, Borkar and Mitter's unified framework of hybrid control [Branicky *et al.* 94] and Hybrid Automata [Alur *et al.* 95]. Results were presented of their equivalence to Continuous I/O machines, and their limitations and differences discussed.

15.1 Contributions

The major contribution of the thesis is the development of a limiting process that marries discrete and continuous time models. This opens up the possibility of new formalisms, such as Continuous I/O machines, which are fundamentally different from any that I am aware of in computer science or engineering.

Continuous I/O machines offer a number of theoretical advantages over existing hybrid representations:

- They have much stronger compositional semantics than the alternatives, based on the continuous extension the synchronous product. So
  - they are more suitable for modelling complex hybrid systems
  - they can be composed with discrete controllers and
  - observer based verification can be applied.
- They describe their output as a continuum rather than a nondeterministic sequence so they do not suffer from Zeno time problems or equivalence issues.
- For every continuous machine output flow there is a deterministic sequence described by shadow signals which captures all the behaviour of the hybrid automata's nondeterministic sequence in a deterministic way.

It was already stated in the introductory chapters that the resulting formalism has a number of practical advantages over existing models:

- every continuous model is guaranteed to have a discrete approximation of arbitrary accuracy
- furthermore that discrete approximation can be implemented using existing compilers to simulate the system's behaviour,
- both continuous model and discrete approximation can be combined with discrete controller models (or indeed implementations if they have been implemented in a synchronous language) to provide complete system descriptions,

- properties of the composed systems can be verified using the technique of synchronous observers [Halbwachs *et al.* 93]. In the case of simple (discrete) properties, this verification process can be carried out automatically with existing tools [Westhead & Nadjm-Tehrani 96].

It also has a number of theoretical advantages because of the continuous description of continuous change. The limiting approach leads to a more elegant solution to the problem of Zeno time. Zeno time only arises because of the attempt to choose a sequence of points from a continuum. If you describe the whole continuum there is no Zeno time problem. Similarly the issues regarding equivalence between machines which were discussed in Chapter 4 also disappear. A Continuous I/O machine produces a deterministic output and so behavioural equivalence is not difficult to determine. Furthermore the results of Chapter 14 demonstrate that the resulting formalism is at least as expressive as existing models.

More importantly, however, the work breaks new theoretical ground in taking the limit of a sequence of automata. The limiting process is very robust and very general. As we saw in Chapters 9 to 12 sequence of machines of the forms discussed need only have convergent delta bounds for it to be guaranteed that there exists a convergent subsequence that tends to a well defined limit machine.

Overall the work brings closer together the understanding of discrete and continuous processes and opens up the possibilities of extending discrete sequential techniques to the study of continuous change.

15.2 Future work

The work presented in this thesis focused on a theoretical problem. Whilst initial decisions were guided by practical concerns the final modelling system was arrived at in order to demonstrate the theoretical results.

Nevertheless this new approach holds significant potential for the modelling of mode switching systems. Providing a formal link between the precise continuous system and its discrete approximation may hold a number of practical and theoretical advantages.

The last example in Chapter 13 illustrated how the discrete approximation can be executed as a simulation of a system. It may also be the case that formal results can be proved in the discrete case that under certain circumstances may hold over the continuous, or *vice versa.*

There is a great deal of work that could follow on from this thesis. It falls into three areas, verification, computational properties and practical application.

15.2.1 Verification

There are a number of approaches to model verification that could be adopted for Continuous I/O machines. They were originally designed with the idea of verification using synchronous observers [Halbwachs *et al.* 93] in mind. An example of this in use is given in [Westhead & Nadjm-Tehrani 96]. However, in general, automatic verification with observers will not currently work in situations which depend on the values of signals. It may be the case, in a restricted class of model, that mathematical reasoning could be used to extend existing techniques to deal with valued signals.

The verification of safety properties can be seen as the reachability problem. In general the problem asks: is there any input flow such that the machine will reach any one of some set of states with any of a set of valuations over its signals set? By simple manipulations of a machine this problem can be translated to a simpler one: for a special state q is there any input flow such that the machine will reach q?

Intuitively it seems feasible that, for a suitably restricted model, some sort of mathematical analysis might be possible which could reduce simple real-valued predicates to Boolean valued signals that could be dealt with using existing techniques. Undecidablity would mean that any such analysis could not be guaranteed to terminate, but it would provide an exact solution in the cases that it did.

An alternative approach might be to use simulations of delta machines to search for violations. In Chapter 13 it was suggested that under certain assumptions that this could be done without loss of rigour and that observer-like technology could be used to identify those areas of the model where this would be necessary.

It should also be possible to apply the approximate verification technique used on

Hybrid Automata [Alur *et al.* 95] that uses polygon approximation of the reachability space of the systems parameters.

15.2.2 Computational properties

The computational properties of hybrid systems are surprisingly unpleasant. Even for extremely simple systems the reachability problem is undecidable [Alur *et al.* 95]. Continuous I/O machines suffer this too, although the new perspective they provide on the problem may lead to alternative classes of decidable machines.

The limiting process itself may also open up new possibilities to take existing computational results which have only been applied to sequences and consider their application to continua. It is interesting to speculate for example, whether results regarding computability and decidability might have analogies in the continuous case.

15.2.3 Practical application

A final area of further work is the practical application of the modelling technique to the design and analysis of real systems. Using theoretical models is essential if they are to develop into useful mathematical models, as Milner points out the establishment of such formalisms

> *... is a different thing from a physical theory... it does not stand or fall by experiment in the conventional sense. But there is a kind of experimental yardstick with which to measure it. People will use it only if it enlightens their design and analysis of systems; therefore the experiment is to determine the extent to which it is useful, the extent to which the design process and analytic methods are indeed improved by the theory.* [Milner 89]

One approach to practical application of these ideas would be to use the limiting process to extend existing calculi to provide continuous descriptions. In this regard LUSTRE is possibly the strongest contender. The advantage of this approach is that the discrete semantics are already available, known by the community and have tool support.

15.3 Final note

To conclude this thesis I would like to draw your attention to the quotation at the chapter head, which I am afraid to say, I feel applies to this work. When developing new ideas so much effort goes in to finding *any* way to express the ideas and prove the results that there is little left to seek the most elegant.

Despite this I feel that the underlying mathematics does have an intrinsic elegance to it and hope that the ideas here might be taken forward to lose a few unnecessary wheels.

Bibliography

[Abadi & Lamport 90] M. Abadi and L. Lamport. Composing specifications. Technical Report 66, DEC (SRC), Palo Alto, California, October 1990.

[Abadi & Lamport 91] M. Abadi and L. Lamport. An old-fashioned recipe for real-time. In W. P. de Roever J.W. Bakker, C. Huizing and G. Rozenberg, editors, *Proc. REX workshop*. LNCS, Springer Verlag, October 1991.

[Addanki *et al.* 91] S. Addanki, R. Cremonini, and J. S. Scott. Graphs of models. *Artificial Intelligence*, 51:145–177, 1991.

[Agre 95] P. E. Agre. Computational research on interaction and agency. *Artificial Intelligence*, 72:1–52, 1995.

[Alur & Dill 94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[Alur & Henzinger 89] R. Alur and T. Henzinger. A really temporal logic. In *FOCS'89*, 1989.

[Alur *et al.* 91] R. Alur, T. Feder, and T. Henzinger. The benifits of relaxing punctuality. In *PODC'91*. IEEE, 1991.

[Alur *et al.* 95] R. Alur, C. Courcoubetis, and N. Habwachs et. al. The algorithmic analysis of hybrid systems. *Theoretical Computer Science B*, 137, January 1995.

[Antsaklis *et al.* 93] P. J. Antsaklis, J. A. Stiver, and M. D. Lemmon. Hybrid systems modeling and autonomous control systems. In Grossman et al. [Grossman *et al.* 93], pages 255–267.

[Aristotle 68] Aristotle. Time. In Gale [Gale 68], pages 9–23.

[Augustine 68] St. Augustine. Some questions about time. In *The Philosophy of Time* [Gale 68], pages 38–54.

[Back *et al.* 93] A. Back, J. Guckenheimer, and M. Myers. A dynamical simulation facility for hybrid systems. In Grossman et al. [Grossman *et al.* 93], pages 366–392.

[Beer 95] R. D. Beer. A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, pages 173–215, 1995.

[Bensoussan & Lions 84] A. Bensoussan and J. L. Lions. *Impulse Control and Quasi-Variational Inequalities.* Gauthier-Villars, Paris, 1984.

[Benveniste *et al.* 93a] A. Benveniste, P. Caspi, N. Halbwachs, and P. Le Guernic. Data-flow synchronous languages. In *A Decade of Concurrency, reflexions and perspectives, REX School/Symposium*, volume 803 of *Lecture Notes in Computer Science*, pages 1–45. Springer Verlag, June 1993.

[Benveniste *et al.* 93b] A. Benveniste, M. Le Borgne, and P. Le-Guernic. Hybrid systems: The signal approach. *Lecture Notes in Computer Science*, 736:230–254, 1993.

[Bergqvist *et al.* 95] C. Bergqvist, Söderquist, and Wernersson. On combining force/torque sensors and electrical sensing for detecting contact errors during assembly. IROS, 1995.

[Berry & Gonthier 92] G. Berry and G. Gonthier. The ESTEREL synchronous programming language: design, semantics, implementation. *Science of Computer Programming*, 19(2):87–152, November 1992.

[Berry 92] G. Berry. A hardware implementation of pure Esterel. *Sadhana, Academy Proceedings in Engineering Sciences, Indian Academy of Sciences*, 17(1):95–130, 1992. Rapport Centre de Mathématiques Appliquées de l'Ecole des Mines de Paris, numéro 06/91.

[Berry 95] G. Berry. The constructive semantics of pure esterel, 1995.

[Berry *et al.* 91] G. Berry, F. Migard, and J-P. Paris. Programming a digital watch in esterel v3_2. Ecole des Mines de Paris. This document is available by FTP from http://www.inria.fr/meije/esterel, June 1991.

[Bouajjani & Lakhnech 96] A. Bouajjani and Y. Lakhnech. Logics vs. automata: the hybrid case. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III. Verification and Control. Proceedings*, number 1066 in

Lecture Notes in Computer Science, pages 531–542, 1996.

[Bouajjani *et al.* 93] A. Bouajjani, R. Echahed, and J. Sifakis. On model checking for real-time properties with durations. In *LICS'93*. IEEE, 1993.

[Bouajjani *et al.* 95] A. Bouajjani, Y. Lakhnech, and R. Robbana. From duration calculus to linear hybrid automata. *Lecture Notes in computer Science*, 939:196–210, 1995.

[Branicky *et al.* 94] M. Branicky, V. Borkar, and S. Mitter. A unified framework for hybrid control. In *Proceedings 33rd Conference on Decision and Control*, pages 4228–4234, December 1994.

[Broad 68] C. Broad. Ostensible temporality. In *The Philosophy of Time* [Gale 68], pages 38–54.

[Brockett 83] R. W. Brockett. Hybrid models for motion control systems. In H. L. Trentelman and J. C. Willems, editors, *Essays in Control*, pages 29–53. Birkäuser, Boston, 1983.

[Broenick & Wijbrands 93] J. F. Broenick and K. C. J. Wijbrands. Describing discontinuities in bond graphs. In *Proc. First Int. Conf. on Bond Graph Modeling*, volume 25 of *SCS Simulation Series*, pages 120–125. ICBGM'93, 1993.

[Brooks 89] R. A. Brooks. A robot that walks; emergent behaviors from a carefully evolved network. A.I. Memo 1091, Massachusettes Institute of Technology Artificial Intelligence Laboratory, February 1989.

[Caspi & Halbwachs 86] P. Caspi and N. Halbwachs. A functional model for describing ans reasoning about time behaviour of computing systems. *Acta Informatica*, 22:1986, 1986.

[Chaochen *et al.* 92] Z. Chaochen, A.P. Ravn, and M.R. Hansen. An extended duration calculus for hybrid real-time systems. In A.P. Ravn R.L. Grossman, A. Nerode and H. Rischel, editors, *Proc. Workshop on Theory of Hybrid Systems*. Springer Verlag, October 1992.

[Dauphin-Tanguy *et al.* 89] G. Dauphin-Tanguy, C. Sueur, and C. Rombaut. Bond-graph approach of communicating phenomena. In R. Husson, editor, *Proc. of Advanced Info. Processing in Automatic Control*, pages 339–343. IFAC, 1989.

[deKleer & Brown 84] J. de Kleer and J. S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

[deKleer 84] J. de Kleer. How circuits work. *Artificial Intelligence*, 24:205–346, 1984.

[Ducreux *et al.* 93] J. P. D. Ducreux, G. Dauphin-Tanguy, and C. Rambaut. Bond-graph modelling of communicating phenomena in power electronics circuits. In *Proc. First Int. Conf. on Bond Graph Modeling*, volume 25 of *SCS Simulation Series*, pages 132–136. ICBGM'93, 1993.

[Einstein 61] A. Einstein. *Relativity — The Special and General Theory.* Bonanza Books, 1961.

[Falkenhainer & Forbus 91] B. Falkenhainer and K. D. Forbus. Compositional modeling: finding the right model for the job. *Artificial Intelligence*, 51:95–143, 1991.

[Gale 68] M. Gale. *The Philosophy of Time.* McMillan & Co. Ltd., 1968.

[Gebben 81] V. D. Gebben. Bond graph for modeling valves and switches — a digital graph element represents two state devices. Tech Brief PB81-970082/LEW-13177, NASA, Lewis Research Center, Cleveland, Ohio, 1981. One page only.

[Grossman *et al.* 93] R. Grossman, A. Nerode, A. Ravn, and H. Rischel, editors. *Hybrid Systems*, volume 736 of *LNCS*, New York, 1993. Springer.

[Grunbaum 63] A. Grunbaum. *Philosophical Probhlems of Space and Time.* Alfred A. Knopf, 1963.

[Hajnicz 96] E. Hajnicz. *Time Structures.* Number 1047 in Lecture Notes in Artificial Intelligence. Springer, 1996.

[Halbwachs & Maraninchi 95] N. Halbwachs and F. Maraninchi. On the symbolic analysis of combinational loops in circuits and synchronous programs. In *Euromicro'95*, Como (Italy), September 1995.

[Halbwachs 93] N. Halbwachs. *Synchronous programming of reactive systems.* Kluwer Academic Pub., 1993.

[Halbwachs *et al.* 91a] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous dataflow programming language lustre. *Proceedings of the IEEE*, 79(9):1305–1320, September 1991.

[Halbwachs *et al.* 91b] N. Halbwachs, P. Raymond, and C. Ratel. Generating efficient code from data-flow programs. In *Third International Symposium on Programming Language Implementation and Logic Programming*, Passau (Germany), August 1991.

[Halbwachs *et al.* 93] N. Halbwachs, F. Lagnier, and P. Raymond. Synchronous observers and the verification of reactive systems. In M. Nivat, C. Rattray, T. Rus, and G. Scollo, editors, *Third Int. Conf. on Algebraic Methodology and Software Technology, AMAST'93.* Workshops in Computing, Springer Verlag, June 1993.

[Harel & Pnueli 85] D. Harel and A. Pnueli. On the development of reactive systems: Logic and models of concurrent systems. In *Proc. NATO Advanced Study Insititute on Logics and Models for Verification and Specification of Concurrent Systems*, volume 13 of *NATO ASI Series F,*, pages 477–498. Springer-Verlang, 1985.

[Harel 86] D. Harel. Statecharts: A visual approach to complex system. Technical Report CS86-02, Weizman Institute of Science, March 1986.

[Harel 87] D. Harel. On the formal semantics of statecharts. In *Proceedings of the second IEEE Symposium on Login in Computer Science*, pages 54–64, 1987.

[Hartmanis & Stearns 66] J. Hartmanis and R.E. Stearns. *Algebraic Structure Theory of Sequential Machines.* Prentice-Hall, 1966.

[Henzinger & Ho 95] T. A. Henzinger and P. Ho. Algorithmic analysis of nonlinear hybrid systems. In *Computer Aided Verification, 7th International Conference, CAV '95*, number 939 in Lecture Notes in Computer Science, pages 225–238, July 1995.

[Karnopp 83] D. C. Karnopp. Computer models of hysteresis in mechanical and magnetic components. *J. of the Franklin Institute*, 316(5):405–415, 1983.

[Karnopp 85] D. C. Karnopp. General method for including rapidly switched devices in dynamical system simulation models. *Trans. of the Society for Computer Simulation*, 2(1):155–168, 1985.

[Kirsh 95] David Kirsh. Intelligent use of space. *Artificial Intelligence*, 73:31–68, 1995.

[Kuc & Siegel 87] R. Kuc and M. Siegel. A physically based simulation model for acoustic sensor robot navigation. *IEEE Trans. PAMI*, 9(6):766–778, 1987.

[Lamport 93] L. Lamport. Hybrid systems in TLA$^+$. *Lecture Notes in Computer Science*, 736:77–102, 1993.

[Landsberg 82] P. Landsberg. *The Enigma of Time.* Adam Hilger Ltd., 1982.

[Lyons & Arbib 89] D. Lyons and M. Arbib. A formal model of computation for sensory-based robotics. *IEEE Transactions on Robotics and Automation*, 5(3), June 1989.

[Lyons & Hendriks 95] D. Lyons and A. Hendriks. Exploiting patterns of interaction to achieve reactive behaviour. *Artificial Intelligence*, 73:117–148, 1995.

[Malcolm 91] C. Malcolm. Behavioural modules in robotic assembly. Teaching paper, March 1991.

[Malcolm *et al.* 89] C. Malcolm, T. Smithers, and J. Hallam. An emerging paradigm in robot architecture. DAI Research Paper 447, Department of Artificial Intelligence University of Edinburgh, 1989.

[Maler *et al.* 92] O. Maler, Z. Manna, and A. Pnueli. From timed to hybrid systems. In *REX workshop on Real-Time: Theory and Practice*, number 600 in LNCS, 1992.

[Maraninchi 91] F. Maraninchi. Argos : a graphical synchronous language for the description of reactive systems. Spectre report c29, LGI-IMAG, Grenoble, mar 1991.

[McTaggart 93] J. McTaggart. The unreality of time. In *The Philosophy of Time* [Poidevin & MacBeath 93], chapter I, pages 23–34.

[Mellor 93] D. H. Mellor. The unreality of tense. In *The Philosphy of Time* [Poidevin & MacBeath 93], chapter III, pages 47–59.

[Milner 89] A. J. R. G. Milner. *Communications and Concurrency.* Prentice-Hall, 1989.

[Milner 83] A. J. R. G. Milner. Calculi for synchrony and asynchrony. *Journal of Theoretical Computer Science*, 25:267–310, 83.

[Moller & Toffs 89] F. Moller and C. Toffs. A temporal calculus of communicating systems. LFCS Report Series ECS-LFCS-89-104, Dept. of Computer Science, University of Edinburgh, 1989.

[Moller & Toffs 91] F. Moller and C. Toffs. Relating processes with respect to speed. LFCS Report Series ECS-LFCS-91-143, Department of Computer Science, University of Edinburgh, 1991.

[Nadjm-Tehrani 94] S. Nadjm-Tehrani. *Reactive Systems in Physical Environments.* Unpublished PhD thesis, Department of Computer Science and Information Science, Linköping University, Sweeden, 1994.

[Nerode & Kohn 93] A. Nerode and W. Kohn. Models for hybrid systems: automata, topologies, controllability. In Grossman et al. [Grossman *et al.* 93], pages 317–356.

[Newton-Smith 80] W. Newton-Smith. *The Structure of Time.* Routledge & Kegan Paul, 1980.

[Nicollin *et al.* 92] X. Nicollin, J. Sifakis, and S. Yovine. Compiling real-time specifications into extended automata. *IEEE Transactions on Software Engineering*, 18(9):794–804, September 1992.

[Poidevin & MacBeath 93] R. Le Poidevin and M. MacBeath. *The Philosophy of Time.* Oxford University Press, 1993.

[Ramadge & Wonham 87] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25(1), January 1987.

[Ramadge & Wonham 89] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1), January 1989.

[Rinderle & Subramaniam 91] J. R. Rinderle and B. L. Subramaniam. Automated bond graph modelling and simplification to support design. *Automated Modeling*, 34:45–68, 1991.

[Rosenschein & Kaelbling 95] S. J. Rosenschein and L. P. Kaelbling. A situated view of representation and control. *Artificial Intelligence*, 73:149–173, 1995.

[Rosenshein & Kaelbling 86] S. J. Rosenshein and L. P. Kaelbling. The synthesis of digital machines with proveable epstemic properties. In *Proceedings of the conference on Theoretical Aspects of Reasoning About Knowledge*, pages 83–89, San Mateo, CA, 1986. Morgan Kaufmann. updated version: Technical Note 412, Artificial Intelligence Centre, SRI International.

[Russell 81] B. Russell. Mathematics and metaphysicians. In *Mysticism and Logic*, pages 59–74. Totowa, NJ, 1981.

[Sandewall 89] E. Sandewall. Combining logic and differential equations for describing systems. In *Proceedings of the first conference on the principles of knowledge representation*, pages 412–420, 1989.

[Sandewall 94] E. Sandewall. *Features and Fluents. The Representation of Knowledge about Dynamical Systems.*, volume I. Oxford University Press, 1994.

[Seddon 87] K. Seddon. *Time.* Croom Helm, 1987.

[Smithers 92] T. Smithers. Are autonomous agents information processing systems? Draft paper prepared as part of the preceedings of the workshop on Emergence, Situatedness, Subsumption, and Symbol Grounding, held at the Corsendonk Priory, Belgium, May 1992.

[Smithers 94] T. Smithers. What the dynamics of adaptive behaviour and cognition might look like in agent-environment interaction systems. In *On the Role of Dynamics and Representation in Adaptive Behaviour and Cognintion.* DRAB'94, 1994.

[Söderman 95] U. Söderman. *Conceptual Modelling of Switching Physical Systems.* Unpublished PhD thesis, Department of Computer and Information Science, Linköping University, 1995.

[Steels 87] L. Steels. Artificial intelligence and complex dynamics. A.I. Memo 88-2, Brussels VUB AI Lab, 1987. Presented at the IFIP workshop on tools, concepts and kbs. Mnt Fuji, Japan.

[Strömberg 94] J. Strömberg. A mode switching modelling philosophy. Unpublished M.Sc. thesis, Linkóping University, 1994.

[Tavernini 87] L. Tavernini. Differential automata and their discrete simulators. *Nonlinear analysis, Theory, Methods, and Applications*, 11(6):665–683, 1987.

[Toffs 89a] C. Toffs. The autosynchronisation of leptothorax acervorum (fabricius) described in wsccs. LFCS Report Series ECS-LFCS-90-128, Department of Computer Science, University of Edinburgh, 1989.

[Toffs 89b] C. Toffs. Timing concurrent processes. LFCS Report Series ECS-LFCS-89-103, Department of Computer Science, University of Edinburgh, 1989.

[Toffs 91] C. Toffs. Task allocation in monomorphic ant species. LFCS Report Series ECS-LFCS-91-144, Department of Computer Science, University of Edinburgh, 1991.

[vanGelder 94] T. van Gelder. What might cognition be if not computation. In *On the Role of Dynamics and Representation in Adaptive Behaviour and Cognintion.* DRAB'94, 1994.

[W. Borutzky & Wijbrands 93] J. F. Broenick W. Borutzky and K. C. J. Wijbrands. Graphical descriptions of physical system models containing discontinuities. In *Proc. of ESM'93*, pages 203–207, 1993.

[Walker *et al.* 97] V.A. Walker, H. Peremans, and J.C.T. Hallam. One tone, two ears, three dimensions: a robotic investigation of the pinnae movements used by rhinolophids and hipposiderid bats. *J. Acoust. Soc. Am. (submitted)*, 1997.

[Westhead & Hallam 96a] M. D. Westhead and John Hallam. Modelling hybrid systems as the limit of discrete computational processes. In *International Conference on Robotics and Automation.* IEEE, 1996.

[Westhead & Hallam 96b] M. D. Westhead and John Hallam. Pushing finite state machines to the limit. Tech. Report 38, Dept. of AI, University of Edinburgh, 1996.

[Westhead & Nadjm-Tehrani 96] M. D. Westhead and S. Nadjm-Tehrani. Verification of embedded systems using synchronous observers. In *proceedings of the 4th international conference on Formal Techniques in Real-Time and Fault-Tolerant Systems FTRTFT'96*, LNCS 1135, pages 405–419. Springer Verlag, September 1996.

[Westhead 92] M. D. Westhead. Linda and the paradigms of parallelisation. Colloquium on "Generic Parallelisation of Algoriths for Control and Simulation, February 1992.

[Westhead 93] M. D. Westhead. Robust intelligent control through the use of a behaviour based control paradigm. IEE Coloquium: Control and Systems group C8 - Robust Control, November 1993.

[Westhead 95] M. D. Westhead. Synchronous systems for behaviour based robot control. In *AISB-95 Workshop on Mobile Robotics.* Society for the Study of Arificial Intelligence and the Simulation of Behaviour, April 1995.

[White 92] M. White. *The Continuous and the Discrete.* Clarendon Press, 1992.

A

Proofs from chapter 7

Lemma A.1 *The operation '$\ominus$' defined over valuations with the same domains is a valid metric.*

Proof To be a valid metric it must have the following properties:

1. $|V_S \ominus W_S| \geq 0$
2. $|V_S \ominus W_S| = 0 \Rightarrow V_S = W_S$
3. $|V_S \ominus W_S| = |W_S \ominus V_S|$
4. $|V_S \ominus W_S| \leq |W_S \ominus U_S| + |U_S \ominus V_S|$

The first three follow directly from the definition. The last one is proved by the following argument.

Consider any signal $\mathtt{s} \in S$; now we show that

$$|V_S(\mathtt{s}) - W_S(\mathtt{s})| \leq |V_S(\mathtt{s}) - U_S(\mathtt{s})| + |U_S(\mathtt{s}) - W_S(\mathtt{s})|$$

For numeric values this is obvious, because '$-$' is a metric over reals. For non-numeric values either:

- $V_S(\mathtt{s}) = W_S(\mathtt{s})$ in which case $|V_S(\mathtt{s}) - W_S(\mathtt{s})| = 0$ and the inequality must hold because of (1) above or
- $V_S(\mathtt{s}) \neq W_S(\mathtt{s})$ in which case either
  - $V_S(\mathtt{s}) = U_S(\mathtt{s}) \neq W_S(\mathtt{s}) \Rightarrow R.H.S. = 1 \;\; L.H.S. = 1$
  - $V_S(\mathtt{s}) \neq U_S(\mathtt{s}) = W_S(\mathtt{s}) \Rightarrow R.H.S. = 1 \;\; L.H.S. = 1$
  - $V_S(\mathtt{s}) \neq U_S(\mathtt{s}) \neq W_S(\mathtt{s}) \Rightarrow R.H.S. = 1 \;\; L.H.S. = 2$

In all three cases the inequality holds. □

There are two obvious ways in which the synchronous product can be defined these are either to zip the two realization flows together (as we have done here) or to take the realization of the union of the two characteristic relations. However the two definitions are equivalent, because of the following lemma.

Lemma A.2 *For two characteristic relations χ_1, χ_2 the following is true*

$$R(\chi_1 \sqcup \chi_2) \equiv R(\chi_1) \amalg R(\chi_2)$$

Proof First we need to prove a sub result, that:

$$(R_1 \sqcup R_2) \sqcup R_3 = R_1 \sqcup (R_2 \sqcup R_3)$$

This is clear since

$$\begin{array}{rcl}
(x,y) \in (R_1 \sqcup R_2) \sqcup R_3 & \Leftrightarrow & (((x \cup y) \downarrow (d_1 \setminus r_2 \cup d_2 \setminus r_1), y \downarrow (r_1 \cup r_2)) \in (R_1 \sqcup R_2) \text{and} \\
& \Leftrightarrow & (((x \cup y) \downarrow d_1, y \downarrow r_1) \in R_1 \text{ and} \\
& & \quad ((x \cup y) \downarrow d_2, y \downarrow r_2) \in R_2) \text{ and} \\
& & \quad ((x \cup y) \downarrow d_3, y \downarrow r_3) \in R_3 \\
& \Leftrightarrow & (x,y) \in R_1 \sqcup (R_2 \sqcup R_3)
\end{array}$$

The main result can then be proved as follows. Let

- $(F, \preceq, \theta) \stackrel{def}{=} R(\chi_1 \sqcup \chi_2)$
- $(F', \preceq', \theta') \stackrel{def}{=} R(\chi_1) \amalg R(\chi_2)$
- $(F_1, \preceq_1, \theta_1) \stackrel{def}{=} R(\chi_1)$
- $(F_2, \preceq_2, \theta_2) \stackrel{def}{=} R(\chi_2)$.

Since the flows are forward and regular we can assume that $(F, \preceq) = (F', \preceq')$, and all that needs to be proved is that $\theta = \theta'$. This is done by induction. First the base case:

$$\begin{array}{rcl}
\theta'(\mathbf{x}_0) & = & \theta_1(\mathbf{x}_0) \sqcup \theta_2(\mathbf{x}_0) \\
& = & \{(\{\}, V_{0_1})\} \sqcup \{(\{\}, V_{0_1})\} \\
& = & \{(\{\}, V_{0_1} \cup V_{0_2})\} \\
& = & \theta(\mathbf{x_0})
\end{array}$$

Now the inductive step, from the definition of a realization we get:

$$\theta(\mathbf{x}) \quad = \quad \{(x, y \downarrow (r_1 \cup r_2)) | (x,y) \in \chi \cup \theta(\overleftarrow{\mathbf{x}})\}$$

Where r_1 and r_2 are the ranges of χ_1 and χ_2 repectively. By induction we can substitute for $\theta(\overleftarrow{\mathbf{x}})$ to give:

$$\begin{array}{rcl}
\theta(\mathbf{x}) & = & \{(x, y \downarrow (r_1 \cup r_2)) | (x,y) \in (\chi_1 \sqcup \chi_2) \sqcup \theta'(\overleftarrow{\mathbf{x}})\} \\
& = & \{(x, y \downarrow (r_1 \cup r_2)) | (x,y) \in (\chi_1 \sqcup \chi_2) \sqcup (\theta_1(\overleftarrow{\mathbf{x}}) \sqcup \theta_2(\overleftarrow{\mathbf{x}}))\} \\
& = & \{(x, y \downarrow (r_1 \cup r_2)) | (x,y) \in (\chi_1 \sqcup \theta_1(\overleftarrow{\mathbf{x}})) \sqcup (\chi_2) \sqcup \theta_2(\overleftarrow{\mathbf{x}}))\} \\
& = & \{(x, y \downarrow r_1) | (x,y) \in (\chi_1 \sqcup \theta_1(\overleftarrow{\mathbf{x}}))\} \sqcup \\
& & \{(x, y \downarrow r_2) | (x,y) \in (\chi_2) \sqcup \theta_2(\overleftarrow{\mathbf{x}}))\} \\
& = & \theta_1(\mathbf{x}) \sqcup \theta_2(\mathbf{x}) \\
& = & \theta'(\mathbf{x})
\end{array}$$

□

B

Proofs from chapter 8

Theorem 8.1 *In a convergent comb c_n over a sequence of flows X_n, if in any branching sequence $\{b_n\}$, there is an infinite subsequence $\{b_i\}$ such that for all i $b_i \geq 2$ then the limit flow $X_{\lim}$ is uncountable.*

Proof This is a proof by contradiction, using the standard technique of diagonalization. We will suppose that the points in the flow can be counted, and then show that regardless of the way they are counted, there is at least one point that has been left uncounted.

First notice that each of the points in a limit flow is uniquely identified by its complete descent. This is clear from the definition of ordering of the limit points, since two points are only equal in the order if they share a complete descent.

So for a comb c_n over flows X_n consider all descents σ and their corresponding branching sequences $\{b_n\}$, each of which has a subsequence with each point $b_i \geq 2$. Since the other branching points add no extra children, we will ignore them, and consider only the $\{b_i\}$s. Let us suppose that $\forall i$, $b_i = 2$, since if it is larger it can only add more points. With $b_i = 2$ the set of descents is the set of infinite strings over $\{0, 1\}$.

Let us assume that all the points are present and suppose that these points can be counted, that means that we can assign a number to index each descent σ call them $\sigma_1, \sigma_2, \cdots$. These descents can then be written out one after the other:

$$\begin{array}{rcl} \sigma_1 & = & a_{11_0}a_{11_1} \quad a_{12_0}a_{12_1} \quad \cdots \\ \sigma_2 & = & a_{21_0}a_{21_0} \quad a_{22_0}a_{22_1} \quad \cdots \\ \sigma_3 & = & a_{31_0}a_{31_0} \quad a_{32_0}a_{32_1} \quad \cdots \\ \vdots & & \end{array}$$

where each $a_{ij_k} \in \{0, 1\}$. Now choose the descent $\sigma' = a'_{1_0}, a'_{1_1}, a'_{2_0} \cdots$ such that:

$$a'_{n_0}a'_{n_1} \neq a_{nn_0}a_{nn_1} \tag{B.1}$$

Notice also that at each point there will be three alternatives which satisfy requirement (B.1).

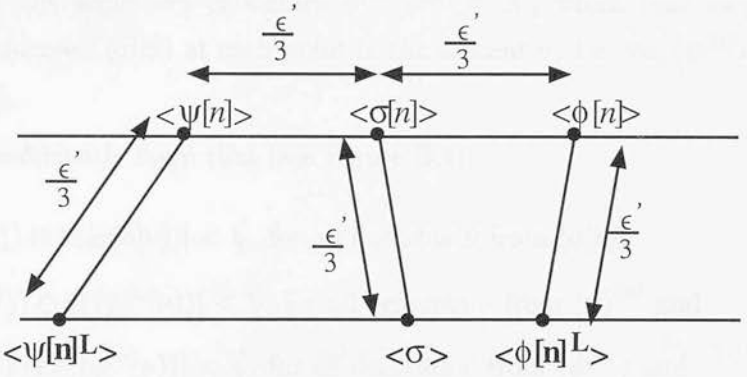


Figure B.1: Illustrating the proof of continuity.

Now this descent σ' defines a descent of the comb. However it may be that σ' is a rightmost descent and so not included in the limit flow. However it has been noted that there were three alternative at choosing every pair of characters in σ' so if it is a rightmost descent, it can be replaced by another string that also satisfies (B.1).

So therefore $\langle \boldsymbol{\sigma'} \rangle$ is a point in the flow $X_{\lim}$, however the element $\langle \sigma' \rangle \notin \{\langle \sigma_1 \rangle, \langle \sigma_2 \rangle, \cdots\}$ because its index differs from every one of those in at least one place so it is an uncounted point, and we have a contradiction.

□

Theorem 8.4 *The limit $X_{\lim}$, if it exists, of a sequence of valuation flows X_n with respect to a comb c_n is θ-continuous at a point $\langle \boldsymbol{\sigma} \rangle \in X_{\lim}$ if:*

- *c_i converges uniformly*
- *for any $\epsilon > 0$ there exists N such that*

$$\left|\theta(\langle \sigma[n] \rangle) \ominus \theta(\langle \overleftarrow{\sigma[n]} \rangle)\right| < \epsilon \text{ and } \left|\theta(\langle \sigma[n] \rangle) \ominus \theta(\langle \overrightarrow{\sigma[n]} \rangle)\right| < \epsilon$$

  for all $n > N$.

Proof We are required to show that for all values of $\epsilon' > 0$, there exist $\mathbf{k}, \mathbf{k'} \in X_{\lim}$ with $\mathbf{k} \prec \langle \boldsymbol{\sigma} \rangle \prec \mathbf{k'}$ such that

$$|\theta_{\lim}(\mathbf{x}) \ominus \theta_{\lim}(\langle \boldsymbol{\sigma} \rangle)| \, \forall \mathbf{x} \in X_{\lim}, \mathbf{k} \prec \mathbf{x} \prec \mathbf{k}.$$

By completeness of c_i the point $\langle \boldsymbol{\sigma} \rangle$ must be the limit of descent from a point in X_0.

Notice that due to monotonicity and completeness of flows, $\forall n$ all the points in the descent to $start(X_{\lim})$ are $start(X_n)$, and all the points in the descent to $end(X_{\lim})$ are $end(X_n)$. So by exclusivity since $\mathbf{a}$ is neither top nor bottom of $X_{\lim}$ all the points $\langle \sigma[n] \rangle$ are neither bottoms nor tops and hence have both a successor and a predecessor.

Now consider the sequences of strings $\psi^{(n)}, \phi^{(n)}, \in X_n$ which lead to points which precede, and succeed $\langle\sigma[n]\rangle$ at each point in the descent σ. i.e. $\forall n, \langle\psi^{(n)}\rangle \approx \overleftarrow{\langle\sigma[n]\rangle}$ and $\langle\phi^{(n)}\rangle \approx \overrightarrow{\langle\sigma[n]\rangle}$.

Now find N sufficiently large that (see Figure B.1):

- $|\theta(\langle\sigma[n]\rangle) \ominus \theta(\langle\sigma[n]\nu\rangle)| < \frac{\epsilon'}{3}$, for all descents ν from $\langle\sigma[n]\rangle$
- $\left|\theta(\langle\psi^{(n)}\rangle) \ominus \theta(\langle\psi^{(n)}\nu\rangle)\right| < \frac{\epsilon'}{3}$, for all descents ν from $\langle\psi\rangle^{(n)}$ and
- $\left|\theta(\langle\phi^{(n)}\rangle \ominus \theta(\langle\phi^{(n)}\nu\rangle)\right| < \frac{\epsilon'}{3}$, for all descents ν from $\langle\phi^{(n)}\rangle$ and
- $\left|\theta(\langle\psi^{(n)}\rangle) \ominus \theta(\langle\sigma[n]\rangle)\right| < \frac{\epsilon'}{3}$,
- $\left|\theta(\langle\phi^{(n)}\rangle) \ominus \theta(\langle\sigma[n]\rangle)\right| < \frac{\epsilon'}{3}$,

for $n > N$. By uniform convergence of X_n and for the last two bullet points, by assumption such an N exists.

Then take $\mathbf{k} = \langle\boldsymbol{\psi}^{(n)L}\rangle$ and $\mathbf{k}' = \langle\boldsymbol{\phi}^{(n)L}\rangle$. Clearly this satisfies $\mathbf{k} \prec \langle\boldsymbol{\sigma}\rangle \prec \mathbf{k}'$. Now observe that by monotonicity of the limit (lemma 8.2), any point $\mathbf{x} \in X_{\lim}$ satisfying $\mathbf{k} \prec \mathbf{x} \prec \langle\boldsymbol{\sigma}\rangle$ must be a descendant of either $\langle\sigma[n]\rangle$ or $\langle\psi^{(n)}\rangle$. In the former case its difference from $\langle\boldsymbol{\sigma}\rangle$ is bounded by $\frac{\epsilon'}{3}$ and in the latter case $\frac{\epsilon'}{3} + \frac{\epsilon'}{3} + \frac{\epsilon'}{3} \leq \epsilon'$ (by the triangle inequality.)

Similarly any point $\mathbf{x} \in X_{\lim}$ satisfying $\mathbf{k}' \succ \mathbf{x} \succ \langle\boldsymbol{\sigma}\rangle$ must be a either a descendant of $\langle\sigma[n]\rangle$ or the point $\langle\boldsymbol{\phi}^{(n)L}\rangle$ itself. In the former case its difference from $\langle\sigma\rangle$ is bounded by $\frac{\epsilon'}{3}$ and in the latter case $\frac{\epsilon'}{3} + \frac{\epsilon'}{3} + \frac{\epsilon'}{3} \leq \epsilon'$ (by triangle ineq.). $\square$

C

Proofs from chapter 9

Theorem 9.1 *For all sequences of delta clocks C_k which have convergent delta bounds there exists a subsequence of clocks C_n which have a well defined limit which is the continuous clock $C_{\lim}$.*

Proof The proof is required to show that over the sequence of clock flows T_k a subsequence T_n with a corresponding comb converges uniformly with even spread.

This involves the following steps:

1. specifying a subsequence and suitable comb
2. proving that the comb is well defined
3. proving that the comb will be uniformly convergent
4. proving that the comb will have even spread.

As always we take $\langle\sigma\rangle_{\mathtt{t}} \stackrel{def}{=} \theta(\langle\sigma\rangle)(\mathtt{t})$.

Specifying the comb First take a subsequence T_n from the sequence of delta clocks T_k such that for each $\max(\Delta)_{n+1} = \frac{1}{2}\min(\Delta)_n$ By convergence of the delta bounds such a subsequence must exist.

Now define a comb c_n over T_n as follows (see Figure C.1.) For each $\mathbf{x} \in T_n$ define a subflow $S_{\mathbf{x}} \sqsubseteq T_{n+1}$, such that $\forall \mathbf{u} \in S_{\mathbf{x}}, \theta_n(\mathbf{x})(\mathtt{t}) - \frac{1}{2}\theta_n(\mathbf{x})(\Delta) < \theta_n(\mathbf{u})(\mathtt{t}) \leq \theta_n(\mathbf{x})(\mathtt{t}) + \frac{1}{2}\theta_n(\vec{\mathbf{x}})(\Delta)$. Take these in order as the children of $\mathbf{x}$, $c(\mathbf{x}, 0) \stackrel{def}{=} start(S), \ldots, c(\mathbf{x}, |S|) \stackrel{def}{=} end(S)$.

Proving it is well defined To show that the comb is well defined, we need to demonstrate four things:

1. For each $\mathbf{x} \in T_n$, there is at least one child

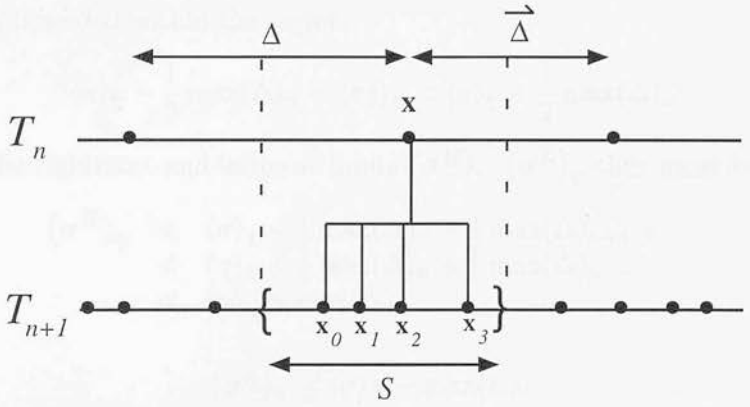


Figure C.1: Illustrating the existence of a general comb over convergent delta bounds where $\mathbf{x}_i \stackrel{def}{=} c(\mathbf{x}, i)$

2. **monotonicity** $c_i(\mathbf{x}, k) \prec c_i(\mathbf{x}', k') \Leftrightarrow \mathbf{x} \prec \mathbf{x}'$ or $(\mathbf{x} \approx \mathbf{x}'$ and $k < k')$ for $\mathbf{x}, \mathbf{x}' \in X, k, k' \in \mathbb{N}$
3. **exclusivity** $c(\mathbf{x}, k) = c(\mathbf{x}', k') \Leftrightarrow \mathbf{x} \approx \mathbf{x}', k = k'$
4. **completeness** $\mathbf{x} \in X_i, i > 0 \Rightarrow \exists \mathbf{x}' \in X_{i-1}$ s.t. $\mathbf{x} = c(\mathbf{x}', k)$ for some k.

Monotonicity is evident from the fact that the children are chosen from the set $S_{\mathbf{x}}$ in order. Exclusivity is guaranteed by the fact that the bounds that delimit the $S_{\mathbf{x}}$ are non-overlapping. Completeness is guaranteed by the fact that the bounds that delimit the $S_{\mathbf{x}}$ cover $\mathbb{R}^+ \cup \{0\}$ since $\forall n, \mathbf{x}_0 \stackrel{def}{=} start(X_n), \theta_n(\mathbf{x})(\mathtt{t}) = 0$.

So it just remains to show the existence of at least one child for each point. In fact (because we need it later) we prove the existence of at least two children. That is for any $\mathbf{x} \in T_n, \exists \mathbf{y}, \mathbf{z} \in T_{n+1}$ such that $x - \frac{1}{2}\theta(\mathbf{x})(\Delta) < y, z \leq x + \frac{1}{2}\theta(\overrightarrow{\mathbf{x}})(\Delta)$

There are two cases:

- If $\mathbf{x} = start(T_n)$ then $c(\mathbf{x}, 0) = start(T_{n+1})$ and has value zero, call this point $\mathbf{y}$. Then $\theta_n(\overrightarrow{\mathbf{y}})(\mathtt{t}) \leq 0 + \max(\Delta)_{n+1} < \frac{1}{2}\min(\Delta)_n \leq x + \frac{1}{2}\theta(\overrightarrow{x})(\Delta)$. So $\overrightarrow{\mathbf{y}} \in S_{\mathbf{x}}$ and we have two points $\overrightarrow{\mathbf{y}} \in S_{\mathbf{x}}$ and $\mathbf{y}$, as required.
- Otherwise observe that $\frac{1}{2}\theta_n(\overleftarrow{\mathbf{x}})(\Delta) + \frac{1}{2}\theta_n(\mathbf{x})(\Delta) \geq \min(\Delta)_n > 2\max(\Delta)_{n+1}$. If there were no points in this region that would imply a distance of at least $2\max(\Delta)_{n+1}$ between points leading to a contradiction. Furthermore suppose there is only one point $\mathbf{y}$ in this range, then either $\theta(\mathbf{y})(\mathtt{t}) - \theta(\overleftarrow{\mathbf{y}})(\mathtt{t}) > \max(\Delta)_{n+1}$ or $\theta(\overrightarrow{\mathbf{y}})(\mathtt{t}) - \theta(\mathbf{y})(\mathtt{t}) > \max(\Delta)_{n+1}$. Either way there is a contradiction. So there must be at least two children.

Uniform Convergence Let $\mathbf{x} \approx \langle\sigma\rangle$ then the N children of $\langle\sigma j\rangle \in T_n$ have values bounded to be within the region:

$$\langle\sigma\rangle_{\mathtt{t}} - \frac{1}{2}\left\langle\overleftarrow{\sigma}\right\rangle_{\Delta} < \langle\sigma j\rangle_{\mathtt{t}} \leq \langle\sigma\rangle_{\mathtt{t}} + \frac{1}{2}\langle\sigma\rangle_{\Delta}$$

and therefore bounded within the region:

$$\langle\sigma\rangle_{\mathtt{t}} - \frac{1}{2}\max(\Delta)_n < \langle\sigma j\rangle_{\mathtt{t}} < \langle\sigma\rangle_{\mathtt{t}} + \frac{1}{2}\max(\Delta)_n$$

So consider the rightmost and leftmost limits $\langle\boldsymbol{\sigma}^{\boldsymbol{R}}\rangle_{\mathtt{t}}, \langle\boldsymbol{\sigma}^{\boldsymbol{L}}\rangle_{\mathtt{t}}$ they must be bounded as follows:

$$\begin{aligned}\langle\boldsymbol{\sigma}^{\boldsymbol{R}}\rangle_{\mathtt{t}} &< \langle\sigma\rangle_{\mathtt{t}} + \tfrac{1}{2}\max(\Delta)_n + \tfrac{1}{2}\max(\Delta)_{n+1} + \ldots \\ &< \langle\sigma\rangle_{\mathtt{t}} + \tfrac{1}{2}\max(\Delta)_n + \tfrac{1}{4}\max(\Delta)_n \ldots \\ &< \langle\sigma\rangle_{\mathtt{t}} + \max(\Delta)_n\end{aligned}$$

Similarly

$$\langle\boldsymbol{\sigma}^{\boldsymbol{L}}\rangle_{\mathtt{t}} \leq \langle\sigma\rangle\,\mathtt{t} - \max(\Delta)_n$$

So by monotonicity of the comb and the limit (Lemma 8.3) all the descents from $\langle\sigma\rangle$, lie within $\pm\max(\Delta)_n$ of $\langle\sigma\rangle_{\mathtt{t}}$. Since $\max(\Delta)$ tends to zero, for any $\epsilon > 0$ we can choose N such that $\forall n > N, 2\max(\Delta)_n < \frac{1}{2}\epsilon$ and hence meet the criterion for uniform convergence.

Even spread It has already been demonstrated that every point has at least two children. For a point $\langle\sigma\rangle \in T_n$ let the leftmost child be $\langle\sigma 0\rangle \in T_{n+1}$ and the rightmost child be $\langle\sigma r\rangle \in T_{n+1}$. Now we show that $\langle\sigma 0\rangle_{\mathtt{t}} < \langle\sigma\rangle_{\mathtt{t}} < \langle\sigma r\rangle_{\mathtt{t}}$.

$$\begin{aligned}&\langle\sigma 0\rangle_{\mathtt{t}} \leq \langle\sigma\rangle_{\mathtt{t}} - \tfrac{1}{2}\left\langle\overleftarrow{\sigma}\right\rangle_{\Delta} + \max(\Delta)_{n+1} \\ &\langle\sigma 0\rangle_{\mathtt{t}} \leq \langle\sigma\rangle_{\mathtt{t}} - \tfrac{1}{2}\min(\Delta)_n + \max(\Delta)_{n+1} \\ &\langle\sigma 0\rangle_{\mathtt{t}} < \langle\sigma\rangle_{\mathtt{t}}\end{aligned}$$

similarly

$$\begin{aligned}&\langle\sigma r\rangle_{\mathtt{t}} \geq \langle\sigma\rangle_{\mathtt{t}} + \tfrac{1}{2}\langle\sigma\rangle_{\Delta} - \max(\Delta)_{n+1} \\ &\langle\sigma r\rangle_{\mathtt{t}} \geq \langle\sigma\rangle_{\mathtt{t}} + \tfrac{1}{2}\min(\Delta)_n - \max(\Delta)_{n+1} \\ &\langle\sigma r\rangle_{\mathtt{t}} > \boldsymbol{\sigma}\mathtt{t}\end{aligned}$$

The spread of the children of $\langle\sigma\rangle \in T_n$ at level $n+1$ must be at least $\min(\Delta)_{n+1}$ since there are at least two children. Now consider the children of $\langle\sigma 0\rangle$. The left most child $\langle\sigma 00\rangle$ must be less than (or equal too in the case of *start(X)* its parent $\langle\sigma 00\rangle_{\mathtt{t}} < \langle\sigma 0\rangle_{\mathtt{t}}$. Similarly for the children of $\langle\sigma r\rangle$, the rightmost child must be greater than its parent $\langle\sigma rr\rangle_{\mathtt{t}} > \langle\sigma r\rangle_{\mathtt{t}}$. So the values of $\langle\sigma 00\rangle$ and $\langle\sigma rr\rangle$ are also separated by at least $\min(\Delta)_{n+1}$. Since each rightmost child cannot be less than its parent, and each leftmost child cannot be greater than its parent the point's spread at the limit must be greater than $\min(\Delta)_n$ □

Lemma 9.2 *Every point in the continuous clock has a unique value.*

Proof

Suppose the contrary, then there must exist $\langle\boldsymbol{\sigma}\rangle, \langle\boldsymbol{\sigma'}\rangle \in T_{\lim}$ where $\langle\boldsymbol{\sigma}\rangle_{\mathtt{t}} = \langle\boldsymbol{\sigma'}\rangle_{\mathtt{t}}$ but $\sigma \neq \sigma'$. (Taking $\langle\boldsymbol{\sigma}\rangle_{\mathtt{t}} \stackrel{def}{=} \theta(\langle\boldsymbol{\sigma}\rangle)(\mathtt{t})$).

Since $\sigma \neq \sigma'$ and they are both infinite strings, they must differ at some finite place, call this n_0. i.e. $\sigma_n = \sigma'_n \forall n < n_0$, but $\sigma_{n_0} \neq \sigma'_{n_0}$.

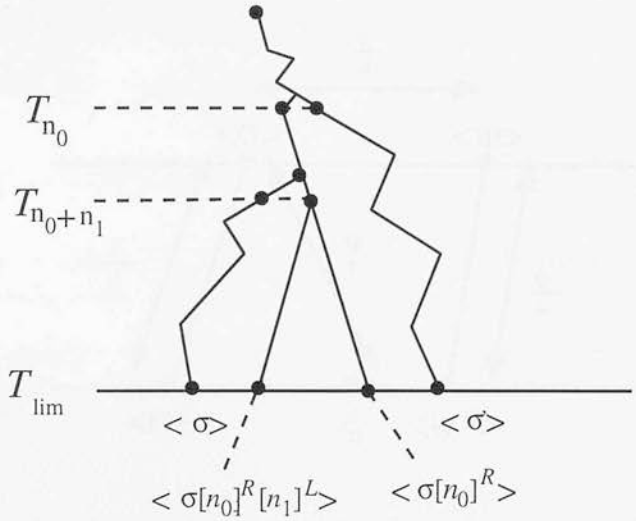


Figure C.2: Illustrating the proof of uniqueness of values in the continuous clock.

Without loss of generality we can assume $\sigma < \sigma'$, so $\sigma[n_0] < \sigma'[n_0]$. Now by definition $\sigma \neq \sigma[n]^R$, so there must exist some n_1 such that $\sigma_{n_0+n_1} \neq r(\langle\sigma[n_0+n_1-1]\rangle)$ so $\sigma[n_0+n_1] \neq \sigma[n_0]^R[n_1]$[1]. (see Figure C.2) By monotonicity $\langle\boldsymbol{\sigma}\rangle \prec \langle\boldsymbol{\sigma}[\boldsymbol{n_0}]^R[\boldsymbol{n_1}]^L\rangle$, and $\langle\boldsymbol{\sigma}[\boldsymbol{n_0}]^R\rangle \prec \langle\boldsymbol{\sigma'}\rangle$. Now by lemma 8.3 $\langle\boldsymbol{\sigma}\rangle_{\mathtt{t}} \leq \langle\boldsymbol{\sigma}[\boldsymbol{n_0}]^R[\boldsymbol{n_1}]^L\rangle_{\mathtt{t}}$ but

$$\langle\boldsymbol{\sigma}[\boldsymbol{n_0}]^R[\boldsymbol{n_1}]^R\rangle_{\mathtt{t}} - \langle\boldsymbol{\sigma}[\boldsymbol{n_0}]^R[\boldsymbol{n_1}]^L\rangle_{\mathtt{t}} = \mathrm{Sp}(\langle\sigma[n_0]^R[n_1]\rangle) = l > 0$$

(note that $\langle\boldsymbol{\sigma}[\boldsymbol{n_0}]^R[\boldsymbol{n_1}]^R\rangle = \langle\boldsymbol{\sigma}[\boldsymbol{n_0}]^R\rangle$), so $\langle\boldsymbol{\sigma}\rangle_{\mathtt{t}} - \langle\boldsymbol{\sigma'}\rangle_{\mathtt{t}} \geq l$, and there is a contradiction. □

Lemma 9.3 *The values of points in the flow T_{lim} are unbounded.*

Proof Suppose $\forall\langle\boldsymbol{\sigma}\rangle \in T_{\mathrm{lim}}$, the value $\langle\boldsymbol{\sigma}\rangle_{\mathtt{t}}$ is bounded by a value L. Consider the spread of points in T_n. In the continuous clock $\mathrm{Sp}(\langle\sigma[n]\rangle) = \langle\boldsymbol{\sigma}[\boldsymbol{n}]^R\rangle_{\mathtt{t}} - \langle\boldsymbol{\sigma}[\boldsymbol{n}]^L\rangle_{\mathtt{t}}$, since it is the sum of the values of Δ dividing up the children at each level, in the clock this is the same as the difference between the first and last child at each level. So because the comb must be spread evenly, $\exists M > 0$ such that $\mathrm{Sp}(\langle\sigma[n]\rangle) > M, \forall\langle\sigma[n]\rangle \in T_n$. By monotonicity of the flows

$$\langle\boldsymbol{\sigma}[\boldsymbol{n}]^L\rangle_{\mathtt{t}} = \langle\overleftarrow{\boldsymbol{\sigma}[\boldsymbol{n}]^L}\rangle_{\mathtt{t}} + \mathrm{Sp}(\langle\sigma[n]\rangle) > \langle\overleftarrow{\boldsymbol{\sigma}[\boldsymbol{n}]^L}\rangle_{\mathtt{t}} + M.$$

So $\langle\boldsymbol{\sigma}[\boldsymbol{n}]^L\rangle_{\mathtt{t}} > iM$ where $i \stackrel{def}{=} \left|Sub^{\prec\langle\sigma[n]\rangle}(T_n)\right|$ since T_n is open on the right, there must exist an $\langle\sigma[n]\rangle$ such that $iM > L$ and there is a contradiction. □

Theorem 9.4 *For any $r_0 \in \mathbb{R}^+ \cup \{0\}$, there is a point $\langle\boldsymbol{\sigma}\rangle \in T_{\mathrm{lim}}$ in the continuous clock such that $\langle\boldsymbol{\sigma}\rangle_{\mathtt{t}} = r_0$.*

Proof First consider the leftmost limit of the point $\mathbf{x}_0 \in T_0$ $\langle 0^L\rangle$. Because of monotonicity and completeness in the comb this descent must contain the bottom of each

[1] This notation deserves a little explaination. $\sigma[n_0]^R[n_1]$ is intended to mean the first n_1 characters of the infinite string $\sigma[n_0]^R$.

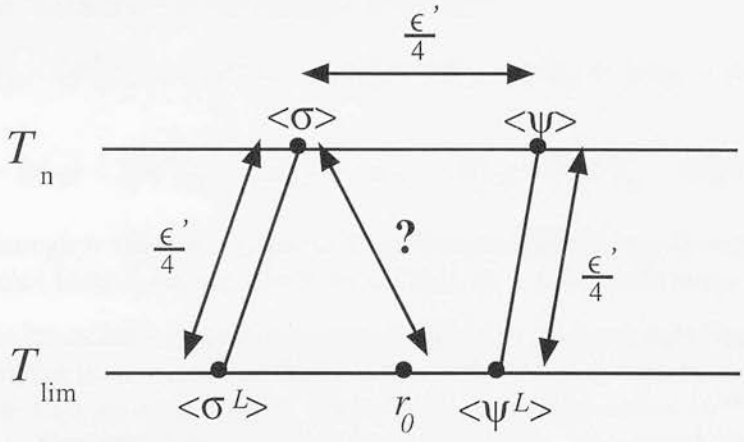


Figure C.3: Illustrating the proof that there is a descent to any r_0. Note: $\left\langle \overleftarrow{\psi} \right\rangle = \langle \sigma \rangle$.

flow in the sequence. By definition of the delta clocks these values will all be zero, and so consequently their limit in T_{lim} must also be zero. So the lemma is true for $r_0 = 0$.

So now consider all $r_0 > 0$; it is necessary to demonstrate the presence of a descent which tends to r_0. This is done by finding a point $\langle \sigma \rangle \in T_n$ for each n such that its leftmost limit is less than r_0 i.e. $\langle \boldsymbol{\sigma}^L \rangle_{\mathtt{t}} < r_0$ but its successor $\overrightarrow{\langle \sigma \rangle}$ has a leftmost limit greater than r_0 i.e.$\langle \overrightarrow{\boldsymbol{\sigma}^L} \rangle_{\mathtt{t}} > r_0$. Then we show that $\langle \sigma \rangle_{\mathtt{t}}$ lies within ϵ of r_0 for suitably large n, showing the existence of a sequence of values tending to r_0. It then remains to be shown that this is a descent.

Every descent value in the clock is unique (by lemma 9.2), and the values in the limit are unbounded (lemma 9.3). So there must be points be points $\langle \rho \rangle \in T_n$ such that $\langle \boldsymbol{\rho}^L \rangle > r_0$ take the minimum over '$\prec$' of these and call it $\langle \psi \rangle$, since the flow is regular $\langle \sigma \rangle \stackrel{def}{=} \overleftarrow{\langle \psi \rangle}$ will exist, and by monotonicity it will have a leftmost limit less than or equal to r_0 (otherwise $\langle \psi \rangle$ is not the minimum $\langle \rho \rangle$).

Now for some $\epsilon > 0$ find a value N such that for $n > N$ the following hold:

- $\left| \langle \sigma \rangle_{\mathtt{t}} - \langle \boldsymbol{\sigma}^L \rangle_{\mathtt{t}} \right| < \frac{\epsilon}{4}$
- $\left| \langle \psi \rangle_{\mathtt{t}} - \langle \boldsymbol{\psi}^L \rangle_{\mathtt{t}} \right| < \frac{\epsilon}{4}$
- $\left| \langle \psi \rangle_{\mathtt{t}} - \langle \sigma \rangle_{\mathtt{t}} \right| = \Delta_n < \frac{\epsilon}{4}$

By convergence of the comb, and $\max(\Delta)_n$, such an N must exist. (see Figure C.3). Now by the triangle inequality

$$\left| r_0 - \langle \sigma \rangle_{\mathtt{t}} \right| < \left| r_0 - \langle \sigma^L \rangle_{\mathtt{t}} \right| + \left| \langle \boldsymbol{\sigma}^L \rangle_{\mathtt{t}} - \langle \sigma \rangle_{\mathtt{t}} \right|$$

and we know that r_0 lies between the values of $\langle \boldsymbol{\sigma}^L \rangle$ and $\langle \boldsymbol{\psi}^L \rangle$ i.e. $\langle \boldsymbol{\sigma}^L \rangle_{\mathtt{t}} \leq r_0 < \langle \boldsymbol{\psi}^L \rangle_{\mathtt{t}}$. So

$$\left| r_0 - \langle \psi^L \rangle_{\mathtt{t}} \right| < \left| \langle \boldsymbol{\sigma}^L \rangle_{\mathtt{t}} - \langle \psi^L \rangle_{\mathtt{t}} \right|$$

and by further application of the triangle inequality:

$$\left|\langle \boldsymbol{\psi}^{\boldsymbol{L}} \rangle_{\mathtt{t}} - \langle \boldsymbol{\sigma}^{\boldsymbol{L}} \rangle_{\mathtt{t}}\right| < \left|\langle \boldsymbol{\psi}^{\boldsymbol{L}} \rangle_{\mathtt{t}} - \langle \psi \rangle_{\mathtt{t}}\right| + \left|\langle \psi \rangle_{\mathtt{t}} - \langle \sigma \rangle_{\mathtt{t}}\right| + \left|\langle \psi \rangle_{\mathtt{t}} - \langle \boldsymbol{\sigma}^{\boldsymbol{L}} \rangle_{\mathtt{t}}\right|$$

so

$$|r_0 - \langle \sigma \rangle_{\mathtt{t}}| < \left|\langle \boldsymbol{\psi}^{\boldsymbol{L}} \rangle_{\mathtt{t}} - \langle \psi \rangle_{\mathtt{t}}\right| + \left|\langle \psi \rangle_{\mathtt{t}} - \langle \sigma \rangle_{\mathtt{t}}\right| + \left|\langle \boldsymbol{\sigma}^{\boldsymbol{L}} \rangle_{\mathtt{t}} - \langle \sigma \rangle_{\mathtt{t}}\right| < \epsilon$$

So for large enough n there is a point in T_n arbitrarily close to r_0. It remains to show that these points form a descent which would lead to a point with value r_0 in T_{lim}.

So given $\langle \sigma \rangle \in T_n$ satisfying the properties required, we need to show that $\langle \sigma' \rangle \in T_{n+1}$ which also satisfies these properties in T_{n+1} is a child of $\langle \sigma \rangle$ i.e. $\exists i$ s.t. $\langle \sigma' \rangle = c(\langle \sigma \rangle, i)$. Let us suppose that no such i exists. Then by monotonicity either $\langle \boldsymbol{\sigma}'^{\boldsymbol{R}} \rangle \prec \langle \boldsymbol{\sigma}^{\boldsymbol{L}} \rangle$ or $\langle \boldsymbol{\sigma}'^{\boldsymbol{L}} \rangle \succeq \langle \boldsymbol{\psi}^{\boldsymbol{L}} \rangle$. In other words if $\langle \sigma' \rangle$ is not a child of $\langle \sigma \rangle$ the closest its descendants can get to r_0 is $\langle \boldsymbol{\sigma}^{\boldsymbol{L}} \rangle$ or $\langle \boldsymbol{\psi}^{\boldsymbol{L}} \rangle$, but we have already seen that we can get arbitrarily close so we have a contradiction. □

D

Proofs from chapter 10

Lemma 10.1 *For a sequence of valuation flows X_n over a signal set S, if $\forall \mathtt{s} \in S, X_n(\mathtt{s})$ converges then so does X_n.*

Proof First notice that because $\forall \mathtt{s} \in S, X_n(\mathtt{s})$ converges, then $\langle \boldsymbol{\sigma} \rangle_{\mathtt{S}}$ is well defined and therefore so is $\theta(\langle \boldsymbol{\sigma} \rangle)$.

It is required to show that $\forall \epsilon > 0, \exists N$ such that:

$$|\theta(\langle \sigma_n \rangle) \ominus \theta(\langle \boldsymbol{\sigma} \rangle)| < \epsilon$$

By assumption we have: $\forall \mathtt{s} \in S$

$$d(\langle \sigma[n] \rangle_{\mathtt{S}}, \langle \boldsymbol{\sigma} \rangle_{\mathtt{S}}) < \epsilon'$$

so let $\epsilon' = \frac{\epsilon}{|S|}$ then

$$\theta(\langle \sigma[n] \rangle) \ominus \theta(\langle \boldsymbol{\sigma} \rangle) = \Sigma_{\mathtt{S} \in S} d(\langle \sigma[n] \rangle_{\mathtt{S}}, \langle \boldsymbol{\sigma} \rangle_{\mathtt{S}}) < \epsilon' |S| = \epsilon.$$

□

The next theorem refers to a sequence of delta flow machines $M_n \stackrel{def}{=} (q0, V, S, X_n)$, as shown in Figure D.1.

Theorem 10.2 *For the sequence of machines M_n if f is continuous then for any comb over which $X_n(\mathtt{t})$ converges, X_n also converges to a limit $X_{\lim}$ in which the values of the signal u are defined as the trajectory $f(t)$.*

Proof It is necessary to show that for any comb c which converges uniformly with even spread over $X_n(\mathtt{t})$, the flow sequence $X_n(\mathtt{u})$ also converges uniformly with even

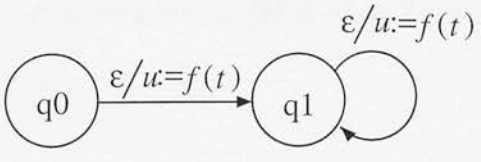


Figure D.1: A trajectory machine (the assignments to t are assumed)

spread at all points $\langle\boldsymbol{\sigma}\rangle \in X_{\lim}$ where f is continuous and that the values of $\mathtt{u}$ are defined in the limit flow $X_{\lim}$ as $u = f(t)$.

First of all notice that spread is a property of the comb which is dependant only on the value of Δ, so even spread over $X_n(\mathtt{u})$ follows immediately from the even spread of $X_n(\mathtt{t})$.

Proving uniform convergence is a little more involved. Continuity of f means that over any closed bounded interval I

$$\forall\epsilon > 0, \exists\delta \text{ s.t. } |f(k) - f(a)| < \epsilon \; \forall k \text{ with } |k - a| < \delta \tag{D.1}$$

for any $a \in I$.

For uniform convergence we require, for some contiguous subflow $Z_{\lim} \sqsubset X_{\lim}$ closed at the top and bottom, that $\forall\epsilon' > 0 \exists N$ s.t.

$$\forall\langle\boldsymbol{\sigma}\rangle \in Z_{\lim}, |\langle\sigma[n]\rangle_{\mathtt{u}} - \langle\boldsymbol{\sigma}\rangle_{\mathtt{u}})| < \epsilon'$$

For an arbitrary $\epsilon' > 0$ we have to show the existence of a suitable N.

Let $\epsilon = \epsilon'$, for ϵ from D.1, this fixes a value for δ. Now $X_n(\mathtt{t})$ is uniformly convergent so we know that $\exists N'$ s.t. $\forall n > N'$

$$|\langle\sigma[n]\rangle_{\mathtt{t}} - \langle\boldsymbol{\sigma}\rangle_{\mathtt{t}}| < \delta$$

So with $N = N'$ and by D.1,

$$|f(\langle\sigma[n]\rangle_{\mathtt{t}}) - f(\langle\boldsymbol{\sigma}\rangle_{\mathtt{t}})| < \epsilon = \epsilon'$$

$\forall\langle\boldsymbol{\sigma}\rangle \in Z_{\lim}$. Since $Z_{\lim}$ was arbitrarily chosen this gives our result. □

Leading on from this result there are a number of corrolaries regarding the continuity of the limit flow $X_{\lim}$. These results refer to a second machine $M'_n \stackrel{def}{=} (q0, V, S', X'_n)$, which is very similar M_n except that there are additional signals $\mathtt{v}$ and $\mathtt{w}$ so $S' \stackrel{def}{=} S \cup \{\mathtt{v}, \mathtt{w}\}$. As before the value of signal $\mathtt{u}$ is given by $u := f(t)$. Here the value of signal $\mathtt{w}$ is described as a function of the signal $\mathtt{v}$ given by $w := h(v)$.

Corollary 10.3

1. *The flow $X_{\lim}(\mathtt{u})$ is θ-continuous (except at start$(X_{\lim})$) iff f is continuous.*
2. *If $X_{\lim}(\mathtt{v})$ is θ-continuous then it can be described as a continuous trajectory where $v = f(t)$ for a continuous function f.*
3. *The flow $X_{\lim}(\mathtt{w})$ is θ-continuous (except at start$(X_{\lim})$) if h is continuous and v is θ-continuous.*

Proof

Part 1 The first proof demonstrates the commonality of the definition of continuity of a function, and θ-continuous of a flow.

First let us prove the 'if' direction of the theorem. For all points $\mathbf{x} \in X_{\lim}$ such that $\mathbf{x} \succ \mathit{start}(X)$, $\mathbf{x}$ satisfies all three criteria for Theorem 8.4:

1. From the definition of the limit the comb must be uniformly convergent
2. We need to show that $d(f(t), f(t-\Delta)) < \epsilon$ below an arbitrary point in the comb. Because of the bounds on Δ and the continuity of f this will hold.
3. The last condition is symmetric to the second.

So is $X_{\lim}(\mathtt{u})$ is θ-continuous.

The 'only if' direction can be established by contradiction. Suppose f is not continuous but $X(\mathtt{u})$ is θ-continuous. So at all points $\mathbf{a} \in X_{\lim}(\mathtt{u})$, it is dense and for every $\epsilon > 0$ there exist points $\mathbf{k}$ and $\mathbf{k}'$ (with $\mathbf{k} \prec \mathbf{a} \prec \mathbf{k}'$) such that $|\theta(\mathbf{x}) - \theta(\mathbf{a})| < \epsilon$ for all $\mathbf{x}$ such that $\mathbf{k} \prec \mathbf{x} \prec \mathbf{k}'$. Take one of the discontinuities of f, that occurs at a point $\mathbf{d}$ with value d The θ-continuity of $X(\mathtt{u})$ implies that for all $\epsilon > 0, \exists \delta$ s.t. $|f(d) - f(z)| < \epsilon, \forall \mathbf{z}, |d - z| < \delta$, simply by finding $\mathbf{k}, \mathbf{k}'$ above and taking $\delta = \min(\theta(\mathbf{k}) - \theta(\mathbf{d}), \theta(\mathbf{k}') - \theta(\mathbf{d}))$. This would mean that f is continuous and so leads to a contradiction.

Part 2 There is an implicit function from the values of $\mathtt{t}$ to the value of $\mathtt{w}$, call it g, defined for all $\mathbf{x} \in X_{\lim}$ by $g(\theta(\mathbf{x})(\mathtt{t})) \stackrel{def}{=} \theta(\mathbf{x})(u)$. By the first proof this function must be continuous.

Part 3 From part 2 there is a continuous function g such that $g(t) = v$. Since the composition of two continuous functions is continuous and $w = h(g(t))$ then by corollary 10.3, $X_{\lim}(\mathtt{w})$ is a θ-continuous flow. □

Lemma 10.4 *In a dense value flow $X_{\lim}$ over a signal set S, if $\forall \mathtt{s} \in S$, $X_{\lim}(\mathtt{s})$ is a θ-continuous flow then $X_{\lim}$ is a θ-continuous flow.*

Proof By θ-continuity of $X_{\lim}(\mathtt{s})$ we have $\forall \langle \boldsymbol{\sigma} \rangle \in X_{\lim}, \forall \mathtt{s} \in S$,

$$\begin{gathered} \exists \mathbf{k}_{\mathtt{S}}, \mathbf{k}'_{\mathtt{S}} \text{ s.t. } |\langle \boldsymbol{\sigma} \rangle_{\mathtt{S}} - \langle \boldsymbol{\psi} \rangle_{\mathtt{S}}| < \epsilon_{\mathtt{S}} \\ \forall \langle \boldsymbol{\psi} \rangle \text{ s.t. } \left| \mathbf{k}_{\mathtt{S}} \prec \langle \boldsymbol{\psi} \rangle \prec \mathbf{k}'_{\mathtt{S}} \right| \end{gathered}$$

let $b \stackrel{def}{=} |S|$. Choose an arbitrary $\epsilon > 0$, and then fix each $\epsilon_{\mathtt{S}} \stackrel{def}{=} \frac{\epsilon}{b}$.

Choose $\mathbf{k}$ to be the least upper bound of $\{\mathbf{k}_{\mathtt{T}}\}$ *i.e.*

$$\mathbf{k} \stackrel{def}{=} \mathbf{k}_{\mathtt{T}} \text{ s.t. } \forall \mathtt{s} \in S \; \mathbf{k}_{\mathtt{S}} \preceq \mathbf{k}_{\mathtt{T}} \prec \langle \boldsymbol{\sigma} \rangle$$

and similarly choose $\mathbf{k}'$ to be the greatest bound of $\{\mathbf{k}'_r\}$ *i.e.*

$$\mathbf{k}' \stackrel{def}{=} \mathbf{k}'_{\mathtt{T}} \text{ s.t. } \forall \mathtt{s} \in S \; \langle \boldsymbol{\sigma} \rangle \preceq \mathbf{k}'_{\mathtt{T}} \prec \mathbf{k}'_{\mathtt{S}}$$

Then for all $\langle\psi\rangle$ such that $\mathbf{k} \prec \langle\psi\rangle \prec \mathbf{k}'$

$$|\theta(\langle\sigma\rangle) \ominus \theta(\langle\psi\rangle)| < \sum_{\mathbf{s}\in S} \epsilon_{\mathbf{s}} = \epsilon,$$

□

The following result establishes sufficient criteria for which the limit of Delta I/O machines is well defined. It refers to a sequence of delta flow machines M_n with output flow X_n and signal set S with $\{\mathtt{t},\mathtt{u}\} \subseteq S$ where, the value of $\mathtt{u}$ updated with respect to an expression:

$$u := \overleftarrow{u} + \Delta f(t, \overleftarrow{u})$$

Theorem 10.5 The limit flow $X_{\lim}$ exists for any comb which converges over the delta clock if f has a bounded partial derivative with respect to its second variable and if the solution of the differential equation:

$$\frac{du}{dt} = f(t,u) \tag{D.2}$$

has a bounded second derivative. Furthermore the flow it describes satifies equation D.2.

Proof The result follows because under these conditions the delta machine is calculating the Euler's method approximation to the differential equation 10.2 and there is a standard result which states that the error in the approximation is bounded by $M_E \max(\Delta)_n$, (for some fixed constant M_E) and thus uniformly convergent.

Extension This extension deals with the case where the assignment of the value of $\mathtt{u}$ given above follows a constrained divider point $\mathbf{k}_n$. An additional error will be introduced into the calculation as a result of the flow starting at a divider point. Because the divider point is constrained, this error will be bounded by $M_{\mathbf{k}} \max(\Delta)_n$. So the total distance $|\langle\sigma[n]\rangle_{\mathtt{u}} - \langle\sigma\rangle_{\mathtt{u}}|$ for any $\langle\sigma[n]\rangle \succ \mathbf{k}_n$ will be bounded by $M_E \max(\Delta)_n + M_{\mathbf{k}} \max(\Delta)_n$, so can be made arbitrarily small, thus giving uniform convergence. □

This next Theorem does not appear in the main body of the text but is included here as an addendum.

Theorem D.1 *Consider a sequence of delta flow machines M_n with output flow X_n and signal set S with $\{\mathtt{t},\mathtt{u}\} \subseteq S$ where, beyond a constrained divider point $\mathbf{k}_n$, the value of $\mathtt{u}$ updated with respect to an expression:*

$$u := \overleftarrow{u} + \Delta f(\overleftarrow{t})$$

for the usual clock signal $\mathtt{t}$. Then for any comb c over which $X_n(\mathtt{t})$ converges uniformly with even spread, $X_n(\mathtt{u})$ will also converge uniformly with even spread. Furthermore the value of $\mathtt{u}$ at a point $\mathbf{z} \in X_{\lim}, \mathbf{z} \succ \mathbf{k}_{\lim}$ will be defined by:

$$\theta(\mathbf{z})(\mathtt{u}) := \theta(\mathbf{k}_{\lim})(\mathtt{u}) + \int_{\theta(\mathbf{k}_{\lim})(\mathtt{t})}^{\theta(\mathbf{z})(\mathtt{t})} f(v)\, dv$$

Proof Let σ be the unique descent to $\mathbf{z}$. Then consider the value of a point $\mathtt{u}$ at any point $\mathbf{x}_n \stackrel{def}{=} \langle \sigma[n] \rangle$.

$$\theta_n(\mathbf{x}_n)(\mathtt{u}) = \theta_n(\mathbf{k}_n)(\mathtt{u}) + \sum_{\mathbf{y} \in Y_n} \theta(\mathbf{y})(\Delta) f(\theta(\mathbf{y})(\mathtt{t}))$$

for $Y_n = \{\mathbf{y} \in X_n | \mathbf{k}_n \prec \mathbf{y} \prec \mathbf{x}_n\}$. This is a Riemann sum from $\mathbf{k}_n$ to $\mathbf{x}_n$, which is close to a Riemann sum between $\mathbf{k}_{\lim}$ and $\mathbf{z}$. The maximum amount by which it differs is:

$$Err \stackrel{def}{=} A \left| \theta(\mathbf{k}_n)(\mathtt{t}) - \theta(\mathbf{k}_{\lim})(\mathtt{t}) \right| + B \left| \theta(\mathbf{x}_n)(\mathtt{t}) - \theta(\mathbf{z})(\mathtt{t}) \right|$$

where

$$A \stackrel{def}{=} \max \left(f(\theta(\mathbf{k}_n)(\mathtt{t})), f(\theta(\mathbf{k}_{\lim})(\mathtt{t})) \right)$$

$$B \stackrel{def}{=} \max \left(f(\theta(\mathbf{x}_n)(\mathtt{t})), f(\theta(\mathbf{z})(\mathtt{t})) \right)$$

However f is being considered over a closed bounded interval, so its value is bounded by a maximum, call it M_f. Then because the sequence of divider points is constrained, and because c is convergent we see that Err is bounded by:

$$Err < M_f M_{\mathbf{k}} \max(\Delta)_n + M_f M_{\mathbf{x}} \max(\Delta)_n$$

where $M_{\mathbf{k}}$ is the divider point constraint bound, and $M_{\mathbf{x}}$ is the bound of uniform convergence of $\mathtt{t}$ descents. Since $\max(\Delta)_n \to 0$ as $n \to \infty$ this error term disappears, and the point tends to the integral given.

It remains to show that the descent tends to this value in a uniform way. In order to do this we show that over some contiguous subflow $Z_{\lim}$ of $X_{\lim}$ closed at top and bottom that $\forall \epsilon, \exists N$s.t.

$$|\theta_n(\mathbf{x}_n)(\mathtt{u}) - \theta_{\lim}(\mathbf{z})(\mathtt{u})| < \epsilon, \forall n > N$$

So first observe that the error bound Err is fixed for all points in $Z_{\lim}$ since $M_{\mathbf{k}}$ and $M_{\mathbf{x}}$ are both fixed and M_f is fixed for any closed bounded interval.

Now consider the Reimann sum from $\mathbf{k}_n$ to any point $\mathbf{x}_{\sigma[\mathtt{n}]} \succ \mathbf{k}_{\lim}$ such that $\mathbf{z} \in Z_{\lim}$. It is necessary to show that the error in the Riemann sum is less than $M_R \max(\Delta)_n$, for any set of δ values. Because f is well behaved, there exists a bound, call it G, on f' over the region $[\theta_{\lim}(\mathbf{k}_n)(\mathtt{t}), \theta_{\lim}(\mathbf{x}_n)(\mathtt{t})]$. Let $Y_n = \{\mathbf{y} | \mathbf{k}_n \preceq \mathbf{y} \preceq \mathbf{x}_n\}$. Then the total error in the Riemann sum over Y_n must be less than

$$\sum_{\mathbf{y} \in Y_n} \frac{1}{2} \theta_n(\mathbf{y})(\Delta)^2 G$$

and

$$G \sum_{\mathbf{y} \in Y_n} \frac{1}{2} \theta_n(\mathbf{y})(\Delta)^2 < G \sum_{\mathbf{y} \in Y_n} \frac{1}{2} \theta_n(\mathbf{y})(\Delta) \max(\Delta)_n < G \left[\sum_{\mathbf{y} \in Y_n} \frac{1}{2} \theta_n(\mathbf{y})(\Delta) \right] \max(\Delta)_n$$

so take

$$G \left[\sum_{\mathbf{y} \in Y_n} \frac{1}{2} \theta_n(\mathbf{y})(\Delta) \right]$$

which is bounded. Let us call the bound M_R.

We can now see that

$$|\theta_n(\mathbf{x}_n)(\mathrm{u}) - \theta_{\lim}(\mathbf{z})(\mathrm{u})| < (M_f M_{\mathbf{k}} + M_f M_{\mathbf{x}} + M_R) \max(\Delta)_n$$

since $\max(\Delta)_n \to 0$ for any $\epsilon > 0$ there is an N beyond which this holds for all n. $\square$

E

Proofs from chapter 11

Lemma 11.1 *For the general transition depicted in Figure 11.1, where the sequence of divider points* $\mathbf{k}_n^{\alpha|\beta}$ *for* $\mathtt{u}$ *is constrained,*

- *If* α *is a continuous trajectory* $f(t)$ *then for any comb* c *which converges over* $X_n(t)$*,the limit* $X_{\text{lim}}(\mathtt{u})$ *is well defined up to (but not necessarily including)* $\mathbf{k}_{\text{lim}}^{\alpha|\beta} \in X_{\text{lim}}$.
- *If* β *is a continuous trajectory* $g(t)$ *then for any comb* c *which converges over* $X_n(t)$*,the limit* $X_{\text{lim}}(\mathtt{u})$ *is well defined from (but not necessarily including)* $\mathbf{k}_{\text{lim}}^{\alpha|\beta} \in X_{\text{lim}}$.

Proof Even spread of c is immediate so we simply need to show uniform convergence of those descents of the comb which lead to the sub flow up to $\mathbf{k}_{\text{lim}}^{\alpha|\beta}$. This is done by showing that for any point to the left of $\mathbf{k}_{\text{lim}}^{\alpha|\beta}$ for sufficiently large N the values of its descent are defined entirely in terms of α.

By uniform convergence of $X_n(\mathtt{t})$ with respect to c, $\forall \epsilon > 0 \exists N$ such that

$$|\langle \sigma[n] \rangle_{\mathtt{t}} - \langle \boldsymbol{\sigma} \rangle \mathtt{t}| < \epsilon, \ \forall n > N$$

for any point $\langle \boldsymbol{\sigma} \rangle \in X_{\text{lim}}$.

Choose a point $\langle \boldsymbol{\sigma} \rangle \in X_{\text{lim}}$ such that $\langle \boldsymbol{\sigma} \rangle \prec \mathbf{k}_{\text{lim}}^{\alpha|\beta}$. Because of uniqueness of descent (Lemma 9.2) $0 < \left| \langle \boldsymbol{\sigma} \rangle_{\mathtt{t}} - \theta_{\text{lim}}(\mathbf{k}_{\text{lim}}^{\alpha|\beta})(\mathtt{t}) \right| \stackrel{def}{=} \delta$

Now we can choose N such that $\forall n > N, |\max(\Delta)_n M + \epsilon| < \delta$ where M is the divider point constraint bound. The value of δ is fixed, so such an N must exist. Beyond this point the value $\mathtt{u}$ will just be defined with respect to α (*i.e.* $f(t)$) and so will converge by lemma 10.2, i.e.

$$\forall n > N, \theta_n(\langle \sigma[n] \rangle_{\mathtt{u}} = f(\langle \sigma[n] \rangle_{\mathtt{t}})$$

The second part can be shown by symmetry. □

Lemma 11.2 *For a set of uncountable piecewise-*θ*-continuous valuation flows* R *which are equivalent almost everywhere the global equivalence flow is well defined.*

Proof We are required to show that for any two uncountable piecewise-θ-continuous flows $X, Y \in R$ that $X \setminus P_X \equiv Y \setminus P_Y$

First notice that P_X, P_Y are regular flows and since $X \cong Y$ we know that

$$X \setminus P_X \cong Y \setminus P_Y \Rightarrow$$
$$(X \setminus P_X) \setminus S'_{XY} \equiv (Y \setminus P_Y) \setminus S'_{YX}$$

what we need to show is that the smallest sets S'_{XY}, S'_{YX} which satisfy this are empty.

Now $S_{XY} \subseteq P_X$ and $S_{YX} \subseteq P_Y$ so if $S_{XY} = P_X$ and $S_{YX} = P_Y$ then we are done.

So suppose that there are additional points in P_X which are not S_{YX}. These points represent points which are equivalent in X and Y but which differ between X and other flows in R. Now it will be demonstrated that for every such point in P_X there is a matching point in P_Y that has been removed from the flow Y. If all these points match then there will be no need to remove any further points from $Y \setminus P_Y$ to make it match $X \setminus P_X$ and so S'_{YX} will be empty.

So choose an arbitrary point $\mathbf{x} \in P_X$ such that $\mathbf{x} \notin S_{XY}$. Now there must exist $Z \in R$ such that $\mathbf{x} \in S_{XZ}$.

Now assume this sequence of flows is timed *i.e.* there is a strictly monotonic clock signal $\mathtt{t}$ with a value at each point. Then let $t_{\mathbf{x}} \stackrel{def}{=} \theta_X(\mathbf{x})(\mathtt{t})$ and let $\mathbf{y} \in Y$ be the point such that $\theta_Y(\mathbf{y})(\mathtt{t}) = t_{\mathbf{x}}$ and similarly let z be the point in Z such that $\theta_Z(\mathbf{z})(\mathtt{t}) = t_{\mathbf{x}}$.

Since $\mathbf{x} \notin S_{XY}$ that implies that $\theta_X(\mathbf{x}) = \theta(\mathbf{y})$, but $\mathbf{x} \in S_{XZ}$ which implies that $\theta_X(\mathbf{x}) = \theta(\mathbf{z})$ so $\theta_Y(\mathbf{y}) \neq \theta(\mathbf{z})$ and therefore $\mathbf{y} \in P_Y$.

The point $\mathbf{x}$ was chosen arbitrarily from $P_X \setminus S_{XY}$ so for every such point there is a matching point $\mathbf{y}$, and so therefore S'_{YX} is empty. By symmetry S'_{XY} must also be empty and the result holds.

It was assumed that the flow was timed; however, this assumption was only necessary to make the proof neater. The temporal metric provided a function that mapped the set of nonequivalence points from different flows to one another. However, because the flows are piecewise-θ-continuous such a function is already available. Because any two flows are equivalent almost everywhere to each other, there is a one-one isotone which maps all the points except the discontinuities to each other. However, because the discontinuities are isolated there is an implicit mapping between these points too. □

F

Proofs for chapter 12

Lemma 12.1 *Consider a characteristic function* $\chi_n \to \chi_{\lim}$ *and a valuation sequence* $V_n \to V_{\lim}$ *with* $V_n \in \mathcal{E}_{d(\chi_n)}$*, and the sequence of valuations* $\chi_n[V_n]$*. Then providing* $\chi_{\lim}$ *is continuous at* $V_{\lim}$*,*

$$\lim_{n\to\infty} \chi_n[V_n] \equiv \chi_{\lim}[V_{\lim}].$$

Proof For some $\epsilon > 0$, choose an N such that:

$$\begin{array}{rcl} |\chi_n[V_n] \ominus \chi_{\lim}[V_n]| & < & \frac{\epsilon}{2} \\ |\chi_{\lim}[V_n] \ominus \chi_{\lim}[V_{\lim}]| & < & \frac{\epsilon}{2} \end{array}$$

By convergence of χ_n, convergence of V_n and continuity of $\chi_{\lim}$ respectively such an N exists. So by the triangle inequality for '$\ominus$':

$$|\chi_n[V_n] \ominus \chi_{\lim}[V_n]| < \epsilon$$

as required. □

Lemma 12.2 *Consider two standard characteristic function sequences* χ_n *and* ψ_n*. Let* χ_n, ψ_n *be uniformly convergent almost everwhere over* Δ_n*, such that* $\chi_n \to \chi_{\lim}, \psi_n \to \psi_{\lim}$*. Let* $\chi_n \cup \psi_n$ *be well defined for all* n*. Then* $\chi_n \cup \psi_n$ *is guaranteed to converge to* $\chi_{\lim} \cup \psi_{\lim}$ *at all points* V_S *where* $\chi_{\lim} \cup \psi_{\lim}[V_S] \downarrow D(\chi^f_{\lim} \cup \psi^f_{\lim})$ *is continuous.*

Proof We are required to show that $\forall \epsilon > 0, \exists N$ such that

$$|(\chi_n \cup \psi_n)[V] \ominus (\chi_{\lim} \cup \psi_{\lim})[V]| < \epsilon$$

At first sight this would appear straightforward; however, there is a problem because of the way in which the union operator is defined over characteristic functions is as a fixed point. The assignment values may provide input to each other, and so care must be taken to show that the limit is approached as expected.

First choose an arbitrary $\epsilon > 0$. Now $(\chi_n \cup \psi_n)^\Delta$ is independant of the fixed partition in that their input comes entirely from $S_{I \cup \overleftarrow{S}}$. Consequently for any V over which χ and ψ are convergent, there exists an N such that

$$\left|(\chi_n \cup \psi_n)^\Delta[V] \ominus (\chi_{\lim} \cup \psi_{\lim})[V]\right| < \frac{\epsilon}{2}$$

The rest of the characteristic function $(\chi_n \cup \psi_n)^f$ is fixed with respect to Δ_n, but may depend on the output of $(\chi_n \cup \psi_n)^\Delta$.

Observe that

$$\begin{aligned}(\chi_n \cup \psi_n)[V] &= ((\chi_n \cup \psi_n)^\Delta \cup (\chi_n \cup \psi_n)^f)[V] \\ &= \mu X.(\chi_n \cup \psi_n)^\Delta[V \cup X] \cup (\chi_n \cup \psi_n)^f[V \cup X] \\ &= (\chi_n \cup \psi_n)^\Delta[V] \cup (\chi_n \cup \psi_n)^f[W_n]\end{aligned}$$

where $W_n \stackrel{def}{=} V \cup (\chi_n \cup \psi_n)^\Delta[V]$. Then see that because of continuity and lemma 12.1, $\exists N$ such that $\forall n > N$

$$\left|(\chi_n \cup \psi_n)^f[W_n] \ominus (\chi_{\lim} \cup \psi_{\lim})^f[W_{\lim}]\right| < \frac{\epsilon}{2}$$

Now notice that because $\ominus$ is a sum of differences that

$$\begin{aligned}|(\chi_n \cup \psi_n)[V] \ominus (\chi_{\lim} \cup \psi_{\lim})[V]| \leq \; & \left|(\chi_n \cup \psi_n)^\Delta[V] \ominus (\chi_{\lim} \cup \psi_{\lim})^\Delta[V]\right| \quad + \\ & \left|(\chi_n \cup \psi_n)^f[V_n] \ominus (\chi_{\lim} \cup \psi_{\lim})^f[V_{\lim}]\right| \quad < \epsilon\end{aligned}$$

as required. □

Theorem 12.4 *For any two delta-flow machine M_Δ, M'_Δ the synchronous product of their limits is equivalent to the limit of their synchronous product i.e.*

$$M_{\lim} \| M'_{\lim} \equiv \lim_{\Delta \to 0} M_\Delta \| M'_\Delta$$

Proof Clearly on both sides the set of states, initial state and signal sets are identical. It remains to show that the two flows are equivalent, i.e.

$$\lim_{\Delta \to 0} (F_{M_\Delta} \amalg F_{M'_\Delta}) \equiv F_{M_{\lim}} \amalg F_{M'_{\lim}}.$$

Let $X = F_{M_{\lim}} \amalg F_{M'_{\lim}}$ and $Y = \lim_{\Delta \to 0}(F_{M_\Delta} \amalg F_{M'_\Delta})$.

First we show $X \sqsubseteq Y$ *i.e.* there exists an isotonic mapping $g : Y \to X$ s.t.

$$\forall \mathbf{y} \in Y \;\; \theta_X(g(y)) \equiv \theta_Y(y).$$

It can be assumed without loss of generality that the machines are timed in which case, this mapping would have to be isotonic. If we can prove the existance of equivalent points the mapping will be well defined. In other words for every $\mathbf{x} \in X$ there is a unique $\mathbf{y} \in Y$ such that $\theta_Y(\mathbf{y}) = \theta_X(\mathbf{x})$

What we do now is to prove that equivalent points exist assuming that the same comb has been used for both limits. If the the limits are the same for any arbitrary comb, then the global equivalence set will also be the same.

Take an arbitrary point $\mathbf{y} \in Y$, let $t_0 \stackrel{def}{=} \theta_Y(\mathbf{y})(\mathtt{t})$. Now by lemma 12.2

$$\theta_Y(\mathbf{y}) = (\chi_{M_\Delta} \cup \chi_{M'_\Delta})^{\mathbf{y}} \equiv (\chi_{M_\Delta})^{\mathbf{y}} \cup (\chi_{M_\Delta})^{\mathbf{y}}$$

Now let $t_p \stackrel{def}{=} (\chi_{M_\Delta})^{\mathbf{y}}(\mathtt{t})$ clearly $(\chi_{M'_\Delta})^{\mathbf{y}}(\mathtt{t}) = t_p$.

There must exist a point $\mathbf{x} \in X$ such that $\theta_X(\mathbf{x})(\mathtt{t}) = t_0$. $\theta(\mathbf{x}) = (\chi_{M_\Delta})^{\mathbf{x}} \cup (\chi_{M_\Delta})^{\mathbf{x}}$. Since t is strictly monotonic we can deduce that $(\chi_{M_\Delta})^{\mathbf{x}} = (\chi_{M_\Delta})^{\mathbf{y}}$ and likewise $(\chi_{M'_\Delta})^{\mathbf{x}} = (\chi_{M'_\Delta})^{\mathbf{y}}$ so therefore:

$$\theta_X(\mathbf{x}) = (\chi_{M_\Delta})^{\mathbf{x}} \cup (\chi_{M'_\Delta})^{\mathbf{x}} = (\chi_{M_\Delta})^{\mathbf{y}} \cup (\chi_{M'_\Delta})^{\mathbf{y}} = (\chi_{M_\Delta} \cup \chi_{M'_\Delta})^{\mathbf{y}} = \theta_Y(\mathbf{y})$$

The point $\mathbf{y}$ was arbitrarily chosen so $X \sqsubseteq Y$, and the mapping is one-to-one and onto so $Y \sqsubseteq X$ and therefore $X \equiv Y$ $\square$

G

Proofs for chapter 14

Theorem 14.1 For any system H described as a BBM model, there exists a Continuous I/O machine M such that the output flow M is equivalent to the flow described by H.

Proof (sketch) First we'll need some new pieces of notation. The double arrow shown in Figure G.1 is short hand for the machine fragment beneath it. This will be used to model the timed transitions of the BBM model.

Let us define a vector valuation as simply a valuation which ranges over each of the vector components. Then define $A(i)$ as being a set of valuations such that $V \in A(i)$ implies that V defines a point in the space A_i, for all i and let $\overline{A(i)}$ be the complement. Similarly let us define $C(i)$ for the space C_i. Let us assume that the controller can signal the occurrence of a discrete change with the signal $k = 1$. Let κ be the valuation $\kappa = \{(k, 1)\}$ and let $\overline{\kappa} = \{(k, v) | v \in \mathbb{R}, v \neq 1\}$. Let us also assume that a function $\eta(x(t))$ or $\eta(x(t), v)$ can be defined such that when $x(t)$ makes a discrete jump to $x'(t) \in S_j$ $\eta(\cdot) \stackrel{def}{=} j$.

Now observe that from the partitioning of the space in H there are two significant states for each S_i. These represent whether $x(t) \in C_i$ in which case the controller may instigate a discrete change, or it is not.

These two states are represented in the machine M shown in Figure G.2. State $q0$ represents the state in which $x \in C_i$. There are two transitions from $q0$, (1) and (2), and there are three looping transitions (0),(3), and (4).

- (0) The looping transition (0) represents takes place unless a change takes place.
- (1) The transition (1) represents what happens if nothing changes transition (2) represents the system dynamics continuously taking x out of C_i and thus to state $q1$.
- (2) and (5) Transitions (2) or (4) takes place when κ becomes 1, that is when a discrete jump is instigated by the controller. Let the new value of x be $x' \in S_j$. Transition (2) takes place when the destination of x falls outside C_j. Transition (4) takes place if it falls inside.
- (3) and (4) take place when $x \in A_i$, *i.e.* an autonomous jump. In which case

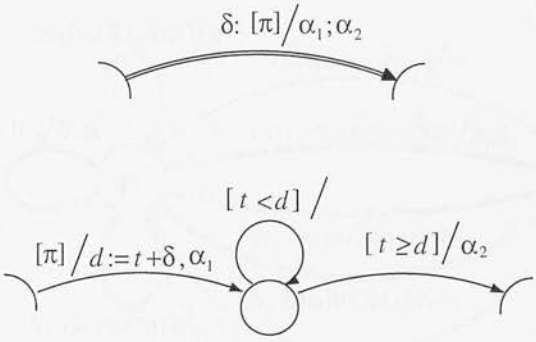


Figure G.1: A delay transition

(3) occurs when the destination of x falls outside C_j and (5) takes place when it falls inside.

Similarly there are 3 transitions from state $q1$, (6), (7) and (8):

- (6) is a looping transition which occurs so long as x remains in $\overline{C(i)} \cup \overline{A(i)}$.
- (7) and (9) if x enters $A(i)$ then an autonomous jump takes place. If the destination is not in C_j (for $j = \eta(x)$) then transition (9) will be taken, otherwise the looping transition (7) will be taken.
- (8) finally transition (8) represents a continuous change of x from $\overline{C_i}$ to C_i.

So the signals in the machine will be as follows (where all but `t,i,j` are vector signals.

- $I_M = \{$`u,v,x'`$\}$
- $O_M = \{$`x,t,i,j`$\}$

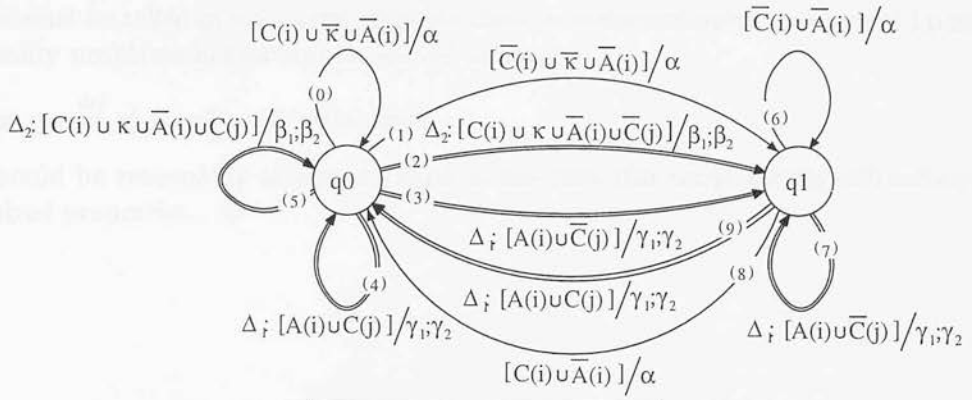
The destination sets D_i do not effect the presence of dynamics so can be ignored in modelling the system.

From the construction it should be clear that M will model the dynamics of H. □

Theorem 14.2 *For any Hybrid Automaton H in which the activities can be described as trajectories, integrator functions or dynamical systems, there exists a Continous I/O Machine M such that there is an isotonic function h which will map every point in any run of H to a point in the flow from the M.*

Proof (sketch) The correspondance between the two types of machine is fairly close. A full proof of the theorem is not offered here, but an outline is given of the first part that illustrates this correspondance.

So for a Hybrid Automaton $H = (Loc, Var, Lab, Edg, Act, Inv)$, I construct a Continuous I/O Machine $M = (S_M, V_M, R(\chi_M))$.



α :	$x := \xi(t)$
β_1 :	$j := \eta(x)$
β_2 :	$x := x', i := j$
γ_1 :	$j := \eta(x)$
γ_2 :	$i := j, x := G(x, v)$
Δ_1 :	$\Delta_1(x, v)$
Δ_2 :	$\Delta_1(x, x')$

Figure G.2: Proof of equivalence to BBM model

Recall that $V_M = \mathcal{V} \cup Q_M$ where $\mathcal{V}$ is the set of values that system parameters take, and Q_M is the set of state values. In M set $\mathcal{V} \stackrel{def}{=} \mathbb{R}$ and $Q_M \stackrel{def}{=} Loc \cup q0_M$. The initial state does not seem to occur explicitly in the description of hybrid automata, but we can add an explicit initial state $q0_M$ with an initialization transition associated with it.

Next let us to define the signal set $S_M = I_M \cup O_M \cup \{\xi\}$, $I_M \stackrel{def}{=} \{\}$ and $O_M = Var$.

Finally let us define the characteristic function χ_M. The characteristic function in M is used to describe two aspects of H — the activities and the transitions. Let us take activities first.

By assumption each activity in H can be represented as a trajectory, integrator function or dynamical system, so for each f call its corresponding representation f'. Then package up all the activities thus represented for each location $l \in Loc$ into an assignment α_l and place this on a looping transition. It just remains to predicate these with the appropriate invariants $Inv(l)$ and add the state transitions which gives the set:

$$\chi'_M \stackrel{def}{=} \{(\{(\xi, l)\} \cup Inv(l), \{(\xi, l)\} \cup \alpha_l) | \forall l \in Loc\}$$

In which each transition looks like:

$$l \xrightarrow{[Inv(l)]/\alpha_l} l$$

Next the transitions are defined, for each transition $e = (l, a, \mu, l')$ in Edg, let us define a transition of M.

$$\chi''_M \stackrel{def}{=} \{(x \cup \{(stsig, l)\}, y \cup \{(stsig, l')\}) | (x, y) \in \mu\}$$

Care must be taken in reassigning signal values on these transition and to avoid possible causality problems mirror signals should be used.

Then $\chi_M \stackrel{def}{=} \chi'_M \cup \chi''_M$ and we're done.

It should be reasonably straightforward to see that this construction will satisfy the required properties. $\square$

Notation

This insert summarizes the notation used, in the second part of the thesis. An index of technical terms is also available at the back for reference.

Flows

$\mathbf{x}$ ..... a flow point
$\prec$ ..... precedes (flow points)
$\preceq$ ..... precedes or equal to (flow points)
$\approx$ ..... equal to (flow points)
$\succeq$ ..... succeeds or equal to (flow points)
$\succ$ ..... succeeds (flow points)
$X = (F, \preceq, \theta)$ ..... a flow for which:
$\mathbf{x} \in X$ ..... $\mathbf{x} \in F$
$X \setminus F'$ ..... $(F \setminus F', \preceq, \theta)$
$|X|$ ..... $|F|$
$\|\mathbb{N}\|$ ..... natural number flow
$\|\mathbb{R}\|$ ..... real number flow
$Sub^{op\mathbf{x}}(X)$ ..... $(F', \preceq, \theta)$ for $F' \stackrel{def}{=} \{\mathbf{y} \in X | \mathbf{y}\ op\ \mathbf{x}\}$
$\overleftarrow{\mathbf{x}}$ ..... predecessor point $(end(Sub^{\prec x}(X)))$
$\overrightarrow{\mathbf{x}}$ ..... successor point $(start(Sub^{\succ x}(X)))$
$\equiv$ ..... equivalence
$\cong$ ..... equivalence almost everywhere
$\sup(X)$ ..... supremum of a flow X
$\inf(X)$ ..... infimum of a flow X
$start(X)$ ..... start point of a flow X
$end(X)$ ..... end point of a flow X
$\mathbf{k}^{\alpha|\beta}$ ..... divider point
$X \amalg_f Y$ ..... zip of X and Y w.r.t. f

Values and signals

$\perp$ ..... undefined value
$\mathcal{V}$ ..... a set of values
$A \downarrow B$ ..... $\{(x, y) \in A | x \notin B\}$
$\mathcal{V}^+$ ..... $\mathcal{V} \cup \{\perp\}$
$\mathcal{D}$ ..... a data set
$\mathcal{S}$ ..... set of all signals
$\mathcal{E}$ ..... set of all valuations
$\mathcal{E}_S$ ..... set of all valuations over the signal set S
$\ominus$ ..... metric over valuations
$\mathtt{s}$ ..... a signal
$\underset{\leftarrow}{s}$ ..... a current signal value
$\overset{\leftarrow}{s}$ ..... a previous signal value
$\overset{\leftarrow}{S}$ ..... previous signal set
$\underset{\leftarrow}{e}$ ..... current machine event
$\overset{\leftarrow}{e}$ ..... previous machine event
X ..... valuation X renamed to previous signals
$D(V)$ ..... domain of valuation V
$R(\chi)$ ..... realization of characteristic function χ
$\sqcup$ ..... union over relations
$\chi[V]$ ..... $\chi \sqcup \{(\{\}, V)\}$
$\overline{\mathtt{x}}$ ..... mirror signal
$\mathtt{x}^{\blacklozenge}$ ..... discrete shadow signal

Strings and Combs

$|\sigma|$ ..... length of string σ
σ_n ..... n^{th} character of σ
$\sigma[m]$ ..... string of first m^{th} characters of σ
σ^L ..... leftmost descent
σ^R ..... rightmost descent
$\mathtt{x}$ ..... flow point in a comb
$\langle\sigma\rangle$ ..... a point in a comb
$\langle\boldsymbol{\sigma}\rangle$ ..... a point at the limit of a comb σ
$\langle\sigma\rangle_{\mathtt{s}}$ ..... value of the signal $\mathtt{s}$ at $\langle\sigma\rangle$
$\langle\boldsymbol{\sigma}\rangle_{\mathtt{s}}$ ..... value of the signal $\mathtt{s}$ at $\langle\boldsymbol{\sigma}\rangle$
$\mathrm{Sp}(\langle\sigma\rangle)$ ..... the spread of a flow

Index

θ-continuous, **56**

accepting, 17
agent system, 4
automaton, 14

branching sequence, **75**

cardinality, 44
characteristic function
    standard, **104**
closed, 52
comb, **71**
    convergence, uniform, **73**
    limit, **72**
comb identifier, 71, **72**
connected, 43, **54**
continuous, **55**
Continuous I/O machine
    synchronous product, **106**
continuous I/O machine, 78
continuum, **55**
controller, 4
convergent delta bounds, **82**
countable, 44
cut, **54**

delta flow machine, **65**
    standard, 104
delta partition, **104**
dense, 43, **52**
descendants, **74**
descent, **72**
    leftmost, **72**
    rightmost, **72**
descent point, **72**
descent value, **72**
discrete shadowing signals, 100
divider point, **92**
    constrained sequence, **93**
dynamic, 30

environment, 4
equivalence flow, 95
equivalence, flow, **51**
equivalent almost everywhere, **95**
event, 27

fixed partition, **104**
flow, 27, **49**
    θ-continuous, **56**
    connected, **54**
    continuous, **55**
    dense, **52**
    end, 28, **52**
    equivalence, **51**
    equivalence, global, **52**
    forward, 52
    monotonic, strictly, 56
    montonic, **56**
    regular, 54
    start, 28, **52**
    structure, 27
    supremum, **56**
flow component, **87**
flow machine, **64**

generating, 17
global equivalence flow, **95**
globally equivalent almost everywhere, **95**
globally equivalent, **52**

hybrid automata, 123

interactive, 18
isotonic, **51**

metric, 28
metric dataset, 55

mirror signal, 98
monotonic, strictly, 56
montonic, **56**

nonequivalent points, 95

order preserving, **51**

plant, 4
proactive, 31

reach, **71**
reactive, 31
reactive systems, 13
root flow, 71

signals, 59
situated agent, 3
spread, **75**
spread, even, **75**
static, 30
string, 69
subflow, **53**
    contiguous, **53**
supremum, **56**
synchronous observer, 21
synchronous product, **64**
synchrony labelling, 28

temporal metric, 29
    external, 31
    internal, 31
temporal system, 27
transformational systems, 13

uncountable, 44

valuation, 50, **60**
    distance metric, 60

zipped, **57**