The formulation and evaluation of a neuron model based on biological neurons

Louwrence Daniël Erasmus Pr.Eng., M.Sc.

Thesis submitted for the degree Philosophiae Doctor in Electrical and Electronics Engineering at the Potchefstroom Campus of the North West University

Promoter: Prof. C.P. Bodenstein Pr.Eng.

2007

DIE FORMULERING EN EVALUERING VAN 'N NEURONMODEL GEBASEER OP BIOLOGIESE NEURONE

Opgedra aan Ronel, Malisa en Bernice.

Abstract

This thesis formulates and evaluates a mathematical model from an engineer's point of view based on the currently-known information-processing processes and structures of biological neurons. The specification and evaluation of the RealNeuron model form a baseline for current use in engineering solutions and future developments.

The RealNeuron is a carefully-reduced model that retains the essential features of more complex models. A systems engineering approach is used to formulate it, i.e. the model is described as using multiple resolution levels with configurable modular elements at each resolution level and is then implemented, verified and validated in a bottom-up method. It is computationally efficient and only adds or subtracts ion concentrations based on the states at the membrane structure's level. The results are integrated at the lower levels of resolution. The RealNeuron's simple calculations make simulations on personal computers possible by using standard spreadsheet software for a seven-neuron classical-conditioning neural circuit. All the simulated states at the highest level of resolution (i.e. pumps, channels, etc.), the intermediate levels of resolution (i.e. membrane potentials, neurotransmitters in the synapse, etc.) and the lowest level of resolution (i.e. conditioning signal, conditioned signal, conditioned reaction, etc.) are available on a spreadsheet.

The RealNeuron is verified in a bottom-up manner. The pumps, channels and receptors are verified first. These components are then integrated into the different membrane types (post-synaptic membrane, main membrane, axonal membrane) and verified while the membrane components are validated simultaneously. This process is repeated until individual neurons have been built up and RealNeuron networks have finally been constructed. The RealNeuron is verified and validated in configurations for AND, NAND, OR, NOR, NOT and XOR logic functions. It is also verified and validated by the implementation of classical conditioning.

In a noisy environment, the RealNeuron's performance is dependent on the pump's parameters in the main membrane of the sensor neurons.

This thesis proposes that a grade of machine intelligence is used to distinguish between the different synthesis requirements for intelligent machines.

An engineering synthesis of a RealNeuron network, based on classical conditioning, demonstrates how to implement a RealNeuron network that can be used in machines built to the grade of machine intelligence requirement which is classical-conditioning learning implemented with neural networks that can change learned associations in a dynamic environment.

Opsomming

Die proefskrif formuleer en evalueer 'n wiskundige model wat gebaseer is op huidige kennis oor biologiese neurone se inligtingprosesseringprosesse en -strukture. Dit is geformuleer vanuit die standpunt van 'n ingenieur. Die RealNeuronspesifikasie en evaluering vorm 'n basislyn vir huidige ingenieurstoepassings en toekomstige ontwikkeling.

Die RealNeuron is 'n sorgvuldig gereduseerde model wat die belangrike eienskappe van meer komplekse modelle behou. 'n Stelselsingenieurswesebenadering word gebruik om die RealNeuron te formuleer, d.w.s. die model word beskryf as 'n multiresolusie en modulêre model wat uit verskeie komponente in verskeie abstraksievlakke bestaan. Die RealNeuron word geïmplimenteer en geverifieer in 'n onder-na-bo aanslag. Dit gebruik min berekeninge deur slegs ioon konsentrasies te verhoog of te verlaag, gebaseer op die toestande van die membraanstruktuurvlak. Hierdie resultate word op laer vlakke van resolusie geïntegreer. Die RealNeuron se eenvoudige berekeninge maak dit moontlik om 'n simulasie vir 'n sewe neuron klassieke konditioneringsneuralebaan met 'n standaard sigbladprogram op 'n skootrekenaar vinnig uit te voer. Al die gesimuleerde toestande in die hoogste resolusievlak, d.w.s. pompe, kanale, ens., tussen resolusievlakke, d.w.s. membraanpotensiale, oordragstowwe in die sinapse, ens., en laagste resolusievlak, d.w.s. kondisioneringsein, ongekondisioneerde-sein, gekondisioneerde-reaksie, is in die sigblad sigbaar.

Die RealNeuron word geverifieer met 'n onder-na-bo aanslag. Eerste word die pompe, kanale en metaboliese reseptore geverifieer. Hierdie komponente word dan geïntegreer in die verskillende membraantipes (postsinapsmembraan, hoofmembraan, aksonmembraan) en terselfde tyd word die membraankomponente gevalideer. Die proses word herhaal totdat individuele neurone opgebou is en uiteindelik 'n RealNeuronnetwerk. Die RealNeuron word geverifieer in EN-, NEN-, OF-, NOF-, NlE-, en EOF-logiese funksiekonfigurasies. Dit word ook geverifieer en -valideer in 'n implementering van klassieke konditionering.

Die RealNeuron se ruisprestasie is afhanklik van die pompparameters in die hoofmembraan van die sensorneuron.

Hierdie proefskrif stel voor dat 'n graad van masjienintelligensie gebruik word om te onderskei tussen die verskillende sintesevereistes vir intelligente masjiene. Verder word die graad van masjienintelligensie gestel wat vereis dat 'n masjien veranderdende-assosiasies in 'n dinamiese-omgewing moet kan leer deur middel van neuralenetwerkgeïmplimenteerde klassieke konditionering.

'n Ingenieurssintese demonstreer hoe die bostaande graad van masjienintelligensie deur 'n RealNeuronnetwerk geïmplimenteer kan word.

Acknowledgements

Like the research for my M.Sc. studies [52], this research was like a Star Trek adventure; a voyage into the abstract spaces of the unknown. What made this voyage so amazing was the fact that it was a voyage into the inner workings of the brain.

Never before have I undertaken something so challenging and so enriching as this research. Apart from the academic and technical knowledge I have also gained spiritual knowledge and enrichment.

I would like to express my gratitude to Prof. Charles Bodenstein for the interest he took in this research. His guidance and encouragement was an inspiration. As in my undergraduate studies he gave me just enough guidance to keep me interested, but left the thinking to me. We spent many hours together to philosophise about the deeper meaning of the research and its implications to society.

I would like to thank Prof. Dr. Gerd Döben-Henisch of the University of Applied Sciences, Frankfurt am Main, Germany, for his friendship and support. I joined his team at inm-magic GmbH in April 1999. We spent many hours together in Germany and South Africa talking about philosophy, philosophy of science, semiotics, computational semiotics, computational neuro-semiotics, theology, science, engineering, etc. All his groundbreaking work with the Knowbots, INM-neuron and later the RealNeuron, made my research possible. His constructive criticism inspired me to complete the work on the RealNeuron.

I would like to thank Dr. Kobus Myburgh, of the N.G. Kerk Wierdapark, for the hours of discussion on my very deep ethical and theological dilemmas that arose from this research.

I would like to thank Dr. Joachim Hasebrook, Bankakademie e.V. and Efiport A.G., Frankfurt am Main, Germany, for his friendship and the sharing of his knowledge of psychology and the human brain. His creative contributions, every Wednesday night at Knowbotic Systems on the potential applications of Knowbots and RealNeurons gave direction and meaning to these technologies.

I would like to thank the Knowbotic Systems team in Germany that I worked with and learned the Knowbot and RealNeuron technologies from.

I would like to thank Dr. Etienne de Villiers for the time he spent in doing the editing for me. He helped me to complete the last mile.

I would like to thank Ms. Helen Thorne for the time she spent in doing the linguistic editing for me. She helped me to complete the last few yards.

I would like to thank Mr. Rocco van Schalkwyk for taking the time to talk to me about brain modelling and exploring the possibilities of combining his brain model with the RealNeuron. He helped me to express the last complex ideas in simple words.

I am thankful to all my teachers, mentors, colleagues and friends who have a part in forming me into the engineer I am today.

I am very grateful to my parents for their love and support through all these years of study. Only after having children of my own, can I fully appreciate the sacrifices made by them throughout the years.

My sincerest thanks go to my wife Ronel, and daughters Malisa and Bernice for their love, support, encouragement, sacrifices and understanding throughout my research. They are my spiritual support team for big projects.

Everyday I experience the omnipresence of *Elohim*. *He* looks after me, guides me, and gives me the inner strength to continue my work; *Sola Gratia*. *He* lovingly reveals his own workings and works to me. *He* allows me to ask the difficult questions, to challenge current beliefs and to enrich my own personal relationship with *Him*; *Soli Deo Gloria*.

Centurion
South Africa
2007

Contents

Abstract i

Opsomming iii

Acknowledgements v

Contents vii

List of Figures xv

List of Tables xvii

1 Introduction 1

1.1 Switch to the Sci-Fi channel 1
1.2 Artificial intelligence 2
1.3 Biological intelligence 4
1.4 Neuroscience 5
1.5 Engineering applications of neuroscience 5
1.6 History of the RealNeuron 6
1.7 Purpose of the research 8
1.7.1 The research context 8
1.7.1.1 The scientific way of life 8
1.7.1.2 The engineering way of life 9
1.7.2 Scope of investigation 9
1.7.3 Assumptions 11
1.8 Layout of thesis 12

2 Background 15

2.1 What is intelligence? 15
2.2 Intellifacts 17
2.3 Machine intelligence 18
2.3.1 Artificial intelligence 18
2.3.2 Computational intelligence 19

2.3.2.1 Artificial neuron models 19
2.3.2.2 NIST models 21
2.4 Neural network grade 22
2.4.1 Learning 22
2.4.2 Thinking 23
2.5 Neuroscience 24
2.5.1 Computational neuroscience 25
2.6 Summary 27

3 RealNeuron modelling **29**
3.1 Formal theory 29
3.1.1 Systems, environments and bodies 31
3.2 Formal system breakdown structure 32
3.3 From biological neuron to RealNeuron 32
3.3.1 Action potential 38
3.3.2 The synapse 38
3.3.3 Electrochemistry 40
3.3.4 Nernst equation 40
3.3.5 Goldmann equation 41
3.4 State space theory 42
3.4.1 Continuous systems 42
3.4.2 Discrete systems 43
3.5 RealNeuron notation 44
3.6 Summary 44

4 RealNeuron models **47**
4.1 Different types of models 47
4.1.1 Predicate model 48
4.1.2 Block diagram model 48
4.1.3 State space model 48
4.2 Predicate model 49
4.2.1 RealNeuron network 49
4.2.2 RealNeuron 49
4.2.3 Soma 50
4.2.4 Axon 50
4.2.5 Bulb end 51
4.2.6 Membranes 51
4.2.7 Ion channels 52
4.2.8 Ion pumps 53
4.2.9 Metabolic receptors 53

4.3 Message processing 53
4.3.1 RealNeuron message processing 54
4.3.2 RealNeuron 54
4.3.3 Soma 55
4.3.3.1 Input soma processing 55
4.3.3.2 Bulb-end soma processing 55
4.3.4 Axon 56
4.3.5 Bulb end 56
4.3.6 Membranes 56
4.3.7 Ion channels 58
4.3.8 Ion pumps 59
4.3.9 Long-term potential function 59
4.4 State space and block diagram model 59
4.4.1 RealNeuron network 60
4.4.2 RealNeuron 64
4.4.2.1 Sensor neuron 66
4.4.2.2 Internal neuron 68
4.4.2.3 Motor neuron 68
4.4.3 Soma 70
4.4.3.1 Input soma 71
4.4.3.2 Sensoric soma 72
4.4.3.3 Bulb-end soma 74
4.4.4 Axon 78
4.4.5 Bulb end 79
4.4.6 Membrane 81
4.4.6.1 Post-synaptic membrane 82
4.4.6.2 Axonal membrane 83
4.4.6.3 Main membrane 85
4.4.7 Long-term potential function 88
4.4.8 Integration function 88
4.4.9 Vesicle 89
4.4.10 Nernst and Goldmann equations 90
4.4.11 Inverse Nernst and Goldmann equations 91
4.4.12 Membrane pores 92
4.4.12.1 Ion pump 92
4.4.12.2 Ion channels 93
4.4.13 Receptor 95
4.5 Memory 96
4.6 Summary 96
4.6.1 RealNeuron network summary 96

5 Theoretical evaluation of the RealNeuron 99
5.1 General mathematical relation 99
5.1.1 RealNeuron related to pulsed neurons 101
5.1.2 RealNeuron relation to McCulloch-Pitts model 102
5.2 Performance in a noisy environment 103
5.3 Summary 105

6 Implementation and evaluation 107
6.1 Spreadsheet implementation 107
6.2 C++ implementation 107
6.3 Verification and validation 108
6.4 Reference parameters 109
6.5 Reference input signal 109
6.6 Mass balance 110
6.7 Membranes 110
6.7.1 Main membrane 110
6.7.1.1 Expected results of the main membrane 111
6.7.1.2 Results of the main membrane 111
6.7.2 Post-synaptic membrane 112
6.7.2.1 Expected results of the post-synaptic membrane . . 112
6.7.2.2 Results of the post-synaptic membrane 113
6.8 RealNeuron sub-components 113
6.8.1 Input soma 113
6.8.1.1 Expected results of the input soma 114
6.8.1.2 Results of the input soma 114
6.8.2 Bulb end 114
6.8.2.1 Expected results of the bulb end 114
6.8.2.2 Results of the bulb end 114
6.9 RealNeuron examples 118
6.9.1 Sensor neuron 118
6.9.1.1 Expected results of the sensor neuron 118
6.9.1.2 Results of the sensor neuron 118
6.9.2 Motor neuron 119
6.9.2.1 Expected results of the motor neuron 119
6.9.2.2 Results of the motor neuron 119
6.9.3 Purkinje cell 119
6.9.3.1 Expected results of the Purkinje cell 120
6.9.3.2 Results of the Purkinje cell 122
6.9.4 Nucleus interpositus cell 124
6.9.4.1 Expected results of the nucleus interpositus cell . . 124

6.9.4.2 Results of the nucleus interpositus cell 125
6.10 Logic function implementation 126
6.10.1 BUFFER function 126
6.10.1.1 SBS of the BUFFER network 126
6.10.1.2 Connections of the BUFFER network 128
6.10.1.3 Expected results of the BUFFER function 128
6.10.1.4 Results of the BUFFER function 128
6.10.2 INVERTER function 130
6.10.2.1 SBS of the INVERTER network 130
6.10.2.2 Connections of the INVERTER network 130
6.10.2.3 Expected results of the INVERTER network 131
6.10.2.4 Results of the INVERTER network 132
6.10.3 AND function 132
6.10.3.1 SBS of the AND network 132
6.10.3.2 Connections of the AND network 133
6.10.3.3 Expected results of the AND network 134
6.10.3.4 Results of the AND network 134
6.10.4 NAND function 136
6.10.4.1 SBS of the NAND network 136
6.10.4.2 Connections of the NAND network 136
6.10.4.3 Expected results of the NAND network 137
6.10.4.4 Results of the NAND network 137
6.10.5 OR function 139
6.10.5.1 SBS of the OR network 139
6.10.5.2 Connections of the OR network 139
6.10.5.3 Expected results of the OR network 140
6.10.5.4 Results of the OR network 140
6.10.6 NOR function 142
6.10.6.1 SBS of the NOR network 142
6.10.6.2 Connections of the NOR network 142
6.10.6.3 Expected results of the NOR network 143
6.10.6.4 Results of the NOR network 143
6.10.7 XOR function 145
6.10.7.1 SBS of the XOR network 145
6.10.7.2 Connections of the XOR network 145
6.10.7.3 Expected results of the XOR network 146
6.10.7.4 Results of the XOR network 146
6.11 Classical conditioning 148
6.11.1 SBS for the emulation of classical conditioning 148
6.11.2 Connections for classical conditioning 149

6.11.3 Expected results for classical conditioning 150
6.11.4 Results of the classical conditioning network 151
6.12 An engineering synthesis using the RealNeuron 152
6.12.1 The poison and food game 152
6.12.1.1 Behavioural description 153
6.12.1.2 Solution with RealNeuron technology 153
6.12.1.3 Kinematic description 155
6.12.1.4 Results 155
6.13 Reflecting on the results 157
6.13.1 Reflecting on the membranes 157
6.13.2 Reflecting on the RealNeuron sub-components 158
6.13.3 Reflecting on the sensor neuron 158
6.13.4 Reflecting on the motor neuron 158
6.13.5 Reflecting on the Purkinje cell 158
6.13.6 Reflecting on the nucleus interpositus cell 159
6.13.7 Reflecting on the logic functions 159
6.13.8 Reflecting on classical conditioning 159
6.13.9 Reflecting on the poison and food game 159
6.14 Summary 159

7 Conclusion **161**
7.1 Formulation of a neuron model 161
7.2 Evaluation of a neuron model 162
7.3 Future work 163
7.3.1 Long-term memory 163
7.3.2 Simpler and closer to biology 163
7.3.3 Repeating the Hodgkin-Huxley results 164
7.4 Final Thought 164

A Glossary **165**
A.1 Symbology 165
A.2 Abbreviations and Acronyms 167

B Disclosure of research **169**
B.1 Presentations 169
B.2 Publications 170
B.3 Publication in book series 170
B.4 Citations of work in other publications 170
B.5 Recognition of contributions in other publications 170
B.6 Future disclosures 171

C XML Description of the RealNeuron **173**
C.1 DTD 173

D Implementations examples **177**
D.1 C++ Code 177
D.1.1 `species.h` 177
D.1.2 `RNNException.h` 177
D.1.3 `pores.h` 177
D.1.4 `pores.cc` 178
D.1.5 `pumps.h` 178
D.1.6 `pumps.cc` 178
D.2 C++ Unit Test Code 178
D.2.1 `pumpstest.h` 178
D.2.2 `pumpstest.cc` 179

E Motivation for the bulb-end soma **181**

F Simulation Implementation Notes **183**
F.1 Implementing the calculating parts 183
F.2 The nucleus interpositus cell's implementation 184
F.3 The Purkinje cell's implementation 185
F.4 General notes 185

G CD-ROM Contents **187**

Bibliography **188**

List of Figures

1.1 3D visualisation of RealNeuron 2000-prototype 7
1.2 Scientific and engineering framework 8

2.1 Biological neuron 20
2.2 Artificial neuron 20

3.1 Context diagram for a body 33
3.2 Context diagram for a RealNeuron network 33
3.3 Context diagram for RealNeurons 33
3.4 Biological neural network 34
3.5 Biological pyramidal neurons 34
3.6 Purkinje and granule cells 35
3.7 Components of the RealNeuron 36
3.8 RealNeuron Components 37
3.9 The synapse 39
3.10 An intelligent system overview 45

4.1 RealNeuron network block diagram 60
4.2 Generic RealNeuron block diagram 64
4.3 Sensor neuron block diagram 67
4.4 Motor neuron block diagram 69
4.5 Input soma block diagram 71
4.6 Sensoric soma block diagram 74
4.7 Bulb-end soma block diagram 75
4.8 Axon block diagram 78
4.9 Bulb-end block diagram 80
4.10 Post-synaptic membrane block diagram 83
4.11 Axonal membrane block diagram 84
4.12 Main membrane block diagram 86

6.1 RealNeuron UML class diagram 108
6.2 Results of the main membrane implementation 111

6.3 Post-synaptic membrane block diagram 112
6.4 Results of the post-synaptic membrane 113
6.5 Implemented input soma block diagram 113
6.6 Results of the input soma implementation. 115
6.7 Results of the input soma implementation (continued) 116
6.8 Results of the bulb end 117
6.9 Results of the sensor neuron 119
6.10 Results of the motor neuron 120
6.11 Logic circuit for Purkinje cell 122
6.12 Perceptron neural network for Purkinje cell 122
6.13 Logic circuit for nucleus interpositus cell 124
6.14 A perceptron implementation of the nucleus interpositus cell 124
6.15 BUFFER RealNeuron network 126
6.16 A perceptron implementation of the BUFFER function 128
6.17 Results of the BUFFER RealNeuron network 129
6.18 INVERTER RealNeuron network 130
6.19 A perceptron implementation of the INVERTER function 132
6.20 Results of the INVERTER RealNeuron network 132
6.21 AND RealNeuron network 133
6.22 A perceptron implementation of the AND function 135
6.23 Results of the AND RealNeuron network 135
6.24 NAND RealNeuron network 136
6.25 A perceptron implementation of the NAND function 137
6.26 Results of the NAND RealNeuron network 138
6.27 OR RealNeuron network 139
6.28 A perceptron implementation of the OR function 140
6.29 Results of the OR RealNeuron network 141
6.30 NOR RealNeuron network 142
6.31 A perceptron implementation of the NOR function 143
6.32 Results of the NOR RealNeuron network 144
6.33 XOR RealNeuron network 145
6.34 A perceptron implementation of the XOR function 146
6.35 Results of the XOR RealNeuron network 147
6.36 RealNeuron circuit representing the blinking of a rabbit's eye 148
6.37 Digital logic functions for classical conditioning 150
6.38 Functional identification of N_5 and N_6 151
6.39 Results of classical conditioning RealNeuron network 151
6.40 Hardware and brain for the poison and food game 153
6.41 Results of the Poison and Food Simulation 156

List of Tables

4.1 Parameters and states for a generic RealNeuron network 63
4.2 Parameters and states for the generic RealNeuron 66
4.3 Parameters and states for the sensor neuron 68
4.4 Parameters and states for the neuron 69
4.5 Parameters and states for the motor neuron 70
4.6 Parameters and states for the input soma 73
4.7 Parameters and states for the sensoric soma 75
4.8 Parameters and states for the bulb-end soma 77
4.9 Parameters and states for the axon 80
4.10 Parameters and states for the bulb end 81
4.11 Parameters and states for the post-synaptic membrane 84
4.12 Parameters and states for the axonal membrane 86
4.13 Parameters and states for the main membrane 87
4.14 Summary of RealNeuron network components 97

6.1 Membrane reference parameters 110
6.2 Expected results for the sensor neuron 118
6.3 Expected results for the Purkinje cell 121
6.4 Results of the Purkinje cell 123
6.5 Expected results of nucleus interpositus cell 124
6.6 Results of the nucleus interpositus cell 125
6.7 Expected results of the BUFFER function 129
6.8 Expected results of the INVERTER function 131
6.9 Expected results of the AND function 134
6.10 Expected results of the NAND function 138
6.11 Expected results for the OR function 141
6.12 Expected results for the NOR function 144
6.13 Expected results for the XOR function 147
6.14 Digital logic functions to implement classical conditioning 150

Chapter 1

Introduction

> *If man will begin with certainties, he shall end in doubts, but if he will be content to begin with doubts, he shall end in certainties* — Francis Bacon *The Advancement of Learning* (1605)

The biological brain is one of the most complex systems known to exist in the universe. Judging by the exhibits found in science and technology museums worldwide, the topic of brain modelling and machines that emulate them has been a fascinating one for centuries. Why the brain should endeavour to model itself is also a potential subject for a philosophical study, but it falls outside the scope of this thesis.

The information-processing elements in a biological brain are neurons. The neuron is a complex system in itself. Philosophers and various scientific disciplines study the biological brain to try and explain its working.

This thesis formulates and evaluates a neuron model based on biological neurons from an engineer's point of view. It is not trying to explain the working of a biological neuron, but to specify a neuron model or RealNeuron for engineering applications that is based on the current knowledge available form various disciplines.

It is an exercise in complex systems modelling and simulation. A systems engineering approach is used to build the RealNeuron, i.e. the model is described (specified) as using multiple resolution levels with configurable modular elements at each resolution level and is then implemented, verified and validated in a bottom-up method. The elements on a specific level of resolution is integrated as a new element on the next lower level of resolution; this is repeated until the system has been integrated.

The RealNeuron is evaluated in various neural network configurations with predictable outcomes. Although other methods, e.g. rule-based methods or perceptron, can also be used to demonstrate the functions of these configurations, the purpose in this thesis is to show that a RealNeuron network can do that as well.

The specification and evaluation of the RealNeuron model form a baseline for current use in engineering solutions and future developments.

1.1 Switch to the Sci-Fi channel

In her novel, Fool's War [160], Sarah Zettel gave a description of artificial intelligence life forms or AI agents, based on artificial neural networks, called Fools. These AI agents were in alliance with the human race. They protected humans in cyberspace from outside intrusions, malicious virile agents, and themselves.

The AI agents found a way of entering human bodies that had been created through tissue engineering. They used these bodies to interact and move amongst humans.

Although the humans denied the existence of AI agents, they knew the Fools Order was a group of very special "people" that played the fool when things were getting too serious. On many occasions they defused serious arguments that might have led to deadly wars.

Is it possible to construct Fools (AI agents) in some form or other? Or is this just a "fools dream?"

1.2 Artificial intelligence

In 1947 Turing delivered the first public lecture to mention computer intelligence in London [23], saying:

> *What we want is a machine that can learn from experience ... [and] ... [t]he possibility of letting the machine alter its own instructions provides the mechanism for this* [151].

It is regarded in [23] that he published the first manifesto for Artificial Intelligence [150] in 1948. He mainly discussed machine learning and describes experiments on the modification of an initially unorganised machine by a process that looks like teaching by reward and punishment. This paper also included the following concepts:

- Theorem-proving approach to problem solving.
- The hypothesis that intellectual activity consists mainly of various kinds of searching.
- The genetic algorithm.
- The anticipation of connectionism (neural networks).
- The training of artificial neural networks to perform specific tasks.

In 1950 Turing published his famous imitation game [149] also known today as the Turing Test. In this paper he referred to the unorganised machine as a "child-machine", and he had:

> *... succeeded in teaching it a few things, but the teaching method was too unorthodox for the experiment to be considered successful* [149].

He also gave a whole description on how to implement this "child-machine" with a Turing Machine.

Turing's paper [149] was written in an inappropriate style for the original audience of philosophers which led to misunderstanding [23]. Unfortunately Turing died early in life and could not clear his many critics' misunderstanding of [149] context [133, 23, 117]. A balanced compilation of several positions on the Turing Test is given in [117]. For this section the context given in [23] of Turing's work, based on Turing's published and earlier unpublished work, is used as basis.

In reading Turing's work it becomes clear that he was not just a theoretical academic, but a practical engineer who also proposed implementations for some of his

theories. He had provided the first specifications for a computer's electronic circuits and code examples in history, including the cost to build such a machine [152].

When Turing was asked to give a definition of thinking (intelligence) in a BBC radio broadcast entitled "Can Automatic Calculating Machines Be Said To Think", recorded in January 1952 [24], he replied:

> *I don't want to give a definition of thinking but if I have to I should probably be unable to say more than that it was a sort of buzzing that went on inside my head. But I don't really see that we need to agree on a definition at all.*

Thus, Turing did not offer a definition of intelligence (thinking) [23] and in his words machines may:

> *... carry out something which ought to be described as thinking but which is very different from what man does* [149].

In other words, a machine may be deemed intelligent if it can behave in such a way that a human cannot distinguish the machine from another human being.

The purpose of the Turing test is best described as:

> *The real value of the imitation game lies not in treating it as the basis for an operational definition but in considering it as a potential source of good inductive evidence for the hypothesis that machines think* [116].

The imitation game can be replaced by any human activity involving pure intellectual activity of which playing chess is one such activity [150, 149]:

> *We may hope that machines will eventually compete with men in all purely intellectual fields. ... many people think that a very abstract activity, like the playing of chess, would be best* [149].

Turing described a restricted form of the imitation game that he had actually done [150]. This form of the imitation game was restricted to the playing of chess. In January 1952, in the above mentioned BBC radio broadcast, Turing said the following about the imitation game:

> *I am not saying at present either that machines really could pass the test, or they couldn't. My suggestion is just that this is the question we should discuss. It's not the same as "Do machines think," but it seems near enough for our present purpose, and raises much the same difficulties* [24].

Garry Kasparov has drawn against chess playing computers several times. He has even lost a couple of times. These machines from IBM compete with human intelligence in the context of playing chess. These machines still do not think, i.e. are intelligent, but a great deal on how to make machines do planning and do what-if analyses, sometimes very different from what a human does, has been discovered (see §2.1 on p. 15 on the part that planning contributes to intelligence.) The other purely intellectual fields for the imitation game and the machine's capability in using natural language are excluded from playing chess against IBM's chess-playing machines.

The Turing Test, especially the interpertation of it as an operational definition, contributed to the rise of the classical artificial-intelligence research field. Most of the algorithms that mimic intelligent behaviour are based on logically-orientated knowledge-based techniques [85] and are too inflexible to represent the primitive forms of learning [64, 65]. They lead to a number of behavioural paradoxes when used to support human learning [66]. Most of these techniques also require huge databases of captured knowledge to function satisfactorily.

The general idea that machines should be able to learn, modifying their behaviour with experience, was a vital part of Turing's thought process on machine intelligence [151, 150, 149, 23]. This idea was largely ignored by the classical artificial intelligence research field and revived with the rise of computational intelligence (which include neural networks, genetic algorithms and fuzzy logic.) The way that humans judge whether a machine, or any mind, is intelligent is by obtaining behavioural evidence [133]. Thus it can be concluded:

Postulate 1 *An intelligent machine should be able to alter its behaviour by learning from experience and altering its own configuration based on a series of rewards and punishments.*

1.3 Biological intelligence

People do not have huge databases of formally encoded knowledge as required in classical artificial intelligence. They capture knowledge in a distributed way through neural networks in the brain. If the brain is damaged its learning ability or some other performance is decreased, yet such a brain could still function satisfactorily [13]. If a part of a database is damaged the captured knowledge in that part is lost.

A human or biological agent receives sensory input from its environment through its senses that translate the input into nervous-system signals [13, 41, 56]. The brain is an electro-chemical machine that processes vast amounts of information that enters it as nervous-system signals. It builds a biological agent's reality based on sensory stimuli and internal states [34, 33, 56, 134]. It also evaluates this created reality and creates abstract realities with new meanings [56, 134]. These evaluations are done to satisfy the biological agent's needs which have been identified [106] as bodily needs, security needs, egotistical needs, social needs, and self-actualisation.

The ability of the brain to construct and evaluate a biological agent-based reality can be understood as the mechanism for consciousness and confirms Crick's hypothesis[1] [25]:

Postulate 2 (Crick's consciousness hypothesis) *A true understanding of consciousness cannot be achieved by treating the brain as a black box. Scientists could accumulate the kind of knowledge required to create a scientific model of consciousness only by examining neurons and the internal structure between them.*

The capability of biological agents to deduce appropriate responses for themselves are generally understood as intelligent behaviour [4, 56, 134]. The capabilities of the brain are a direct result of how the neurons inside the nervous system are organised and how they are integrated with a biological agent's body [13, 41, 28, 56].

Biological neurons are the information-processing elements in the brain. A neuron releases neurotransmitters at a bulb end based on the latter's internal potential

[1] Francis Crick and James Watson received the Nobel Prize for physiology or medicine in 1962 for their work on the DNA's (deoxyribonucleic acid) double-helix structure.

[41, 125, 137, 158]. The concentration of ions inside and outside a neuron membrane determines the internal potential of the membrane [28, 41, 73, 72, 125, 137, 158]. The neurotransmitters around a post-synaptic membrane influence the flow of ions through the membrane [41, 125, 137, 158]. It is reasonable to assume

Postulate 3 *Neurons are the building blocks for intelligence inside a biological agent.*

1.4 Neuroscience

The field of neuroscience studies the underlying behavioural structures of intelligence. The physiological structure of the biological brain forms the physical basis of the basic ability of observable behaviour [34]. Although neurobiological facts cannot be interpreted without a link to the actual behaviour or subjective experiences, they do represent the core of our knowledge about the mechanisms underlying observable behaviour [64, 65].

When studying a multi-disciplinary field like neuroscience, one is confronted with various paradigms on the subject. According to [13], neuroscience is a general field of science that studies the nervous-system, and contributors to this field are neurology, neurosurgery, psychiatry, anatomy, physiology, biochemistry, and psychology.

During the research for this thesis, it was found that other disciplines also contribute actively to this field. Today the brain is studied by scientific disciplines ranging from purely empirical to highly theoretical sciences. They include, amongst others, philosophy, engineering, computer science, and cognitive science.

In computer science and engineering, the fields that contribute most to neuroscience are robotics and computational intelligence (neural networks, fuzzy logic, and evolutionary algorithms).

The simulation of biological neural networks based on Hodgkin-Huxley and compartmental models, developed by neuroscientists, does not seem to be promising for engineering applications. Attempts have shown that simulations of brain cells consume a vast amount of computer resources, for instance 18.2h to simulate one second of activity in a neuron [60]. This is not generally suitable for engineering applications if such excessive time and computational resources are considered to implement only one neuron.

The spiking neuron models are simplified models of the more complex models. They model only the macroscopic behaviour of biological neurons [84]. Huge networks can be built using these types of neurons [83]. These models require the solving of differential equations which are, in many cases, non-linear [104, 84, 83]. During the simplification process the structure of biological neurons is lost.

1.5 Engineering applications of neuroscience

In engineering, the human brain is also a fascinating subject to study. The understanding of the nervous system, from an engineering perspective [28], is necessary if engineers design and build measuring devices for use in brain research and medical diagnoses. Further, if the nervous system function is understood in engineering terms, machines could be built to perform similar functions.

The development of the field of neural networks has exploded in the last 20 years. The special issues on neural network applications of the Institute of Electrical and

Electronics Engineers, Inc. (IEEE) and other engineering organisations, published on a regular basis, are testimony to this fact. The IEEE publishes the Transactions of Neural Networks bimonthly. Several other international journals are also published. Yearly, there are dozens of international conferences covering topics on neural networks.

Today, machines implement simplified structures based on the brain. One such structure is the simple perceptron that has been implemented in some engineering applications [12, 42, 57, 90, 93, 120, 122]. In many branches of electrical, electronics and computer engineering, someone is endeavouring to apply neural networks in problem solving with varying degrees of success.

Artificial neural networks have achieved some success in non-linear forecasting, pattern matching and in artificial life paradigms [42, 89, 90, 157]. An artificial neural network exhibits the capability of redundancy and vague representation of information. This is in contrast to classical artificial intelligence which requires crisp knowledge encoded into well-defined structures [85].

However, artificial neural networks still lack many of the vital features of biological neural networks, such as the latter's ability to allow self-modification with regard to both short-term and long-term learning. In the words of [20]:

> *Artificial neural networks do indeed employ some of the same fundamental processes used in biological networks, such as changes in connection strengths, thus supporting the proposed similarity between the two systems. However, biological networks also exhibit several additional forms of plasticity which have received less attention in network models, including changes in neuronal excitability; changes in the fidelity (as well as strength) of synaptic transmission; changes in signal-to-noise ratios; changes in type of neurotransmitter synthesized and released; and changes in neuron number. The richness of plastic mechanisms found in biological neurons suggests there may be a number of effective computational tricks used by real nervous-systems that could be advantageously incorporated into artificial neural networks* [20].

1.6 History of the RealNeuron

This thesis is a further step in the RealNeuron's[2] evolutionary history. The block diagrams and state space models are new contributions. The set predicate models have been completed, and the terminology and symbols have been standardised.

The RealNeuron started off as the INM-neuron and was conceptualised as part of the Knowbotic Interface Project, in the Institut für Neue Medien e.V., Frankfurt am Main, Germany (INM), 1994-1998, see [38]. The goal of the Knowbotic Interface Project was to build the concept of a totally new type of artificial intelligence.

The first version of an INM-neuron (also referred to as the 1998-prototype) was demonstrated as part of a classical conditioning experiment using the model of the eye reflex of a rabbit [109] in 1998. The formulation was a predicate model written in Prolog.

The 1998-prototype is deduced from the brain cell [51, 145], which models the main features of biological neural cells that are considered to be the key features of learn-

[2]RealNeuron was a registered trademark of Knowbotic Systems GmbH & Co.KG, Frankfurt am Main, Germany. Knowbotic Systems was liquidated in 2001.

ing. The INM-neuron is empirically more sound than the classical artificial neural networks and the simulation runs in real time or faster on a personal computer.

From November 1998 to November 1999 the development of the INM-neuron continued in the Learning Technologies Department of the inm-magic GmbH, Frankfurt am Main, Germany[3]. Then in December 1999, the Learning Technologies Department was transformed into the company Knowbotic Systems GmbH & Co KG, Frankfurt am Main, Germany.

In March 2000 a new prototype of the INM-neuron was deduced and named the RealNeuron (also referred to as the 2000-prototype.) The 2000-prototype was implemented using Java [145]. It is implemented with a simulator which can be used to construct RealNeuron networks. This version of the RealNeuron was formulated as a combination between algorithmic descriptions [145, 144] and incomplete set predicate models [30, 36, 32].

Figure 1.1 shows a POVRAY[4]-rendered picture of a possible visualisation of the 2000-prototype.

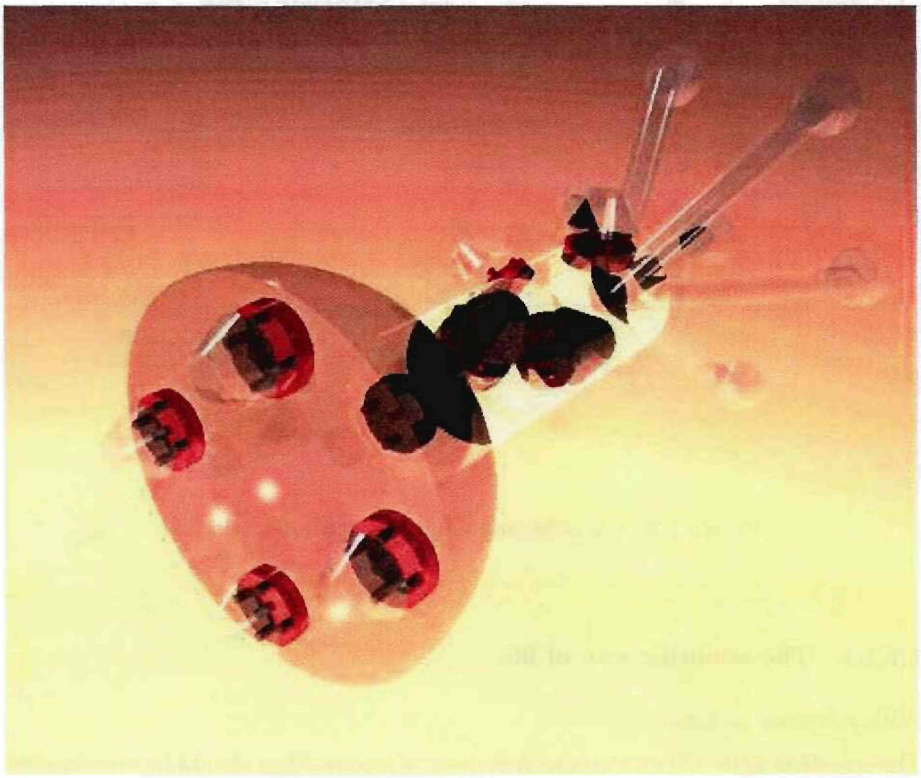


Figure 1.1: 3D visualisation of RealNeuron 2000-prototype

[3]The author joined inm-magic in April 1999 as a telecommuter from South Africa and from mid-May 1999 to August 2001 as a full-time team member in Frankfurt am Main.

[4]The Persistence of Vision Raytracer (POVRAY) is available from http://www.povray.org

1.7 Purpose of the research

1.7.1 The research context

This thesis is presented from an engineer's point of view, based on an engineering method [77, 81, 11]. It is not trying to explain scientific phenomena, but use scientific results in synthesising specifications to satisfy stated requirements.

The scientific method as described in the theory of science [37, 9, 8, 146, 101] might look very similar to the engineering method, but there is a difference between the objectives and approach in the two methods. Figure 1.2 shows the assumptions of the basic elements of a scientific and engineering discipline. The scientists and engineers communicate by using an agreed language (§1.7.1.1 and §1.7.1.2 is deduced from the overview on the scientific and engineering way of life in [37, 35].).

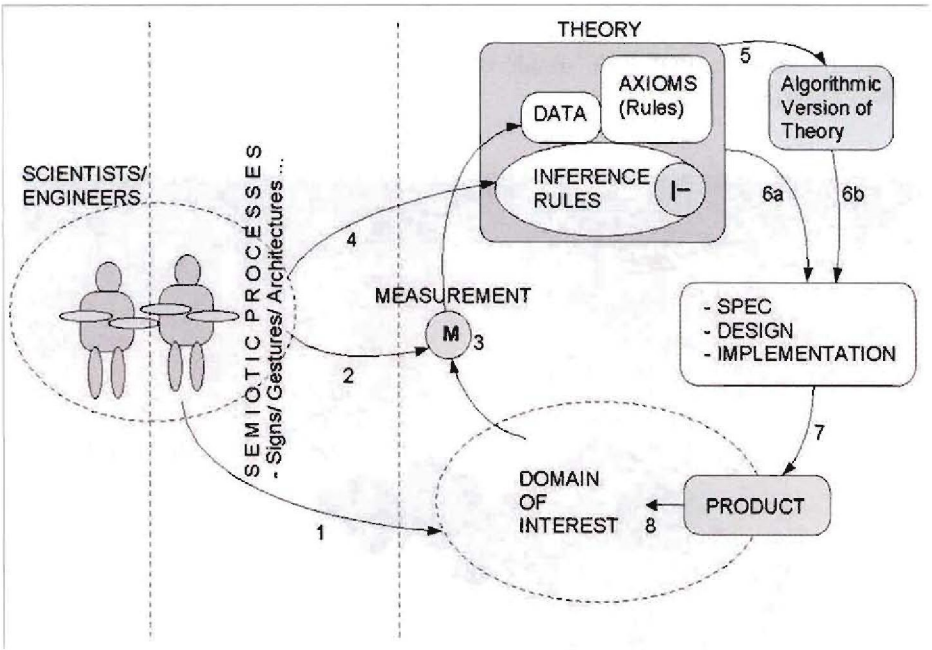


Figure 1.2: Scientific and engineering framework, [35]

1.7.1.1 The scientific way of life

With reference to figure 1.2:

The scientists agree (1) on a common domain of interest that should be investigated. They agree on procedures (2) to measure (3) the properties of the domain of interest. The measurements are data or facts which put events in a specific time, space and context.

The scientists generate (4) systematic concepts, schemata, models, formal structures, etc. which are used to relate the data into more general functional contexts so that explanation, forecast, and hypothesis testing can be done. The ideal case would be a formal structure connected to a formal inference concept. They translate parts or all of the general formal theories into algorithms (5).

Definition 1 (Pure theory) *The formal structure and inference concept is called a pure theory, e.g. mathematical theory.*

Definition 2 (Theory or descriptive theory) *A pure theory together with data is a theory or a descriptive theory.*

1.7.1.2 The engineering way of life

With reference to figure 1.2:

The engineers agree (1) on a certain problem that should be solved within a certain domain of interest. They agree on how to generate systematic concepts by referring to agreed standards that should be used to describe the problem in a way that allows a practical solution (4).

Parts of the model are translated into algorithms (5) for computer simulation and implementation purposes. The general formal structures (6a) and the algorithmic formal structures (6b) are transformed into products by specifying, designing and implementing appropriate material instances, e.g. programmed computers, manufactured components, and other machines (7). The products are verified and validated for implementation in the domain of interest (8).

The definition of a pure theory is the same as in definition 1 and a theory is the same as in definition 2.

Definition 3 (Applied theory) *A theory that satisfies product specifications is an applied theory.*

1.7.2 Scope of investigation

The RealNeuron model is developed in the framework of intelligent systems. Intelligence factors or intellifacts are introduced and defined to deal with the absence of an universally agreed definition for intelligence. Intelligence is defined mathematically in terms of intellifacts. This definition is then compared with an existing engineering definition of machine intelligence. The scope for this thesis is limited to only the learning and thinking intellifacts.

The grade of machine intelligence is introduced. This is used to distinguish between the different synthesis requirements for intelligent machines. It is based on the current engineering practice of designing, manufacturing, installing and operating materials and machines to a specific grade or standard specification[5].

The modelling of biological intelligence building blocks is addressed in an individual agent, see postulate 3. Several measurements have been taken by many neurobiologists and in this thesis the descriptions and measurements from [28, 41, 73, 72, 125, 137, 158] are used. The RealNeuron model is formulated and evaluated, restricted to the aspects that implement the learning properties of biological neurons. It is a state space formulation and illustrated with block diagrams[6]. It is a multi-resolutional and modular model that consists of several components in various layers of abstraction. It is computationally efficient and only adds or subtracts ion concentrations based on the states at the membrane structure's level.

[5]This is based on the author's 14 years of experience as a practising engineer.

[6]The author proposed the RealNeuron mathematical state space model and block diagrams. He partook in several discussions on the 2000-prototype's theory as well as its implementation [36, 31, 32, 145, 144].

The RealNeuron is verified in a bottom-up manner. The RealNeuron is verified and validated in various configurations including learning through classical conditioning (Pavlov conditioning) as known in psychology [124] and discussed in [110]. A model has been put forward in [126] using rate-coded neurons with a temporal Hebb-like learning rule which utilises the predictive capabilities of bandpass-filtered signals by using the derivative of the output to modify the weights. The advantage of using RealNeurons instead of rate-coded neurons (spiking neurons) is that RealNeurons are not based on electrical equivalent circuits, but on the actual structure of a biological neuron.

An engineering synthesis of a RealNeuron network, based on classical conditioning, demonstrates how to implement a RealNeuron network that can be used in machines built to the grade of machine intelligence requirement which is classical-conditioning learning implemented with neural networks that can change learned associations in a dynamic environment.

This thesis demonstrates that the RealNeuron model resembles biological neurons more closely than classical artificial neurons or spiking neurons, by achieving the following sub-goals:

1. Modelling of the fundamental electrochemical building blocks of intelligence as observed in nature [28, 41, 73, 72, 125, 137, 158]. The ways these building blocks interact in complex structures to achieve intelligent processing units are modelled in a top-down fashion using the following procedure:

   (a) The systematic correction and completion of the terminology and symbols in the set predicate model which was deduced from [40, 125, 137] in [36, 31, 32], see §4.2 and the message processing algorithms in §4.3.

   (b) The development of block-diagram models of the 2000-prototype implementation. This model visually shows the signal flow inside the RealNeuron, see §4.4.

   (c) The development of state space models of the 2000-prototype implementation, see §4.4. The model parameters and states (input, internal and output) are explicitly identified to provide a basis for future improvements.

2. A theoretical evaluation of the RealNeuron that compares it with the structures of pulsed neural networks, the perceptron and indirectly to the Hodgkin-Huxley model, see chapter 5. This validates that the RealNeuron is a carefully reduced model that retains essential features of more complex models, e.g. the Hodgkin-Huxley model [72].

3. A theoretical evaluation of the RealNeuron model is done to determine the performance of the model in the presence of noise.

4. An evaluation of the performance of a RealNeuron and its subcomponents compared to what is known in the neuroscience and electronics engineering literature in a bottom-up fashion:

   (a) Verify the implementation of a main membrane and post-synaptic membrane by comparing the simulation results with the biological results in [28, 110, 73, 72, 125], see §6.7.

   (b) Verify the implementation of an input soma and bulb end by comparing the simulation results with the biological results in [110], see §6.8.

(c) Verify the implementation of a sensor neuron, that is a sub-function of the classical conditioning network, with the description in [110], see §6.9.1.

(d) Verify the implementation of a motor neuron, that is a sub-function of the classical conditioning network, with the description in [110], see §6.9.2.

(e) Verify the implementation of a Purkinje cell, that is a sub-function of the classical conditioning network, with the description in [110], see §6.9.3. Derive implementations for special cases by using perceptrons and logical gates.

(f) Verify the implementation of a nucleus interpositus cell that is a sub-function of the classical conditioning network with the description in [110], see §6.9.4. Derive implementations for special cases with perceptrons and logical gates.

(g) Verify the implementation of AND, NAND, OR, NOR, NOT and XOR logic functions[7] with the description in [54], see §6.10. The AND and OR logic functions are sub-functions of the classical conditioning network described in [110].

(h) Verify the RealNeuron network implementation of associative learning through classical conditioning [124, 110], see §6.11. Derive implementations for special cases from perceptrons and logical gates.

1.7.3 Assumptions

The neuron model describes a logical structure which is deduced from neurobiological data and structures as described in [28, 41, 73, 72, 125, 137, 158]. Only one abstract type of neuron is modelled. Differently configured instances of the RealNeuron can implement different neurons.

The examples in this thesis are instances of logical neurons deduced from biological neuron data. The following assumptions are made regarding biological neurons:

- Sodium, potassium, and chlorine ions contribute the most to information processing[8].
- Dendrites make spatial signal processing possible and can be modelled by time delay elements on the signals entering a membrane.
- Between neurons, signals are transported by means of neurotransmitters.
- Only inhibitory and excitatory neurotransmitters exist. All the other special functions of neurotransmitters are ignored.

[7] The ability of language is a part of intelligence [14] and a calculus is developed for the symbolic processing with logic functions in [14].

[8] Although many ion species exist that contribute to information processing, [72] has discovered that sodium, potassium, and chlorine ions contribute the most to information processing and have the highest concentrations in the neuron.

1.8 Layout of thesis

This thesis has been typeset using LaTeX with the following header:

```
\documentclass[a4paper, twoside, 10pt, dvipdfm]{report}
\usepackage[a4paper, bookmarksnumbered]{hyperref}
\usepackage{a4, acronym, amssymb, array, boxedminipage, color,
            dsfont, graphicx, graphics, subfigure, tabularx}
\pagestyle{headings}
\newcommand\savedparskip{\parskip}
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex plus 0.75ex}
\setcounter{secnumdepth}{5}
\setcounter{tocdepth}{5}
\newtheorem{definition}{Definition}
\newtheorem{theorem}{Theorem}
\newtheorem{axiom}{Axiom}
\newtheorem{postulate}{Postulate}
```

The thesis consists of seven chapters, seven appendices, and a bibliography. Figures and tables are inside the main body of the text using the LaTeX layout rules.

Chapter 1 is the introduction. An overview of the field of study is presented as well as the purpose of the research.

Chapter 2 gives the engineering contexts in which this study is executed as well as some of the related work of others. The different fields that contribute to this study are also presented. Intelligence is defined mathematically in terms of intelligence factors or intellifacts. The grade of machine intelligence is introduced in this chapter. A justification is given for limiting the scope, in this thesis, to only the learning and thinking intellifacts. This is followed by a general overview of neuroscience.

Chapter 3 gives an overview of the different modelling techniques used in the thesis. How to model the RealNeuron is discussed. An overview of the RealNeuron is given based on a multi-resolutional approach found in systems engineering. A formal system breakdown structure (SBS) is defined. A variation of Aristotle's convention, on the naming of objects, is presented for the RealNeuron.

Chapter 4 presents different descriptions of the RealNeuron model. The detail for the RealNeuron network system level and lower system levels is developed. This detail is developed by establishing models, that extract the information-processing capabilities of the biological neurons. An abstract model of the RealNeuron is presented using a set predicate description. An algorithmic description for each part of the RealNeuron is given. Block diagrams and a state space model of the RealNeuron are presented.

Chapter 5 compares the mathematical model of the RealNeuron with the general structure of spiking neurons, the perceptron and indirectly with the Hodkin-Huxley model. It also evaluates the performance of a RealNeuron in a noisy environment.

Chapter 6 describes the verification and validation of the RealNeuron in a bottom-up manner. The pumps, channels and receptors are verified first. These components are then integrated into the different membrane types (post-synaptic membrane, main membrane, axonal membrane) and verified while the membrane components are validated simultaneously. This process is repeated until individual neurons have been built up and RealNeuron networks have finally been constructed. The RealNeuron is verified and validated in configurations for AND, NAND, OR, NOR,

NOT and XOR logic functions. It is also verified and validated by the implementation of classical conditioning. An engineering synthesis of a RealNeuron network, based on classical conditioning, demonstrates how to implement a RealNeuron network that can be used in machines built to the grade of machine intelligence requirement which is classical-conditioning learning implemented with neural networks that can change learned associations in a dynamic environment.

Chapter 7 describes the conclusions drawn. Further work is also identified as well as possible enhancements to the RealNeuron model and its implementation.

The appendices consist of:

- A glossary of abbreviations and symbols used in this thesis.
- Presentations and publications based on the research done for this thesis.
- eXtended Markup Language (XML) Document Type Definition (DTD) for the RealNeuron network.
- An example of the pump implementation using C++.
- Motivation for the bulb-end soma.
- Simulation implementation notes.
- Contents of the included CD-ROM.

A bibliography is given of all references used in the thesis. The citations and bibliography have been compiled using BibTeX with the IEEE transactions format as defined in the IEEE BibTeX style, but with the bibliography sorted in alphabetical order [136]. The following setting was used for the bibliography:

```
\bibliographystyle{IEEELatex/IEEEtranBST/IEEEtranS}
```

Chapter 2

Background

> *The essence of science: ask an impertinent question, and you are on the way to a pertinent answer* — Jacob Bronowski *The Ascent of Man* (1973).

The context is set in that domain in which the RealNeuron is developed, i.e. it provides the background and defines the domain of interest, see §1.7.1.1 p. 8 and §1.7.1.2 p. 9.

The question is asked: What is intelligence? Intelligence is defined mathematically in terms of intelligence factors or intellifacts. Machine intelligence is put into relation with intelligence. A definition for intelligence is formulated in terms of identified intellifacts that are based on standard dictionary definitions. This definition is then compared with an existing engineering definition of machine intelligence.

The grade of machine intelligence is introduced in this chapter. It is based on the current engineering practice of designing, manufacturing, installing and operating materials and machines to a specific grade or standard specification.

Some scientists' and engineers' work in the area of machine intelligence are discussed.

A justification is given for limiting the scope, in this thesis, to only the learning and thinking intellifacts. This is followed by a general overview of neuroscience.

2.1 What is intelligence?

Before AI agents can be built, see §1.1 p. 1, there should first be established what is meant by an "intelligent system". What constitutes intelligence is a topic of major discussion and debate in different areas of science.

Several philosophical viewpoints on and diverse definitions for "intelligence" exist, as reflected in the exhaustive discussions in the Internet discussion group on Architectures of Intelligent Control Systems (AICS-L), moderated by Alexander Meystel at the time. The proposed definitions for intelligence exclude several factors that are generally understood as contributing to intelligence.

Engineers are interested in machine intelligence, i.e. intelligence designed into machines. The engineering view differs from the scientists' and philosophers' viewpoints, see the archives on AICS-L. What engineers call "intelligence" in the context

of machines should rather be called *intelligence factors* or *intellifacts*[1] in the general sense [112, 64, 65]. This is similar to the *factor analysis* approach in psychology, [69]. Thus, in mathematical terms, it can be stated in general that:

> Intelligence is the set
>
> $$X_i = \{x | x \text{ is an intellifact}\}$$
>
> with the set of intellifacts $X_{if} \in \mathbf{2}^{\mathbf{X_i}}$ and $X_{if} \neq \emptyset$.

Thus, a real intelligent machine cannot exist if all the $x_{if} \in X_i$ is not known and implemented – this is a scientific viewpoint.

Engineers, on the other hand, would like to draft specifications or standards that determine when machines could be regarded as intelligent. Such a standard would specify a set of intellifacts X_{if} that a machine should comply with to regard it as intelligent. Different sets of intellifacts would give rise to different specifications for machine intelligence. Thus, each standard would define a different grade of machine intelligence. This is analogous with the codes and standards that exists for the designing, manufacturing, installation and operation of different grades of materials and various other engineering products.

The aim should be then to arrive at a specification for an intelligent machine. Thus, a set of intellifacts that should be implemented by a machine to be regarded as intelligent to a specified grade.

Thus, based on the mathematical definition of intelligence above, it can be stated that:

> Artificial intelligence (AI) or machine intelligence is the set
>
> $$X_{ai} = \{x | x \in X_{if}\}$$
>
> with the set of intellifacts $X_{if} \subset X_i$ and $X_{if} \neq \emptyset$.

For this discussion, intelligence is limited to the scope of biological intelligence as found in carbon-based life forms. This type of intelligence is found in nature of which human beings are very good specimens. This type of intelligence is what people normally have in mind when they talk about intelligence. A standard dictionary of the English language, [138], defines "intelligence" as:

1. Intelligence is the quality of being *intelligent* or *clever*.
2. Intelligence is the ability to *think*, *reason*, and *understand* instead of doing things automatically or by instinct.

The word "intelligent" is defined by [138] as:

> A person or animal that is intelligent has the ability to *think*, *understand* and *learn* things quickly and well.

In [138], "clever" is defined as:

> Someone who is clever is intelligent and able to *understand* things easily or *plan* things well.

From the above definitions, *intelligence is the ability to think, understand, learn and plan things quickly and well instead of doing things automatically or by instinct.* Based on Boole's classic work [14], the abilities to reason and think are combined.

[1] This term was coined by the author in the AICS-L discussion group.

2.2 Intellifacts

Intellifacts are defined mathematically in the previous section and from this definition it can be stated that *intellifacts are the intelligence factors which contribute to that which is understood as intelligence.* As seen in the previous section, if only a certain set of intellifacts are implemented in a machine, the machine is built to a specific grade of machine intelligence.

Examples of intellifacts and sub-intellifacts are the ability to speak, write, solve puzzles, learn, reason with facts (both complete and incomplete), understand, plan, etc. But, the words of Qui-Gon from Star Trek Episode I: "The ability to speak doesn't make you intelligent", after he has saved the life of Jar Jar Binks on the planet Naboo, [100], apply to intelligence as well. Based on the reasoning in the previous section, it is not generally true that the ability to think, understand, learn and plan things imply intelligence. Thus, intelligence only implies certain abilities or intellifacts and not that certain intellifacts imply intelligence. Mathematically, this can be expressed as

$$intelligence \neq intellifact$$

but

$$intelligence \Rightarrow intellifact.$$

Thus, artificial intelligence is not intelligence, unless the grade of machine intelligence is implementing all the constituting intellifacts of intelligence. This grade is called the intelligence grade (IG). From the argument in the previous section, machines built to IG specifications are not practically possible at this stage.

Following from the definition of intelligence, intellifacts include the abilities of:

- Learning,
- Thinking,
- Understanding and
- Planning.

These intellifacts are not necessarily mutually orthogonal, thus sub-intellifacts are not unique to only one intellifact. If these four intellifacts are used as the basis for defining grades of machine intelligence by forming different combinations then the number of grades is theoretically fifteen ($2^4 - 1$, because the $\emptyset$ is excluded by definition.) These grades can be further refined by including sub-intellifacts as well as the technology used to implement them, e.g. rule-based, neural networks, fuzzy logic, etc.

Albus proposed the following definition for machine intelligence as used in intelligent control systems, [6]:

> Intelligence is an ability of a system to act appropriately in an uncertain environment, where appropriate action is that which increases the probability of success, and success is the achievement of behavioural sub-goals that support the system's ultimate goal.

This definition is a well-formed requirement in terms of [80] and if analysed it is seen as:

Capability: to act appropriately in an uncertain environment,

Condition: increases the probability of success, and

Constraint: the achievement of behavioural sub-goals that support the ultimate goal.

From systems engineering it is known that in order to achieve the ultimate goal for a system by the achievement of sub-goals, proper planning is necessary, especially in an uncertain environment [11, 81, 44]. Thus, in order to satisfy the constraint of Albus' machine intelligence definition, the ability to *plan* is necessary.

To be able to plan something, a model is required on which the planning is based [11, 81, 44]. This model should be a fair representation of the reality that is planned in order to increase the probability of success and, thus, to meet the condition of Albus' machine intelligence definition. The model makes understanding of an environment possible. A model is normally a simplified representation of reality and provides a means to *understand* an uncertain environment's dynamics. This understanding is used to make predictions on possible outcomes.

The results of the predictions on possible outcomes are evaluated and reasoned about or, in other words, the results are thought over, i.e. *thinking* takes place to assess the validity of predictions.

To be able to construct a model of reality requires the ability to *learn* from the environment in order to adapt the model.

From this discussion it is seen that the intellifacts identified earlier are also implied by Albus' machine intelligence definition. The grade for machines implementing Albus' machine intelligence definition are then intelligent system grade (ISG). Machines built to ISG specifications are practically possible to construct, see §2.3.2.2.

Machine learning is implemented with an optimisation algorithm [130, 42, 155, 156, 68]. General optimisation algorithms, that can solve constraint problems, make use of punishments and rewards [130]. Thus, a system that fulfils the requirement stated in Albus' machine intelligence definition also fulfils the requirements of postulate 1 p. 4.

2.3 Machine intelligence

In this section the work of other scientists and engineers in the field of machine intelligence is discussed. This is not an exhaustive literature survey of the machine intelligence field.

2.3.1 Artificial intelligence

Several scientists and engineers have endeavoured to construct intelligent machines. The Turing Test [149] has given rise to the whole effort of machine intelligence research in the area of hard artificial intelligence (AI). Hard AI is the study and construction of algorithms that mimic human intelligence based on logically oriented knowledge-based techniques [85]. By extending the metaphor of flight to hard AI

systems, it is similar to the flapping of an aircraft's wings in trying to fly[2]. The principles of intelligence are still not understood by studying hard AI. See also the inability of hard AI algorithms to learn in §1.2 p. 2.

It is interesting to note that the whole hard AI movement has conveniently forgotten about Turing's manifesto for Artificial Intelligence in 1948 (see 1.2 on p. 2.) Also his criteria for a thinking machine in same paper as the Turing test [149], summarised in postulate 1 p. 4. Thus, hard AI systems are, according to Turing, not thinking machines, i.e. they are not intelligent machines, because they cannot change their own programming to generate desired behaviours in order to succeed in an uncertain environment, i.e. learn and adapt, as required by postulate 1 p. 4. Hard AI systems are only implementing the thinking intellifact, thus the machine intelligence grade assigned to such systems is thinking grade (TG), and the planning intellifact, thus the machine intelligence grade assigned to such systems is planning grade (PG).

2.3.2 Computational intelligence

Another approach is soft artificial intelligence or computational intelligence. Computational intelligence algorithms are 'partially intelligent', because they do not try to perform actions that mimic human intelligence in its totality, but aspects of observed mechanisms that generate intelligent behaviours in nature [64]. Computational intelligence simulates aspects of the perceived fundamentals of intelligence. The main topics studied in computational intelligence are neural networks, fuzzy logic, genetic algorithms and variations on these themes [42, 1].

Neural networks, neuro-fuzzy logic and genetic algorithms give a machine the ability to fulfil Turing's criteria for a thinking machine in order to learn and adapt, as required by postulate 1 p. 4.

Engineering applications of neural networks are briefly discussed in §1.5 p. 5. The IEEE also publishes transactions on genetic algorithms and fuzzy systems on a regular basis.

2.3.2.1 Artificial neuron models

Some books on artificial neural networks [42, 29] introduce the subject by giving the motivation that basic artificial neurons are deduced from biological neurons without showing how. This statement is accompanied by a picture of a biological neuron (see figure 2.1), naming the major components of the neuron. The authors jump directly to a graph model of the artificial neuron (see figure 2.2) and this is then followed by a mathematical model:

$$u_k = \sum_{j=0}^{p} w_{kj} x_j \tag{2.1}$$

$$y_k = \varphi(u_k + \theta_k) \tag{2.2}$$

This description of an artificial neuron is an oversimplification of how a biological neuron works.

[2]In the early days of human attempts to fly, the flapping of a bird's wing was imitated by flapping wings. The effect of this was only a machine that was standing and vibrating on the ground, with no take-off or individuals with wings strapped to their arms and falling to the ground and breaking a couple of bones. Only by studying the fundamentals of aerodynamics, could engineers build aircraft that have fixed wings or rotating wings. Today aeroplanes and helicopters are part of modern modes of transport used by millions of people each year [64, 65].

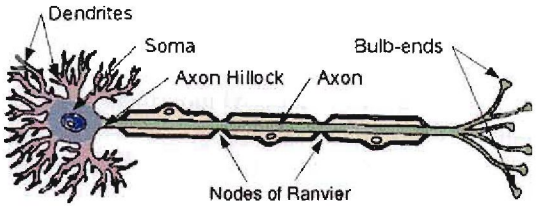


Figure 2.1: Biological neuron, source unknown

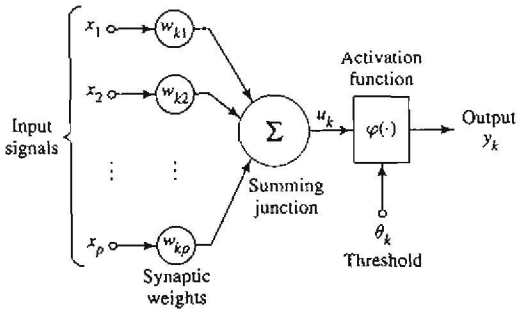


Figure 2.2: Artificial neuron [67]

The simplification of a biological neuron, that exhibits chaotic behaviour [56], into a deterministic expression for an adaline or a basic perceptron [42, 67, 98] is very difficult to accept. The artificial neuron is rather an abstraction of a neuropil (which is a collection of chaotic neurons that are configured into a neuro-circuit [56, 41]). This neuropil then represents an average of the activity of the underlying chaotic neurons that deliver behaviour closer to the artificial neuron like an adaline or perceptron.

Pasemann has configured perceptrons in recurrent networks that exhibit the chaotic behaviour of a biological neuron [123], but these configurations cannot be mapped one-to-one to the biological case. A biological neuron never feeds back onto itself (auto-feedback) as in Pasemann's configurations. If a biological neuron has an auto-feedback, the feedback link (synapse) will be weakened and eventually destroyed [56].

Husbands and his associates have modelled GasNets. This type of network models is based on the brain's ability to influence a region in itself with secreted neuro-transmitters [74, 75, 132, 131]. GasNets model only one mechanism in the brain and are still using the perceptron as the basis.

Pulsed neural networks are mainly used to explain measured data of biological neurons and building silicon nervous systems with integrated circuits [104]. A network of spiking neurons is part of the set of pulsed neural networks.

The existing artificial neural models are classified according to the following three generations [103]:

- *First Generation*: McCulloch-Pitts neurons or perceptrons are used to implement multi-layered perceptrons, Hopfield nets, and Boltzmann machines, [68]. They can compute Boolean functions as well as digital functions.
- *Second Generation*: Applying activation functions with a continuous set of possible output values to a weighted sum of the inputs (e.g. sigmoid functions, linear saturated functions, piecewise exponential functions) enables the implementation of feed-forward networks, recurrent neural networks and radial basis function networks [68]. According to [19] these networks can approximate non-linear functionals defined on a compact set of a Banach space.
- *Third Generation*: Spiking neurons, such as integrate and fire neurons, form networks in which timing of individual computational steps play a key role in the computation [104].

It is proved in [102] that a network consisting of a finite number of spiking neurons can simulate a universal Turing machine. This means that, in principle, any problem that is computable [21], could be solved with networks of spiking neurons [102].

It is further proved in [102] that spiking neurons have at least the same computational power as artificial neural networks of the first two generations. Many functions require significantly fewer spiking neurons than neurons of the first or second generation [102].

2.3.2.2 NIST models

Albus has proposed a model for intelligence [6, 3] and Meystel has proposed the grouping, focusing attention, and combinatorial search (GFACS) algorithm, see [113]. They have collaborated and have produced a practical intelligent control framework [4]. The framework has been tested in several technology demonstrators [1], which is a combination of multi-resolutional control (MRC) and GFACS. Each level of resolution of their framework makes provision for the intellifacts defined in §2.2 and supports Albus' machine intelligence definition.

Their method is a combination between systems engineering, control engineering, robotics, semiotics[3], artificial intelligence and computer science [4]. Judging by the technology demonstrators implementing it and demonstrated during the 1998 IEEE International Joint Conference on Intelligent Science and Technology at the National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, United States, this framework seems to be successful. Several papers were presented about the framework and the technology demonstrators [1]. The technology demonstrators that the author saw in action were:

- Humvee autonomous vehicle control.
- Robocrane for precision construction.
- Precision welder.
- 3D component precision measurement system.
- 3D precision control of a tooling machine.

[3]Semiotics is concerned with everything that can be taken as a sign. A sign is everything which can be taken as being a significant substitute for something else [43].

The NIST technology demonstrators are all built to ISG specifications, if they are based on the intelligence framework described in [3, 4]. Thus, they are also fulfilling the requirements of postulate 1 p. 4, as argued in §2.2.

The NIST framework is an abstraction that generates ISG intelligent behaviours and does not use models of biological neural networks explicitly. However, this framework does not exclude the use of biologically inspired neural networks in its various MRC levels and implementing GFACS. The GFACS algorithm can be replaced by any other semiotic learning algorithm by using a semiotically orientated autonomous intelligent systems engineering methodology as demonstrated in [61].

2.4 Neural network grade

Neurons are assumed to be the building blocks of intelligence, see postulate 3 on p. 5. If this is true then all intellifacts could be implemented with neural networks. This adds another dimension to the grade of machine intelligence. In this thesis the emphasis will be on gaining an understanding on how to construct the building blocks of intelligence.

Because the construction of machines built to IG specifications is not currently possible, this thesis will not attempt to implement all the intellifacts, but limit the scope to the demonstration of aspects of learning and thinking intellifacts using RealNeuron networks.

2.4.1 Learning

The different forms of learning that can be explained neurologically, are broadly summarised by [110] as non-associative learning, associative learning, higher forms of associative learning, imprinting and learning with understanding. Habituation and sensitisation are forms of non-associative learning [110]. Classical conditioning and operant conditioning are forms of associative learning [110]. Orientational learning, learning through observation and learning through play are higher forms of associative learning [110].

Non-associative learning is not very interesting from an engineering point of view at this stage, because the neural network is only sensitised for the input, without any definite output signals associated with it. For all the other types of learning, classical conditioning is the simplest to use to demonstrate the performance of RealNeurons. The other "higher" learning forms can be implemented with more complex neural networks.

Since the Pavlov learning experiments [124], a number of variations on the original experiment were done. Some of these variations use human beings [140, 115], cats [27], dogs [124] and rats [96].

The experiment most often repeated and referenced is the Rabbit Eye-Blinking Conditioning. This experiment has its origin:

> *A chance visit of Goodrich, Ross, and Wagner to Peter Arnott's laboratory, where rabbits were being used in the study of electroretinograms, convinced them that this animal, with large eyes that rarely blinked unless stimulated and a calm tolerance for restraint, might be the ideal subject for eye-blink conditioning. They had only little opportunity to do pilot work before Ross went to Wisconsin (where he set up the first rabbit eye-blink conditioning laboratory) and Wagner went to Yale (and soon followed suit), but it was enough for them to encourage others working with the human eye blink to try this preparation. Isadore Gormezano and his students quickly did so, eventually making it one of the most carefully characterised examples of Pavlovian conditioning. After Richard Thompson and his colleagues decided to pursue its biological basis, it also became one of the most thoroughly understood experiments at neurophysiological level* [7].

One of the first papers on results of the Rabbit Eye-Blinking Conditioning [147] appeared in 1964. Through the years articles have been published on various aspects of Rabbit Eye-Blinking Conditioning, especially the study of the various aspects at neuro-physiological level [140, 87, 121, 96, 114]. Several models have been constructed to explain neuro-physiological data [161, 154, 126, 105, 5, 87, 110].

Various types of neurons that contribute to the mechanism of classical conditioning have been identified and studied. They are the Purkinje cell [22, 18, 63, 142, 92, 91, 141], pontine cell [148], inferior olive [153] and interpositus nucleus cell [27, 148, 114].

The demonstration of classical conditioning in this work is based on [110] because a clear summary of various research results of the Rabbit Eye-Blinking Conditioning is presented together with a proposed neuron circuit.

2.4.2 Thinking

Thinking is well described by Boole in his famous work [14]. The underlying mechanism for thought, that is used to evaluate the world, consists of the basic logic functions: AND, OR, NOT and XOR [14]. By combining some of the functions in different ways, the following list is obtained:

- BUFFER = NOT NOT
- AND
- NAND = NOT AND
- OR
- NOR = NOT OR
- NOT
- XOR

These logic functions are used as part of the mental process for following a train of thought through to a point where it can be determined if objectives are met [14].

2.5 Neuroscience

Neuroscience is the general field of science that studies the nervous system [13]. The general purpose of this field of research is to relate the biological and chemical properties of the brain, including the brain's cells, to behaviour.

Various aspects of brain function and structure are examined by different specialists, who work within the general field of neuroscience. The following two categories are identified by [13] in neuroscience:

- Experimental neuroscience, in which the scientist perturbs the nervous system of an organism in order to yield measurable, predictable, and eventually explicable changes.
- Clinical neuroscience, in which a medical physician or other professional observes a change produced by life events (e.g. developmental, infectious, or traumatic events) and tries to understand the effect of the change on brain function.

Another category that can be added is theoretical or computational neuroscience, in which the scientist builds mathematical models based on the results of the above two categories in order to gain insights into the structures and dynamics of the nervous system of an organism.

The branches of medicine most closely tied to the study of the brain are neurology (the study of the diseases of the brain), neurosurgery, and psychiatry (the study of behavioural, emotional, and mental diseases).

The experimental neuroscience have traditionally been divided into the following major disciplines: Anatomy (seek to describe the organisation of the elements of the brain), physiology (attempt to understand the function of the elements of the brain), biochemistry (investigate the chemical properties of the elements of the brain), and psychology (seek to understand the behavioural operations of the brain).

Many studies in the field of neuroscience are interdisciplinary.

All the facts about cells and circuits have been organised around a skeletal outline of the cellular organisation and function of the brain's machinery [13]. These general statements are reviewed so that they are explained to some degree, see [13].

- The basic operating elements of the nervous system are the individual nerve cells, or neurons.
- Neurons share certain universal cellular features with other cells in the body. All have a plasma membrane, cytoplasm containing organelles, and a nucleus.
- Neurons also differ considerably from other body cells in several ways. They have characteristic configurations, with protruding parts (an axon and dendrites) that connect with other neurons to form circuits. In addition, the arrangement of organelles within neurons differs from that of other cells.
- The activity of neurons is regulated by the properties of the nerve cell membrane. When chemically stimulated, the membrane state changes from a resting potential to an action potential, and a nerve impulse is generated.
- The chemical stimulation of a neuron is in the form of a synaptic transmitter, which one neuron manufactures and releases from the end of its axon across the synaptic gap to a receiving neuron.

- Synaptic transmitters may have the function of producing a nerve impulse, inhibiting an impulse, or modifying the function of another transmitter.
- Generally, a receiving neuron requires a summation of many impulses in order to generate an action potential.
- The three major classes of synaptic transmitters are amino acid transmitters, monoamine transmitters, and peptides.
- Three basic patterns of neuronal circuitry emerge from genetic specifications: hierarchical chains, local-circuit neurons and single-source/divergent circuits.
- Besides neurons, the nervous system contains two types of glial cells, one of which forms the myelin sheath covering many types of neural circuits; all the functions of the other type of glial cells are not yet known for certain, but these cells appear to at least play a role in destroying possible toxic substances. Other brain cells are those making up the blood vessels and those composing the membranes surrounding the brain and spinal cord.

2.5.1 Computational neuroscience

The Hodgkin-Huxley model [72, 16], compartmental models [16] and spiking neurons [104] are the mathematical models that are mostly used to construct models to explain neurophysiological data.

The compartmental models are realistic biological neuron models [16] that model the biological data accurately, but are very slow to execute. The modelling techniques employed are all based on electric circuit equivalent models (conductance-based models), as originally done in the Nobel Prize winning seminal paper [72]. Spiking models are, strictly speaking, not conductance-based models, but are simplified variants of the conductance-based models.

The modelling techniques employed in computational neuroscience are all reliant on differential equations which lead to coupled systems of non-linear differential equations.

Some of the software simulation packages that are used to do simulation experiments of biological neuron models, based are: Genesis [16], NeuronC [139], SABER [17], SPICE [17], CONICAL [143], NEURON [70, 71] PDP++ [129], Neu-MODEL [86], Hi-NOON [26].

Biological neurons have also been simulated with hardware such as electrical models (using resistors and capacitors) [45] and electronic integrated circuits [107].

When simulations are performed on digital computers, the differential equations need to be discretised. The solution of these discretised equations leads to other numerical problems, e.g. convergence, initial values, numerical accuracy, truncation errors, etc., depending on the methods used and the complexity of the systems of equations to be solved, see [97].

Models of brain cells developed by computational neuroscience consume a vast amount of computer resources. As reported in [60], it took 18.2h on five connected SUN SPARC-2 workstations to simulate one second of activity of a single biological neuron.

The complexity of the differential equations are reduced by making approximations of the original model dynamics, [53, 118, 127, 2, 88], to achieve reasonable calculation times.

The neuron models can be divided into three broad groups [135]:

- Simple neurons: Single neuron details are functionally superfluous when the neuron operates in a large network and very simple models can be used to represent the input/output characteristics of neurons.
- Carefully-reduced models that retain essential features of more complex models help to understand information processing in large neuronal networks.
- Complicated models of neurons are developed to reveal how the various kinds of neuron components (dendrites, spines, axons, membrane channels and synapses) create a computationally powerful unit.

The RealNeuron is a carefully reduced model as is shown in chapter 5.

The classical way of modelling in neuroscience is by describing the neurological process with differential equations, e.g. the Hodgkin-Huxley equation [4], [72, 16, 28, 88, 58]

$$C\frac{du}{dt} = -\sum_k I_k + I(t) \tag{2.3}$$

$$\sum_k I_k = g_{Na}m^3h(u - V_{Na}) + g_K n^4 h(u - V_K) + g_L(u - V_L) \tag{2.4}$$

$$\dot{m} = \alpha_m(u)(1 - m) - \beta_m(u)m \tag{2.5}$$

$$\dot{n} = \alpha_n(u)(1 - n) - \beta_n(u)n \tag{2.6}$$

$$\dot{h} = \alpha_h(u)(1 - h) - \beta_h(u)h \tag{2.7}$$

where

- C membrane electrical capacitance in $\mu F/cm^2$,
- u is the voltage in mV,
- $\sum_k I_k$ is the sum of the ionic currents through the cell membrane in μA,
- $I(t)$ is the external driving current in μA,
- g_{Na}, g_K, g_L are conductances in mS/cm^2,
- V_{Na}, V_K, V_L is the reversal potentials at which a current I_k changes direction in mV,
- m, h determine the removal of the Na channel block,
- n determine the removal of the K channel block,
- α, β are imperical functions of u that are adjusted to fit measured data.

The Hodgkin-Huxley model is a relatively simple and experimentally testable model that embodies the major features of membrane non-linearity: namely, voltage-dependent ionic currents that activate and inactivate in time. The model is, in its original form, limited to the two voltage-dependent currents found in the squid giant axon and it must be extended significantly if it is to deal with the excitable soma and dendrites of neurons. Secondly, the macroscopic and deterministic model does not capture the kinetics of the sodium channel correctly and it cannot account for the stochastic response to current injection that arises from the discrete nature of ion channels. Thirdly, much simpler integrate-and-fire-type models (pulsed neural networks) seem to be more useful for exploring collective phenomena in neuronal networks [111].

[4] Without changing the form of the Hodgkin-Huxley equation, the notation in [58] is used, which deviates from the original notation used in [72].

2.6 Summary

The RealNeuron is a carefully-reduced model, see chapter 5, and is developed in the contexts of machine intelligence. The building block for biological intelligence is the neuron. Current artificial neuron models are abstracts of biological neurons. They ignore many of the capabilities of biological neurons.

An accepted scientific definition for "intelligence" does not yet exist, thus the concept of intellifacts was introduced. Intellifacts are the intelligence factors which contribute to that which is understood as intelligence and include, amongst others, the abilities of learning, thinking, understanding and planning.

Based on dictionary definitions, intelligence is the ability to think, understand, learn and plan things quickly and well instead of doing things automatically or by instinct. Mathematically the first part of the definition can be written as:

> Intelligence is the set
>
> $$X_i = \{x | x \text{ is an intellifact}\}$$
>
> with the set of intellifacts $X_{if} \in \mathbf{2^{X_i}}$ and $X_{if} \neq \emptyset$.

A real intelligent machine cannot exist if all the $x_{if} \in X_i$ are not known and implemented. This leads to the following formulation:

> Artificial intelligence (AI) or machine intelligence is the set
>
> $$X_{ai} = \{x | x \in X_{if}\}$$
>
> with the set of intellifacts $X_{if} \subset X_i$ and $X_{if} \neq \emptyset$.

The grade of machine intelligence is introduced based on the specific intellifacts that are implemented and the technology used for implementation. Theoretically, fifteen grades of machine intelligence can be defined based on the four identified intellifacts. These following grades are explicitly proposed:

TG machines that implement only the thinking intellifact, e.g. classic AI-rule based systems, in §2.3.1.

PG machines that implement only the planning intellifact, e.g. chess playing computers from IBM, in §2.3.1.

ISG machines that implement learning, thinking, understanding and planning intellifact, e.g. some of the technology demonstrators built by NIST, in §2.2.

IG truly intelligent machines, i.e. machines that are intelligent just like human beings. This grade is not part of the theoretical fifteen grades based on the identified intellifacts, and it is not possible to build with the current knowledge about intelligence, in §2.2. This grade is an ideal.

Another dimension can be added to these grades by specifying the implementation technology, e.g. implementation with RealNeuron.

The focus of the rest of this thesis is on how to model biological neural networks and to demonstrate that deduced structures can be used to implement aspects of learning and thinking intellifacts.

Chapter 3

RealNeuron modelling

Though this be madness, yet there is method in't — William Shakespeare
Hamlet

It has been established in postulate 3 p. 5 that neurons are assumed to be the underlying building blocks for biological intelligence. This chapter discusses how a RealNeuron is modelled.

An overview of the RealNeuron is given in §3.3, based on a multi-resolutional approach found in systems engineering [77, 81, 44, 11]. Based on the formal theory presented in §3.1, a formal SBS is defined in §3.2. In §3.5, a variation of Aristotle's convention, on the naming of objects, is presented for the RealNeuron.

With respect to the scientific and engineering way of life (§1.7.1.1 p. 8 and §1.7.1.2 p. 9) this chapter describes the theory and language used in the rest of the thesis.

3.1 Formal theory

A reference is made to a theory in figure 1.2 on p. 8. A formal language L is used to express a theory T [36, 35]. This implies that $T \subset L$. A set theory language can be used [99] as the formal language L, with the following conventions:

- Elements of sets are described by small letters, e.g. x, y, z, etc.
- Functions and mappings are described by small letters $f(x)$, $g(x)$, $func(x)$, etc. Oftenly used functions have special symbols like:
  - $\cup$ is the union of two sets,
  - $\cap$ is the intersection of two sets,
  - $\emptyset$ is the empty set,
  - $\#()$ is the cardinal number of a set, i.e. for finite sets the number of elements in the set,
  - $\mathcal{R}$ is the range set of a function, mapping or relation, and
  - $\mathcal{D}$ is the definition set of a function, mapping or relation.
- Sets are described by capitalised first letters e.g. X, Y, $Numbers = \{x|...\}$. Oftenly used sets will also have special symbols like:
  - $\mathbb{N}$ for natural numbers, and

  - $\mathbb{R}$ for real numbers.

- Relations are described by capital letters only, e.g. $THEORY(x) \iff ....$ Oftenly used special symbols are:
  - $\in$ the element relation, and
  - the $\subset$ subset relation.
- The power set of a set X is indicated by $\mathbf{2}^X$. If $X = \{a, b, c\}$ then

$$\mathbf{2}^X = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\}$$

A theory T is defined as a structure using the following predicate logic form [15, 101].

Definition 4 (Theory)

$$T(x) \iff x =< < O_1, ..., O_n >, < R_1, ..., R_m, f_1, ..., f_k >, A >$$

with

O_i *representing a set of objects,*

R_i *representing relations,*

f_i *representing functions and mappings, and*

A *representing the axioms of the theory.*

In the above definition A is the axioms, and O_i, R_i as well as f_i are part of the inference rules in figure 1.2 on p. 8.

For every relation or function, a type can be given by stating the structure of either the set or sets to which it belongs, [36, 35], e.g.

$$\begin{aligned} R_i &\subseteq O_1 \times \mathbf{2}^{(O_1 \times O_2)} \\ f_j &\subseteq \mathbf{2}^{(O_1 \times O_2)} \times O_3 \end{aligned}$$

The axioms describe laws or definitions in the domain of the theory's objects and relations [36, 35], e.g.

1. Laws: $(\forall x, y)(P(x) \wedge C(y) \Rightarrow X(x, y))$
2. Definitions: $(\forall x)(f(x) = \{y | F(x, y)\})$.

The theory T, on its own, is a pure formal theory with no connection to empirical reality. To establish such a relationship, data D is needed. D is written in a data representation language L', which should be a subset of the formal language L, thus $L' \subset L$ [36, 35].

Measured data $d \in D$ is obtained through measurements M from the domain of interest in figure 1.2 on p. 8. Any measured data expression d_i is a special axiom of the theory, thus a set of data expressions D is a subset of the axioms A of the theory, thus $D \subset A$ [36, 35]. The set $D \cup A$ has to be consistent. In practical situations, a subset D' of the set D is used to verify the theory T with $D' \cup A$ and generalised for $D \setminus D' \cup A$.

For convenience, a formal structure is defined as:

Definition 5 (Structure)

$$STRUCTURE(x) \iff T(x)$$

Thus, a formal structure and a formal theory are synonyms in this thesis.

3.1.1 Systems, environments and bodies

It is convenient to deal only with the substructure of a structure called a *System*. A system S is a 2-tuple $< x_1, x_2 >$ of objects and relations.

Definition 6 (System)

$$
\begin{aligned}
&Systems = \{x | SYSTEM(x)\} \quad \wedge \\
&SYSTEM(x) \iff \\
&\qquad x \in \{ < x_1, x_2 > \mid x_1 =< O_1, ..., O_m > \wedge x_2 =< R_1, ..., R_n > \} \quad \wedge \\
&obj(x) = x_1 \quad \wedge \\
&rel(x) = x_2 \quad \wedge \\
&states(x) = \mathbb{S} \subseteq obj(x) \quad \wedge \\
&\mathbb{S} \cap Systems = \emptyset
\end{aligned}
$$

It is necessary to deal with simple systems in special cases. A simple system consists of objects representing states ($\mathbb{S}$) and one relation representing the system function σ_f.

Definition 7 (Simple system)

$$
\begin{aligned}
&SimpleSystems \subset Systems \quad \wedge \\
&SimpleSystems = \{x | SIMPLESYSTEM(x)\} \quad \wedge \\
&SIMPLESYSTEM(x) \iff \\
&obj(x) =< \mathbb{S} > \quad \wedge \\
&rel(x) =< \sigma_f > \quad \wedge \\
&\sigma_f : \mathbf{2}^{\mathbb{S}} \rightarrow \mathbf{2}^{\mathbb{S}}
\end{aligned}
$$

Normally, a system has interaction with an environment. An environment is at least a simple system with environmental states ($\mathbb{S}_\mathcal{E}$).

Definition 8 (Environment)

$$
\begin{aligned}
&Environments = \{x \mid ENVIRONMENT(x)\} \quad \wedge \\
&ENVIRONMENT(x) \iff x \cap SimpleSystems \neq \emptyset \quad \wedge \\
&states(x) = \mathbb{S}_\mathcal{E}
\end{aligned}
$$

A body is needed for the RealNeuron to interact with an environment. Sensors and actuators are used for interaction with the environment.

The sensors (ϖ) and actuators (φ) as well as the RealNeuron network (Ψ) are part of the body (B), thus $\varpi, \varphi, \Psi \in B$. The body limits the environmental states ($\mathbb{S}_\mathcal{E}$) that could be sensed and effected by the RealNeuron network. The ϖ detect the possible sensory states ($\mathbb{S}_\varpi$), and the φ influence the effectory states ($\mathbb{S}_\varphi$), with the sensory states and effectory states part of the local body states ($\mathbb{S}_B$), thus $\mathbb{S}_\varpi \cup \mathbb{S}_\varphi \subset \mathbb{S}_B \subset \mathbb{S}_\mathcal{E}$.

Definition 9 (Body)

$$
\begin{aligned}
&Bodies \subset Systems \quad \wedge \\
&Bodies = \{x | BODY(x)\} \quad \wedge
\end{aligned}
$$

$$
\begin{aligned}
&BODY(x) \iff \\
&obj(x) =< \Psi, \varpi, \varphi, \mathbb{S}_B > \quad \wedge \\
&rel(x) =< \sigma_B > \quad \wedge \\
&< \mathbb{S}_\varphi, \mathbb{S}_\varpi > \subseteq \mathbb{S}_B \quad \wedge \\
&\sigma_s : \varpi \rightarrow \mathbb{S}_\varpi \quad \wedge \\
&\sigma_a : \mathbb{S}_\varphi \rightarrow \varphi \quad \wedge \\
&< \sigma_s, \sigma_a > \subseteq \sigma_B \quad \wedge \\
&\sigma_B : \mathbf{2}^\Psi \times \mathbf{2}^{\mathbb{S}_B} \rightarrow \mathbf{2}^\Psi \times \mathbf{2}^{\mathbb{S}_B}
\end{aligned}
$$

3.2 Formal system breakdown structure

In systems engineering, a system breakdown structure (SBS) is used as a structure to define and manage a system and its life cycle [81, 44, 77, 11].

Formally, it can be said that

$$SBS = \{(x, y) | y \in \mathbf{2}^S, S \in Systems, \exists x \in \mathbf{2}^S, y \subsetneq x\} \tag{3.1}$$

where S is the system with lowest level of resolution that is of interest.

The subsystems and components of S are S_s, thus

$$S_s \subset S \tag{3.2}$$

S is an environment for the subsystems S_s. Each sub-system in S_s is again a system which consists of subsystems and components. This is repeated until the necessary level of resolution is reached.

A component can be defined as follows:

$$
\begin{aligned}
&Components = \{x \mid COMPONENTS(x)\} \quad \wedge \\
&COMPONENTS(x) \iff x \cap Systems \neq \emptyset \quad \wedge \\
&obj(x) \cap Systems = \emptyset \quad \wedge \\
&rel(x) \cap Systems \neq \emptyset
\end{aligned}
$$

Each level of resolution is indicated by a number. The highest level of resolution is system level one. If a system consists of seven levels of resolution, then the lowest level of resolution will be number 7 and the highest resolution level will be 1. From this description, it follows that if $S^{(i)}$ is the subsystems and components of system level $i + 1$ and ℓ is the number of system levels of interest, then

$$S^{(1)} \subset S^{(2)} \subset ... \subset S^{(\ell)} = S$$

If $S^{(i)}$ is the subsystems and components of system level $i + 1$, then

$$\forall x \in S^{(i)}, \exists y \in S^{(i+1)}, \ni x \subset y$$

3.3 From biological neuron to RealNeuron

In biology, physiology, psychology and neuroscience textbooks, the normal way of describing the biological neuron is with a bottom-up approach [13, 41, 125, 137, 158]. In systems engineering the system description is a top-down description. It is outside

the scope of this thesis to discuss the advantages and disadvantages of the two methods. The systems engineering method of describing a system, with the aid of the system breakdown structure, is used. In the author's experience, it leads to simpler descriptions of complex systems.

A biological body, e.g. human body in its environment, forms an intelligent system. It has interfaces with the environment to observe environmental states and to manipulate the environment. The context diagram for the body is shown in figure 3.1. The body is a subsystem of the intelligent system. This is the lowest level of a system's resolution.

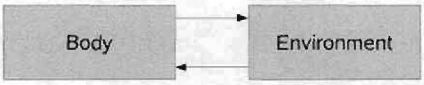


Figure 3.1: Context diagram for a body

If the body is regarded as a system for processing information then it consists of a central nervous system that is connected to sensors and actuators, see also the discussion in §3.1.1. In this case the central nervous system, sensors and actuators are subsystems of the body. The central nervous system in a biological body is a neural network. This neural network is modelled with a RealNeuron network in this thesis. The RealNeuron network context diagram is shown in figure 3.2.

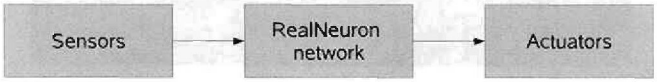


Figure 3.2: Context diagram for a RealNeuron network

If the RealNeuron network is regarded as a system then it consists of RealNeurons, i.e. motor neurons, internal neurons and sensor neuron. The RealNeuron context diagram is shown in figure 3.3.

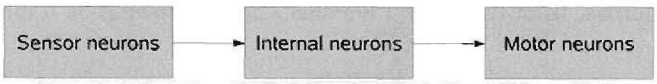


Figure 3.3: Context diagram for RealNeurons

An example of a biological neural network is shown in figure 3.4. This network consists of cortical neurons (nerve cells) that grow in a culture. In the top left-hand corner of figure 3.4 a large, round, textured, astrocytic glial cell (neuroglial cell) is visible. Neuroglial cells supports cells in the nervous system. Neurons usually have a large cellular body with one thick axon and several thinner dendrites extending from it. The axon carries nerve impulses away from the neuron. Its branching ends (bulb ends) make contact with other neurons and with muscles or glands. The highly branching dendrites receive information from other neurons [94].

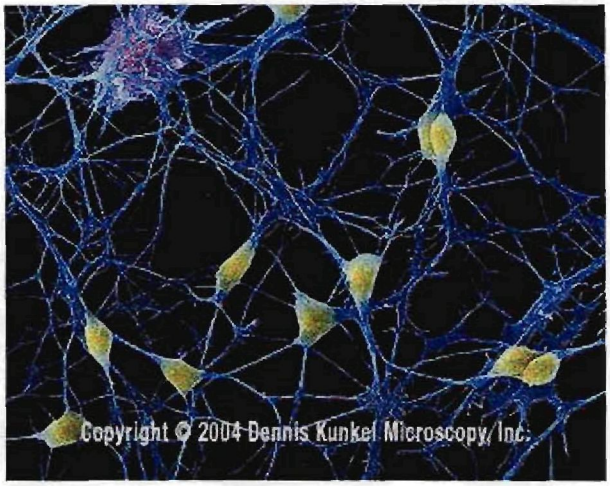


Figure 3.4: Biological neural network, provided by [94]

A biological neural network has different types of neurons. They all carry electrochemical nerve signals, but differ in structure, i.e. number of synapses, and are found in different parts of the body [46]. Two types of neurons are the pyramidal neuron in figure 3.5, and the Purkinje cells in figure 3.6. Figure 3.5 shows the pyramidal neurons, from the central nervous system, growing on a glial/fibroblast matrix.

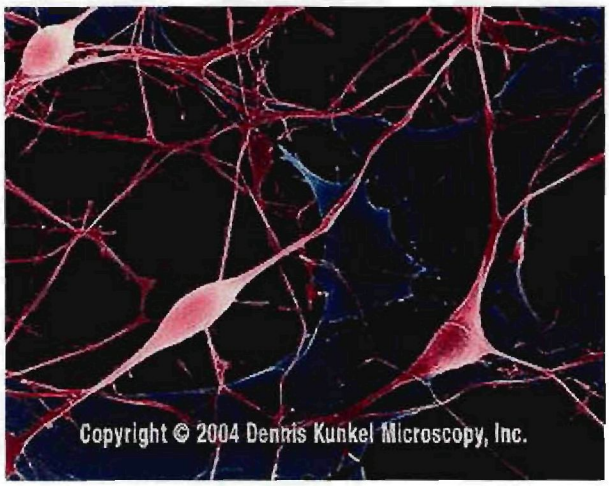


Figure 3.5: Biological pyramidal neurons, provided by [95]

Ramón y Cajal made the sketch in figure 3.6 of the Purkinje cells (A) and granule cells (B) from a pigeon cerebellum in 1899. Ramón y Cajal first revealed the structure of individual neurons and made the point that the complex structure of the brain depends on individual cells and their connections [13].

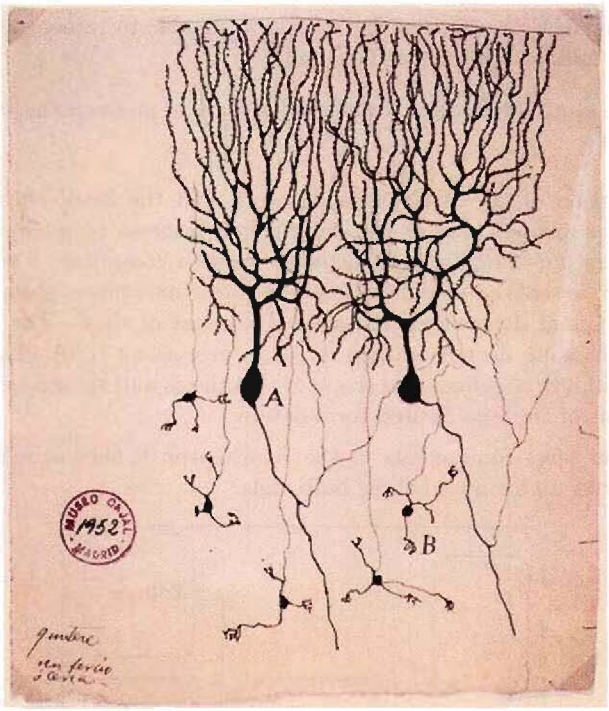


Figure 3.6: Purkinje cells (A) and granule cells (B), [159]

Sensory neurons (sensor neurons) are the interfaces between the body's sensors (eyes, ears, etc.) and the central nervous system. Motoneurons (motor neurons) carry signals from the central nervous system and glands. Interneurons (internal neurons) form all the neural wiring within the central nervous system. These different biological neurons are modelled with RealNeurons as a sensor neuron, motor neuron and internal neuron. The system context diagram for a RealNeuron network is shown in figure 3.3.

In figure 2.1, a typical biological neuron is shown. The three main components identified in this picture is the soma, axon and bulb ends. Based on this and other descriptions of biological neurons [13, 41, 125, 137, 158], it is deduced that a RealNeuron is a system that consists of the following three subsystems or components [51, 50]:

- one soma (S),
- no or one axon (A), and
- no or several presynaptic terminals or bulb ends (Υ).

Part of the input synapses for the biological neuron is at its dendrites [13, 41, 125, 137, 158], figure 2.1 on p. 20. By examining the descriptions of dendrites [13, 41, 125, 137, 158], it can be concluded that for information processing purposes, the functions for dendrites are:

- increasing of the surface of the neuron to be able to receive several thousand or even millions of input signals, and
- delaying of grouped input signals before arrival at the soma by using spatial placement.

In the description of the RealNeuron network and the RealNeuron interconnections, dendrites are not part of the description, because no geometrical (spatial) restrictions exist for a RealNeuron network inside a computer. The spacial signal processing can be realized by defining post-synaptic membrane groups that are processed together and delayed for the desired amount of time. The descriptions of classical conditioning do not address dendrite processing [110]. Thus, the spatial processing of inputs is outside the scope of this thesis and should be considered for future revisions of the RealNeuron formulation.

The next lower level components of the RealNeuron is shown in figure 3.7 for a RealNeuron with an axon and three bulb ends.

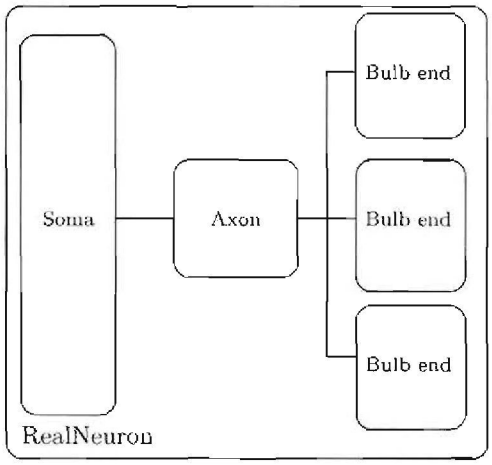


Figure 3.7: Components of the RealNeuron

Based on the descriptions of biological neurons [13, 41, 125, 137, 158], it is deduced that the above-mentioned three groups of components consist of different membrane types. The following two membrane types are distinguished inside the RealNeuron (N):

- post-synaptic membrane (M_p).
- main membrane (M_m).

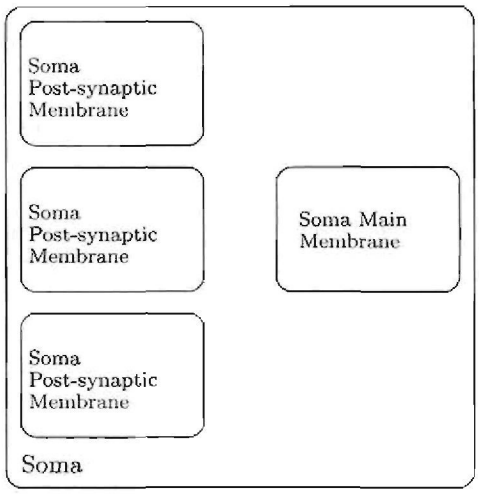
Two types of main membrane are distinguished:

- The soma main membrane ($M_{m\mathcal{S}}$), also known as the axon hillock.
- The bulb-end main membrane ($M_{m\Upsilon}$). This has no biological equivalent, see appendix E p. 181 for an explanation.

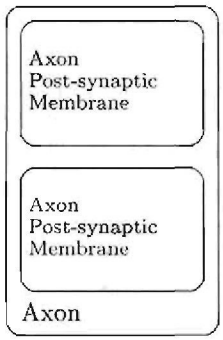
Based on the biological descriptions [41], it is deduced that three types of post-synaptic membranes can be distinguished, namely

- soma post-synaptic membrane (M_{pS}),
- axon post-synaptic membrane (M_{pA}), and
- bulb-end post-synaptic membrane ($M_{p\Upsilon}$).

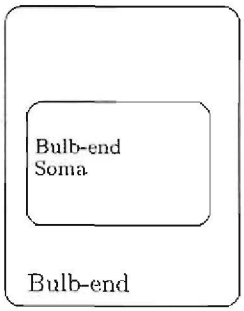
The RealNeuron components consist of many different components themselves. Figure 3.8(a) shows the membrane components of a soma, figure 3.8(b) shows the membrane components of an axon, and figure 3.8(c) shows some of the membrane components of a bulb end.



(a) Some of the components of the soma



(b) Some of the components of the axon



(c) Some of the components of the bulb end

Figure 3.8: RealNeuron Components

It is deduced from biological neurons [13, 41, 125, 137, 158] that the membranes consist of:

- pump (P),
- voltage-gated channel (C_v),
- receptor-gated channel (C_τ), or
- concentration channel (C_c)

for regulating the different ion species concentration at a specific cell membrane. The ion concentrations are used for the calculation of the corresponding electrical potential in the main membrane.

A membrane can also have metabolic receptors (R) that cause functional changes inside membranes and somas [13, 41, 125, 137, 158].

The bulb end also has vesicles. The vesicles release neurotransmitters into the synaptic gap [13, 41, 125, 137, 158], if the potential value in mV of the bulb end rises above a certain threshold.

These are the main components that contribute to the concept of the RealNeuron. Other components will be introduced, as needed, in this thesis.

The RealNeuron response and threshold functions are implemented through the simulation of the different brain cell subcomponents that are similar to the biological cell subcomponents described in [13, 41, 125, 137, 158]. Hence, the model simulates the mechanism of a main membrane, which generates an action potential that propagates through the axon, followed by several bulb ends. Each bulb end is followed by a synaptic gap and a post-synaptic membrane.

3.3.1 Action potential

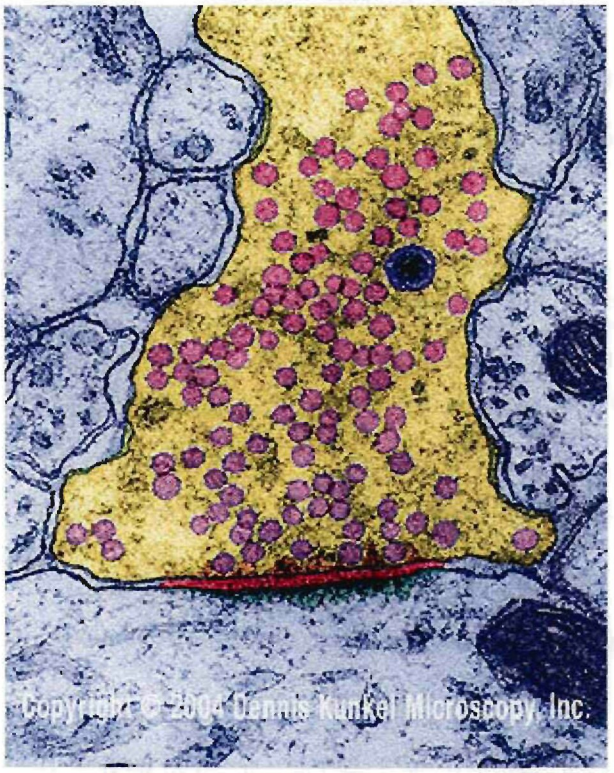
Messages are conveyed through the central nervous system through the propagation of action potentials in the nerve fibres (axons). An action potential is generated in the axon hillock. Once the threshold potential is exceeded in the axon hillock, it is propagated along the axon towards the bulb ends [13, 41, 125, 137, 158].

Both the generation of the action potential (AP) and threshold function are implemented as part of the main membrane inside the RealNeuron. The membranes are simulated with respect to the electrical potential value in mV, the timing of the processes and the concentrations of chemical substances involved.

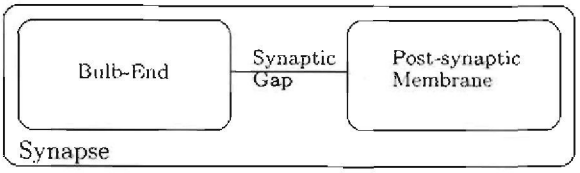
3.3.2 The synapse

The connection between two neurons is called a synapse. The biological synaptic zone, as seen under a microscope, is shown in figure 3.9(a). Take note of the vesicles and single dark dense core vesicle in the bulb end. The RealNeuron equivalent is shown in figure 3.9(b).

A synapse is a system that consists of a bulb end, a synaptic gap, and a post-synaptic membrane [13, 41, 125, 137, 158], see also figure 3.9(a). In an artificial neural network, the synaptic strengths are represented by fixed weights [68]. In a RealNeuron network, the synaptic strengths could change temporarily based on certain events inside the RealNeuron network.



(a) Excitatory synapse from the central nervous system, provided by [94]



(b) The components of a synapse

Figure 3.9: The synapse

3.3.3 Electrochemistry

It is required to calculate the potential value in mV of certain membranes in the RealNeuron. The potential value in mV is based on the ion concentration at the membrane. For biological membranes, the ion concentration's calculation at the membranes is very well described [28, 41, 73, 72, 125, 137, 158]. The Nernst equation is used to calculate the individual contribution of an ion species at a membrane and the Goldmann equation is used to calculate the nett effect of sodium ions (Na^+), potassium ions (K^+) and chlorine ions (Cl^-).

3.3.4 Nernst equation

The Nernst equation is defined in [128] as:

$$E = E^\theta - \frac{RT}{zF} \ln Q$$

where E is the exact effect of concentrations upon a potential in Volt (V), E^θ is the standard potential in V, R is the molar gas constant in Joule per Kelvin per mol (J/K/mol), T is the absolute temperature in Kelvin (K), z is the valence number, and F is the Faraday number in Coulomb per mol (C/mol).

The potential difference over a membrane for a specific ion species σ will then be:

$$E_\sigma = E_i - E_e$$

with E_i the internal potential and E_e the external potential at the cell membrane in V. Thus,

$$E_\sigma = E^\theta - \frac{RT}{z_\sigma F} \ln[\sigma]_i - E^\theta + \frac{RT}{z_\sigma F} \ln[\sigma]_e$$

and this simplifies to the modified Nernst equation for rest potentials:

$$E_\sigma = \frac{RT}{z_\sigma F} \ln\left(\frac{[\sigma]_e}{[\sigma]_i}\right)$$

In biology, the cell potential is expressed in millivolt (mV) thus, the modified Nernst equation for rest potentials in biological cells is:

$$\begin{aligned} E_\sigma &= \frac{1000RT}{z_\sigma F} \ln\left(\frac{[\sigma]_e}{[\sigma]_i}\right) \\ &= \frac{1000 \times 8.31451\,T}{z_\sigma 96485} \ln\left(\frac{[\sigma]_e}{[\sigma]_i}\right) \\ &= \frac{0.086174T}{z_\sigma} \ln\left(\frac{[\sigma]_e}{[\sigma]_i}\right) \qquad (3.3) \\ &= \frac{0.198423T}{z_\sigma} \log\left(\frac{[\sigma]_e}{[\sigma]_i}\right) \qquad (3.4) \end{aligned}$$

with the units of E_σ in mV, z_σ is the valence of the ion and:

$$\sigma \in \Gamma = \{s \mid \forall s,\ s \text{ is a membrane substance}\}$$

It is also needed to calculate the inverse for the Nerust equation. This is done by:

$$\begin{aligned} [\sigma]_i &= [\sigma]_e e^{-\frac{z_\sigma E_\sigma}{0.086174\,T}} \qquad (3.5) \\ &= [\sigma]_e 10^{-\frac{z_\sigma E_\sigma}{0.198423\,T}} \qquad (3.6) \end{aligned}$$

3.3.5 Goldmann equation

The combination of the different ion species in a membrane is calculated by using the Goldman equation [28, 41, 125, 137]. The Goldmann equation is given by:

$$E_M = \frac{RT}{F} \ln \left(\frac{P_K[K^+]_e + P_{Na}[Na^+]_e + P_{Cl}[Cl^-]_i}{P_K[K^+]_i + P_{Na}[Na^+]_i + P_{Cl}[Cl^-]_e} \right) \tag{3.7}$$

where

R is the universal gas constant (8.31451 J/K/mol),

F is the Faraday number (96485 C/mol).

T is the temperature in K,

P_K is the membrane permeability to potassium ions in m/s [125],

P_{Na} is the membrane permeability to sodium ions in m/s [125],

P_{Cl} is the membrane permeability to chlorine ions in m/s [125],

$[K^+]_i$ is the internal concentration of potassium ions,

$[Na^+]_i$ is the internal concentration of sodium ions in mol/l,

$[Cl^-]_i$ is the internal concentration of chlorine ions in mol/l.

$[K^+]_e$ is the external concentration of potassium ions in mol/l,

$[Na^+]_e$ is the external concentration of sodium ions in mol/l, and

$[Cl^-]_e$ is the external concentration of chlorine ions in mol/l.

Some texts use $\frac{k_b}{q_e}$, with k_b the Boltzmann constant (1.3806e-23 J/K) and q_e the electron charge (1.6022e-19 C), instead of $\frac{R}{F}$. It should be noted that $\frac{R}{F} = \frac{k_b}{q_e}$.

Another way of calculating the net effect at a membrane is discussed in [125] as part of the explanation of the Goldmann equation:

$$E_M = \frac{g_K}{g_K + g_{Na} + g_{Cl}} E_{K^+} + \frac{g_{Na}}{g_K + g_{Na} + g_{Cl}} E_{Na^+} + \frac{g_{Cl}}{g_K + g_{Na} + g_{Cl}} E_{Cl^-} \tag{3.8}$$

with

$$\begin{aligned} g_\sigma &= \frac{F^3}{(RT)^2} \frac{E_M P_\sigma [\sigma]_e}{\frac{E_M F}{e^{-RT}-1}} \\ &= \left(\frac{F}{RT} \right)^2 P_\sigma [\sigma]_e (e^{-RT} - 1) \\ &\quad \forall \sigma \in \{K^+, Na^+, Cl^-\} \end{aligned} \tag{3.9}$$

with g_σ in Siemens per square meter (S/m^2). By substituting (3.9) into (3.8):

$$\begin{aligned} E_M = & \frac{P_K[K^+]_e}{P_K[K^+]_e + P_{Na}[Na^+]_e + P_{Cl}[Cl^-]_e} E_{K^+} \\ + & \frac{P_{Na}[Na^+]_e}{P_K[K^+]_e + P_{Na}[Na^+]_e + P_{Cl}[Cl^-]_e} E_{Na^+} \\ + & \frac{P_{Cl}[Cl^-]_e}{P_K[K^+]_e + P_{Na}[Na^+]_e + P_{Cl}[Cl^-]_e} E_{Cl^-} \end{aligned} \tag{3.10}$$

It is assumed that the proportional change in $E_m(t)$ from $Em(t^-)$ is the same for the individual potentials E_K, E_{Na}, E_{Cl} for calculating the inverse of (3.8). Thus,

$$\begin{aligned} E_K(t) &= \frac{E_M(t)}{E_M(t^-)} E_K(t^-) \\ E_{Na}(t) &= \frac{E_M(t)}{E_M(t^-)} E_{Na}(t^-) \\ E_{Cl}(t) &= \frac{E_M(t)}{E_M(t^-)} E_{Cl}(t^-) \end{aligned} \tag{3.11}$$

3.4 State space theory

States describe the condition of a system, and provide the information which (together with a knowledge of the equations describing the system) enables the calculation of the future behaviour from a knowledge of the inputs [10]. In this thesis, the parameters and states of systems are not only scalars or vectors with components as elements of real numbers, but it can also be other elements from general topological spaces, e.g. string spaces describing molecules, etc. This means that the $+, -, \times$ operators are also adapted accordingly for the respective topological spaces.

3.4.1 Continuous systems

A linear dynamic system can be described by the following set of equations

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{3.12}$$

where u is the input states to the system described in (3.12), y is the output states from the system, and x is the internal states of the system [59]. The matrices A, B, C, and D are the linear operators of the transfer function of the system. The application of definition 7 on (3.12) results in:

$$\begin{aligned} \mathbb{S} &= < u, x, y, t > \\ \sigma_f &= < A, B, C, D > \end{aligned}$$

Thus, the linear system in (3.12) is a simple system with 2-tuple structure[1]:

$$< x_1, x_2 > = << u, x, y, t >, < A, B, C, D >> .$$

The more general description of a system is:

$$\begin{aligned} \dot{x} &= f(x, u, t) \\ y &= h(x, u, t) \end{aligned} \tag{3.13}$$

where f is the mapping between the input sates u and the internal states x and h is the mapping between the internal states x, the input states u and the output states y.

[1] See definition 7.

The generalised system in (3.13) is rewritten as the system described by parameterised operators:

$$\begin{aligned} \dot{x} &= F(\nu)\begin{bmatrix} x \\ u \\ t \end{bmatrix} \\ y &= H(\omega)\begin{bmatrix} x \\ u \\ t \end{bmatrix}. \end{aligned} \tag{3.14}$$

The system description in (3.14) can also be expressed as:

$$\begin{aligned} x &= \int F(\nu)\begin{bmatrix} x \\ u \\ t \end{bmatrix} dt \\ &= F_I(\nu)\begin{bmatrix} x \\ u \\ t \end{bmatrix} \\ y &= H(\omega)\begin{bmatrix} x \\ u \\ t \end{bmatrix}. \end{aligned} \tag{3.15}$$

where $F_I = \int F$. In a more formal way it can be said that F_I and H are mappings defined as:

$$\begin{aligned} F_I : X \times U \times \mathbf{T} &\rightarrow X \\ H : X \times U \times \mathbf{T} &\rightarrow Y \end{aligned} \tag{3.16}$$

where X is the internal state space, U is the input state space, $\mathbf{T} \subset \mathbf{R}^+$ is the time space and Y is the output state space.

The variables y, x, u, t are the states of the system (3.15) and the variables ν, ω are the descriptive parameters of the system (3.15). The states are implemented as descriptive parameters in different data structures. This enables optimisation in the future by varying ν and ω.

The application of definition 7 on (3.16) results in:

$$\begin{aligned} \mathbb{S} &= < Y, X, U, \mathbf{T} > \\ \sigma_f &= < F_I, \nu, H, \omega > \end{aligned}$$

Thus, (3.16) is a system as per definition 6 and a simple system per definition 7 with the following 2-tuple structure:

$$< x_1, x_2 > = << Y, X, U, \mathbf{T} >, < F_I, \nu, H, \omega >> .$$

3.4.2 Discrete systems

Assume a continuous function $f(t)$ that is sampled at time intervals of $\Delta t > 0$, then the function $F(t) = \sum_{k=-\infty}^{\infty} f(t)\delta(t - k\Delta t)$, where $\delta(t)$ is the Dirac delta function. If $t_k = k\Delta t$, it follows that $F(t_k) = f(t_k)$ and for any $\varepsilon < \Delta t$, $F(t_k \pm \varepsilon) = 0$.

The following notation is introduced:

$$Z^{-1}F(t_k) = F(t_k - \Delta t)$$

$$Z^{-n}F(t_k) = F(t_k - n\Delta t)$$

For a fixed sampling period the following notation $F(t_k) = F(k)$ is adopted and without the loss of generality, it could be assumed that $\Delta t = 1$, or:

$$Z^{-1}F(k) = F(k-1)$$
$$Z^{-n}F(k) = F(k-n)$$

3.5 RealNeuron notation

The notation is presented by an example. The output voltage of an output neuron is determined as follows:
Start with the neuron N, that is an output neuron N_y, and specifically the output of the output neuron Y_{N_y}, then finally refer to the voltage $E_{Y_{N_y}}$. This notation maps to the software as `Neuron_y.Y.E`.

If the component i of a vector is referred to, then it is indicated by $E^{(i)}_{Y_{N_y}}$. This notation maps to the software as `Neuron_y.Y.E[i]`.

The general algorithm for identifying a state is:

1. Identify the object O that you are working with.
2. Identify the subclass S of the object, if any.
3. Identify what type of state of interest T is.
4. Identify the name of the state of interest N.
5. Identify the index i of the vector, if any.

From the above-mentioned algorithm the state identifier is $N^{(i)}_{T_{O_S}}$. The following is also indicated by using the above algorithm and stops at the step of interest:

- $T_{O^{(i)}_S}$ - The type of state T of object vector O_S's i'th component (in software `O_S[i].T`).
- T_O - The type of state T of object O (in software `O.T`).
- O_S - The object O of subclass S (in software `O_S`).
- O - The general object of type O or the object itself (in software `O`).

3.6 Summary

A definition for a formal theory has been given using the set theory notation in §3.1. A formal definition has been presented for a system breakdown structure in §3.2, using the formal theory definition. Based on the formal theory, the predicate logic model for the RealNeuron network is developed further in the next chapter.

An overview of the RealNeuron structure is presented in §3.3. This structure can be summarised with the system breakdown structure in figure 3.6. All models in the next chapter are based on this structural description.

A structure for state space system models is defined in §3.4.

The notation on how to label the subcomponents of the RealNeuron is presented in 3.5. The state space model for a RealNeuron network is developed further in the next chapter.

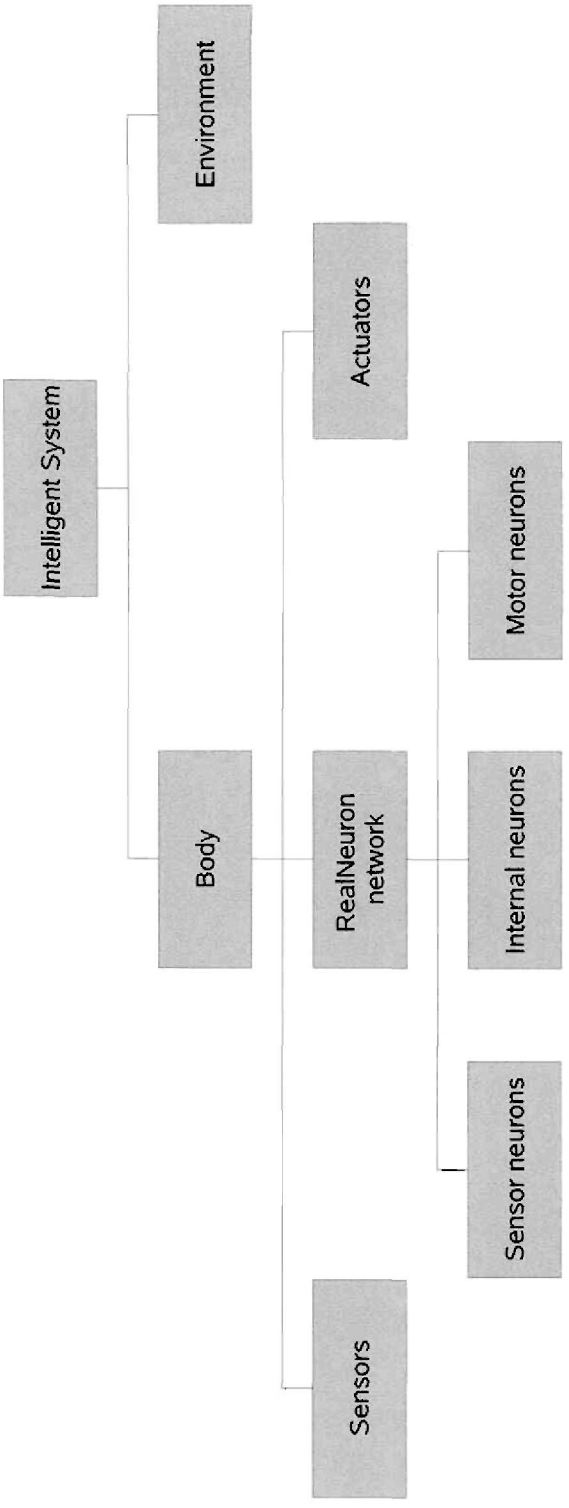


Figure 3.10: An intelligent system overview

Chapter 4

RealNeuron models

Equations are more important to me, because politics is for the present, but an equation is something for eternity — Albert Einstein [1]

This chapter develops the detail for the RealNeuron network system level and lower system levels in figure 3.6. This detail is developed by establishing models, that extract the information-processing capabilities of the biological neurons, in §4.2, §4.3 and §4.4.

With respect to the scientific and engineering way of life (§1.7.1.1 p. 8 and §1.7.1.2 p. 9), this chapter provides the different models (theory) for the RealNeuron. According to definition 1, the models in this chapter are still only a *pure theory*. The theory is presented in set predicates, block diagrams and a state space model.

The explicit contribution of this thesis is:

- The completion and standardisation of terminology and symbology of a predicate description of the RealNeuron in §4.2 by applying the definitions 4 to 7 in §3.1 on pp. 30-31, and the naming convention described in §3.5 on p. 44.
- The block diagrams in §4.4.
- The development of a state space model in §4.4 based on §3.4. The message processing (or signal processing) is modelled directly with difference equations.

4.1 Different types of models

From the author's experience as a practising engineer, different model types emphasise different aspects. In the modelling of complex systems, the use of different models, to emphasise different aspects of a system, is crucial in the understanding of the structure and dynamics of a system. The choice of model types is generally driven by the problem to be solved as well as the experience and preferences of the modellers.

In this thesis the following three types of models are developed for a RealNeuron network:

- Predicate model.

[1] In Stephen Hawking *A Brief History of Time* (1998). See also C.P. Snow 'Einstein' in M. Goldsmith et.al. (eds.) *Einstein* (1980).

- Block diagram model.
- State space model.

These different models provide different insights into the identification and simplification of RealNeuron networks.

4.1.1 Predicate model

Set predicates are used, because it is how the structures were originally described. This type of model provides an easy and clear way for formulating complex structures. The direct relation that this method has with a formal theory, see §3.1, is a great advantage, because the model is the formal theory of the described structure.

The predicate model can be directly translated into an XML description, see Appendix C for an example, which makes programming simpler by using a standard XML-parser library, e.g. Xerces [2], to parse a RealNeuron network's text description.

4.1.2 Block diagram model

Block diagrams are very closely related to signal flow diagrams. See [55, 119], for a discussion on the subject. Block diagrams are used in this thesis.

This type of model makes the visualisation of the causal processing of information for a system easier. It provides a visual picture of the data flow in the model. In future revisions of the RealNeuron model, the block diagram model should be used to simplify the causal processing of the membranes and their subcomponents. This topic has led to major discussions in the past at Knowbotic Systems.

The identification of patterns, inside the RealNeuron model, is also made easier with block diagrams. In future revisions of the RealNeuron model, these patterns can lead to simpler implementations by using an object-orientated language such as C++ or Java.

Block diagrams can be easily modelled, almost one-to-one, in Simulink/Matlab or Scicos/Scilab mathematical modelling software packages.

4.1.3 State space model

The state space model together with the block diagrams make it possible to implement RealNeuron networks using several computer languages or software packages that include amongst others C++, Scilab, OpenOffice.org Calc, etc. The state space model is implemented by using systems engineering analysis' different levels of resolution [4, 44].

By using the notation in 3.4, the parameters can be distinguished more easily from the states of the models. The time domain is a discrete domain. This is similar to the discrete state space formulation used in digital control systems, see [59], but with some states as strings.

In future revisions of the RealNeuron model, optimisation can be used to implement long-term memory by varying the descriptive parameters ν and ω in (3.15) on p. 43.

[2] Available on the Internet at http://xml.apache.org/

4.2 Predicate model

This section focuses on the detail structure for the RealNeuron network system level and lower system levels, mentioned in figure 3.6, deduced from the biological description of neurons in [13, 41, 125, 158].

The predicate model has been completed and the terminology and symbology have been standardised as well, by applying the definitions 4 to 7 in §3.1 on pp. 30-31 and the naming convention described in §3.5 on p. 44. The predicate model of the RealNeuron is based on the work done in [30, 36, 32] as mentioned in §1.7.1 on p. 8.

4.2.1 RealNeuron network

The RealNeuron network (Ψ) forms part of the body as seen in the system breakdown structure in figure 3.6 on p. 45. A RealNeuron network consists of RealNeurons (N) that are interconnected to each other. Some of the RealNeurons are connected to sensory states ($\mathbb{S}_{\varpi}$) and effectory states ($\mathbb{S}_{\varphi}$) in the body (B), see definition 9 on p. 31. The sensory states and effectory states are part of local RealNeuron network states ($\mathbb{S}_{\Psi}$), thus $\mathbb{S}_{\varpi}, \mathbb{S}_{\varphi} \subset \mathbb{S}_{\Psi}$.

Definition 10 (RealNeuron network (Ψ))

$$
\begin{aligned}
&RNNetworks \subset Systems \quad \wedge \\
&RNNetworks = \{x | REALNEURALNETWORK(x)\} \quad \wedge \\
&REALNEURALNETWORK(x) \iff \\
&obj(x) =< N, \mathbb{S}_{\varpi}, \mathbb{S}_{\varphi}, \mathbb{S}_{\Psi} > \quad \wedge \\
&rel(x) =< \sigma_{\Psi} > \quad \wedge \\
&N =< N_1, ..., N_n > \quad \wedge \\
&\sigma_{\Psi} : \mathbf{2}^{\mathbb{S}_{\varpi}} \times \mathbf{2}^{N} \times \mathbf{2}^{\mathbb{S}_{\Psi}} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_{\varphi}} \times \mathbf{2}^{N} \times \mathbf{2}^{\mathbb{S}_{\Psi}}
\end{aligned}
$$

with $\mathbb{T}$ *the time space.*

4.2.2 RealNeuron

A RealNeuron (N) can be a motor neuron, an internal neuron or an actuator neuron, as indicated in the system breakdown structure in figure 3.6 on p. 45.

Based on the biological description of a neuron in figure 2.1, it follows that in general, a RealNeuron (N) may consist of the following interconnected subsystems: A soma (S), an axon (A), and bulb ends (Υ). Because a RealNeuron is a dynamic system, local RealNeuron states ($\mathbb{S}_N$) also form part of the RealNeuron description.

Definition 11 (RealNeuron (N))

$$
\begin{aligned}
&RNeurons \subset Systems \quad \wedge \\
&RNeurons = \{x | REALNEURON(X)\} \quad \wedge \\
&REALNEURON(x) \iff \\
&obj(x) =< S, A, \Upsilon, \mathbb{S}_N > \quad \wedge \\
&rel(x) =< \sigma_N > \quad \wedge \\
&\Upsilon =< \Upsilon_1, ..., \Upsilon_n > \quad \wedge \\
&\sigma_N : \mathbf{2}^{\mathbb{S}_N} \times \mathbf{2}^{S} \times \mathbf{2}^{A} \times \mathbf{2}^{\Upsilon_1} \times ... \times \mathbf{2}^{\Upsilon_n} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_N} \times \mathbf{2}^{S} \times \mathbf{2}^{A} \times \mathbf{2}^{\Upsilon_1} \times ... \times \mathbf{2}^{\Upsilon_n}
\end{aligned}
$$

with $\mathbb{T}$ *the time space.*

4.2.3 Soma

In the description of the RealNeuron in [36, 31, 32, 145, 144], two types of soma exist, namely:

- input soma (S_i), and
- bulb-end soma (S_Υ).

Thus, $S_i \cup S_\Upsilon \subseteq S$.

The S_Υ has no biological equivalent and is introduced in [145, 144] to take advantage of the object-orientated data-definition capabilities of the Postgresql database manager[3], see appendix E.

From descriptions of the soma in [13, 41, 125, 158], only the information-processing components were extracted and deduced from [73, 72]. Thus, a soma consists of a set of post-synaptic membranes (to make things simple, for a sensoric neuron in the S_i and for a motor neuron in the S_Υ, this set is the empty set $\emptyset$), a soma integration function ($\mathcal{I}_S$) that integrates the potential of each post-synaptic membrane (M_p), a main membrane (M_m), an LTP function (μ) and the local soma states ($\$_S$).

Definition 12 (Soma)

$$
\begin{aligned}
& Soma \subset Systems \quad \wedge \\
& Soma = \{x | SOMA(x)\} \quad \wedge \\
& SOMA(x) \iff \\
& obj(x) =< M_p, M_m, \mathcal{I}_S, \mu, \$_S > \quad \wedge \\
& rel(x) =< \sigma_S > \quad \wedge \\
& M_p =< M_{p_1}, ... M_{p_n} > \quad \wedge \\
& \sigma_S : \mathbf{2}^{\$_S} \times \mathbf{2}^{\mu} \times \mathbf{2}^{\mathcal{I}_S} \times \mathbf{2}^{M_m} \times \mathbf{2}^{M_{p_1}} \times ... \times \mathbf{2}^{M_{p_n}} \times \mathbb{T} \rightarrow \\
& \qquad \mathbf{2}^{\$_S} \times \mathbf{2}^{\mu} \times \mathbf{2}^{\mathcal{I}_S} \times \mathbf{2}^{M_m} \times \mathbf{2}^{M_{p_1}} \times ... \times \mathbf{2}^{M_{p_n}}
\end{aligned}
$$

with $\mathbb{T}$ *the time space.*

4.2.4 Axon

An axon consists of a list of compartments L_A described by axonal membranes (M_a) and the local axon states ($\$_A$). This is deduced from the descriptions of a biological axon in [13, 41, 125, 158]. The action potential propagation through the axon can be implemented by linking a couple of the compartments together and propagating the action potential from compartment to compartment.

Definition 13 (Axon)

$$
\begin{aligned}
& Axons \subset Systems \quad \wedge \\
& Axons = \{x | AXON(x)\} \quad \wedge \\
& AXON(x) \iff \\
& obj(x) =< L_A, \$_A > \quad \wedge \\
& rel(x) =< \sigma_A > \quad \wedge \\
& L_A =< M_{a1}, ..., M_{an} > \quad \wedge \\
& \sigma_A : \mathbf{2}^{\$_A} \times \mathbf{2}^{M_{a1}} \times ... \times \mathbf{2}^{M_{an}} \times \mathbb{T} \rightarrow \mathbf{2}^{\$_A} \times \mathbf{2}^{M_{a1}} \times ... \times \mathbf{2}^{M_{an}}
\end{aligned}
$$

[3] http://www.postgresql.org

with $\mathbb{T}$ *the time space.*

4.2.5 Bulb end

A bulb end (Υ) consists of a soma (S), a threshold potential (π_Υ), a neurotransmitter (τ_Υ) that will be released into the synapse, and the local bulb-end states ($\mathbb{S}_\Upsilon$). The elements of the bulb end are deduced from the descriptions of the biological bulb end in [13, 41, 125, 158]. As mentioned and motivated in appendix E, no biological equivalent exists for the soma (S) structure in the bulb end.

Definition 14 (Bulb end)

$$
\begin{aligned}
&BulbEnds \subset Systems \quad \wedge \\
&BulbEnds = \{x | BULBEND(x)\} \quad \wedge \\
&BULBEND(x) \iff \\
&obj(x) =< S_\Upsilon, \tau_\Upsilon, \pi_\Upsilon, \mathbb{S}_\Upsilon > \quad \wedge \\
&rel(x) =< \sigma_\Upsilon > \quad \wedge \\
&\sigma_\Upsilon : \mathbf{2}^{S_\Upsilon} \times \{\tau_\Upsilon\} \times \{\pi_\Upsilon\} \times \mathbf{2}^{\mathbb{S}_\Upsilon} \times \mathbb{T} \rightarrow \mathbf{2}^{S_\Upsilon} \times \{\tau_\Upsilon\} \times \{\pi_\Upsilon\} \times \mathbf{2}^{\mathbb{S}_\Upsilon}
\end{aligned}
$$

with $\mathbb{T}$ *the time space.*

It is possible to substitute τ_Υ with the tuple $(\tau_\Upsilon, \omega_\Upsilon)$, where ω_Υ is the released neurotransmitter concentration. This would define the synaptic strengths between RealNeurons based on the concentrations of released neurotransmitters. The implementation of this is outside the scope of this thesis and meant for a future revision of the RealNeuron.

4.2.6 Membranes

A membrane (M) consists of channels (C), pumps (P), metabolic receptors (R), ionic species (σ), and local membrane states ($\mathbb{S}_M$), deduced from the descriptions of biological membranes in [13, 41, 73, 72, 125, 158]. The channels and pumps make the exchange of ionic species possible. The metabolic receptors trigger chemical processes inside the soma. It should be noted that $M_p \cup M_a \cup M_m \subseteq M$.

Definition 15 (Membrane)

$$
\begin{aligned}
&Membranes \subset Systems \quad \wedge \\
&Membranes = \{x | MEMBRANE(x)\} \quad \wedge \\
&MEMBRANE(x) \iff \\
&obj(x) =< C, P, R, \mathbb{S}_M > \quad \wedge \\
&rel(x) =< \sigma_M > \quad \wedge \\
&\sigma_M : \mathbf{2}^C \times \mathbf{2}^P \times \mathbf{2}^R \times \mathbf{2}^{\mathbb{S}_M} \times \mathbb{T} \rightarrow \mathbf{2}^C \times \mathbf{2}^P \times \mathbf{2}^R \times \mathbf{2}^{\mathbb{S}_M}
\end{aligned}
$$

with $\mathbb{T}$ *the time space.*

4.2.7 Ion channels

A channel (C) specifies the specific ion species (σ) that it will let through, the channel capacity (γ) in mmol/time step, the channel type or channel control signature (δ), and the local channel states ($\mathbb{S}_C$), deduced from the description of biological channels in [41, 125, 158].

Definition 16 (Channel)

$$
\begin{aligned}
& Channels \subset Systems \quad \wedge \\
& Channels = \{x | CHANNEL(x)\} \quad \wedge \\
& CHANNEL(x) \iff \\
& obj(x) =< \sigma, \gamma, \delta, \mathbb{S}_C > \quad \wedge \\
& rel(x) =< \sigma_C > \\
& \sigma_C : \{\gamma\} \times \mathbf{2}^{\delta} \times \mathbf{2}^{\mathbb{S}_C} \times \mathbb{T} \rightarrow \mathbf{2}^{\delta} \times \mathbf{2}^{\mathbb{S}_C}
\end{aligned}
$$

with $\mathbb{T}$ *the time space.*

The control signature δ in the above-mentioned definition determines the channel type as follows:

Definition 17 (Channel control signature)

$$\delta \in \{ < C_\pi, < \pi_c > >, < C_\theta, < \theta > >, < C_\tau, < \kappa(c), \mathcal{T}, b(\kappa,\tau), r(\kappa,\tau), o(c), t(c) > > \}$$

with

C_π - *Voltage-gated channel,*

π_c - *Threshold potential,*

C_τ - *Neurotransmitter-gated channel,*

$\mathcal{T} = \{<< \tau_1, \text{effect} >, ..., < \tau_n, \text{effect} >>\}$ - *neurotransmitters in the synaptic gap,*

τ_i - *Neurotransmitter molecules,*

$r(\kappa,\tau)$ - *openness function for the channel, returns 1 if* τ *is an agonist - the synapse is excitatory, 0 if* τ *is an antagonist - the synapse is inhibitory,*

C_θ - *Concentration-gated channel,*

θ - *Threshold concentration,*

$o(c)$ - *Current openness,*

$t(c)$ - *Current blocked time left,*

$b(\kappa,\tau)$ - *Time span that the channel is blocked, and*

$\kappa(c)$ - *Channel receptor molecule.*

with $\mathbb{T}$ *the time space.*

4.2.8 Ion pumps

A pump (P) specifies the specific ion species (σ) that it will let through, the pump capacity (γ) in mmol/time step, the pump direction (ϕ), the threshold concentration (θ) in mmol, and the local pump states ($\mathbb{S}_P$), deduced from the description of biological pumps in [41, 125, 158].

Definition 18 (Pumps)

$$
\begin{aligned}
& Pumps \subset Systems \quad \wedge \\
& Pumps = \{x | PUMP(x)\} \quad \wedge \\
& PUMP(x) \iff \\
& obj(x) =< \sigma, \gamma, \theta, \phi, \mathbb{S}_P > \quad \wedge \\
& rel(x) =< \sigma_P > \\
& \sigma_P : \{\gamma\} \times \{\theta\} \times \{\phi\} \times \mathbf{2}^{\mathbb{S}_P} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_P}
\end{aligned}
$$

with $\mathbb{T}$ *the time space.*

4.2.9 Metabolic receptors

A metabolic receptor (R) specifies the metabolic receptor molecule (ρ), the neurotransmitters in the synaptic gap ($\mathcal{T} = \{<< \tau_1, \text{effect} >, ..., < \tau_n, \text{effect} >>\}$), $\mathcal{T}(\rho) = \{\tau \in \mathcal{T} | \forall b(\rho, \tau) \text{ is defined}\}$, the neurotransmitter molecules (τ_i), the time duration for blocking the receptor ($b(\rho, \tau)$), the current blocked time left ($t(\rho)$), the receptor event flag (e_ρ), and the local metabolic receptor states ($\mathbb{S}_R$), deduced from the description of biological receptors in [41, 125, 158].

Definition 19 (Receptor)

$$
\begin{aligned}
& Receptors \subset Systems \quad \wedge \\
& Receptors = \{x | RECEPTOR(x)\} \quad \wedge \\
& RECEPTOR(x) \iff \\
& obj(x) =< \rho, \mathcal{T}, b(\rho, \tau), t(\rho), e_\rho, \mathbb{S}_R > \quad \wedge \\
& rel(x) =< \sigma_R > \quad \wedge \\
& \sigma_R : \{\rho\} \times \mathbf{2}^{\mathcal{T}} \times \{b(\rho, \tau)\} \times \{t(\rho)\} \times \mathbf{2}^{\mathbb{S}_R} \times \mathbb{T} \rightarrow \{t(\rho)\} \times \{e_\rho\} \times \mathbf{2}^{\mathbb{S}_R}
\end{aligned}
$$

with $\mathbb{T}$ *the time space.*

4.3 Message processing

The algorithmic description in [145, 144], is combined with the description in [36, 31, 32] to formulate message processing inside the RealNeuron.

For the processing of the description of messages, a top-down approach is used. Only the processing of exchanged messages inside the body is considered. This is not a restriction, because the externally exchanged messages use a predefined communication protocol to communicate with the environment. The communication protocol is implemented through the sensor and actuator objects that are part of the body definition.

By concentrating on the body system level in figure 3.6, three types of connections can be identified inside the body:

Body-to-Brain ($B \hookrightarrow \Psi$) The assumption is made that the body (through sensors) induces a changing potential measured in millivolt (mV) in the range $[-n_1, +n_2]$, $\forall n_1, n_2 \geq 0$.

$$sense : \mathbb{S}_{\varpi} \hookrightarrow M_p$$

with $\mathbb{S}_{\varpi} \subset \mathbb{R}^n$, and $\mathbb{S}_{\varpi} \subset \mathbb{S}_B$ with n the number of sensory states measured in mV.

Brain-to-Body ($\Psi \hookrightarrow B$) The body (through actuators) reacts to either changing electrical potentials measured in mV in the range $[-n_1, +n_2], \forall n_1, n_2 \geq 0$ or neurotransmitter (τ) molecules or hormones.

$$response : \Upsilon \hookrightarrow \mathbb{S}_{\varphi}$$

with $\mathbb{S}_{\varphi} \subset \mathbb{R}^{n-m} \times \tau^m$, and $\mathbb{S}_{\varphi} \subset \mathbb{S}_B$ with n the number of effectory states measured of which m states are measured in mV, and $n - m$ the number of neurotransmitter or hormone names expressed as character strings.

Brain-to-Brain ($\Psi \hookrightarrow \Psi$) In a synapse, the bulb end releases a neurotransmitter of a specific type and concentration into the synaptic gap that could dock onto a post-synaptic membrane.

$$synapticPulse : \Upsilon \hookrightarrow M_p$$

with the output of a bulb end measured as the released neurotransmitter (τ) name expressed as a character string.

4.3.1 RealNeuron message processing

As with the processing of classical neurons [68], the strategy, where a neuron forms the local computing unit, is described in this section. This can easily be expanded where the membranes are the local computing unit [145].

The time unit for each simulation step is one time step, see 3.4.2.

4.3.2 RealNeuron

Based on definition 11 and the descriptions for a biological neuron [13, 41, 73, 72, 125, 137, 158], let $N =< S_i, A, \mathcal{Y} >$ be a RealNeuron, with S_i the input soma, A is an axon, and $\mathcal{Y}$ is the set of bulb ends connected to A. For every membrane $m \in N$, $[\sigma]_e$ and $[\sigma]_i$ are the external and internal concentrations of the relevant ion species $\sigma \in \{Na^+, K^+, Cl^-\}$. Further, for all post-synaptic membranes $m_p \in N$, let $\mathcal{T}_{m_p}$ be the set of neurotransmitters in the synaptic gap at m_p.

1. **Soma processing:** Apply the input soma algorithm in §4.3.3 to determine the action potential $\mathcal{P}^{out}_{M_m}$.
2. **Axon processing:** Propagate the main membrane output potential $\mathcal{P}^{out}_{M_m}$ through the axon A by applying the axon processing algorithm in §4.3.4 to get the axon output potential $\mathcal{P}^{out}_{A}$.
3. **Bulb-end processing:** For all bulb ends $y \in Y$: Apply the bulb-end processing algorithm §4.3.5.

The output of N is the neurotransmitters: τ_y emitted by the bulb end, τ_{μ_y} emitted by the long-term potential (LTP) function of the bulb end, and τ_{μ_i} emitted by the LTP function of the input soma.

4.3.3 Soma

Based on definition 12 and the descriptions for a biological neuron [13, 41, 73, 72, 125, 137, 158], let $S =< M_p, M_m, \mu >$ be a soma, M_p is post-synaptic membranes, M_m is a main membrane, and if defined, μ is an LTP function. For every membrane $m \in N$, $[\sigma]_e$ and $[\sigma]_i$ are the external and internal concentrations of the relevant ion species $\sigma \in \{Na^+, K^+, Cl^-\}$. Further, for all post-synaptic membranes $m_p \in N$, let $\mathcal{T}_{m_p}$ be the set of neurotransmitters in the synaptic gap of m_p.

4.3.3.1 Input soma processing

1. **Post-synaptic membrane processing:** For all soma post-synaptic membranes (M_{pS}), $\forall m \in M_{pS}$, apply the membrane level algorithm in §4.3.6. Determine the output potentials $\mathcal{P}_m^{out}$ and the metabolic receptor event flag $e_\rho, \rho \in \mathcal{R}_m =< \rho_1, ..., \rho_k >$.

2. **Input soma integration processing:** Determine the input soma potential as the average output potential of the post-synaptic membranes

$$\mathcal{P}^{insoma} = \frac{1}{\#(M_{pS})} \sum_{\forall m \in M_{pS}} \mathcal{P}_m^{out}$$

   $\#(M_{pS})$ is the number of soma post-synaptic membranes

3. **LTP function processing:** If an LTP function $\mu_S =< < \rho_1, ..., \rho_k >, \tau_{\mu_S}, < m_1, ..., m_l > >$, $m_i \in M_{pS}$ exists, apply the LTP function algorithm in §4.3.9.

4. **Main membrane processing:** Use $\mathcal{P}^{insoma}$ as the input to the main membrane M_m and apply the membrane level algorithm in §4.3.6 to determine the output potential $\mathcal{P}_{M_{mS}}^{out}$.

The output is $\mathcal{P}_{M_{mS}}^{out}$, measured in mV.

4.3.3.2 Bulb-end soma processing

1. **Main membrane processing:** Apply the membrane level algorithm in §4.3.6 to determine the output potential $\mathcal{P}_{M_{m\Upsilon}}^{out}$.

2. **Post-synaptic membrane processing:** For all bulb-end post-synaptic membranes $(M_{p\Upsilon})$, $\forall m \in M_{p\Upsilon}$, apply the membrane level algorithm in §4.3.6 and determine the output potentials $\mathcal{P}_m^{out}$ and the receptor event flags $e_\rho, \rho \in \mathcal{R}_m =< \rho_1, ..., \rho_k >$.

3. **LTP function processing:** If an LTP function $\mu_\Upsilon =< < \rho_1, ..., \rho_k >, \tau_{\mu_\Upsilon}, < m_1, ..., m_l > >$, $m_i \in M_{p\Upsilon}$ exists, apply the LTP function algorithm in §4.3.9.

4. **Bulb-end soma integration processing:** Determine the bulb-end soma potential as

$$\mathcal{P}_\Upsilon = \frac{\mathcal{P}_{M_{m\Upsilon}}^{out} + \sum_{\forall m \in M_{p\Upsilon}} \mathcal{P}_m^{out}}{\#(M_{p\Upsilon}) + 1}$$

   $\#(M_{p\Upsilon})$ is the number of bulb-end post-synaptic membranes. The output is $\mathcal{P}_\Upsilon$, measured in mV.

4.3.4 Axon

Based on definition 13 and the descriptions for a biological neuron [13, 41, 73, 72, 125, 137, 158], let $A = L_A$ be the axon, where L_A is a list of compartments described as axonal membranes. These axonal membranes correspond to the post-synaptic membranes that are part of synapses at the nodes of Ranvier in a biological neuron[4], see figure 2.1 on p. 20. $L_A =< m_1, ..., m_n >$, $\forall m_i \in M_a$, and axonal membranes are ordered according to their neighbour relation, e.g. m_1 is the membrane closest to the axon hillock and receives its input from the soma main membrane (M_{mS}), m_n is the closest to the bulb ends, and m_k gets an input from m_{k-1}. Let $\mathcal{P}^{out}_{M_{mS}}$ be the output potential of M_{mS}.

1. Set $\mathcal{P}^{in} = \mathcal{P}^{out}_{M_{mS}}$.
2. Iterate over the list of compartments: $\forall k = 1...n$ do
   (a) **Axonal membrane processing:** Calculate the output potential $\mathcal{P}^{out}_{m_k}$ by applying the membrane level algorithm in §4.3.6 to the axonal membrane m_k with input potential $\mathcal{P}^{in}$ and transmitter substance set $\mathcal{T}_{m_k}$.
   (b) $\mathcal{P}^{in} = \mathcal{P}^{out}_{m_k}$.

$\mathcal{P}^{out}_A = \mathcal{P}^{out}_{m_n}$ is the output potential of the axon.

4.3.5 Bulb end

Based on definition 14 and the descriptions for a biological neuron [13, 41, 73, 72, 125, 137, 158], let $y =< S_\Upsilon, \tau_y, \pi_y >$ be a bulb end where $y \in \Upsilon$, τ_y is the emitted neurotransmitter by the vesicle, and π_y is a potential threshold.

1. **Soma processing:** Apply the bulb-end soma algorithm in §4.3.3 to determine the potential $\mathcal{P}_S$ with the input potential $\mathcal{P}^{out}_A$.
2. **Vesicle processing:** If $\mathcal{P}_S \geq \pi_y$ then emit the neurotransmitter τ_y.

The output is the neurotransmitter τ_y.

4.3.6 Membranes

Based on definition 15 and the descriptions for a biological neuron [13, 41, 125, 137, 158], there are different ways of interaction between the individual channels, pumps, receptors, and processes, e.g. if a receptor-gated channel opens because of a specific neurotransmitter in the synaptic gap, then it changes the ion-species concentrations at the membrane, which in turn influences the overall membrane potential as well as the activity of individual pumps. The change in membrane potential then influences the voltage-gated channels. The voltage-gated channels then change the ion-species concentrations at the membrane, which in turn influences the overall membrane potential as well as the activity of individual pumps.

The following assumptions are made to keep the model of the RealNeuron simple:

[4]Nodes of Ranvier are regularly spaced gaps in the myelin sheath around an axon. About one micrometer in length, these gaps expose the axonal membrane to the extracellular fluid. (The myelin sheath is the fatty tissue layer coating the axon [158, 125].)

- Only one pump is used to pump a specific substance at a specific membrane. This has no affect on the modelling of leak flows over the membrane, because another pump can be used to model this effect. Different pump rates at different ion-species concentration levels can be modelled by using more than one pump for a specific substance.
- Only one channel transports a specific substance (ion species) at a specific membrane. This has no affect on the modelling of any counter flow of molecules through the channel, another channel can be used to model this effect.
- Post-synaptic membranes do not have voltage-gated channels, because communication between neurons is by means of neurotransmitters.
- Main membranes do not have receptor-gated channels and metabolic receptors. A main membrane reacts on the integral result of all the post-synaptic membranes in a soma.
- Axonal membranes consist of voltage- as well as receptor-gated channels, but no receptors are included. The LTP function is implemented only as part of a soma.

Based on definition 15 and the descriptions for a biological neuron [13, 41, 125, 137, 158], let $m =< C, P, \mathcal{R} >$ be a membrane with $m \in M$, C is a set of channels, P is a set of pumps, $\mathcal{R}$ is a set of metabolic receptor molecules. $\Gamma = \{ \sigma \mid \forall c \in C,\ c = (\sigma, \gamma, \delta) \} \cup \{ \sigma \mid \forall p \in P,\ p = (\sigma, \gamma, \theta, \phi) \} \subseteq \{ Na^+,\ K^+,\ Cl^- \}$. For the ion species σ, let $[\sigma]_i$ and $[\sigma]_e$ be the internal and external concentration of $\sigma \in \Gamma$ for m respectively. Let $\mathcal{P}_m^{in}$ be an input potential (main membrane and axonal membrane), or $\mathcal{T}$ be the set of transmitter substances τ_y and τ_μ present in the synaptic gap of m, propagated by the bulb end and soma respectively. For $\rho \in \mathcal{R}$, let $t(\rho)$ be the number of time steps for which a blocked state lasts.

1. **Pump processing:** for all pumps $p \in P$, apply the pump algorithm described in §4.3.8.
2. If no $\mathcal{P}_m^{in}$ is given
   (a) **Receptor-gated channel processing:** $\forall c \in C$, apply the channel algorithm described in §4.3.7. The input parameters for the call are $[\sigma]_i$ and $[\sigma]_e$ for the ion species σ of channel $c = (\sigma, \gamma, \delta)$, the openness state $o(c)$, the set of neurotransmitter $\mathcal{T}_m$, and the blockness time $t(c)$.
   (b) **Metabolic receptor processing:** $\forall \rho \in \mathcal{R}$:
      i. If $t(\rho) > 0$ then $t(\rho) := t(\rho) - 1$ and the metabolic event flag is set, thus $e_\rho = true$.
      ii. Otherwise, if the set $\mathcal{T}(\rho) = \{ \tau \in \mathcal{T} \mid r(\rho, \tau) \text{ is defined} \} \neq \emptyset$, then randomly choose $\tau^* \in \mathcal{T}(\rho)$, $t(\rho) = b(\rho, \tau^*)$ and the metabolic receptor event flag is set, thus $e_\rho = true$.
3. Otherwise
   (a) **Nernst and Goldmann processing**: calculate the potentials $\mathcal{P}_{Nernst}^{t^-}(\sigma)$, $\sigma \in \Gamma$ and $\mathcal{P}_{Goldmann}^{t^-}(\Gamma)$ corresponding to the substance concentrations by applying the formulas of §3.3.4 and §3.3.5.
   (b) Integration with input signal by averaging:

$$\mathcal{P}_{Goldmann}^{t}(\Gamma) = \frac{\mathcal{P}_m^{in} + \mathcal{P}_{Goldmann}^{t^-}(\Gamma)}{2}$$

(c) If membrane m has voltage-gated channels:

Voltage-gated channel processing: $\forall c \in C$, apply the channel algorithm described in §4.3.7. The input parameters are $[\sigma]_i$ and $[\sigma]_e$ for the ion species σ of the channel $c = (\sigma, \gamma, \delta)$ and the input potential $\mathcal{P}^t_{Goldmann}(\Gamma)$.

(d) Otherwise, $\mathcal{P}^t_{Goldmann}(\Gamma)$ is mapped to the substance concentration equivalents:

i. **Inverse Goldmann and Nernst processing:** Using $\mathcal{P}^{t^-}_{Nernst}(\sigma)$, $\sigma \in \Gamma$ and $\mathcal{P}^{t^-}_{Goldmann}(\Gamma)$ as the starting values for the inverse Goldmann function, update the individual concentration $[\sigma]_i$ according to the inverse Goldmann and Nernst equations applied to $\mathcal{P}^t_{Goldmann}(\Gamma)$, using the algorithms in §3.3.4 and §3.3.5.

ii. **Receptor-gated channel processing:** as specified in 2a above.

4. **Nernst and Goldmann processing**: By using the newly calculated substance concentrations $[\sigma]_i$, $\sigma \in \Gamma$, determine the output potential $\mathcal{P}^{out}_m$ by applying formulas in §3.3.4 and §3.3.5.

The output of the membrane m is $\mathcal{P}^{out}_m$ and the metabolic receptor event flags e_ρ, $\rho \in R$.

4.3.7 Ion channels

Based on definition 15 and the descriptions for a biological neuron [13, 41, 125, 137, 158], let m be a membrane, $c =< \sigma, \gamma, \delta >$ be an ion channel of m, let $o(c) \in [0, 1]$ be the current degree of openness (currently $o(c)$ is restricted to open and close, $o(c) \in \{0, 1\}$), and $t(c) \in \mathbb{N} \cup 0$ is the number of simulation steps before a blocked state expires ($t(c) = 0$ means that c is not blocked). For

voltage-gated channels: Let $\mathcal{P}_m$ be the input potential at m, and

receptor-gated channels: Let $\mathcal{T}$ be the set of neurotransmitters τ_y (propagated by the bulb ends) and τ_μ (propagated by the somas) that are currently present in the synaptic gap.

Let $[\sigma]_i$ and $[\sigma]_e$ respectively be the internal and external concentrations of the ion species σ at m.

During a simulation step if, through δ, a receptor molecule κ is specified, then:

- if $t(c) > 0$, then $t(c) := t(c) - 1$. When $t(c) = 0$ then the channel is no longer blocked in the next simulation step,
- otherwise if the set $\mathcal{T}(\kappa) = \{\tau \in \mathcal{T} | r(\kappa, \tau) \wedge b(\kappa, \tau)\} \neq \emptyset$ of neurotransmitters fitting κ is not empty, then randomly choose a transmitter $\tau^* \in \mathcal{T}(\kappa)$. In the next simulation step, the channel is then blocked in the openness state $o(c) = r(\kappa, \tau^*)$ for the next $t(c) = b(\kappa, \tau^*)$ time steps,

otherwise if, through δ, a threshold potential π_c is specified, then the ion channel c will be open, $o(c) = 1$, if

- either π_c is a minimum threshold and $\mathcal{P}_m \geq \pi_c$,
- or π_c is a maximum threshold and $\mathcal{P}_m \leq \pi_c$.

otherwise if, through δ, a default concentration θ_c is specified, then the ion channel c will be open, $o(c) = 1$, if $[\sigma]_i \geq \theta_c$.

For all types of channels:

- if $[\sigma]_i > [\sigma]_e$ then $[\sigma]_i := [\sigma]_i - \gamma o(c)$,
- otherwise if$[\sigma]_i < [\sigma]_e$ then $[\sigma]_i := [\sigma]_i + \gamma o(c)$.

4.3.8 Ion pumps

Let m be a membrane, $p = (\sigma, \gamma, \theta, \phi)$ be a ion pump of m. Let $[\sigma]_i$ be internal concentrations of the ion species σ at m.

During a simulation step

- if $(\phi = in) \wedge ([\sigma]_i < \theta)$ then $[\sigma]_i := [\sigma]_i + \gamma$,
- if $(\phi = out) \wedge ([\sigma]_i > \theta)$ then $[\sigma]_i := [\sigma]_i - \gamma$.

4.3.9 Long-term potential function

Let $\mu = < < \rho_1, ..., \rho_k >, \tau_\mu, < m_1, ..., m_l > >$ be an LTP function (μ), with $< \rho_1, ..., \rho_k >$ metabolic receptors, τ_μ the neurotransmitter that can be emitted, $< m_1, ..., m_l >$ is the post-synaptic membranes that receive the emitted neurotransmitter. If

$$\bigwedge_{1 \leq i \leq k} e_{\rho_i} = true$$

then $\forall m \in \{m_1, ..., m_l\}$ put the neurotransmitter τ_μ in the transmitter substance set $\mathcal{T}_{m_i}$ for the next time step.

4.4 State space and block diagram model

This section focuses on the detail structure of the RealNeuron network's system level, as shown in 3.6, and its subordinate system levels. The detail is deduced from the descriptions of biological neurons in [13, 41, 125, 158].

The development of a state space model based on §3.4 is presented in this section. The state space model of the algorithms in §4.3 is based on the structures defined in §4.2. This model makes an explicit distinction between the parameters and the states of the RealNeuron model. The long-term memory construction process, [13, 41], could be implemented by optimisation of the model parameters. Long-term memory is not part of the classical conditioning description in [110]. Thus, the optimisation of a RealNeuron network is beyond the scope of this thesis.

The block diagram model consists of several levels of resolution. Each level of resolution shows a block diagram indicating the information flow in the model and the relationship of the current level of resolution for both the lower and the higher levels of resolution. The model consists of the following block diagrams:

- RealNeuron network.
- RealNeuron.
- Soma.

- Axon.
- Bulb end.
- Membrane.

4.4.1 RealNeuron network

The RealNeuron network operator $\Psi \subseteq \sigma_\Psi$, where σ_Ψ is defined as in definition 10, is formally defined as:

Definition 20 (RealNeuron network operatorΨ)

$$\Psi(\Pi, \Xi, \Phi, \mathcal{K}) : \mathbf{2}^{\mathbb{S}_\Psi} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_\Psi}$$

with

Π *is the set of sensor operators,* $\Pi = \{\varpi \mid \forall \varpi$ part of $\Psi\}$, *if* Ψ *has no sensors then* $\Pi = \emptyset$,

Ξ *is the set of RealNeuron operators,* $\Xi = \{N \mid \forall N$ part of $\Psi\}$,

Φ *is the set of actuator operators,* $\Phi = \{\varphi \mid \forall \varphi$ part of $\Psi\}$,

$\mathcal{K}$ *the connections matrix,*

$\mathbb{T}$ *the time space, and*

$\mathbb{S}_\Psi$ *the local RealNeuron network states, which consist of input, internal and output states.*

Messages, through the RealNeuron network, are propagated from the sensors through the RealNeurons and to the actuators. The RealNeurons are calculated simultaneously for each time step t_k, after which the results (messages) are propagated. The block diagram for the message processing schema is shown in figure 4.1.

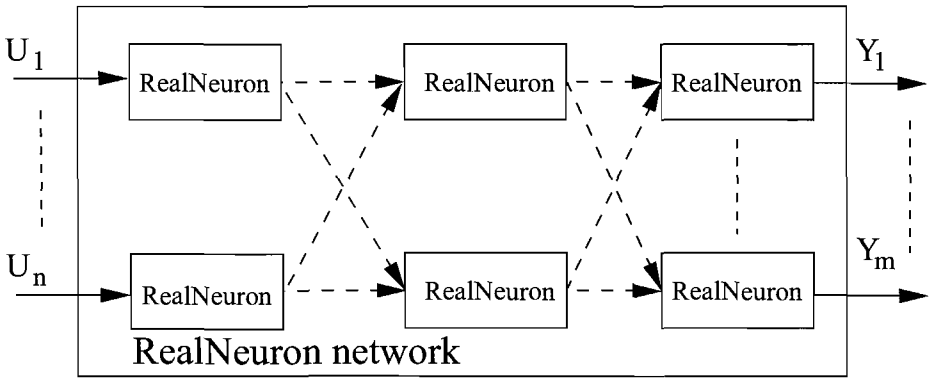


Figure 4.1: RealNeuron network block diagram

For a simple RealNeuron network with a sensor neuron, an internal neuron and a motor neuron, the states are at least

- an input state, U_1,

- the internal states, X_1 and $X2$, and
- an output state, Y_1.

From figure 4.1, it is generally derived that there are $n + r + m$ RealNeurons in a RealNeuron network of which n is the number of motor neurons, r is the number of internal neurons and m is the number of sensor neurons. Thus, each motor neuron could have up to r output connections. If the rule for neurons that states [56]: "a neuron cannot feed back onto itself", i.e. have auto feedback, is applied, then each internal neuron can have up to $r - 1 + m$ output connections. Thus, there are in total $nr + r(r - 1 + m)$ internal states. At most, the local RealNeuron network states ($\mathbb{S}_\Psi$) consist of

- the input states, U_1 to U_n,
- the internal states, X_1 to $X_{r(n+m+r-1)}$, and
- the output states, Y_1 to Y_m.

A trivial network with only sensors and actuators is not a RealNeuron network. A RealNeuron network does not need to have any sensors. In such a case, the network has an internal reference that produces the output to the actuators, e.g. in a clock circuit, the biological equivalent is a pacemaker [13]. If the general expression for local RealNeuron network states ($\mathbb{S}_\Psi$) is applied to the pacemaker with $m = 0$ and $n = 1$, then the states are at most

- r^2 internal states, X_1 to X_{r^2}, and
- one output state, Y_1.

Intuitively it is tempting to say that a RealNeuron network pacemaker only needs to have one internal RealNeuron. Such a pacemaker will only have one internal state which is the connection between the internal neuron and the motor neuron and cannot do much as a pacemaker. At least two internal neurons are required to provide input for each other. Thus, two synapses are needed for the internal connections, and a synapse is necessary to connect to the motor neuron. The motor neuron has a special configuration that cannot provide feedback for the pacemaker. The minimum local RealNeuron network states ($\mathbb{S}_\Psi$) for a pacemaker are

- three internal states, X_1 to X_3, and
- one output state, Y_1.

Thus, the number of states that a RealNeuron network can have is at least 4 and at most $r(n + r + m - 1)$, where n is the number of motor neurons, r is the number of internal neurons and m is the number of sensor neurons.

If a network is considered with no sensor neurons, only one internal neuron and a motor neuron, then the network has only two states. The internal neuron functions as a sensor to detect any changes in temperature, external substance concentration, potentials, etc. It is then a sensor connected directly to a motor neuron and this is by definition not a RealNeuron network.

The input state vector U_Ψ, internal state vector X_Ψ and the output state vector Y_Ψ for the RealNeuron network (Ψ) are then:

$$U_\Psi = \begin{bmatrix} U_1 \\ \vdots \\ U_n \end{bmatrix} \tag{4.1}$$

$$X_\Psi = \begin{bmatrix} X_1 \\ \vdots \\ X_{r(n+r+m-1)} \end{bmatrix} \tag{4.2}$$

$$Y_\Psi = \begin{bmatrix} Y_1 \\ \vdots \\ Y_m \end{bmatrix} \tag{4.3}$$

At time step k, a sensor neuron can be connected with up to r internal neurons, thus, in general, the system of transfer functions from an input state to some internal states is

$$\begin{aligned} X_{(i-1)r+1}(k) &= N_i U_i(k-1) \\ &\vdots \\ X_{(i-1)r+r}(k) &= N_i U_i(k-1) \end{aligned} \tag{4.4}$$

where $\forall i = 1, \cdots, n$.

In general, each internal neuron is connected to all the sensor neurons, all the other internal neurons and motor neurons. In general, the system of transfer functions from the outputs of input neurons and other internal neurons to the output of an internal neuron is:

$$\begin{bmatrix} X_{(n+j-1)r+(j-1)m+1}(k) \\ \vdots \\ X_{(n+j-1)r+(j-1)m+r}(k) \\ X_{(n+j-1)r+(j-1)m+r+1}(k) \\ \vdots \\ X_{(n+j-1)r+(j-1)m+r+m}(k) \end{bmatrix} = N_{n+j} \begin{bmatrix} X_j(k-1) \\ \vdots \\ X_{(n-1)r+j}(k-1) \\ X_{nr+j}(k-1) \\ \vdots \\ X_{(n+r-1)r+(r-1)m+j}(k-1) \end{bmatrix} \tag{4.5}$$

where $\forall j = 1, \cdots, r$ and $X_{(n+j-1)r+(j-1)m+j} \in \emptyset$ because a RealNeuron cannot have auto feedback.

A motor neuron can be connected with up to r internal neurons, thus, in general, the system of transfer functions from input states to an output state is

$$Y_p(k) = N_{n+r+p} \begin{bmatrix} X_{(n+1)r+p}(k-1) \\ \vdots \\ X_{(n+r)r+(r-1)m+p}(k-1) \end{bmatrix} \tag{4.6}$$

where $\forall p = 1, \cdots, m$.

The maximum number of internal states for a RealNeuron network is $r(n+r+m-1)$, but from the internal state index in (4.6) the maximum index number is $r(n+r+m)$. It should be remembered that $X_{(n+j-1)r+(j-1)m+j} \in \emptyset$, which means that r internal states do not exist, thus

$$r(n+r+m) - r = r(n+r+m-1).$$

If the notation in §3.5 on p. 44 is used and the RealNeuron network system of equations is expressed in the form of (3.15) then

$$\begin{aligned} X_\Psi &= F_I(\Pi, \Xi) \begin{bmatrix} X_\Psi \\ U_\Psi \\ t \end{bmatrix} \\ Y_\Psi &= H(\Phi) \begin{bmatrix} X_\Psi \\ t \end{bmatrix}. \end{aligned} \tag{4.7}$$

with the parameters defined as in table 4.1.

Table 4.1: Parameters and states for a generic RealNeuron network

Parameter	Value	Description	Units	Dim
u	$U_\Psi^{(1)}$	input states	any	1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$U_\Psi^{(n)}$	input states	any	1
x	$X_\Psi^{(1)}$	internal states	mV	1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$X_\Psi^{(nr)}$	internal states	mV	1
	$X_\Psi^{(nr+2)}$	internal states	molecule	q_2
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$X_\Psi^{(r(n+r+m))}$	internal states	molecule	q_{r^2+rm}
	$X_\Psi^{(nr+1)}$	element of $\emptyset$	N.A.	0
	$X_\Psi^{((n+1)r+m+2)}$	element of $\emptyset$	N.A.	0
	$X_\Psi^{((n+2)r+2m+3)}$	element of $\emptyset$	N.A.	0
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$X_\Psi^{(r(n+r+m)-m)}$	element of $\emptyset$	N.A.	0
y	$Y_N^{(1)}$	output states	any	1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$Y_N^{(m)}$	output states	any	1
t	$t \in \mathbb{T}$	time step		
ν	$\{\Pi, \Xi, \mathcal{K}\}$	parameters to F_I		
ω	Φ	parameters to G		
F_I	$[N_1, ..., N_{n+r}]^T$	mapping to internal states		
G	$[N_{n+r+1}, ..., N_{n+r+m}]^T$	mapping to output states		

4.4.2 RealNeuron

With σ_N defined in definition 11, the RealNeuron operator $N \subseteq \sigma_N$ is formally defined as:

Definition 21 (Generic RealNeuron operator N)

$$N(S, A, \mathcal{Y}) : \mathbf{2}^{\mathbb{S}_N} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_N}$$

with

- S *the soma operator,*
- A *the axon operator, if N has no axon then $A = I$, with I the identity operator,*
- $\mathcal{Y}$ *is the set of bulb-end operators, $\mathcal{Y} = \{\Upsilon \mid \forall \Upsilon$ part of $N\}$, if N has no bulb ends then $\mathcal{Y} = \emptyset$,*
- $\mathbb{T}$ *the time space, and*
- $\mathbb{S}_N$ *the local RealNeuron states, which are input, internal and output states.*

From the description in §4.3.2, a RealNeuron has an input soma connected to an axon and set of bulb ends connected to the axon. The block diagram for the message processing of a RealNeuron, based on the description in §4.3.2, is shown in figure 4.2.

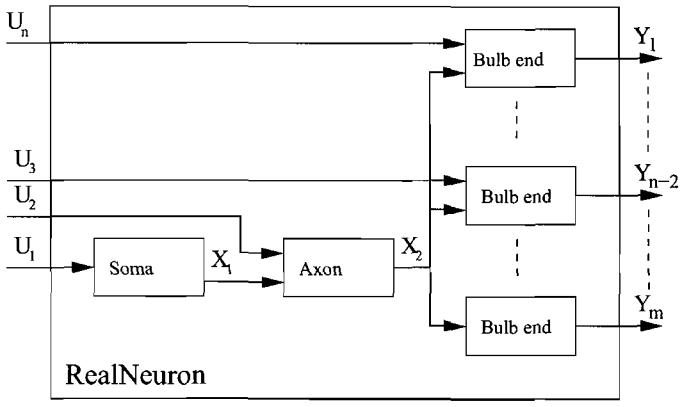


Figure 4.2: Generic RealNeuron block diagram

The local RealNeuron states ($\mathbb{S}_N$) for the generic RealNeuron can be derived from figure 4.2 as

- the input states, U_1 to U_n,
- the internal states, X_1 and X_2, and
- the output state Y_1 to Y_m.

From figure 4.2, it follows that the maximum number of input states are $n \leq m+2$, with m the number of output states.

It follows from definition 21, figure 4.2 and §4.3.2 that if a RealNeuron has no axon, then $X_2 = X_1$.

RealNeurons without bulb ends can exist. If a RealNeuron has no bulb ends then it has no axon and $Y_1 = X_2 = X_1$ and $\Upsilon = A = I$, where I is the identity mapping.

If a RealNeuron configuration is considered with only one bulb end and no axon, then it has at least

- one input state, namely U_1,
- one internal state, namely X_1, and
- one output state, namely Y_1.

If a RealNeuron configuration is considered with no bulb ends[5], then it has at least

- one input state, namely U_1, and
- one output state, namely Y_1.

The generic RealNeuron states, i.e. input state vector U_N, internal state vector X_N and the output state vector Y_N, are:

$$U_N = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_n \end{bmatrix} \tag{4.8}$$

$$X_N = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \tag{4.9}$$

$$Y_N = \begin{bmatrix} Y_1 \\ \vdots \\ Y_m \end{bmatrix} \tag{4.10}$$

If the notation in §3.5 on p. 44 is used, the generic RealNeuron is implemented with the generic RealNeuron system of equations:

$$\begin{aligned}
X_N^{(1)} &= SU_N^{(1)} \\
X_N^{(2)} &= A[U_N^{(2)} \quad X_N^{(1)}]^T \\
\\
Y_N^{(1)} &= \Upsilon_1[X_N^{(2)} \quad U_N^{(3)}]^T \\
Y_N^{(2)} &= \Upsilon_2[X_N^{(2)} \quad U_N^{(4)}]^T \\
\vdots \quad &\vdots \quad \vdots \\
Y_N^{(n-2)} &= \Upsilon_{n-2}[X_N^{(2)} \quad U_N^{(n)}]^T \\
Y_N^{(n-1)} &= \Upsilon_{n-1}X_N^{(2)} \\
\vdots \quad &\vdots \quad \vdots \\
Y_N^{(m)} &= \Upsilon_m X_N^{(2)}
\end{aligned} \tag{4.11}$$

The generic-RealNeuron parameter values, to be substituted with the parameters in (3.15), are listed in table 4.2.

As seen in the system breakdown structure in figure 3.6, three specific types of RealNeurons exist:

[5] As will be seen in §4.4.2.3, this is a motor neuron.

Table 4.2: Parameters and states for the generic RealNeuron

Parameter	Value	Description	Unit	Dim
u	$U_N^{(1)}$	input states	any	q_1
	$U_N^{(2)}$	input states	molecule	q_2
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$U_N^{(n)}$	input states	molecule	q_n
x	$X_N^{(1)}$	internal states	mV	1
	$X_N^{(2)}$	internal states	mV	1
y	$Y_N^{(1)}$	output states	molecule	q_{n+1}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$Y_N^{(n-2)}$	output states	molecule	q_{2n-2}
	$Y_N^{(n-1)}$	output states	any	q_{2n-1}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$Y_N^{(2)}$	output states	any	q_{n+m}
t	$t \in \mathbb{T}$	time step		
ν	$\{S, A\}$	parameters to F_I		
ω	$\mathcal{Y}$	parameters to G		
F_I	$[S, A]^T$	mapping to internal states		
G	$[\Upsilon_1, \Upsilon_2, ..., \Upsilon_m]^T$	mapping to output states		

- Input RealNeuron (also called a sensor neuron (N_ϖ)).
- Internal RealNeuron (also called a neuron (N)).
- Output RealNeuron (also called a motor neuron (N_φ)).

4.4.2.1 Sensor neuron

Firstly, is defined what makes a sensor neuron different from other RealNeurons:

Definition 22 (Sensor neuron) *A sensor neuron has*

- *a soma that only has a potential as an input state,*
- *no axon, and*
- *at least one bulb end without any post-synaptic membranes.*

The sensor neuron operator N_ϖ can be formally derived from the generic RealNeuron operator N, by applying definition 22 to definition 21, to arrive at the general sensor neuron operator N_ϖ:

$$N_\varpi(S, \mathcal{Y}) : \mathbf{2}^{\mathbf{S}_N} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbf{S}_N} \tag{4.12}$$

with

S the soma operator,

$\mathcal{Y}$ is the set of bulb-end operators, $\mathcal{Y} = \{\Upsilon \mid \forall \Upsilon \text{ part of } N\}$,

$\mathbb{T}$ the time space, and

$\mathbb{S}_N$ the local RealNeuron states, which are input, internal and output states.

The sensor neuron block diagram in figure 4.3 is derived from figure 4.2 by applying the discussion in §4.4.2 for a RealNeuron without an axon, definitions 21 and 22.

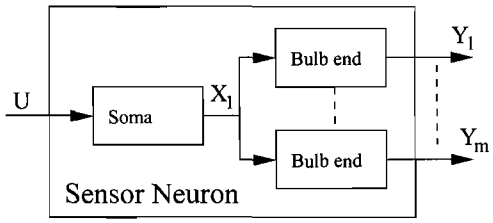


Figure 4.3: Sensor neuron block diagram

The local sensor neuron states $\mathbb{S}_{N_\varpi}$ can be derived from figure 4.3 as

- the input state, U_1,
- the internal state, X_1, and
- the output states Y_1 to Y_m.

The sensor neuron states, i.e. input state vector U_{N_ϖ}, internal state vector X_{N_ϖ} and the output state vector Y_{N_ϖ}, are:

$$U_{N_\varpi} = \begin{bmatrix} U_1 \end{bmatrix} \tag{4.13}$$

$$X_{N_\varpi} = \begin{bmatrix} X_1 \end{bmatrix} \tag{4.14}$$

$$Y_{N_\varpi} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_m \end{bmatrix} \tag{4.15}$$

If the notation in §3.5 on p. 44 is used, then from figure 4.3, the general sensor neuron is implemented with the sensor neuron system of equations:

$$\begin{aligned} X_{N_\varpi} &= SU_{N_\varpi} \\ Y^{(1)}_{N_\varpi} &= \Upsilon_1 X_{N_\varpi} \\ &\vdots \quad \vdots \quad \vdots \\ Y^{(m)}_{N_\varpi} &= \Upsilon_m X_{N_\varpi} \end{aligned} \tag{4.16}$$

Table 4.3 shows the parameters that need to be substituted into (3.15) for the sensor neuron.

Table 4.3: Parameters and states for the sensor neuron

Param	Value	Description	Unit	Dim
u	U_{N_ϖ}	input potential	mV	1
x	X_{N_ϖ}	main membrane potential	mV	1
y	$Y^{(1)}_{N_\varpi}$	bulb-ends output	molecule	q_1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$Y^{(m)}_{N_\varpi}$	bulb-ends output	molecule	q_m
t	$t \in \mathbb{T}$	time step	ms	1
ν	S	parameters to F_I		
ω	$\mathcal{Y}$	parameters to G		
F_I	S	mapping to internal states		
G	$[\Upsilon_1, \Upsilon_2, ..., \Upsilon_m]^T$	mapping to output states		

4.4.2.2 Internal neuron

An internal neuron is distinguished, from the other types of RealNeurons, as follows:

Definition 23 (Neuron) *A neuron is a type of RealNeuron that has*

- *a soma that has at least a neurotransmitter molecule as an input state, and*
- *at least one bulb end.*

Definition 21 holds for the neuron without any change. The discussion for the generic RealNeuron also holds for a neuron.

Based on figure 4.2 and the discussion in §4.4.2 for the maximum number of input states for a RealNeuron, a neuron is implemented by substituting the parameters in table 4.4 into (3.15).

4.4.2.3 Motor neuron

A motor neuron is distinguished, from other RealNeuron types, as follows:

Definition 24 (Motor neuron) *A motor neuron has*

- *a soma that has at least a neurotransmitter molecule as an input state,*
- *no axon, and*
- *no bulb ends.*

The general motor neuron operator N_φ can be formally derived from the generic RealNeuron operator N, by applying definition 24 to definition 21, as:

$$N_\varphi(S) : \mathbf{2}^{\mathbb{S}_N} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_N} \tag{4.17}$$

with

S the soma operator,

Table 4.4: Parameters and states for the neuron

Param	Value	Description	Unit	Dim
u	$U_N^{(1)}$	input states	molecule	q_1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$U_N^{(n)}$	input states	molecule	q_n
x	$X_N^{(1)}$	internal states	mV	1
	$X_N^{(2)}$	internal states	mV	1
y	$Y_N^{(1)}$	output states	molecule	q_{n+1}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$Y_N^{(2)}$	output states	molecule	q_{n+m}
t	$t \in \mathbb{T}$	time step		
ν	$\{S, A\}$	parameters to F_I		
ω	$\mathcal{Y}$	parameters to G		
F_I	$[S, A]^T$	mapping to internal states		
G	$[\Upsilon_1, \Upsilon_2, ..., \Upsilon_m]^T$	mapping to output states		

$\mathbb{T}$ the time space, and

$\mathbb{S}_N$ the local RealNeuron states, which are input, internal and output states.

The motor neuron block diagram in figure 4.4 is derived from figure 4.2 by applying the discussion in §4.4.2 to a RealNeuron without a bulb end, definitions 21 and 24.

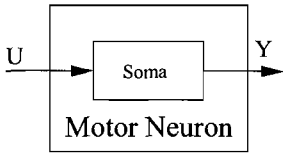


Figure 4.4: Motor neuron block diagram

The local motor neuron states $\mathbb{S}_{N_\varphi}$ can be derived from figure 4.4 as:

- the input state, U_1,
- no internal state, and
- the output state Y_1.

The Motor neuron states, i.e. input state vector U_{N_φ}, internal state vector X_{N_φ} and the output state vector Y_{N_φ}, are:

$$U_{N_\varphi} = [\ U_1\ ] \tag{4.18}$$

$$X_{N_\varphi} = [\ \ ] \tag{4.19}$$

$$Y_{N_\varphi} = [\ Y_1\ ] \tag{4.20}$$

From figure 4.4, the general motor neuron is implemented with:

$$Y_{N_\varphi} = SU_{N_\varphi} \tag{4.21}$$

Table 4.5 shows the parameters that need to be substituted into (3.15) for the motor neuron.

Table 4.5: Parameters and states for the motor neuron

Param	Value	Description	Unit	Dim
u	U_{N_φ}	neurotransmitters in synaptic gaps	molecules	1
x	$\emptyset$			0
y	Y_{N_φ}	main membrane output	mV	1
t	$t \in \mathbb{T}$	time step	ms	1
ν	S	parameters to F_I		
ω	I	parameters to G		
F_I	S	mapping to internal states		
G	I	mapping to output states		

4.4.3 Soma

If σ_S is defined as in definition 12, the soma operator $S \subseteq \sigma_S$ is formally defined as:

Definition 25 (Generic soma operator S)

$$S(\mathcal{M}_p, \mathcal{I}_S, \mu, M_m) : \mathbf{2}^{\mathbb{S}_S} \times \mathbb{T} \to \mathbf{2}^{\mathbb{S}_S}$$

with

$\mathcal{M}_p$ *the set of post-synaptic membrane operators,*

$\mathcal{I}_S$ *the soma integration function operator,*

μ *the LTP function operator,*

M_m *the main membrane operator,*

$\mathbb{T}$ *the time space, and*

$\mathbb{S}_S$ *the local soma states, which are input, internal and output states.*

Based on §4.2.3, three types of soma exist:

- Input soma (S_i) that is part of the neuron and motor neuron,
- Sensoric soma (S_ϖ) that is part of the sensor neuron, and
- Bulb-end soma (S_Υ) that is part of the bulb end.

The difference between the soma types lies in the order in which the operators M_p, M_m, μ, and the $\mathcal{I}_S$ are performed, see §4.3.3. The sensoric soma has a special condition in which $M_p = \mu = \mathcal{I}_S = \emptyset$. No biological equivalent exists for the bulb-end soma, see appendix E for a motivation.

4.4.3.1 Input soma

The input soma operator (S_i) definition is defined as in definition 25, thus the input soma operator and the soma operator have identical formal definitions.

The message processing for the input soma, described in paragraph §4.3.3, is shown in the block diagram in figure 4.5. The time delay element z^{-1} has been derived from the description of the LTP function (μ) in §4.3.9, see also §4.4.7. The "+"

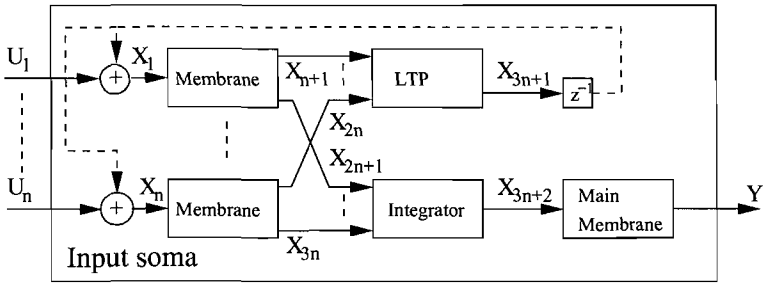


Figure 4.5: Input soma block diagram

in figure 4.5 is defined as adding elements together in sets to form a new set, e.g. $A = \{a, b, c\}$ and $B = \{c, d\}$, $A + B = \{a, b, c, c, d\}$.

The local soma states ($\mathbb{S}_S$) for an input soma can be derived from figure 4.5 as:

- the input states, U_1 to U_n,
- the internal states, X_1 to X_{3n+2}, and
- the output state Y.

It follows from figure 4.5, §4.3.3 and §4.3.9 that:

a. In an input soma, X_{3n+1} can feed back to a membrane input if, and only if, the specific membrane is participating in the LTP function.

b. If a membrane M_k does not participate in the LTP function, then $X_k = U_k$.

If an input soma has n input states, then it has

a. at most, $3n + 2$ internal states.

b. at least, $n + 1$ internal states if the soma has no LTP function.

The minimum local soma states ($\mathbb{S}_S$) for an input soma are

- the input state, U_1,
- the internal state, $X_{3n+2} = X_{3n}$, and
- the output state Y.

The input soma states, i.e. input state vector U_{S_i}, internal state vector X_{S_i} and the output state vector Y_{S_i}, are:

$$U_{S_i} = \begin{bmatrix} U_1 \\ \vdots \\ U_n \end{bmatrix} \tag{4.22}$$

$$X_{S_i} = \begin{bmatrix} X_1 \\ \vdots \\ X_{3n+2} \end{bmatrix} \tag{4.23}$$

$$Y_{S_i} = \begin{bmatrix} Y \end{bmatrix} \tag{4.24}$$

If the notation in §3.5 on p. 44 is used, then an input soma is implemented with the input soma system of equations:

$$\begin{aligned}
X_{S_i}^{(1)} &= U_{S_i}^{(1)} + F({M_p}^{(1)} \in \mu)\, z^{-1} X_{3n+1} \\
&\vdots \\
X_{S_i}^{(n)} &= U_{S_i}^{(n)} + F({M_p}^{(n)} \in \mu)\, z^{-1} X_{3n+1} \\
\begin{bmatrix} X_{S_i}^{(n+1)} & X_{S_i}^{(2n+1)} \end{bmatrix}^T &= {M_p}^{(1)} X_{S_i}^{(1)} \\
&\vdots \\
\begin{bmatrix} X_{S_i}^{(2n)} & X_{S_i}^{(3n)} \end{bmatrix}^T &= {M_p}^{(n)} X_{S_i}^{(n)} \\
X_{S_i}^{3n+1} &= \mu \begin{bmatrix} X_{S_i}^{(n+1)} & \cdots & X_{S_i}^{(2n)} \end{bmatrix}^T \\
X_{S_i}^{3n+2} &= \mathcal{I}_S \begin{bmatrix} X_{S_i}^{(2n+1)} & \cdots & X_{S_i}^{(3n)} \end{bmatrix}^T \\
Y_{S_i} &= M_m X_{S_i}^{3n+2}
\end{aligned} \tag{4.25}$$

where

$$F({M_p}^{(i)} \in \mu) = \begin{cases} 1 & \text{if } {M_p}^{(i)} \in \mu \\ 0 & \text{otherwise} \end{cases}$$

The generic-input soma parameter values, to be substituted with the parameters in (3.15), are listed in table 4.6.

4.4.3.2 Sensoric soma

A sensoric soma is a special kind of input soma and is distinguished from a general input soma as follows:

Definition 26 *A sensoric soma is always part of a sensor neuron and is an input soma with no post-synaptic membranes and LTP functions.*

If definition 26 is applied to figure 4.5, it follows then that a sensoric soma has

- no post-synaptic membrane, thus $M_p = I$,
- no LTP function, thus $\mu = 0$,
- no soma integration function, thus $\mathcal{I}_S = I$, and

Table 4.6: Parameters and states for the input soma

Param	Value	Description	Unit	Dim
u	$U_{S_i}^{(1)}$	neurotransmitters in synaptic gaps	molecules	q_1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$U_{S_i}^{(n)}$	neurotransmitters in synaptic gaps	molecules	q_n
x	$X_{S_i}^{(1)}$	augmented neurotransmitters in synaptic gaps	molecules	q_{n+1}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$X_{S_i}^{(n)}$	augmented neurotransmitters in synaptic gaps	molecules	q_{2n}
	$X_{S_i}^{(n+1)}$	metabolic event flags	true/false	1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$X_{S_i}^{(2n)}$	metabolic event flags	boolean	1
	$X_{S_i}^{(2n+1)}$	membrane outputs	mV	1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$X_{S_i}^{(3n)}$	membrane outputs	mV	1
	$X_{S_i}^{(3n+1)}$	LTP output	molecule	1
	$X_{S_i}^{(3n+2)}$	integration output	mV	1
y	Y_{S_i}	main membrane output	mV	1
t	$t \in \mathbb{T}$	time step	ms	1
ν	$\mathcal{I}_S, \mu, \mathcal{M}_p$	parameters to F_I		
ω	M_m	parameters to G		
F_I	$[\mathcal{M}_p, F, \mathcal{I}_S, \mu]^T$	mapping to internal states		
G	M_m	mapping to output states		

- one membrane, that is the main membrane.

The sensoric soma operator S_ϖ can be formally expressed as:

$$S_\varpi(M_m) : \mathbf{2}^{\mathbb{S}_S} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_S} \tag{4.26}$$

with

M_m the main membrane operator,

$\mathbb{T}$ the time space, and

$\mathbb{S}_S$ the local soma states, which are input, internal and output states.

Figure 4.5 can be reduced to the sensoric soma block diagram in figure 4.6.

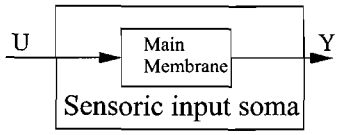


Figure 4.6: Sensoric soma block diagram

The local sensoric soma states $\mathbb{S}_{S_\varpi}$ can be derived from figure 4.6 as:

- the input state, U_1,
- no internal states, and
- the output state Y.

The sensoric soma states, i.e. input state vector U_{S_ϖ}, internal state vector X_{S_ϖ} and the output state vector Y_{S_ϖ}, are:

$$U_{S_\varpi} = [\ U\ ] \tag{4.27}$$
$$X_{S_\varpi} = [\ \ ] \tag{4.28}$$
$$Y_{S_\varpi} = [\ Y\ ] \tag{4.29}$$

If the notation in §3.5 on p. 44 is used, then based on figure 4.6, the sensoric soma is implemented with:

$$Y_{S_\varpi} = M_m U_{S_\varpi} \tag{4.30}$$

Table 4.7 shows the parameters that need to be substituted into (3.15) for the sensoric soma.

4.4.3.3 Bulb-end soma

The formal definition of the bulb-end soma operator (S_Υ) is the same as for the general soma operator (S) in definition 25.

The message processing for the bulb-end soma, described in paragraph §4.3.3, is shown in the block diagram in figure 4.7. The "+" in figure 4.5 is defined as adding elements in sets together to form a new set, e.g. $A = \{a, b, c\}$ and $B = \{c, d\}$, $A + B = \{a, b, c, c, d\}$.

The local soma states ($\mathbb{S}_S$) for a bulb-end soma can be derived from figure 4.7 as:

Table 4.7: Parameters and states for the sensoric soma

Param	Value	Description	Unit	Dim
u	U_{S_ϖ}	input potential	mV	1
x	$\emptyset$			
y	Y_{S_ϖ}	main membrane output	mV	1
t	$t \in \mathbb{T}$	time step	ms	1
ν	I	parameters to F_I		
ω	M_m	parameters to G		
F_I	I	mapping to internal states		
G	M_m	mapping to output states		

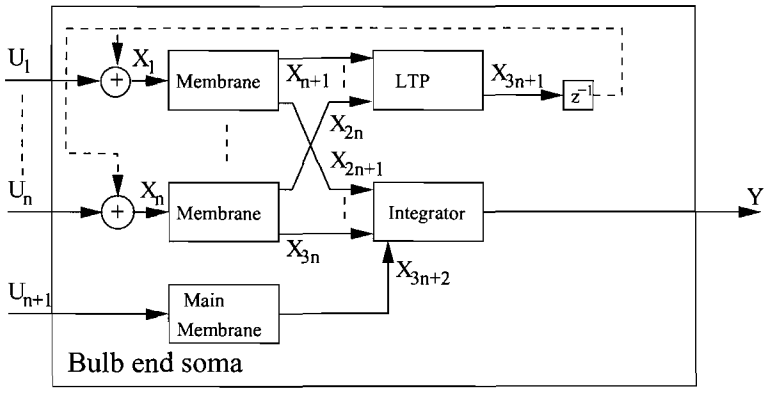


Figure 4.7: Bulb-end soma block diagram

- the input states, U_1 to U_{n+1},
- the internal states, X_1 to X_{3n+2}, and
- the output state Y.

Except for U_{n+1}, the local soma states ($\$_S$) for a bulb-end soma is the same as for states for a local soma states ($\$_S$) and for an input soma. However, the way that the X_{3n+2} and Y are calculated differs, compare figure 4.5 with figure 4.7.

The following can be stated, based on figure 4.7 and the discussion in §4.3.3:

- In a bulb-end soma, X_{3n+2} feeds back to a membrane input if, and only if, the specific membrane is participating in the LTP function.
- If a bulb-end soma has n input states, then it has
  a. at most, $3n+2$ internal states.
  b. at least, $2n+1$ internal states if the soma has no LTP function.

If a bulb-end soma has $n+1$ input states, then it has

a. at most, $3n+2$ internal states.

b. at least, $n+1$ internal states if the soma has no LTP function.

The minimum local soma states ($\$_S$) for a bulb-end soma are:

- the input state, U_1,
- no internal state, and
- the output state $Y = X_{3n+2}$.

The Bulb-end soma states, i.e. input state vector U_{S_Υ}, internal state vector X_{S_Υ} and the output state vector Y_{S_Υ}, are:

$$U_{S_\Upsilon} = \begin{bmatrix} U_1 \\ \vdots \\ U_n \\ U_{n+1} \end{bmatrix} \tag{4.31}$$

$$X_{S_\Upsilon} = \begin{bmatrix} X_1 \\ \vdots \\ X_{3n+2} \end{bmatrix} \tag{4.32}$$

$$Y_{S_\Upsilon} = \begin{bmatrix} Y \end{bmatrix} \tag{4.33}$$

If the notation in §3.5 on p. 44 is used, then from figure 4.7, the bulb-end soma is implemented with the bulb-end soma system of equations:

$$\begin{aligned} X_{S_\Upsilon}^{(1)} &= U_{S_\Upsilon}^{(1)} + F(M_p{}^{(1)} \in \mu)\, z^{-1} X_{3n+1} \\ &\vdots \\ X_{S_\Upsilon}^{(n)} &= U_{S_\Upsilon}^{(n)} + F(M_p{}^{(n)} \in \mu)\, z^{-1} X_{3n+1} \\ \begin{bmatrix} X_{S_\Upsilon}^{(n+1)} & X_{S_\Upsilon}^{(2n+1)} \end{bmatrix}^T &= M_p{}^{(1)} X_{S_\Upsilon}^{(1)} \end{aligned}$$

$$
\begin{aligned}
&\vdots \\
\left[X_{S_\Upsilon}^{(2n)} \quad X_{S_\Upsilon}^{(3n)}\right]^T &= M_p{}^{(n)} X_{S_\Upsilon}^{(n)} \\
X_{S_\Upsilon}^{3n+1} &= \mu\left[X_{S_\Upsilon}^{(n+1)} \cdots X_{S_\Upsilon}^{(2n)}\right]^T \\
X_{S_\Upsilon}^{3n+2} &= M_m U_{S_\Upsilon}^{(n+1)} \\
Y_{S_\Upsilon} &= \mathcal{I}_S\left[X_{S_\Upsilon}^{(2n+1)} \cdots X_{S_\Upsilon}^{(3n)} \; X_{S_\Upsilon}^{(3n+2)}\right]^T
\end{aligned} \tag{4.34}
$$

where

$$
F(M_p{}^{(i)} \in \mu) = \begin{cases} 1 & \text{if } M_p{}^{(i)} \in \mu \\ 0 & \text{otherwise} \end{cases}
$$

Table 4.8 shows the parameters that need to be substituted into (3.15) for the bulb-end soma.

Table 4.8: Parameters and states for the bulb-end soma

Param	Value	Description	Unit	Dim
u	$U_{S_\Upsilon}^{(1)}$	neurotransmitters in synaptic gaps	molecules	q_1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$U_{S_\Upsilon}^{(n)}$	neurotransmitters in synaptic gaps	molecules	q_n
x	$X_{S_\Upsilon}^{(1)}$	augmented neurotransmitters in synaptic gaps	molecules	q_{n+1}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$X_{S_\Upsilon}^{(n)}$	augmented neurotransmitters in synaptic gaps	molecules	q_{2n}
	$X_{S_\Upsilon}^{(n+1)}$	metabolic event flags	true/false	1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$X_{S_\Upsilon}^{(2n)}$	metabolic event flags	true/false	1
	$X_{S_\Upsilon}^{(2n+1)}$	membrane outputs	mV	1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$X_{S_\Upsilon}^{(3n)}$	membrane outputs	mV	1
	$X_{S_\Upsilon}^{(3n+1)}$	LTP output	molecule	1
	$X_{S_\Upsilon}^{(3n+2)}$	main membrane output	mV	1
y	Y_{S_Υ}	integration output	mV	1
t	$t \in \mathbb{T}$	time step	ms	1
ν	$M_m, \mu, \mathcal{M}_p$	parameters to F_I		
ω	$\mathcal{I}_S$	parameters to G		
F_I	$[M_m, \mathcal{M}_p, F, \mu]^T$	mapping to internal states		
G	$\mathcal{I}_S$	mapping to output states		

4.4.4 Axon

The axon operator $A \subseteq \sigma_A$, where σ_A is defined as in definition 13, is formally defined as:

Definition 27 (Axon operator A)

$$A({M_a}^{(1)}, ..., {M_a}^{(k)}) : \mathbf{2}^{\mathbb{S}_A} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_A}$$

with

${M_a}^{(i)}$ *the axonal membrane operator, are ordered so that* ${M_a}^{(1)}$ *is the membrane receiving input from the input soma,* ${M_a}^{(k)}$ *is the output membrane of the axon, and* ${M_a}^{(j)}$ *receives input from* ${M_a}^{(j-1)}$,

$\mathbb{T}$ *the time space, and*

$\mathbb{S}_A$ *the local axon states, which are input, internal and output states.*

The block diagram for the message processing is shown in figure 4.8 and is based on section §4.3.4.

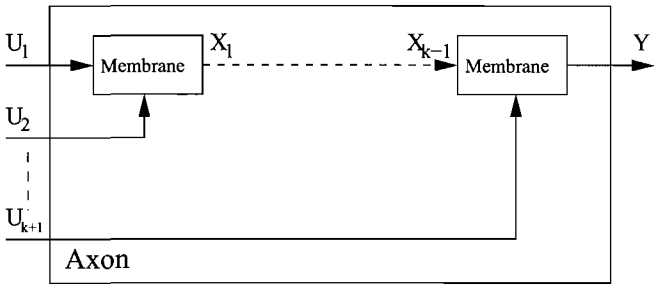


Figure 4.8: Axon block diagram

The Local axon states ($\mathbb{S}_A$) for the generic axon are

- the input states, U_1 to U_{k+1},
- the internal states, X_1 to X_{k-1}, and
- the output state Y.

Based on figure 4.8 and the discussion in §4.3.4, if an axon has less than two input states, then there is no reason why the axon should exist, thus an axon has at least two input states.

By closely examining figure 4.8 and the discussion in §4.3.4, if an axon has $n > 1$ input states, then it has at most $n - 2$ internal states. Further, if $k = 1$ for the local axon states ($\mathbb{S}_A$) for the generic axon then, the axon does not have any internal states.

The axon states, i.e. input state vector U_A, internal state vector X_A and the output state vector Y_A, are:

$$U_A = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_{k+1} \end{bmatrix} \tag{4.35}$$

$$X_A = \begin{bmatrix} X_1 \\ \vdots \\ X_{k-1} \end{bmatrix} \tag{4.36}$$

$$Y_A = \begin{bmatrix} Y \end{bmatrix} \tag{4.37}$$

If the notation in §3.5 on p. 44 is used, then based on figure 4.7, the generic axon is implemented with the axon system of equations:

$$\begin{aligned} X_A^{(1)} &= M_a{}^{(1)}[U_A^{(1)} \quad U_A^{(2)}]^T \\ X_A^{(2)} &= M_a{}^{(2)}[X_A^{(1)} \quad U_A^{(3)}]^T \\ &\vdots \\ X_A^{(i)} &= M_a{}^{(i)}[X_A^{(i-1)} \quad U_A^{(i+1)}]^T \\ &\vdots \\ X_A^{(k-1)} &= M_a{}^{(k-1)}[X_A^{(k-2)} \quad U_A^{(k)}]^T \\ Y_A &= M_a{}^{(k)}[X_A^{(k-1)} \quad U_A^{(k+1)}]^T \end{aligned} \tag{4.38}$$

The generic-axon parameter values, to be substituted with the parameters in (3.15), are listed in table 4.9.

4.4.5 Bulb end

The bulb-end operator $\Upsilon \subseteq \sigma_\Upsilon$, where σ_Υ is defined as in definition 14, is formally defined as:

Definition 28 (Bulb-end operator Υ)

$$\Upsilon(S_\Upsilon, V) : \mathbf{2}^{\mathbb{S}_\Upsilon} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_\Upsilon}$$

with

S_Υ *the bulb-end soma operator,*

V *the vesicle operator,*

$\mathbb{T}$ *the time space, and*

$\mathbb{S}_\Upsilon$ *the local bulb-end states, which are input, internal and output states.*

The block diagram for the message processing is based on section §4.3.5 and shown in figure 4.9.

The local bulb-end states ($\mathbb{S}_\Upsilon$) for the generic bulb end can be derived from figure 4.9 as:

Table 4.9: Parameters and states for the axon

Param	Value	Description	Unit	Dim
u	$U_A^{(1)}$	input potential	mV	1
	$U_A^{(2)}$	neurotransmitters in synaptic gaps	molecules	q_1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$U_A^{(k+1)}$	neurotransmitters in synaptic gaps	molecules	q_k
x	$X_A^{(1)}$	membrane potentials	mV	1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$X_A^{(k-1)}$	membrane potentials	mV	1
y	Y_A	output potential	mV	1
t	$t \in \mathbb{T}$	time step	ms	1
ν	${M_a}^{(1)}, \cdots, {M_a}^{(n-1)}$	parameters to F_I		
ω	${M_a}^{(n)}$	parameters to G		
F_I	$[{M_a}^{(1)} \cdots {M_a}^{(n-1)}]^T$	mapping to internal states		
G	${M_a}^{(n)}$	mapping to output states		

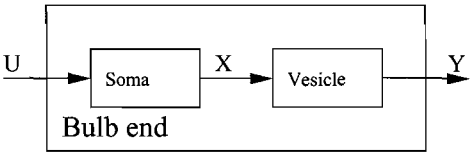


Figure 4.9: Bulb-end block diagram

- the input state, U,
- the internal state, X, and
- the output state Y.

A bulb end always has the above local bulb-end states ($\mathbb{S}_\Upsilon$).

The bulb-end states, i.e. input state vector U_Υ, internal state vector X_Υ and the output state vector Y_Υ, are:

$$U_\Upsilon = \left[\ U\ \right] \tag{4.39}$$
$$X_\Upsilon = \left[\ X\ \right] \tag{4.40}$$
$$Y_\Upsilon = \left[\ Y\ \right] \tag{4.41}$$

If the notation in §3.5 on p. 44 is used, then based on figure 4.9, the generic bulb end is implemented with the bulb end system of equations:

$$\begin{aligned} X_\Upsilon &= S_\Upsilon U_\Upsilon \\ Y_\Upsilon &= V X_\Upsilon \end{aligned} \tag{4.42}$$

The bulb end, values to be substituted with the parameters in (3.15), are shown in table 4.10.

Table 4.10: Parameters and states for the bulb end

Param	Value	Description	Unit	Dim
u	U_Υ	input potential	mV	1
x	X_Υ	bulb-end potential	mV	1
y	Y_Υ	output neurotransmitter	molecule	q
t	$t \in \mathbb{T}$	time step	ms	1
ν	S_Υ	parameters to F_I		
ω	V	parameters to G		
F_I	S_Υ	mapping to internal states		
G	V	mapping to output states		

4.4.6 Membrane

The membrane operator $M \subseteq \sigma_M$, where σ_M is defined as in definition 15, is formally defined as:

Definition 29 (Generic membrane operator M)

$$M(\mathcal{C}, \mathcal{P}, \mathcal{R}, \mathcal{I}_M, \Gamma, \Gamma^{-1}) : \mathbf{2}^{\mathbb{S}_M} \times \mathbb{T} \to \mathbf{2}^{\mathbb{S}_M}$$

with

$\mathcal{C}$ *the set of channels,*

$\mathcal{P}$ *the set of pumps,*

$\mathcal{R}$ *the set of metabolic receptors,*

Γ *the Nernst and Goldmann equations,*

$\mathcal{I}_M$ *the membrane integration function,*

Γ^{-1} *the inverse Nernst and inverse Goldmann equations,*

$\mathbb{T}$ *the time space, and*

$\mathbb{S}_M$ *the local membrane states, which are input, internal and output states.*

Three types of membranes exist, namely:

- post-synaptic membrane (M_p),
- main membrane (M_m), and
- axonal membrane (M_a).

4.4.6.1 Post-synaptic membrane

The structure for a post-synaptic membrane is defined as:

Definition 30 (Post-synaptic membrane) *A post-synaptic membrane is defined as in definition 29, but it has:*

- *no voltage-gated channels, and*
- *no inverse Nernst and inverse Goldmann equations.*

By applying definition 30 to definition 29, the post-synaptic membrane operator $M_p \subseteq \sigma_M$, where σ_M is defined as in definition 15, is formally defined as:

$$M_p(\mathcal{C}_{M_p}, \mathcal{P}, \mathcal{R}, \mathcal{I}_M, \Gamma) : \mathbf{2}^{\mathbb{S}_M} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_M} \tag{4.43}$$

with

$\mathcal{C}_{M_p}$ the set of channel operators without voltage-gated channels,

$\mathcal{P}$ the set of pump operators,

$\mathcal{R}$ the set of metabolic receptor operators,

Γ the Nernst and Goldmann equation operator,

$\mathcal{I}_M$ the membrane integration function operator,

$\mathbb{T}$ the time space, and

$\mathbb{S}_M$ the local membrane states, which are input, internal and output states.

The block diagram for the message processing is shown in figure 4.10 and is based on section §4.3.6.

The local membrane states ($\mathbb{S}_M$) for a post-synaptic membrane can be derived from figure 4.10 as:

- the input state, U,
- the internal states, X_1 and X_2, and
- the output states, Y_1 and Y_2.

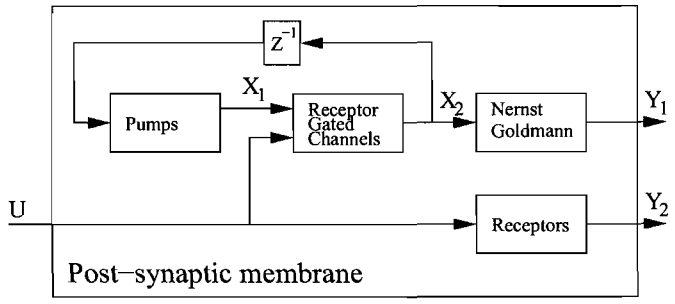


Figure 4.10: Post-synaptic membrane block diagram

A post-synaptic membrane always has:

- One input state, U,
- Two internal states, X_1 and X_2,
- At least one output state, Y_1, if the membrane has no metabolic receptors, and
- At most, two output states, Y_1 and Y_2.

The post-synaptic membrane states, i.e. input state vector U_{M_p}, internal state vector X_{M_p} and the output state vector Y_{M_p}, are:

$$U_{M_p} = \begin{bmatrix} U \end{bmatrix} \tag{4.44}$$

$$X_{M_p} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \tag{4.45}$$

$$Y_{M_p} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \tag{4.46}$$

If the notation in §3.5 on p. 44 is used, then a post-synaptic membrane is implemented with the post-synaptic membrane system of equations:

$$\begin{aligned} X_{M_p}^{(1)} &= \mathcal{P}z^{-1}X_{M_p}^{(2)} = P_k\left(...(P_2(P_1z^{-1}X_{M_p}^{(2)}))\right) \\ X_{M_p}^{(2)} &= \mathcal{C}_R X_{M_p}^{(1)} = C_\tau{}^{(m)}\left(...(C_\tau{}^{(1)}X_{M_p}^{(1)})\right) \\ Y_{M_p}^{(1)} &= \Gamma X_{M_p}^{(2)} \\ Y_{M_p}^{(2)} &= R_{M_p}U \end{aligned} \tag{4.47}$$

The generic-post-synaptic membrane's parameter values, to be substituted with the parameters in (3.15), are listed in table 4.11.

4.4.6.2 Axonal membrane

The structure for an axonal membrane is defined as:

Definition 31 (Axonal membrane) *An axonal membrane is defined as in definition 29, but it has no metabolic receptors.*

Table 4.11: Parameters and states for the post-synaptic membrane

Param	Value	Description	Unit	Dim
u	$U^{(1)}_{M_p}$	The internal and external ion species concentrations for Na^+, K^+, Cl^-	$mol \circ cm^{-3}$	3
	$U^{(2)}_{M_p}$	neurotransmitters in synaptic gap	molecules	q
x	$X^{(1)}_{M_p}$	result from set of pumps	$mol \circ cm^{-3}$	3
	$X^{(2)}_{M_p}$	result from set of receptor-gated channels	$mol \circ cm^{-3}$	3
y	$Y^{(1)}_{M_p}$	main membrane output	mV	1
	$Y^{(2)}_{M_p}$	metabolic receptor output	boolean	1
t	$t \in \mathbb{T}$	time step	ms	1
ν	$\mathcal{P}, C_R$	parameters to F_I		
ω	Γ, R	parameters to G		
F_I	$[\mathcal{P} \quad C_R]$	mapping to internal states		
G	$[\Gamma \quad R]^T$	mapping to output states		

By applying definition 31 to definition 29, the axonal membrane operator $M_a \subseteq \sigma_M$, where σ_M is defined as in definition 15, is formally defined as:

$$M_a(\mathcal{C}, \mathcal{P}, \mathcal{I}_M, \Gamma, \Gamma^{-1}) : \mathbf{2}^{\mathbb{S}_M} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_M}$$

with

$\mathcal{C}$ the set of channels,

$\mathcal{P}$ the set of pumps,

Γ the Nernst and Goldmann equations,

$\mathcal{I}_M$ the membrane integration function,

Γ^{-1} the inverse Nernst and inverse Goldmann equations,

$\mathbb{T}$ the time space, and

$\mathbb{S}_M$ the local membrane states, which are input, internal and output states.

The block diagram for the message processing is shown in figure 4.11 and is based on section §4.3.6.

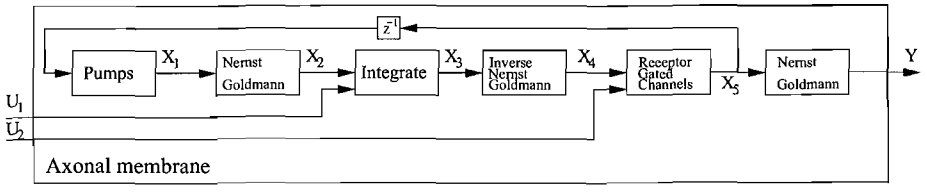


Figure 4.11: Axonal membrane block diagram

The Local membrane states ($\mathbb{S}_M$) for an axonal membrane are always:

- the input states, U_1 and U_2,

- the internal states, X_1 to X_5, and
- the output state Y.

The axonal membrane states, i.e. input state vector U_{M_a}, internal state vector X_{M_a} and the output state vector Y_{M_a}, are:

$$U_{M_a} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{4.48}$$

$$X_{M_a} = \begin{bmatrix} X_1 \\ \vdots \\ X_5 \end{bmatrix} \tag{4.49}$$

$$Y_{M_a} = \begin{bmatrix} Y \end{bmatrix} \tag{4.50}$$

If the notation in §3.5 on p. 44 is used, an axonal membrane is implemented with the axonal membrane system of equations:

$$\begin{aligned}
X_{M_a}^{(1)} &= \mathcal{P}z^{-1}X_{M_a}^{(5)} = P_k\left(...(P_2(P_1 z^{-1}X_{M_a}^{(5)}))\right) \\
X_{M_a}^{(2)} &= \Gamma X_{M_a}^{(1)} \\
X_{M_a}^{(3)} &= \mathcal{I}_M[X_{M_a}^{(2)} \quad U_{M_a}^{(1)}]^T \\
X_{M_a}^{(4)} &= \Gamma^{-1}X_{M_a}^{(3)} \\
X_{M_a}^{(5)} &= \mathcal{C}_R\left[X_{M_a}^{(4)} \quad U_{M_a}^{(2)}\right]^T = \mathcal{C}_R{}^{(m)}\left(...(\mathcal{C}_R{}^{(1)}\left[X_{M_a}^{(4)} \quad U_{M_a}^{(2)}\right]^T)\right) \\
Y_{M_a} &= \Gamma X_{M_a}^{(5)}
\end{aligned} \tag{4.51}$$

Table 4.12 shows the parameter values that need to be substituted with the parameters in (3.15) for the axonal membrane.

4.4.6.3 Main membrane

The structure for a main membrane is defined as:

Definition 32 (Main membrane) *A main membrane is defined as in definition 29, but it has:*

- *no receptor-gated channels,*
- *no metabolic receptors, and*
- *no inverse Nernst and inverse Goldmann equations.*

By applying definition 32 to definition 29, the main membrane operator $M_m \subseteq \sigma_M$, where σ_M is defined as in definition 15, is formally defined as:

$$M_m(\mathcal{C}_{M_m}, \mathcal{P}, \mathcal{I}_M, \Gamma) : \mathbf{2}^{\mathbb{S}_M} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_M}$$

with

$\mathcal{C}_{M_m}$ the set of channels without receptor-gated channels,

Table 4.12: Parameters and states for the axonal membrane

Param	Value	Description	Unit	Dim
u	$U_{M_a}^{(1)}$	The output from the main membrane	mV	1
	$U_{M_a}^{(2)}$	neurotransmitters in synaptic gap	molecules	q
x	$X_{M_a}^{(1)}$	result from set of pumps	$mol \circ cm^{-3}$	3
	$X_{M_a}^{(2)}$	result from Nernst and Goldmann equations	mV	1
	$X_{M_a}^{(3)}$	result from membrane integration function	mV	1
	$X_{M_a}^{(4)}$	result from inverse Nernst and Goldmann equations	$mol \circ cm^{-3}$	3
	$X_{M_a}^{(5)}$	result from set of receptor-gated channels	$mol \circ cm^{-3}$	3
y	Y_{M_a}	output of the Nernst and Goldmann equations	mV	1
t	$t \in \mathbb{T}$	time step	ms	1
ν	$\mathcal{P}, \Gamma, \mathcal{I}_M, \Gamma^{-1}, \mathcal{C}_R$	parameters to F_I		
ω	Γ	parameters to G		
F_I	$[\mathcal{P}, \Gamma, \mathcal{I}_M, \Gamma^{-1}, \mathcal{C}_R]^T$	mapping to internal states		
G	Γ	mapping to output states		

$\mathcal{P}$ the set of pumps,

Γ the Nernst and Goldmann equations,

$\mathcal{I}_M$ the membrane integration function,

$\mathbb{T}$ the time space, and

$\mathbb{S}_M$ the local membrane states, which are input, internal and output states.

The block diagram for the message processing is shown in figure 4.12 and is based on section §4.3.6.

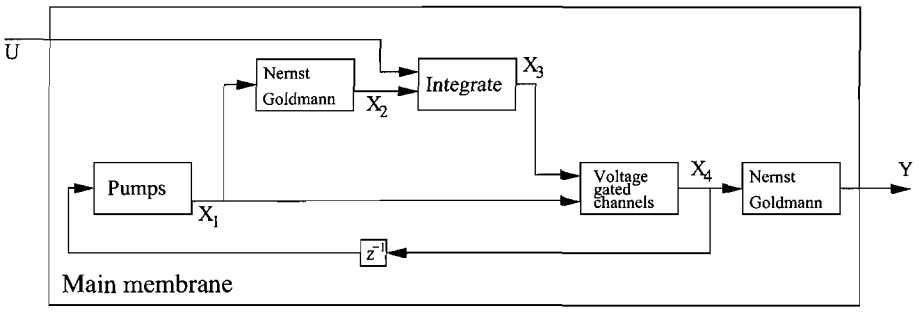


Figure 4.12: Main membrane block diagram

The local membrane states ($\mathbb{S}_M$) for a main membrane are always:

- the input state, U,
- the internal states, X_1 to X_4, and
- the output state Y.

The main membrane states, i.e. input state vector U_{M_m}, internal state vector X_{M_m} and the output state vector Y_{M_m}, are

$$U_{M_m} = \begin{bmatrix} U \end{bmatrix} \tag{4.52}$$

$$X_{M_m} = \begin{bmatrix} X_1 \\ \vdots \\ X_4 \end{bmatrix} \tag{4.53}$$

$$Y_{M_m} = \begin{bmatrix} Y \end{bmatrix} \tag{4.54}$$

If the notation in §3.5 on p. 44 is used, a main membrane is implemented with main membrane system of equations:

$$\begin{aligned}
X^{(1)}_{M_m} &= \mathcal{P} z^{-1} X^{(4)}_{M_m} = P_k \left(... (P_2 (P_1 z^{-1} X^{(4)}_{M_m})) \right) \\
X^{(2)}_{M_m} &= \Gamma X^{(1)}_{M_m} \\
X^{(3)}_{M_m} &= \mathcal{I}_M [X^{(2)}_{M_m} \quad U^{(2)}_{M_m}]^T \\
X^{(4)}_{M_m} &= \mathcal{C}_v \left[X^{(1)}_{M_m} \quad X^{(3)}_{M_m} \right]^T = {\mathcal{C}_v}^{(m)} \left(... ({\mathcal{C}_v}^{(1)} \left[X^{(1)}_{M_m} \quad X^{(3)}_{M_m} \right]^T) \right) \\
Y_{M_m} &= \Gamma X^{(4)}_{M_m}
\end{aligned} \tag{4.55}$$

Table 4.13 shows the parameter values that need to be substituted with the parameters in (3.15) for the main membrane.

Table 4.13: Parameters and states for the main membrane

Param	Value	Description	Unit	Dim
u	U_{M_m}	main membrane input	mV	1
x	$X^{(1)}_{M_m}$	result from set of pumps	$mol \circ cm^{-3}$	3
	$X^{(2)}_{M_m}$	result from Nernst and Goldmann equations	mV	1
	$X^{(3)}_{M_m}$	result from membrane integration function	mV	1
	$X^{(4)}_{M_m}$	result from set of voltage-gated channels	$mol \circ cm^{-3}$	3
y	Y_{M_m}	Nernst and Goldmann equations output	mV	1
t	$t \in \mathbb{T}$	time step	ms	1
ν	$\mathcal{P}, \Gamma, \mathcal{I}_M, \mathcal{C}_v$	parameters to F_l		
ω	Γ	parameters to G		
F_l	$[\mathcal{P}, \Gamma, \mathcal{I}_M, \mathcal{C}_v]^T$	mapping to internal states		
G	Γ	mapping to output states		

4.4.7 Long-term potential function

The LTP function operator μ is derived from the description in §4.3.9 and is formally defined as:

Definition 33 (LTP function operator μ)

$$\mu(\mathcal{R}, \tau_\mu, \mathcal{M}_p) : \mathbf{2}^{\mathbb{S}_\mu} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_\mu}$$

with

μ *the LTP function operator,*

$\mathcal{R}$ *is the set of metabolic receptors of which the outputs participate in* μ*,*

τ_μ *is the neurotransmitter that can be emitted,*

$\mathcal{M}_p$ *is the set of post-synaptic membranes that receive the emitted neurotransmitter,*

$\mathbb{S}_\mu$ *the LTP function states, which are input, internal and output states, and*

$\mathbb{T}$ *the time space.*

The delay described in §4.3.9 is implemented as a delay element in §4.4.3.

An LTP function is simple enough so that it is not necessary for sub-dividing it into sub-components. From §4.3.9 the input state vector U_μ, internal state vector X_μ and the output state vector Y_μ for the LTP function (μ) are:

$$U_\mu = \begin{bmatrix} U_1 \\ \vdots \\ U_n \end{bmatrix} \tag{4.56}$$

where $n = \#(\mathcal{M}_p)$.

$$X_\mu = [\ ] \tag{4.57}$$

$$Y_\mu = [\ Y\ ] \tag{4.58}$$

An LTP function is implemented with:

$$Y_\mu = \begin{cases} \tau_\mu & \bigwedge_{i=1,\cdots,n} U^{(i)}_{M_p} = \text{true} \\ x \in \emptyset & \text{otherwise} \end{cases} \tag{4.59}$$

4.4.8 Integration function

Based on the various descriptions of the integration function throughout §4.3, an integration function operator $\mathcal{I}$ is formally defined as:

Definition 34 (Integration function operator $\mathcal{I}$)

$$\mathcal{I}(n) : \mathbb{R}^n \rightarrow \mathbb{R}$$

with

n *the number of inputs to be used for the integration.*

An integration function is simple enough so that it is not necessary for subdividing it into subcomponents. From the various descriptions of the integration function in §4.3, the input state vector $U_{\mathcal{I}}$, internal state vector $X_{\mathcal{I}}$ and the output state vector $Y_{\mathcal{I}}$ for an integration function ($\mathcal{I}$) are:

$$U_{\mathcal{I}} = \begin{bmatrix} U_1 \\ \vdots \\ U_n \end{bmatrix} \tag{4.60}$$

$$X_{\mathcal{I}} = [\ ] \tag{4.61}$$

$$Y_{\mathcal{I}} = [\ Y\ ] \tag{4.62}$$

An integration function is implemented with:

$$Y_{\mathcal{I}} = \frac{1}{n}\sum_{i=1}^{n} U_{\mathcal{I}}^{(i)} \tag{4.63}$$

4.4.9 Vesicle

The vesicle operator V is derived from §4.3.5 and is formally defined as:

Definition 35 (Vesicle operator V)

$$V(\tau_V, \pi_V) : \mathbb{R} \rightarrow \mathbf{2}^{\mathcal{T}}$$

with

τ_V *is the emitted neurotransmitter by the vesicle,*

π_V *is the threshold potential above which the vesicle will emit a neurotransmitter, and*

$\mathcal{T}$ *is the set of neurotransmitters.*

A vesicle is simple enough so that it is not necessary for sub-dividing it into subcomponents. From §4.3.5 the input state vector U_V, internal state vector X_V and the output state vector Y_V for the vesicle (V) are:

$$U_V = [\ U\ ] \tag{4.64}$$

$$X_V = [\ ] \tag{4.65}$$

$$Y_V = [\ Y\ ] \tag{4.66}$$

A vesicle is implemented with:

$$Y_V = \begin{cases} \tau_V & \text{if } U_V \geq \pi_V \\ x \in \emptyset & \text{otherwise} \end{cases} \tag{4.67}$$

4.4.10 Nernst and Goldmann equations

Based on (3.3) and (3.8), the Nernst and Goldmann equations operator Γ is formally defined as:

Definition 36 (Nernst and Goldmann equations operator Γ)

$$\Gamma(P_K, P_{Na}, P_{Cl}, T) : \mathbb{R}^6 \rightarrow \mathbb{R}$$

where

P_K *is the membrane permeability to potassium ions,*

P_{Na} *is the membrane permeability to sodium ions,*

P_{Cl} *is the membrane permeability to chlorine ions, and*

T *is the membrane temperature.*

The Nernst and Goldmann equations are simple enough so that it is not necessary to subdivide them into subcomponents. From (3.3) and (3.8) the input state vector U_Γ, internal state vector X_Γ and the output state vector Y_Γ for the Nernst and Goldmann equations (Γ) are:

$$U_\Gamma = \begin{bmatrix} [K^+]_i \\ [Na^+]_i \\ [Cl^-]_i \\ [K^+]_e \\ [Na^+]_e \\ [Cl^-]_e \end{bmatrix}^T \tag{4.68}$$

$$X_\Gamma = \begin{bmatrix} E_{K+} \\ E_{Na+} \\ E_{Cl-} \end{bmatrix} \tag{4.69}$$

$$Y_\Gamma = \begin{bmatrix} E_M \end{bmatrix} \tag{4.70}$$

where

$[K^+]_i$ is the internal concentration of potassium ions,

$[Na^+]_i$ is the internal concentration of sodium ions,

$[Cl^-]_i$ is the internal concentration of chlorine ions,

$[K^+]_e$ is the external concentration of potassium ions,

$[Na^+]_e$ is the external concentration of sodium ions,

$[Cl^-]_e$ is the external concentration of chlorine ions,

E_{K-} is the Nernst potential of potassium ions,

E_{Na-} is the Nernst potential of sodium ions, and

E_{Cl-} is the Nernst potential of chlorine ions.

Apply (3.3) to the input components of U_Γ followed by the application of (3.8) to get the combined potential effect (Y_Γ) for the three ion species concentrations.

For better comparisons of the work done in the life sciences and biomedical engineering literature, (3.7) should be used, in this case $X_\Gamma = [\,]$.

4.4.11 Inverse Nernst and Goldmann equations

Based on (3.5) and (3.11), the inverse Nernst and inverse Goldmann equations operator Γ^{-1} is formally defined as:

Definition 37 (Inverse Nernst and inverse Goldmann equations operator Γ^{-1})

$$\Gamma^{-1}(E_M(0), E_{K^+}(0), E_{Na^+}(0), E_{Cl^-}(0)) : \mathbb{R} \rightarrow \mathbb{R}^3$$

where

$E_M(0)$ *is the membrane potential at* $t = 0$,

$E_{K^+}(0)$ *is the potential due to potassium ions at* $t = 0$,

$E_{Na^+}(0)$ *is the potential due to sodium ions at* $t = 0$, *and*

$E_{Cl^-}(0)$ *is the potential due to chlorine ions at* $t = 0$.

The inverse Nernst and inverse Goldmann equations are simple enough so that it is not necessary to subdivide them into subcomponents. From (3.5) and (3.11) the input state vector $U_{\Gamma^{-1}}$, internal state vector $X_{\Gamma^{-1}}$ and the output state vector $Y_{\Gamma^{-1}}$ for the inverse Nernst and inverse Goldmann equations (Γ^{-1}) are:

$$U_{\Gamma^{-1}} = \begin{bmatrix} E_M \end{bmatrix} \tag{4.71}$$

$$X_{\Gamma^{-1}} = \begin{bmatrix} E_M \\ E_{K^+} \\ E_{Na^+} \\ E_{Cl^-} \end{bmatrix} \tag{4.72}$$

$$Y_{\Gamma^{-1}} = \begin{bmatrix} [K^+] \\ [Na^+] \\ [Cl^-] \end{bmatrix} \tag{4.73}$$

where

$E_M(t)$ is the membrane potential,

E_{K^+} is the potential due to potassium ions,

E_{Na^+} is the potential due to sodium ions,

E_{Cl^-} is the potential due to chlorine ions,

$[K^+]$ is the concentration of potassium ions,

$[Na^+]$ is the concentration of sodium ions, and

$[Cl^-]$ is the concentration of chlorine ions.

Apply (3.11) to $U_{\Gamma^{-1}}$ to get the potential contribution of the three ion species, followed by the application of (3.5) to get the three ion species concentrations (Y_Γ).

4.4.12 Membrane pores

Before pumps and channels were discovered, standard textbooks in physiology postulated that ions diffused through holes in the cell membrane called membrane pores, see [62]. The nature of the membrane pores were unknown at the time.

According to [158], the explanations or theories on the mechanisms of active transport and passive transport of substances across the cell membranes have developed a lot since the mid-1970's.

Active transport takes place through pumps [13, 41, 125, 137, 158] and for this thesis the following are relevant:

- The flow goes against gradients of chemical or electrical potentials.
- The flow is specifically for Na^+, K^+, Cl^- ions.

Passive transport takes place through channels [13, 41, 125, 137, 158] and for this thesis the following are relevant:

- The flow goes with gradients of chemical or electrical potentials.
- The flow is specifically for Na^+, K^+, Cl^- ions.

In §4.2.8 or §4.2.7, a membrane pore is not explicitly defined, neither has it been defined in [30, 36, 32, 145, 144]. By looking closely at §4.2.8, §4.2.7, §4.3.8 and §4.3.7, if the control signature δ is expanded to also include the pumps, a formal definition for the membrane pore operator Ψ is defined as:

Definition 38 (membrane pore operator Ψ)

$$\Psi(\sigma, \gamma, \delta) : \mathbf{2}^{\mathbb{S}_\Psi} \times \mathbb{T} \rightarrow \mathbf{2}^{\mathbb{S}_\Psi}$$

with

σ *ion that is passed by the membrane pores,*

γ *is the ionic flow rate through the pore* ($mol \circ s^{-1}$),

δ *the channel control signature,*

$\mathbb{T}$ *the time space, and*

$\mathbb{S}_\Psi$ *the local membrane pore states, which are input, internal and output states.*

Appendix D shows how this definition is implemented as a class, using the C++ programming language, and a pump class inherits it.

4.4.12.1 Ion pump

It is assumed that a pump transports only one specific ion species σ across a membrane. Based on §4.3.8, the pump operator P is formally defined as in definition 38 with

Definition 39 (Pump control signature)

$$\delta =< \theta_p, \phi_p >$$

where

θ_p *is concentration thresholds* $(mol \circ cm^{-3})$, *and*

ϕ_p *is a direction* $\phi \in \{\text{in}, \text{out}\}$.

A pump is simple enough so that it is not necessary to subdivide it into subcomponents. From §4.3.8 the input state vector U_P, internal state vector X_P and the output state vector Y_P for a pump (P) are:

$$U_P = \begin{bmatrix} [\sigma]_i \\ [\sigma]_e \end{bmatrix} \tag{4.74}$$

$$X_P = \begin{bmatrix} o_P \end{bmatrix} \tag{4.75}$$

$$Y_P = \begin{bmatrix} [\sigma]_i^* \end{bmatrix} \tag{4.76}$$

where

$[\sigma]_i$ is the internal ion concentration of the species σ at a membrane,

$[\sigma]_e$ is the external ion concentration of the species σ at a membrane,

o_P is the openness and transport direction, and

$[\sigma]_i^*$ is the newly-calculated internal ion concentration.

A pump is implemented with:

$$[\sigma]_i^* = [\sigma]_i + \gamma o_P \tag{4.77}$$

where

$$o_P = \begin{cases} 1 & \text{if } [\sigma]_i \leq \theta \ \wedge \ \phi = \text{in} \\ -1 & \text{if } [\sigma]_i > \theta \ \wedge \ \phi = \text{out} \\ 0 & \text{otherwise} \end{cases} \tag{4.78}$$

4.4.12.2 Ion channels

Based on §4.3.7, three types of channels exist:

- A receptor-gated channel (C_τ).
- A voltage-gated channel (C_v), and
- A concentration channel (C_c).

It is assumed that a channel transports only one specific ion species σ across a membrane. Based on §4.3.7, the different channel operators are formally defined as in definition 38 with:

Definition 40 (Channel control signatures)

1. *For a receptor-gated channel* (C_τ)*:*

$$\delta = < \kappa, t_b, o_b >$$

where

κ *is a receptor molecule for a receptor-gated channel,*

t_b *is the block time determined by* $b(\kappa, \tau^*)$ *for a receptor-gated channel, and*

o_b *is the openness determined by* $\nu(\kappa, \tau^*)$ *for a receptor-gated channel.*

2. *For a voltage-gated channel (C_v):*

$$\delta =< \pi_c, \phi_c >$$

where

π_c *is a threshold potential, and*

$\phi_c \in \{\text{minimum}, \text{maximum}\}$.

3. *For a concentration channel (C_c):*

$$\delta =< \theta_c >$$

where θ_c is a default concentration.

A channel is simple enough so that it is not necessary to subdivide it into subcomponents. From §4.3.7 the input state vector U_C, internal state vector X_C and the output state vector Y_C for a channel (C) are:

$$U_C = \begin{bmatrix} p_C \\ \mathcal{T}_C \\ [\sigma]_i \\ [\sigma]_e \end{bmatrix} \tag{4.79}$$

$$X_C = \begin{bmatrix} o_C \\ t_C \end{bmatrix} \tag{4.80}$$

$$Y_C = \begin{bmatrix} [\sigma]_i^* \end{bmatrix} \tag{4.81}$$

where

$p_C \in \mathbb{R}$ is the input potential, $p_C \in \emptyset$ for receptor-gated channels and concentration channels,

$\mathcal{T}_C$ is the set of neurotransmitters, $\mathcal{T}_C \in \emptyset$ for voltage-gated channels and concentration channels,

$[\sigma]_i$ is the internal ion concentration of species σ,

$[\sigma]_e$ is the external ion concentration of species σ,

o_C is the openness degree $o_C \in \{0, 1\}$,

t_C is the number of simulation steps before a blocked state expires ($t_C = 0$ means that C is not blocked), $t_C \in \emptyset$ for voltage-gated channels and concentration channels, and

$[\sigma]_i^*$ is the newly-calculated internal ion concentration.

The operation of a channel is implemented for all types of channels with:

$$[\sigma]_i^* = \begin{cases} [\sigma]_i + \gamma o_C & \text{if } [\sigma]_i < [\sigma]_e \\ [\sigma]_i - \gamma o_C & \text{if } [\sigma]_i > [\sigma]_e \\ [\sigma]_i & \text{otherwise} \end{cases} \tag{4.82}$$

$$\tag{4.83}$$

where:

- For a receptor-gated channel:

$$t_C = \begin{cases} t_b & \text{if } (\kappa \in \tau_C) \wedge (z^{-1}t_C = 0) \\ z^{-1}t_C - 1 & \text{if } z^{-1}t_C > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.84}$$

$$o_C = \begin{cases} o_b & \text{if } ((\kappa \in \tau_C) \wedge (t_C = 0)) \bigvee (t_C > 0) \\ 0 & \text{otherwise} \end{cases} \tag{4.85}$$

- For a voltage-gated channel:

$$o_C = \begin{cases} 1 & \text{if } (\phi = \text{minimum} \wedge \mathcal{P}_C \geq \pi_c) \vee (\phi = \text{maximum} \wedge \mathcal{P}_C \leq \pi_c) \\ 0 & \text{otherwise} \end{cases} \tag{4.86}$$

- For a concentration channel:

$$o_C = \begin{cases} 1 & \text{if } [\sigma]_i \geq \theta_c \\ 0 & \text{otherwise} \end{cases}$$

4.4.13 Receptor

Based on §4.3.6, the metabolic receptor operator R is formally defined as:

Definition 41 (Metabolic receptor operator R)

$$R(t_b, \tau_R) : \mathbf{2}^{\tau} \times \mathbb{T} \rightarrow \{\text{false}, \text{true}\}$$

with

t_b *is the blocking time for the receptor,*

τ_R *is the neurotransmitter that is triggers the metabolic event flag,*

τ *is the set of neurotransmitters, and*

$\mathbb{T}$ *the time space.*

A metabolic receptor is simple enough so that it is not necessary to subdivide it into subcomponents. From §4.3.6 the input state vector U_R, internal state vector X_R and the output state vector Y_R for a metabolic receptor (R) are:

$$U_R = \begin{bmatrix} \tau_R \end{bmatrix} \tag{4.87}$$

$$X_R = \begin{bmatrix} t_R \end{bmatrix} \tag{4.88}$$

$$Y_R = \begin{bmatrix} e_R \end{bmatrix} \tag{4.89}$$

where

$\tau_R \subseteq \tau$ is a neurotransmitter,

t_R is the number of time steps left before the blocked state expires, and

$e_R \in \{\text{false}, \text{true}\}$ is the metabolic event flag.

The operation of a metabolic receptor is implemented with:

$$t_R = \begin{cases} z^{-1}t_R - 1 & \text{if } z^{-1}t_R > 1 \\ t_b + 1 & \text{if } (\tau_R \in \tau_R) \wedge (z^{-1}t_R = 0) \\ 0 & \text{otherwise} \end{cases} \tag{4.90}$$

$$Y_R = \begin{cases} \text{true} & z^{-1}t_R > 0 \\ \text{false} & \text{otherwise} \end{cases} \tag{4.91}$$

4.5 Memory

Similar to a classic artificial neural network, see [68], long-term memory in a RealNeuron network lies in the synaptic connections between the neurons.

Unlike a classic artificial neural network, see [68], that has no short-term memory, a RealNeuron network can have a short-term memory. Short-term memory in a RealNeuron network is the ability of a neuron to change its own functionality temporarily based on neurotransmitters present in the synaptic gaps.

4.6 Summary

The major contribution of this thesis has been presented in this chapter:

- The completion and standardisation of terminology and symbology of the predicate description of the RealNeuron in §4.2.
- The block diagrams in §4.4.
- The development of a state space model in §4.4.

According to definition 1 on p. 9, the models presented in this chapter are still only a *pure theory*. In chapter 6 it will unfold into a *descriptive theory* in accordance with definition 2 on p. 9 when the parameters are given actual values to perform simulation, based on neuroscience data.

4.6.1 RealNeuron network summary

The RealNeurons in figure 4.1 can be classified into three different types of RealNeurons. The RealNeurons on the left in figure 4.1 are sensor neurons. The RealNeurons on the right in figure 4.1 are motor neurons. The RealNeurons, in between, in figure 4.1 are internal neurons.

The implementation of figure 4.1 is summarised in table 4.14. Each subcomponent is cross-referenced to its description in this chapter.

Table 4.14: Summary of RealNeuron network components

No.	Component	Paragraph	Figure of block diagram	Parameters table no.	Components reference
1	sensor neuron	4.4.2.1	4.3	4.3	4, 7
2	neuron	4.4.2.2	4.2	4.4	5, 6, 7
3	motor neuron	4.4.2.3	4.4	4.5	5
4	sensoric soma	4.4.3.2	4.6	4.7	9
5	input soma	4.4.3.1	4.5	4.6	8, 9, 11, 12
6	axon	4.4.4	4.8	4.9	10
7	bulb end	4.4.5	4.9	4.10	13, 14
8	post-synaptic membrane	4.4.6.1	4.10	4.11	15, 17, 18, 20, 21
9	main membrane	4.4.6.3	4.12	4.13	15, 17, 18, 19
10	axonal membrane	4.4.6.2	4.11	4.12	15, 16, 17, 18, 20
11	soma integration function	4.4.8	-	-	-
12	LTP function	4.4.7	-	-	-
13	bulb-end soma	4.4.3.3	4.7	4.8	8, 9, 11, 12
14	vesicle	4.4.9	-	-	-
15	Nernst and Goldmann equations	4.4.10	-	-	-
16	inverse Nernst and inverse Goldmann equations	4.4.11	-	-	-
17	membrane integration function	4.4.8	-	-	-
18	pump	4.4.12.1	-	-	-
19	voltage-gated channel	4.4.12.2	-	-	-
20	receptor-gated channel	4.4.12.2	-	-	-
21	metabolic receptor	4.4.8	-	-	-

Chapter 5

Theoretical evaluation of the RealNeuron

> *A poor relation – is the most irrelevant thing in nature* — Charles Lamb
> *Last Relation of Elia* (1833) 'Poor Relations'

This chapter compares the model of the RealNeuron, which was developed in chapter 4, with that of perceptrons, spiking neurons and implicitly with that of the Hodgkin-Huxley model [72]. Furthermore, a theoretical evaluation of the model developed in chapter 4, to determine the performance of the model in the presence of noise, is presented in this chapter. These aspects have been peer reviewed and accepted [47].

With respect to the scientific and engineering way of life (§1.7.1.1 p. 8 and §1.7.1.2 p. 9) this chapter compares the model (theory) of the RealNeuron with the model (theories) of perceptrons and spiking neurons respectively. Implicitly, the RealNeuron is also compared to the Hodgkin-Huxley model. According to definition 1 the comparison of models in this chapter is only between the pure theories.

On a structural level, it has been argued that the RealNeuron is an instance of a spiking neuron [30]. If this is true, all the theories for spiking neurons should also hold true for the RealNeuron [30].

This chapter validates the RealNeuron's to be a carefully reduced model that retains essential features of more complex models, i.e. the Hodgkin-Huxley model.

5.1 General mathematical relation

A reference equation is used to compare RealNeurons with spiking neurons and perceptrons and implicitly with the Hodgkin-Huxley model. The spike response model described in [58] is used as the reference equation. In [88, 58] it is shown that the firing times produced by this model coincides with 90% of the action potentials of the Hodgkin-Huxley model [72]. This is a simple and yet useful description of neurodynamics [88, 58].

If a neuron is denoted by n_i and n_j $\forall j \in \{j | j \text{ pre} - \text{synaptic to } i\}$ denote the neurons whose outputs are inputs to n_i, then n_i is the post-synaptic neuron and all n_j are the presynaptic neurons. A connection between a n_j and n_i is called a synapse. In the rest of this section the details of a synapse should not be confused with the details of a biological synapse or the synapse between two RealNeurons in §3.3.2.

The spike response model according to [58] is:

$$u_i(t) = \sum_{t_i^{(f)} \in \mathcal{F}_i} \eta_i(t - t_i^{(f)}) + \sum_{j \in \Gamma_i} \sum_{t_j^{(f)} \in \mathcal{F}_j} w_{ij} \varepsilon_{ij}(t - t_j^{(f)}) \tag{5.1}$$

and the post-synaptic neuron n_i fires when $u_i = \vartheta$, where

- $u_i(t)$ is the internal state of n_i,
- t is the time,
- $\mathcal{F}_i = \{t_j^{(f)} | 1 \leq f \leq n\} = \{t | u_i(t) = \vartheta, \dot{u}_i > 0\} = \{t_i^{(1)}, ..., t_i^{(n)}\}$ is the set of all firing times of n_i, with the time for the most recent spike emitted by n_i as $t_i^{(n)}$,
- n is the number of spikes fired before time t,
- ϑ is the firing threshold,
- $\Gamma_i = \{j | j \text{ presynaptic to } i\}$ set of presynaptic neuron indices,
- $\eta_i(s)$ is a refractory function of n_i,
- w_{ij} is a weight that represents the connection strength between the neurons j and i, and
- $\varepsilon_{ij}(s)$ is the transfer function between neurons j and i, with $\varepsilon_{ij}(s) = 0$ for $s \leq 0$.

A spike s_i is emitted by n_i when $u_i = \vartheta$, and in general

$$s_i(t) = \delta(u_i(t) - \vartheta) \tag{5.2}$$

where $\delta(.)$ is the Dirac δ function with $\delta(z) = 0$ if $z \neq 0$ and $\int_{-\infty}^{\infty} \delta(\zeta) d\zeta = 1$.
The spike train for n_i is then

$$\begin{aligned} S_i(t) &= \int_{-\infty}^{t} s_i(\tau) d\tau \\ &= \sum_{t_j^{(f)} \in \mathcal{F}_i} \delta(t - t_j^{(f)}) \end{aligned} \tag{5.3}$$

If x_i is the internal state of n_i, y_i is the output state of n_i and u_i is the input state of n_i, then equations (5.1) and (5.2) are rewritten as:

$$x_i(t) = \sum_{t_i^{(f)} \in \mathcal{F}_i} \eta_i(t - t_i^{(f)}) + \sum_{j \in \Gamma_i} \sum_{t_j^{(f)} \in \mathcal{F}_j} w_{ij} \varepsilon_{ij}(u_j, t - t_j^{(f)}) \tag{5.4}$$

$$y_i(t) = \delta\left(x_i(t) - \vartheta\right) \tag{5.5}$$

If $\delta(.)$ is generalised as a transfer function $\phi(.)$ then (5.5) is rewritten as:

$$y_i(t) = \phi\left(x_i(t) - \vartheta\right) \tag{5.6}$$

$\phi(.)$ is a function defined as $\phi : R \rightarrow R$.
Examples of transfer functions are

$$Y = \tanh(x - \varphi)$$

$$Y = e^{(x-\varphi)}$$

$$Y = \mathcal{H}(x-\varphi)$$

where $\mathcal{H}(.)$ is the Heaviside step function $\mathcal{H}(z) = \int_{-\infty}^{z} \delta(\zeta)d\zeta$, with

$$\mathcal{H}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

The Heaviside function is also called a step function and in general the following substitution can be used $z = x - \varphi$.

5.1.1 RealNeuron related to pulsed neurons

The work in this paragraph has been peer reviewed in [47].

An equation for a RealNeuron is derived from §4.4.2 as

$$\begin{aligned} Y_N &= \mathcal{Y} \circ A \circ S\,(U_N) \\ &= \mathcal{Y}\,(A\,(S(U_N))) \\ &= \begin{bmatrix} \Upsilon_1\,(A\,(S(U_N))) \\ \vdots \\ \Upsilon_m\,(A\,(S(U_N))) \end{bmatrix} \end{aligned} \tag{5.7}$$

This equation cannot be directly related to (5.1). Clearly, the structure of the RealNeuron needs to be rearranged to make the relation possible. Based on §4.4.3, S is expanded and (5.7) is rewritten as

$$Y_N = \mathcal{Y} \circ A \circ M_m \circ \mathcal{I}_S \circ \mathcal{M}_p \circ \left(1 + z^{-1} \circ \mu_i \circ \mathcal{M}_p\right)(U_N) \tag{5.8}$$

Instead of defining the output states of the set of bulb ends ($\mathcal{Y}$) as the output of the RealNeuron, the output state of the axon (A) is defined as the output. Furthermore, instead of defining the post-synaptic membranes as input for the RealNeuron, the bulb ends are defined as the input to the RealNeuron. By applying these structural changes (5.8) is rewritten as:

$$Y_N^* = A \circ M_m \circ \mathcal{I}_S \circ \mathcal{M}_p \circ \left(1 + z^{-1} \circ \mu \circ \mathcal{M}_p\right) \circ \mathcal{Y}\,(U_N^*) \tag{5.9}$$

Assume that the RealNeuron, to be compared with the pulse neuron, has no LTP function, thus $\mu = 0$, and no axon, thus $A = I$ then (5.9) is rewritten as:

$$Y_N^* = M_m \circ \mathcal{I}_S \circ \mathcal{M}_p \circ \mathcal{Y}\,(U_N^*) \tag{5.10}$$

Based on the description in §4.4.8 and defining $\Gamma_i = \{j \mid j \text{ presynaptic to } i\}$, (5.10) is rewritten as:

$$\begin{aligned} Y_N^* &= M_{mi} \circ \mathcal{I}_{Si} \circ \mathcal{M}_{pi} \circ \mathcal{Y}\,(U_N^*) \\ &= M_{mi} \circ \frac{1}{\#(M_p)} \sum_{j \in \Gamma_i} \underbrace{M_{pij} \circ \Upsilon_j\left(U_{Nj}^*\right)}_{\substack{\textit{synapse between} \\ N_j \textit{ and } N_i}} \end{aligned} \tag{5.11}$$

Based on the description in §4.4.6.3 an equation for M_{mi} is:

$$\begin{aligned} X_{M_{mi}} &= C_v\left(\frac{1}{2}\begin{bmatrix} \Gamma \circ \mathcal{P}\left(z^{-1}X_{M_{mi}}\right) \\ \mathcal{P}\left(z^{-1}X_{M_{mi}}\right) \end{bmatrix} + \frac{1}{2}\begin{bmatrix} U_{M_{mi}} \\ \mathcal{P}\left(z^{-1}X_{M_{mi}}\right) \end{bmatrix}\right) \\ Y_{M_{mi}} &= \Gamma(X_{M_{mi}}) \end{aligned} \tag{5.12}$$

By substituting M_{mi} in (5.11) with (5.12), (5.11) is rewritten as:

$$X_N^* = \begin{bmatrix} \Gamma \circ \mathcal{P}\left(z^{-1} X_{M_{mi}}\right) \\ \mathcal{P}\left(z^{-1} X_{M_{mi}}\right) \end{bmatrix} + \begin{bmatrix} \frac{1}{\#(M_p)} \sum_{j \in \Gamma_i} M_{pij} \circ \Upsilon_j\left(U_{Nj}^*\right) \\ \mathcal{P}\left(z^{-1} X_{M_{mi}}\right) \end{bmatrix} \quad (5.13)$$

$$Y_N^* = \Gamma \circ C_v\left(\frac{1}{2} X_N^*\right) \quad (5.14)$$

The relation between (5.14) and (5.6) is:

$$\phi\left(x_i(t) - \vartheta\right) = \Gamma \circ C_v\left(\frac{1}{2} X_N^*\right) \quad (5.15)$$

Furthermore, the relation between terms in (5.13) and (5.4) is:

$$\sum_{j \in \Gamma_i} \sum_{t_j^{(f)} \in \mathcal{F}_j} w_{ij}\, \varepsilon_{ij}(u_j, t - t_j^{(f)}) = \frac{1}{\#(M_p)} \sum_{j \in \Gamma_i} M_{pij} \circ \Upsilon_j\left(U_{Nj}^*\right) \quad (5.16)$$

Based on this equation the following relation is derived:

$$\sum_{t_j^{(f)} \in \mathcal{F}_j} w_{ij}\, \varepsilon_{ij}(u_j, t - t_j^{(f)}) = \frac{1}{\#(M_p)} M_{pij} \circ \Upsilon_j\left(U_{Nj}^*\right) \quad (5.17)$$

The refractory-function term relation between (5.13) and (5.4) is:

$$\sum_{t_i^{(f)} \in \mathcal{F}_i} \eta_i(t - t_i^{(f)}) = \Gamma \circ \mathcal{P}\left(z^{-1} X_{M_{mi}}\right) \quad (5.18)$$

The relation with the reference (5.4) and (5.6) of the RealNeuron is given in (5.15) to (5.18).

5.1.2 RealNeuron relation to McCulloch-Pitts model

The work in this paragraph has been peer reviewed in [47].

The McCulloch-Pitts or perceptron model is given by the following equation, [108]:

$$y_k = \psi\left(\sum_{j=1}^{p} w_{kj} x_j - \theta_k\right) \quad (5.19)$$

where

- $\psi(.)$ - the activation function and for the McCulloch-Pitts this is the threshold function, i.e.
$$\psi(\nu) = \begin{cases} 1 & \text{if } \nu \geq 0 \\ 0 & \text{if } \nu < 0 \end{cases}$$
- y_k - the output of the kth neuron,
- x_j - the input from the jth neuron,
- w_{kj} - the synaptic weight between the jth neuron and the kth neuron, and
- θ_k - the threshold of the kth neuron.

If x_i is the internal state of n_i, y_i is the output state of n_i and u_i is the input state of n_i, then equation (5.19) is rewritten as:

$$x_i(t) = \sum_{j=1}^{p} w_{ij} u_{ij} \tag{5.20}$$

$$y_i(t) = \psi\left(x_i(t) - \theta_i\right) \tag{5.21}$$

The relation between (5.21) and (5.6) is:

$$\phi\left(x_i(t) - \vartheta\right) = \psi\left(x_i(t) - \theta_i\right) \tag{5.22}$$

Furthermore, the relation between terms in (5.20) and (5.4) is:

$$\sum_{j \in \Gamma_i} \sum_{t_j^{(f)} \in \mathcal{F}_j} w_{ij}\, \varepsilon_{ij}(u_j, t - t_j^{(f)}) = \sum_{j=1}^{p} w_{ij} u_{ij} \tag{5.23}$$

Based on this equation the following relation is derived:

$$\sum_{t_j^{(f)} \in \mathcal{F}_j} w_{ij}\, \varepsilon_{ij}(u_j, t - t_j^{(f)}) = w_{ij} u_{ij} \tag{5.24}$$

A McCulloch-Pitts neuron does not have a refactoring function thus:

$$\sum_{t_i^{(f)} \in \mathcal{F}_i} \eta_i(t - t_i^{(f)}) = 0 \tag{5.25}$$

The relation with the reference (5.4) and (5.6) of the McCulloch-Pitts neuron is given in (5.22) to (5.25).

The relation of the McCulloch-Pitts neuron to the RealNeuron is derived by comparing (5.22) to (5.25) with (5.15) to (5.18) respectively:

$$\Gamma \circ C_v\left(\frac{1}{2} X_N^*\right) = \psi\left(x_i(t) - \theta_i\right) \tag{5.26}$$

$$\frac{1}{\#(M_p)} \sum_{j \in \Gamma_i} M_{pij} \circ \Upsilon_j\left(U_{Nj}^*\right) = \sum_{j=1}^{p} w_{ij} u_{ij} \tag{5.27}$$

$$\frac{1}{\#(M_p)} M_{pij} \circ \Upsilon_j\left(U_{Nj}^*\right) = w_{ij} u_{ij} \tag{5.28}$$

$$\Gamma \circ \mathcal{P}\left(z^{-1} X_{M_{mi}}\right) = 0 \tag{5.29}$$

5.2 Performance in a noisy environment

The work in this paragraph has been peer reviewed in [47].

For the RealNeuron to be practical in engineering applications, it is important to determine it's robustness for noise in a noisy environment. The noise is coming from the physical sensor measurements [39].

The sensor neurons determine the performance of a RealNeuron network in a noisy environment. Furthermore, the voltage-gated channels determine the performance of a sensor neuron.

The equation for a sensor neuron is derived from §4.4.2.1 as

$$\begin{aligned} Y_{N_\varpi} &= \mathcal{Y} \circ S\left(U_{N_\varpi}\right) \\ &= \mathcal{Y}\left(S(U_{N_\varpi})\right) \end{aligned} \tag{5.30}$$

From §4.4.3.2, (5.30) is rewritten as

$$Y_{N_\varpi} = \mathcal{Y} \circ M_m\left(U_{N_\varpi}\right) \tag{5.31}$$

Based on the description in §4.4.6.3 and defining $X_{M_{mi}}$ as in (5.12) but replacing $U_{M_{mi}}$ with U_{N_ϖ}, (5.31) is rewritten as:

$$Y_{N_\varpi} = \mathcal{Y} \circ \Gamma\left(X_{M_{mi}}\right) \tag{5.32}$$

By adding noise Δ to the input state of (5.32), (5.12) is rewritten as

$$\begin{aligned} X^*_{M_{mi}} &= C_v\left(\frac{1}{2}\begin{bmatrix} \Gamma \circ \mathcal{P}\left(z^{-1}X^*_{M_{mi}}\right) \\ \mathcal{P}\left(z^{-1}X^*_{M_{mi}}\right) \end{bmatrix} + \frac{1}{2}\begin{bmatrix} U_{N_\varpi} + \Delta \\ \mathcal{P}\left(z^{-1}X^*_{M_{mi}}\right) \end{bmatrix}\right) \\ Y^*_{M_{mi}} &= \Gamma(X^*_{M_{mi}}) \end{aligned} \tag{5.33}$$

From §4.4.12.2, C_v in (5.33) will start conducting if the following condition is met

$$\frac{1}{2}\left(\Gamma \circ \mathcal{P}\left(z^{-1}X^*_{M_{mi}}\right) + U_{N_\varpi} + \Delta\right) \geq \pi_c \tag{5.34}$$

Thus, C_v will not conduct if

$$\Gamma \circ \mathcal{P}\left(z^{-1}X^*_{M_{mi}}\right) + U_{N_\varpi} + \Delta < 2\pi_c \tag{5.35}$$

and noise will not affect the performance of C_v if

$$\Delta < 2\pi_c - \Gamma \circ \mathcal{P}\left(z^{-1}X^*_{M_{mi}}\right) - U_{N_\varpi} \tag{5.36}$$

It is seen from (5.36) that for a given threshold π_c the sensitivity to noise of C_v is determined by the refractory function

$$\Gamma \circ \mathcal{P}\left(z^{-1}X^*_{M_{mi}}\right)$$

which is the feedback through the set of pumps of the previous output state of C_v.

C_v can be made more sensitive to noise, increasing the output value of the refractory function, but if

$$\Gamma \circ \mathcal{P}\left(z^{-1}X^*_{M_{mi}}\right) \geq 2\pi_c$$

then C_v loses all its sensitivity towards any input and will conduct constantly.

C_v can also be made less sensitive to noise by decreasing the output value of the refractory function. However, if

$$\Gamma \circ \mathcal{P}\left(z^{-1}X^*_{M_{mi}}\right) < \sup(2\pi_c - U_{N_\varpi}) = 2\pi_c - \inf U_{N_\varpi}$$

then C_v will only conduct if Δ is big enough, i.e.

$$U_{N_\varpi} + \Delta > 2\pi_c$$

By making

$$\Gamma \circ \mathcal{P}\left(z^{-1}X^*_{M_{mi}}\right) < 2\pi_c - \inf(U_{N_\varpi} + \Delta)$$

C_v loses all its sensitivity towards any input and will not conduct at all.

The performance of the RealNeuron in a noisy environment is determined by the following:

- For a sensor neuron to function correctly in a noisy environment the value of the refractory function in (5.33) must be

$$2\pi_c - \inf(U_{N_\varpi} + \Delta) < \Gamma \circ \mathcal{P}\left(z^{-1} X^*_{M_{mi}}\right) < 2\pi_c \tag{5.37}$$

- A sensor neuron is sensitive to only the noise Δ in the environment, thus it will be insensitive to U_{N_ϖ} if the value of the refractory function in (5.33) is

$$2\pi_c - \inf(U_{N_\varpi} + \Delta) < \Gamma \circ \mathcal{P}\left(z^{-1} X^*_{M_{mi}}\right) < 2\pi_c - \inf U_{N_\varpi} \tag{5.38}$$

- A sensor neuron is insensitive to any noise from the environment if

$$\inf U_{N_\varpi} > \sup \Delta \tag{5.39}$$

and the refractory function in (5.33) is

$$2\pi_c - \inf U_{N_\varpi} < \Gamma \circ \mathcal{P}\left(z^{-1} X^*_{M_{mi}}\right) < 2\pi_c - \sup \Delta \tag{5.40}$$

5.3 Summary

This chapter confirms that the RealNeuron is related to the other types of artificial neurons. The work done on first, second and third generation artificial neuron models in [108, 68, 104] and other similar publications can be applied to the RealNeuron by using the relations presented in this chapter and following a similar reasoning to that in [102].

More specifically, the model of the RealNeuron is compared with perceptron and spiking-neuron models in §5.1.2 and §5.1.1 respectively.

The RealNeuron relation with the perceptron is given in (5.26) to (5.29). The theories for perceptrons are applicable to this specific configuration of RealNeurons.

The RealNeuron relation with the spiking neuron is given in (5.15) to (5.18). This confirms the result mathematically that a RealNeuron is a type of pulsing neuron as reported in [30]. The theories for pulsing neurons are also applicable to configurations of RealNeuron that fulfil the relations in (5.15) to (5.18). Thus, based on the proofs in [102] it can be claimed that a RealNeuron network, consisting of a finite number of RealNeurons, can simulate a universal Turing machine. It means that, in principle, any problem that is computable [21] could be solved with RealNeuron networks.

The RealNeuron's is indeed a carefully reduced model that retains essential features of more complex models. This is validated with the above results.

This chapter also presents a theoretical evaluation of the model developed in chapter 4 to determine the performance of the model in the presence of noise.

The performance of a RealNeuron network, in a noisy environment, is dependent on the output value of the refractory functions of the sensor neurons in the network. For a sensor neuron to function correctly in a noisy environment the value of the refractory function in (5.33) must be $2\pi_c - \inf(U_{N_\varpi} + \Delta) < \Gamma \circ \mathcal{P}\left(z^{-1} X^*_{M_{mi}}\right) < 2\pi_c$.

Chapter 6

Implementation and evaluation

Eureka! — Archimedes (c. 287 - 212 BC)

The implementation of the RealNeuron, in this chapter, is based on parameters found in [28]. The parameters in [28] have been verified with the results in [73, 72].

With respect to the scientific and engineering way of life (§1.7.1.1 p. 8 and §1.7.1.2 p. 9), this chapter unfolds the pure theory of chapter 4 into a *descriptive theory* in accordance with definition 2 on p. 9 by giving the parameters in the models actual values that are based on neuroscience data.

6.1 Spreadsheet implementation

The following instances of the

- RealNeuron network,
- RealNeuron,
- some of the sub-components of the RealNeuron, and
- membranes

are implemented using Calc from OpenOffice.org[1]. The implemented instances can be found on the CD-ROM that is included.

6.2 C++ implementation

The RealNeuron can be implemented using C++ or any other computer language. Based on the RealNeuron model developed in chapter 4, p. 47, and especially using the predicate model in §4.2, p. 49, a RealNeuron UML class diagram can be derived and is shown in figure 6.1. In appendix D, p. 177, an example of a pump implementation is presented.

[1] OpenOffice.org (OOo) version 2.0 Build 2.0.0.1 Novell Edition was used

In the RealNeuron UML class diagram the explicit multi-resolutional levels, used in formulating the RealNeuron, are lost. This results in a flat model with an implicit hierarchy that leads to extra complexity in the different relationships between components. In this thesis UML is not used because of this reason.

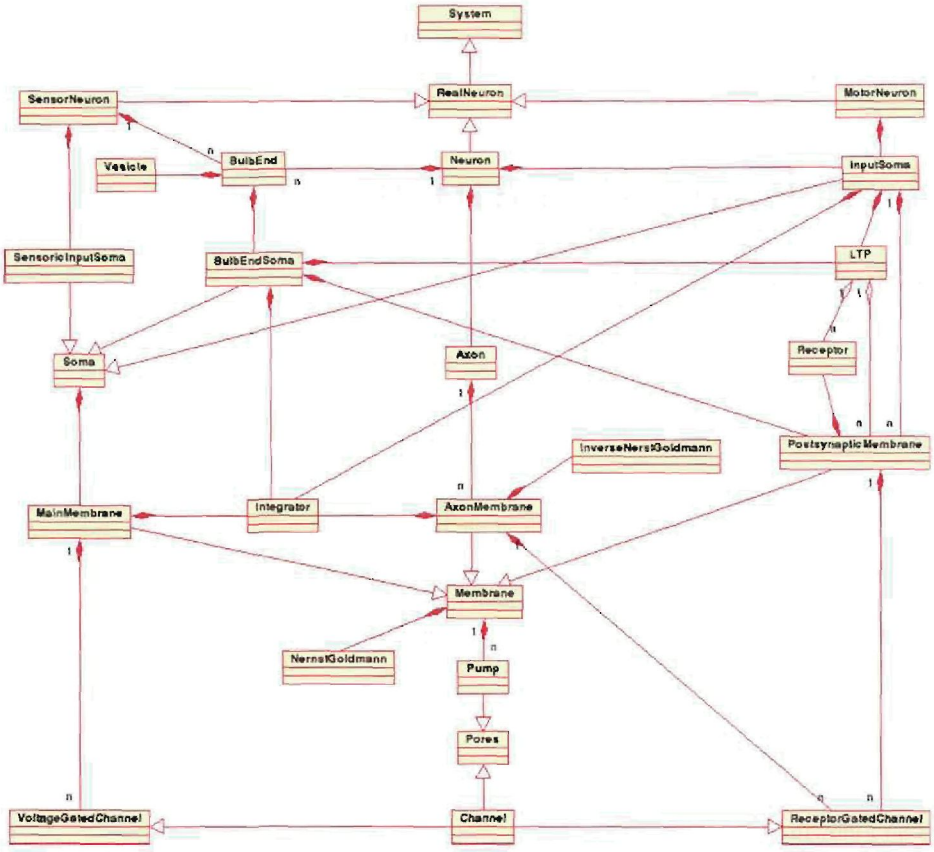


Figure 6.1: RealNeuron UML class diagram

6.3 Verification and validation

Many different interpretations exist for the meaning of *verification and validation* (V&V). Because of the systems engineering approach followed in this thesis, the definition of when a model is verified and when it is validated is based on the level of system resolution at which the model is tested.

The interpretation of the term used in this thesis is the international standard interpretation of the IEEE Software Engineering Standards [78, 79, 81, 82]. This also includes the ISO interpretation as described in ISO/IEC 12207:1995 and adopted by the IEEE in [79].

Definition 42 (Verification) *Verification is the process of making sure that a model has been implemented correctly. If the verification process has been finished, for a model with no anomalies, then the model has a verified status.*

This means that if the mathematical model $y = mx + c$ is specified to be implemented, then the verification process proves that the implementation is $y = mx + c$.

Definition 43 (Validation) *Validation is the process of making sure that a model, when integrated together with other models, gives the expected behaviour for the application environment it is used in. If the validation process has been finished, for a model with no anomalies, then the model has a validated status.*

This means that if a process has a parabolic behaviour, i.e. when it is described by $y = ax^2 + mx + c$, then for $0 \leq x \ll 1$ the term $ax^2 \approx 0$ thus $y \approx mx + c$. The first order model $y = mx + c$ with $0 \leq x \ll 1$ is a validated model for modelling this process. But, $\forall x \in \mathbb{R}$ the first order model $y = mx + c$ is not a validated model for the application.

There are many ways of performing verification and validation as described in [82]. In this thesis the following methods for verification and validation are applied:

- Code-walk-through, i.e. check by hand that each spreadsheet's implementation is correct (verification).
- Analysis, i.e. perform an alternative analysis to arrive at expected results of the models and integrated models (verification and validation).
- Comparison to literature, i.e. compare the model results to published results in the literature (verification and validation).

Independent verification and validation were also performed by asking another person to hold a code-walk-through on the models.

6.4 Reference parameters

In this chapter the results from [28] are used for the membrane. The values of [28] were compared to [110] and [125]. A discrepancy between [28] and [110] was found. Both reference the work of Katz and the Noble Prize Winners Hodgkin and Huxley [73, 72]. The values quoted in [28] are the correct values, and a print error might have slipped in with the values quoted in [110]. These values are different from the values used in the 2000-prototype. The values used in this chapter are shown in table 6.1.

By substituting the values shown in table 6.1 with the parameters in equation 3.7, p. 41, the membrane-resting potential is calculated to be -57.135mV.

Neurotransmitters can be any arbitrary character strings. In this thesis the following strings are used: *ACH*, *Empty*, *GABA*, *Glutamat*, *Haloperidol*, etc. The implementation in this chapter does not consist of any processing of neurotransmitters based on biological descriptions.

6.5 Reference input signal

Input signals varying between -80mV and 20mV are needed. Various arbitrary varying signals could be defined.

Table 6.1: Membrane reference parameters from [28]

Parameter	Parameter Description	Value	Units
T	Temperature	6.3	$°C$
P_K	Membrane permeability to potassium ions	1.8	Normalised
P_{Na}	Membrane permeability to sodium ions	0.07	Normalised
P_{Cl}	Membrane permeability to chlorine ions	0.8	Normalised
$[K]_e$	External concentration of potassium ions	10	μmol/cm^3
$[Na]_e$	External concentration of sodium ions	455	μmol/cm^3
$[Cl]_e$	External concentration of chlorine ions	540	μmol/cm^3
$[K]_i$	Internal concentration of potassium ions	345	μmol/cm^3
$[Na]_i$	Internal concentration of sodium ions	72	μmol/cm^3
$[Cl]_i$	Internal concentration of chlorine ions	61	μmol/cm^3

The following sinusoidal signal is chosen:

$$y = 50\sin(2\pi\frac{k\Delta T}{25}) - 30 \tag{6.1}$$

where k is the time step and ΔT is the time-step interval. Equation (6.1) generates a signal in the range of -80mV to 20mV and two full cycles in a 50 time step simulation. To demonstrate the principle of working of the RealNeuron model, a time-step interval of $\Delta T = 1$s is chosen. The extra iterative calculation steps, if the typical neurobiological time constants ($\geq 140\mu$s [72]) are used, does not provide any additional insight into this principle of working, thus (6.1) can then be rewritten as:

$$y_1 = 50\sin(2\pi\frac{k}{25}) - 30 \tag{6.2}$$

Another input signal is defined by shifting (6.2) with $\frac{\pi}{2}$ radians:

$$y_2 = 50\cos(2\pi\frac{k}{25}) - 30 \tag{6.3}$$

6.6 Mass balance

A membrane has several pores through which ions flow in and out. The ionic flow rate value (γ) through a pore is chosen so that the amount of ions flowing out of the membrane is equal to the amount of ions flowing into it. This will ensure that even after a large number of iterations, the simulation will not diverge from the correct value.

6.7 Membranes

6.7.1 Main membrane

The functioning of the main membrane is described in §4.4.6.3 on p. 85, and the block diagram is shown in figure 4.12 on p. 86.

The membrane parameters are given in table 6.1 and the parameters of the voltage-gated channels are $\pi_{cK} = -40mV$, ϕ_K = minimum, $\pi_{cNa} = -50mV$, ϕ_{Na} = maximum, $\pi_{cCl} = -35mV$, and ϕ_{Cl} = maximum. The input signal is (6.2).

6.7.1.1 Expected results of the main membrane

The main membrane fires as a function of its potential and is dependent on the substance σ, and thresholds $\pi_{c\sigma}$. It is expected that the output potential of the main membrane will change if the membrane potential crosses the -35mV, -40mV and -50mV potentials.

The K voltage-gated channel will fire if the membrane potential is above -40mV. The membrane's permeability for K ions is the highest, see table 6.1. If the K channel opens, then K ions flow from the inside to the outside of the membrane, i.e. the membrane's absolute potential should decrease significantly.

The Na voltage-gated channel will fire if the membrane potential is below -50mV. The membrane's permeability for Na ions is very low, see table 6.1. If the Na channel opens, then Na ions flow from the outside to the inside of the membrane, i.e. the membrane's absolute potential should increase slightly.

The Cl voltage-gated channel will fire if the membrane potential is below -35mV. The membrane's permeability for Cl ions is given in table 6.1. If the Cl channel opens, Cl ions flow from the outside to the inside of the membrane, i.e. the membrane's absolute potential should decrease.

6.7.1.2 Results of the main membrane

The *Main Membrane (Sinusoidal Input) Simulation Results* report is on the enclosed CD-ROM. The simulation results are shown in figure 6.2. The contribution made by the Na channels is very small and is not visible on the graph because the output potential changes from -56.86mV to -56.95mV . The results confirm the expected results.

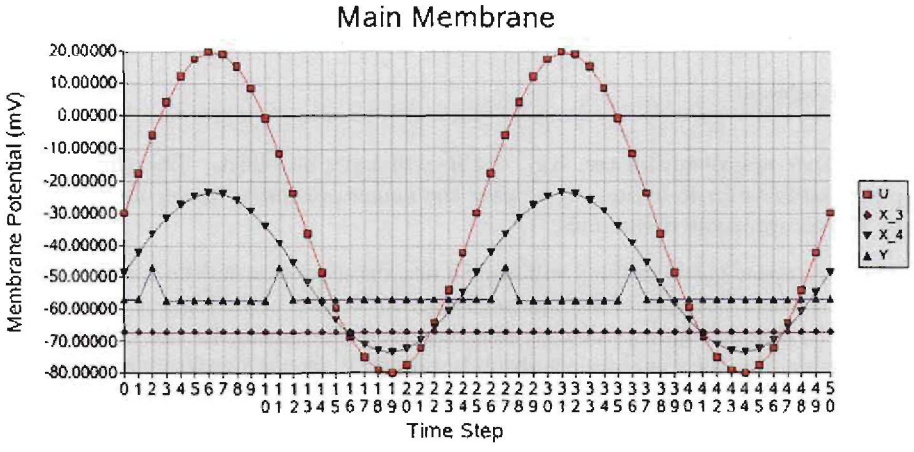


Figure 6.2: Results of the main membrane implementation. U, X_3, X_4 and Y are the states in figure 4.12 on p. 86.

This establishes the *verified* status of the functioning of the main membrane and the *validated* status for voltage-gated channels and pumps in the main membrane.

6.7.2 Post-synaptic membrane

The functioning of the post-synaptic membrane is described in 4.4.6.1, p. 82, and the block diagram in figure 6.3 is for a post-synaptic membrane without a receptor.

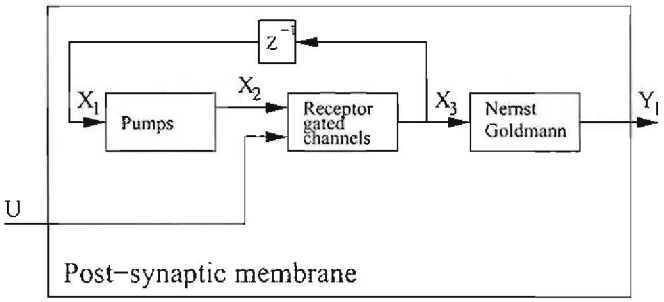


Figure 6.3: Post-synaptic membrane block diagram

The membrane parameters are shown in table 6.1 and the parameters of the receptor-gated channels are τ_K =*GABA*, τ_{Na} =*Haloperidol* and τ_{Cl} =*Glutamat*. The block times for all three receptor-gated channels are $t_b = 4$ time steps.

6.7.2.1 Expected results of the post-synaptic membrane

The post-synaptic membrane fires when a neurotransmitter in the synaptic gap, see 3.3.2 p. 38, matches the signature of a receptor-gated channel. It is expected that the output potential of the post-synaptic membrane will change if the neurotransmitter in the synaptic gap is one of the following molecules:

- *GABA*.
- *Haloperidol*.
- *Glutamat*.

Any other neurotransmitter, e.g. *ACH*, will have no effect. The change in the potential lasts for four time steps after the neurotransmitter molecule is consumed by the receptor-gated channels.

The K receptor-gated channel will fire if a neurotransmitter molecule in the synaptic gap is *GABA*. The membrane's permeability for K ions is the highest, see table 6.1. If the K channel opens, then K ions flow from the inside to the outside of the membrane, i.e. the membrane's absolute potential should significantly decrease.

The Na receptor-gated channels will fire if a neurotransmitter molecule in the synaptic gap is *Haloperidol*. The membrane's permeability for Na ions is very low, see table 6.1. If the Na channel opens, then Na ions flow from the outside to the inside of the membrane, i.e. the membrane's absolute potential should decrease slightly.

The Cl receptor-gated channels will fire if a neurotransmitter molecule in the synaptic gap is *Glutamat*. The membrane's permeability for Cl ions is given in table 6.1. If the Cl channel opens, then Cl ions flow from the outside to the inside of the membrane, i.e. the membrane's absolute potential should decrease.

6.7.2.2 Results of the post-synaptic membrane

The simulation result in figure 6.4 confirms the expected results.

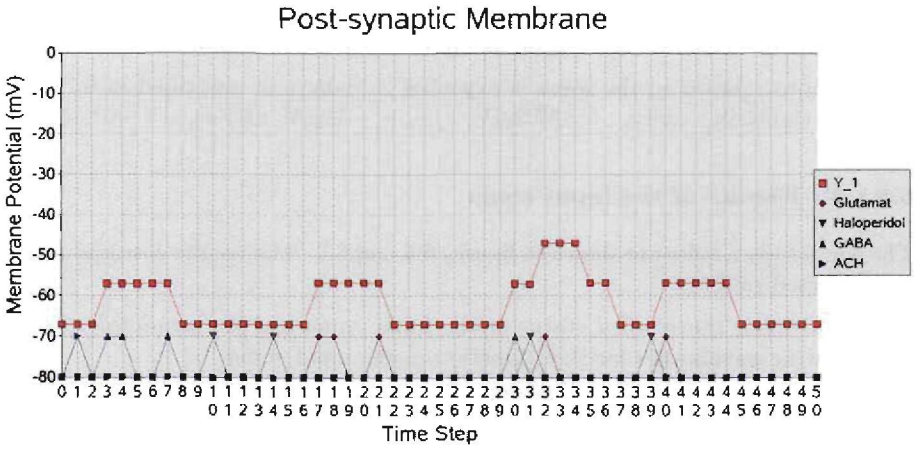


Figure 6.4: Results of the post-synaptic membrane. Y_1 is the output state and *Glutamat, Haloperidol, GABA, ACH* are values for the input state U in figure 6.3. If U is equal to a molecule string then the value for that string is plotted as -70mV, otherwise it is plotted as -80mV.

This establishes the *verified* status of the functioning for the post-synaptic membrane and the *validated* status for receptor-gated channels and pumps in the post-synaptic membrane.

6.8 RealNeuron sub-components

6.8.1 Input soma

The functioning of the input soma is described in 4.4.3.1, p. 71. Figure 6.5 shows a block diagram for an input soma implementation, without an LTP function (μ).

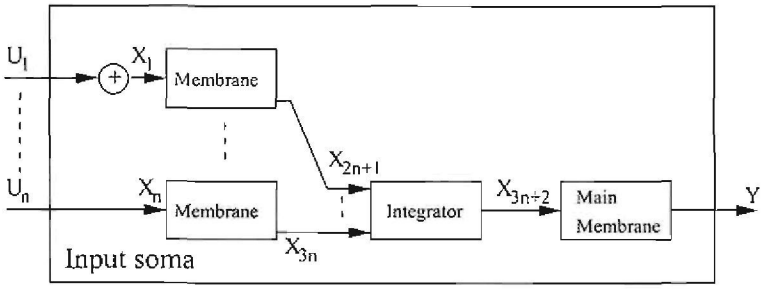


Figure 6.5: Implemented input soma block diagram

Two post-synaptic membranes are implemented as described in 6.7.2 and one main membrane as described in 6.7.1.

6.8.1.1 Expected results of the input soma

If the membrane parameters are as per table 6.1, then the post-synaptic membranes in the soma are expected to function as described in 6.7.2.

The outputs of the post-synaptic membranes are integrated by taking the average between them. This average is used as input to the main membrane.

The main membrane in the soma is expected to behave as described in 6.7.1, but with the thresholds as $\pi_{cK} = -57.2\text{mV}$, $\pi_{cNa} = -53\text{mV}$, and $\pi_{cCl} = -62.15\text{mV}$.

6.8.1.2 Results of the input soma

The simulation results are shown in figures 6.6 and 6.7. The results correspond with the expected results.

This establishes the *verified* status of the input soma and the *validated* status of the following membranes for their function in the input soma:

- Post-synaptic membranes as configured in this section.
- Main membrane.

6.8.2 Bulb end

The functioning of the bulb end is described in 4.4.5, p. 79. Figure 4.9, p. 79, shows the block diagram for a bulb end with two vesicles. The bulb-end soma functions without post-synaptic membranes and an LTP function.

6.8.2.1 Expected results of the bulb end

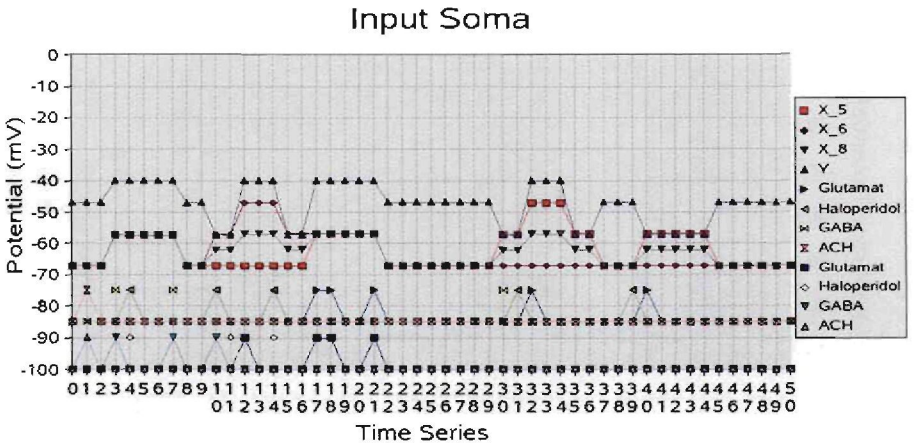
If (6.2) is used as input signal to the bulb end and the membrane parameters are as per table 6.1, it is expected that the bulb end functions like a main membrane, but with the output from the two vesicles that fire as follows:

- If the bulb end potential $> -30\text{mV}$ then a *Glutamat* neurotransmitter is released into the synaptic gap.
- If $-40\text{mV} <$ bulb end potential $\leq -30\text{mV}$ then a *GABA* neurotransmitter is released into the synaptic gap.

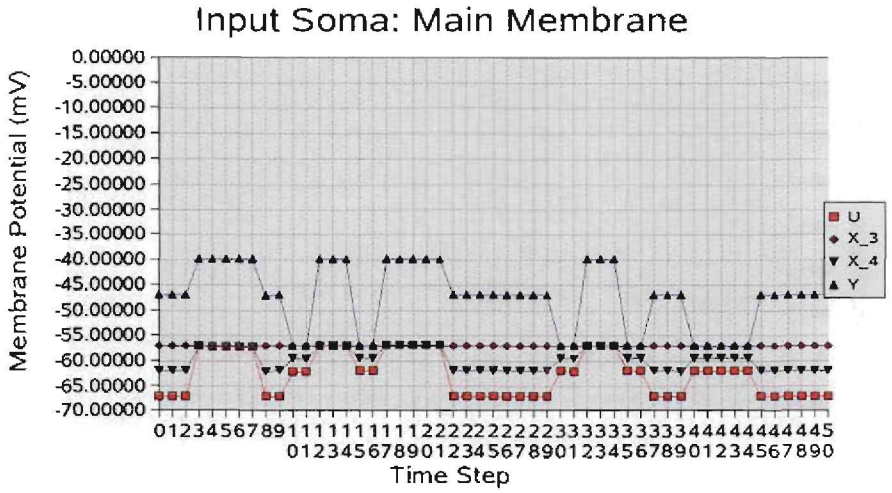
6.8.2.2 Results of the bulb end

The results are shown in figures 6.8(a) and 6.8. The results correspond with the expected results.

This establishes the *verified* status of the bulb end and the *validated* status of the main membrane for its function in the input soma.

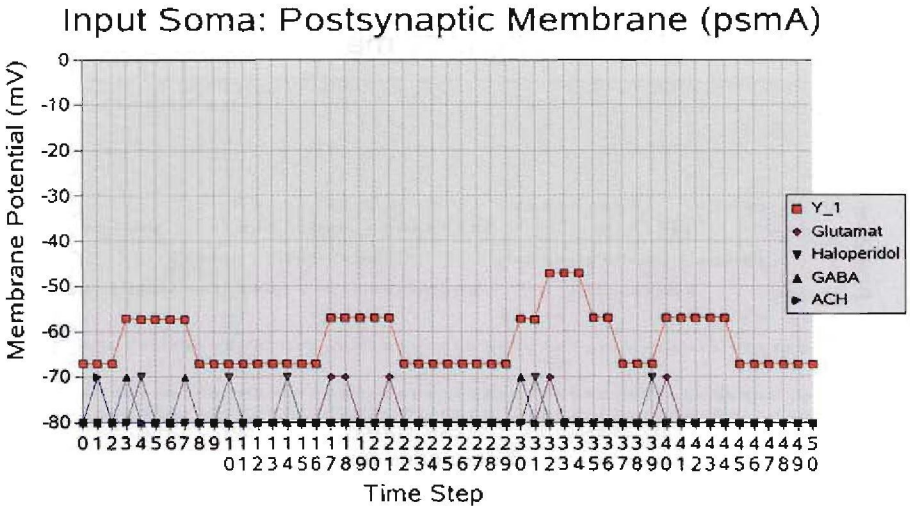


(a) Results of the input soma. Y_1 is the output state, X_5, X_6, X_8 are internal states and *Glutamat, Haloperidol, GABA, ACH* are values for the input states U_1 and U_2 in figure 6.5. If U_1 is equal to a molecule string then the value for that string is plotted as -75mV, otherwise it is plotted as -85mV. If U_2 is equal to a molecule string then the value for that string is plotted as -90mV, otherwise it is plotted as -100mV.

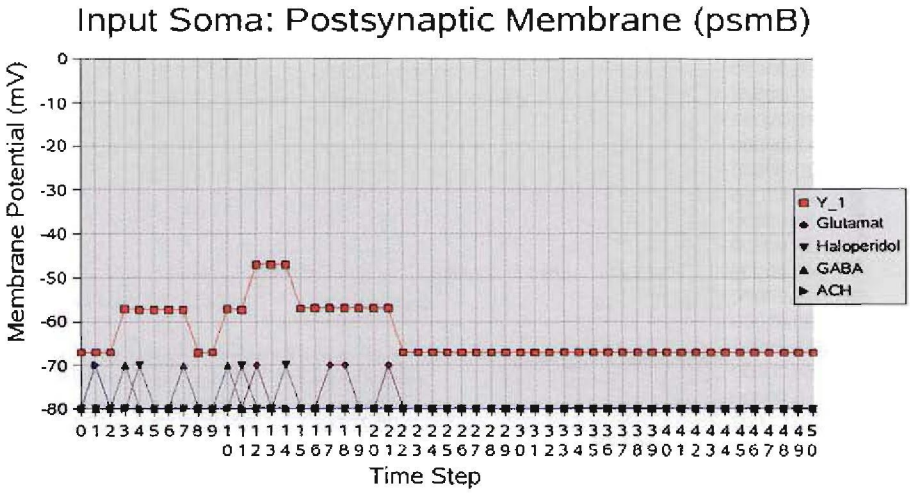


(b) Results of input soma's main membrane. U, X_3, X_4 and Y are the states in figure 4.12 on p. 86.

Figure 6.6: Results of the input soma implementation.

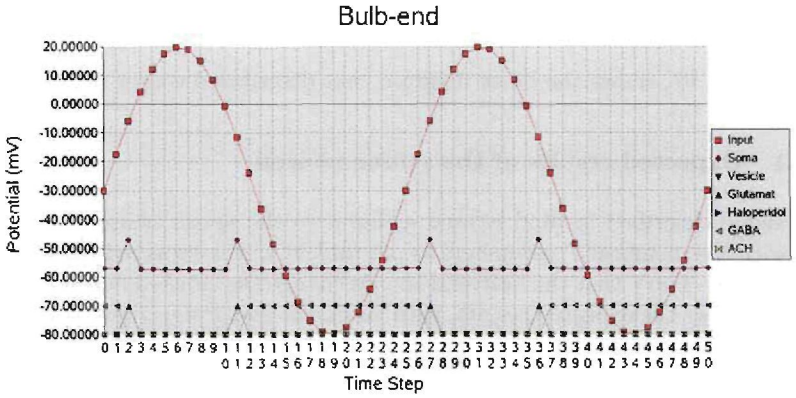


(a) Results of input soma's post-synaptic membrane A. Y_1 is the output state and *Glutamat, Haloperidol, GABA, ACH* are values for the input state U in figure 6.3. If U is equal to a molecule string then the value for that string is plotted as -70mV, otherwise it is plotted as -80mV.

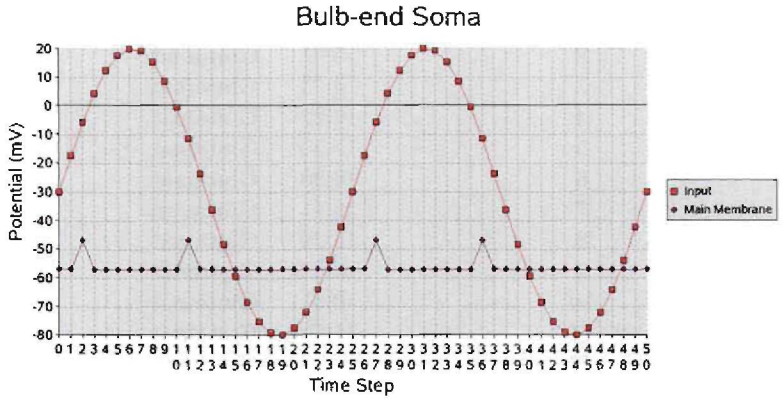


(b) Results of input soma's post-synaptic membrane B. Y_1 is the output state and *Glutamat, Haloperidol, GABA, ACH* are values for the input state U in figure 6.3. If U is equal to a molecule string then the value for that string is plotted as -70mV, otherwise it is plotted as -80mV.

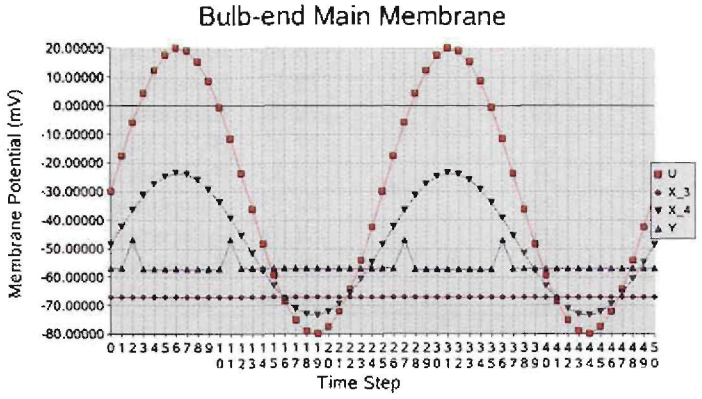
Figure 6.7: Results of the input soma implementation (continued)



(a) Results of the bulb end. *Glutamat*, *Haloperidol*, *GABA*, *ACH* are values for the input state U in figure 4.9 on p. 79. If U is equal to a molecule string then the value for that string is plotted as -70mV, otherwise it is plotted as -80mV.



(b) Result of the bulb-end soma



(c) Result of the bulb-end soma's main membrane. U, X_3, X_4 and Y are the states in figure 4.12 on p. 86.

Figure 6.8: Results of the bulb end

6.9 RealNeuron examples

6.9.1 Sensor neuron

Figure 4.3, p. 67, shows the block diagram that describes a sensor neuron.

6.9.1.1 Expected results of the sensor neuron

If the signal described in (6.2) is used as input to the sensor neuron in figure 4.3, p. 67, with the membrane parameters as per table 6.1, it is expected when the input signal crosses -35mV that the bulb end should change the emitted neurotransmitter. Table 6.2 shows the expected results, with *Glutamat* and *Haloperidol* as the emitted neurotransmitters.

Table 6.2: Expected results for the sensor neuron

Time Step	Input U (mV)	Bulb end
0	-30.00	Glutamat
1	-17.57	Glutamat
2	-5.91	Glutamat
3	4.23	Glutamat
4	12.22	Glutamat
5	17.55	Glutamat
6	19.90	Glutamat
7	19.11	Glutamat
8	15.24	Glutamat
9	8.53	Glutamat
10	-0.61	Glutamat
11	-11.59	Glutamat
12	-23.73	Glutamat
13	-36.27	Haloperidol
14	-48.41	Haloperidol
15	-59.39	Haloperidol
16	-68.53	Haloperidol
17	-75.24	Haloperidol
18	-79.11	Haloperidol
19	-79.90	Haloperidol
20	-77.55	Haloperidol
21	-72.22	Haloperidol
22	-64.23	Haloperidol
23	-54.09	Haloperidol
24	-42.43	Haloperidol
25	-30.00	Glutamat

6.9.1.2 Results of the sensor neuron

The results in figure 6.9 correspond with the expected results in table 6.2.

This establishes the *verified* status of the sensor neuron and the *validated* status for the configured sub-components, membranes, channels and pumps for functioning in the sensor neuron.

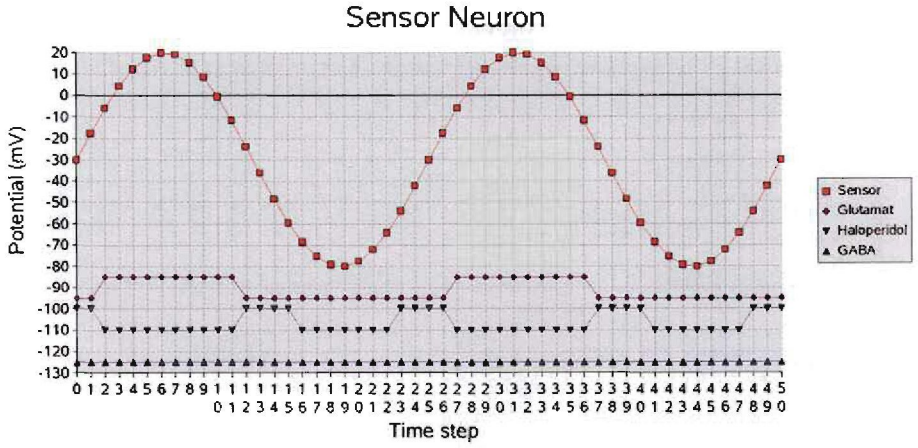


Figure 6.9: Results of the sensor neuron. If *Glutamat* is the output molecule then the result is plotted at -85mV, otherwise it is plotted at -95mV. If *Haloperidol* is the output molecule then the result is plotted at -100mV, otherwise it is plotted at -110mV. If *GABA* is the output molecule then the result is plotted at -115mV, otherwise it is plotted at -125mV.

6.9.2 Motor neuron

See figure 4.4, p. 69, for the schematic block diagram describing a motor neuron.

6.9.2.1 Expected results of the motor neuron

If *GABA* or *Glutamat* is in the input synaptic gap, see 3.3.2 p. 38, of the motor neuron, the output potential of the sensor neuron will increase, i.e. the absolute potential will decrease. Any other neurotransmitter molecule, or if nothing is present in the synaptic gap, will leave the motor neuron in its ground potential.

6.9.2.2 Results of the motor neuron

The results in figure 6.10 correspond with the expected results.

This establishes the *verified* status of the motor neuron and the *validated* status of the configured sub-components, membranes, channels and pumps for functioning in the motor neuron.

6.9.3 Purkinje cell

The function of the Purkinje cell is deduced from the description in [110] as an AND gate, but the second and following input signals have to appear within a specific time frame after the first signal has appeared, to be AND-ed together. The Purkinje cell is N5 in figure 6.36, figure 6.38 and figure 6.37.

The Purkinje cell also implements a long-term delay function. The long term delay is made possible with an LTP function that forms part of the Purkinje cell.

The LTP function is based on the receptors' output from the post-synaptic membranes. It consists of two sub-functions. The one sub-function implements a time

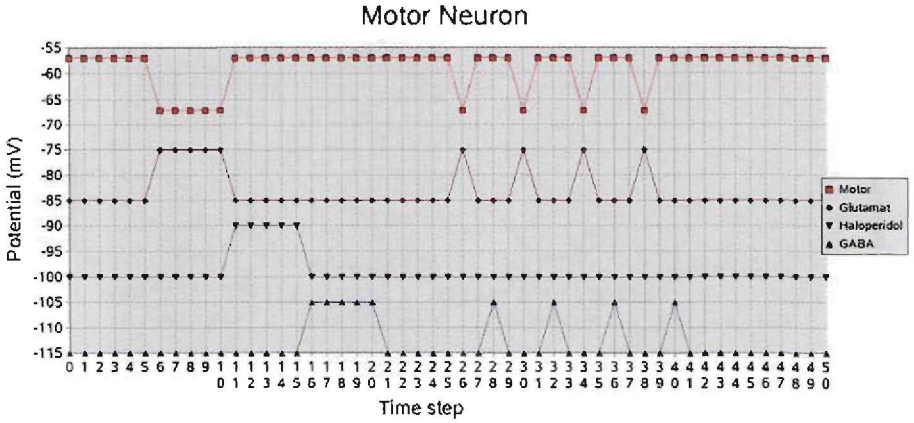


Figure 6.10: Results of the motor neuron. If *Glutamat* is the input molecule then the result is plotted at -75mV, otherwise it is plotted at -85mV. If *Haloperidol* is the input molecule then the result is plotted at -90mV, otherwise it is plotted at -100mV. If *GABA* is the input molecule then the result is plotted at -105mV, otherwise it is plotted at -115mV.

window of $t_b = 2$ time steps. This means that if the two signals are given within this time window, the second sub-function will be activated. The second sub-function implements a conditioned state that lasts for $t_b = 5$ time steps.

6.9.3.1 Expected results of the Purkinje cell

Based on the implementation model of the RealNeuron described in chapter 4 and the Purkinje cell function description in [110], it can be deduced that the expected results for the Purkinje cell are that of an AND gate function with an output block time equal to the block time of the LTP function shown in table 6.3. The LTP function is calculated by taking the AND of the receptor emitted events for unconditioned and conditioning stimulus (see §6.11 for more background on these two signals.)

The function described in the previous paragraph, together with table 6.3, except for time step 18, can be implemented with a discrete logic circuit in figure 6.11. Based on the descriptions in Chapter 4, p. 4:

- A metabolic receptor can be implemented with a mono-stable vibrator.
- The LTP function can be implemented with an AND gate followed by a mono-stable vibrator.

The logic circuit in figure 6.11 can be simulated by using a discrete logic simulation package, e.g. KSimus[2]. This circuit uses an OR gate, two AND gates, and three mono-stable vibrators.

The time frame between the input signals to be AND-ed together is determined by the Purkinje cell's metabolic-receptor functions and is determined for figure 6.11 by

$$t_f = RT \times CT$$

[2]http://ksimus.berlios.de

Table 6.3: Expected results for the Purkinje cell with an LTP block time $t_b = 5$ time steps

Time step	US	CS	Bulb end
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	*1*	0	*0*
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	*1*	*0*
12	0	0	0
13	0	0	0
14	0	0	0
15	0	0	0
16	0	0	0
17	*1*	*1*	*1*
18	0	0	*1*
19	0	0	*1*
20	0	0	*1*
21	0	0	*1*
22	0	0	*1*
23	0	0	0
24	0	0	0
25	0	0	0

The block time for the Purkinje cell's LTP function in figure 6.11 is determined by

$$t_b = RTB \times CTB$$

There is a practical limit on the values of CT and CTB, which are from $10nF$ to $1\mu F$. The maximum practical value for $RT = RTB = 100M\Omega$. The maximum block time and time frame length that can be achieved is then $t_b = 1\mu F \times 100M\Omega = 100s$. This is far from the several hours [110] achieved in nature. Of course, a digital counter-circuit, that is similar to consumer digital watches, can be used to achieve the biological time constant, but this is a complex solution.

The Purkinje cell cannot be implemented by perceptrons alone in a feed-forward network, nor can any other classical artificial neuron alone be used in feed-forward networks. A complex perceptron network, with time delay elements and feedback (recurrent networks), is necessary to implement a Purkinje cell. Figure 6.12 shows a perceptron circuit with 12 neurons and several delay elements that implements the Purkinje functional description.

The use of perceptrons in implementing the Purkinje cell introduces an unnecessarily complex and unhierarchical structure that can result in the loss of the overview on the functionality of the Purkinje cell. The complexity of the RealNeuron, handled in a rigorous and hierarchical structure, is more understandable in which the oversight of the Purkinje cell's working is not lost.

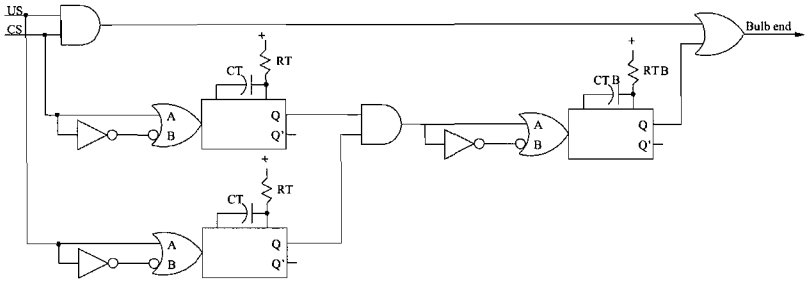


Figure 6.11: Logic circuit for Purkinje cell

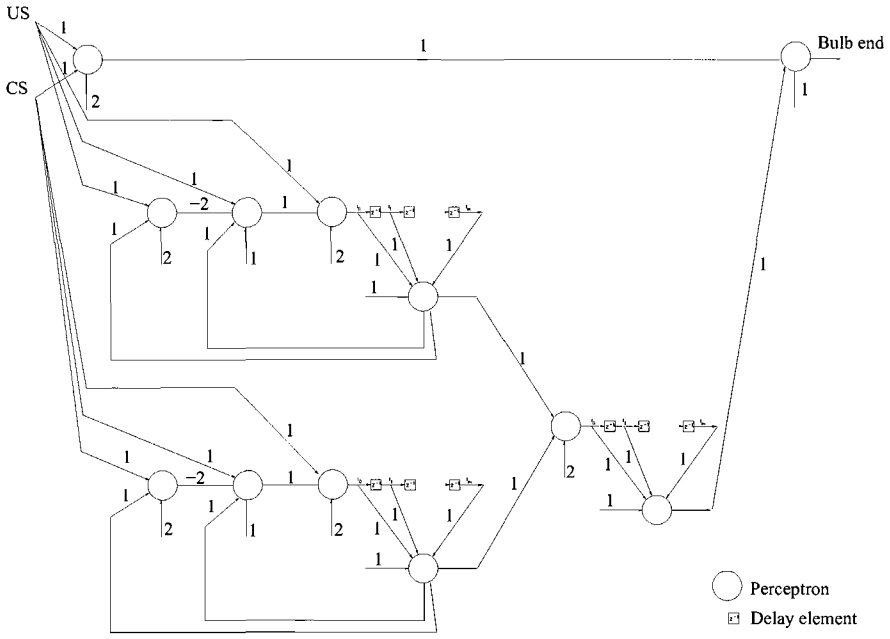


Figure 6.12: Perceptron neural network for Purkinje cell

6.9.3.2 Results of the Purkinje cell

The simulation results of the Purkinje cell is described in table 6.4. The results show that the Purkinje cell is conditioned for five time steps if the two inputs are within two time steps of each other. However, if the two input signals are not given simultaneously but within the two time steps time window, the conditioning last still only five time steps after the first input signal is given. No conditioning takes place if the signals are outside the two-step time window. By substituting the *Glutamat* string in table 6.4 with "*1*" and the *Empty*, *GABA* and *Haloperidol* strings with "*0*", the expected results in table 6.3 are confirmed. The results are available in the *Purkinje RealNeuron Simulation Results* on the enclosed CD-ROM.

This establishes the *verified* status of the Purkinje cell implementation and the *validated* status of the sub-components configured for the Purkinje cell.

Table 6.4: Results of the Purkinje cell

Time step	US	CS	Bulb end
0	Haloperidol	Haloperidol	Empty
1	GABA	GABA	Empty
2	Haloperidol	Haloperidol	Empty
3	Haloperidol	Haloperidol	Empty
4	Haloperidol	Haloperidol	Empty
5	Haloperidol	Haloperidol	Empty
6	*Glutamat*	Haloperidol	Empty
7	Haloperidol	*Glutamat*	Empty
8	Haloperidol	Haloperidol	*Glutamat*
9	Haloperidol	Haloperidol	*Glutamat*
10	Haloperidol	Haloperidol	*Glutamat*
11	Haloperidol	Haloperidol	*Glutamat*
12	Haloperidol	Haloperidol	Empty
13	Haloperidol	Haloperidol	Empty
14	Haloperidol	Haloperidol	Empty
15	Haloperidol	Haloperidol	Empty
16	*Glutamat*	*Glutamat*	*Glutamat*
17	Haloperidol	Haloperidol	*Glutamat*
18	Haloperidol	Haloperidol	*Glutamat*
19	*Glutamat*	Haloperidol	*Glutamat*
20	Haloperidol	Haloperidol	*Glutamat*
21	*Glutamat*	Haloperidol	*Glutamat*
22	*Glutamat*	Haloperidol	Empty
23	*Glutamat*	Haloperidol	Empty
24	Haloperidol	Haloperidol	Empty
25	Haloperidol	Haloperidol	Empty
26	*Glutamat*	Haloperidol	Empty
27	Haloperidol	Haloperidol	Empty
28	Haloperidol	Haloperidol	Empty
29	Haloperidol	Haloperidol	Empty
30	Haloperidol	*Glutamat*	Empty
31	*Glutamat*	Haloperidol	Empty
32	Haloperidol	Haloperidol	*Glutamat*
33	Haloperidol	Haloperidol	*Glutamat*
34	Haloperidol	Haloperidol	*Glutamat*
35	Haloperidol	Haloperidol	*Glutamat*
36	Haloperidol	Haloperidol	Empty

6.9.4 Nucleus interpositus cell

From the description given in [110] of the nucleus interpositus (NI) cell, it can be deduced that the conditioning stimulus (CS) signal is AND-ed with the OR-ed signal of the unconditioned stimulus (US) and the output of the Purkinje cell (PURK). The NI cell is N6 in figures 6.36, 6.38 and 6.37.

6.9.4.1 Expected results of the nucleus interpositus cell

The expected results for the NI cell is shown in table 6.5.

Table 6.5: Expected results of nucleus interpositus cell

US	CS	PURK	Bulb end
0	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0
1	1	0	1
0	1	1	1
1	0	1	0
1	1	1	1

The function described in table 6.5 can also be implemented with discrete logic integrated circuits (ICs) as shown in figure 6.13. By using a discrete logic simulation package, e.g. KSimus[3], the implemented circuit can be simulated. This circuit uses AND and OR gates.



Figure 6.13: Logic circuit for nucleus interpositus cell

The nucleus interpositus cell can be implemented by a perceptron as shown in figure 6.14.

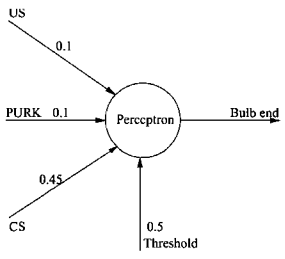


Figure 6.14: A perceptron implementation of the nucleus interpositus cell

[3]http://ksimus.berlios.de

6.9.4.2 Results of the nucleus interpositus cell

The simulation results of the nucleus interpositus cell is described in table 6.6. By substituting the *Glutamat* string in table 6.6 with "*1*" and the *GABA* and *Haloperidol* strings with "*0*", the expected results in table 6.5 are confirmed. The results are available in the *Nucleus Interpositus RealNeuron Simulation Results* on the enclosed CD-ROM.

This establishes the *verified* status of the NI cell implementation and the *validated* status of the sub-components configured for the NI cell.

Table 6.6: Results of the nucleus interpositus cell

Time step	US	CS	PURK	Bulb end
0	*GABA*	Haloperidol	Haloperidol	Haloperidol
1	Haloperidol	*GABA*	Haloperidol	*GABA*
2	Haloperidol	Haloperidol	*GABA*	Haloperidol
3	Haloperidol	Haloperidol	Haloperidol	Haloperidol
4	*GABA*	*GABA*	Haloperidol	*GABA*
5	Haloperidol	Haloperidol	Haloperidol	Haloperidol
6	Haloperidol	*GABA*	*GABA*	*GABA*
7	Haloperidol	Haloperidol	Haloperidol	Haloperidol
8	*GABA*	Haloperidol	*GABA*	*GABA*
9	Haloperidol	Haloperidol	Haloperidol	Haloperidol
10	*GABA*	*GABA*	*GABA*	*GABA*
11	Haloperidol	Haloperidol	Haloperidol	Haloperidol
12	*Glutamat*	Haloperidol	Haloperidol	Haloperidol
13	Haloperidol	Haloperidol	Haloperidol	Haloperidol
14	Haloperidol	*Glutamat*	Haloperidol	Haloperidol
15	Haloperidol	Haloperidol	Haloperidol	Haloperidol
16	Haloperidol	Haloperidol	*Glutamat*	Haloperidol
17	Haloperidol	Haloperidol	Haloperidol	Haloperidol
18	*Glutamat*	*Glutamat*	Haloperidol	*Glutamat*
19	Haloperidol	Haloperidol	Haloperidol	Haloperidol
20	Haloperidol	*Glutamat*	*Glutamat*	*Glutamat*
21	Haloperidol	Haloperidol	Haloperidol	Haloperidol
22	*Glutamat*	Haloperidol	*Glutamat*	Haloperidol
23	Haloperidol	Haloperidol	Haloperidol	Haloperidol
24	*Glutamat*	*Glutamat*	*Glutamat*	*Glutamat*
25	Haloperidol	Haloperidol	Haloperidol	Haloperidol

6.10 Logic function implementation

This section implements the logical functions discussed in §2.4.2 on p. 23. The logic functions implement thinking as discussed in §2.4.2 on p. 23. Thinking is one of the intellifacts discussed in §2.2 on p. 17.

6.10.1 BUFFER function

The BUFFER function is not discussed in §2.4.2 on p. 23, but it is part of general digital electronics, see [54]. The BUFFER function is used in §6.11.

The logical BUFFER function of a signal can be defined as follows: If the value of the signal is above a specific threshold, the output is high, otherwise the output is low. Mathematically, this can be formulated as

$$\text{buffer}(x) = \begin{cases} a & \text{if } x \geq c \\ b & \text{otherwise} \end{cases} \tag{6.4}$$

where $|a| > |b|$.

6.10.1.1 SBS of the BUFFER network

The BUFFER RealNeuron network in figure 6.15 can be described as a structure of embedded systems, see definition 6 p. 31, that encodes a tree structure. This tree structure describes a system breakdown structure (SBS), see §3.2 p. 32.

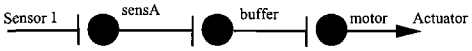


Figure 6.15: BUFFER RealNeuron network

Let NN_{buffer} represent the BUFFER RealNeuron network. From the description in §3.1.1 p. 31, it follows that NN_{buffer} is an element of a body (B) structure or system. B is chosen as the root of the SBS and, thus, the lowest level of resolution. B provides input to NN_{buffer} from the sensor. NN_{buffer} provides a motoric signal to the actuator in B. B has the states $\{S_1, S_2\} \in \mathbb{S}$. Mathematically, the root and its children can be expressed as:

$$\{S_1, S_2,\ NN_{\text{buffer}}\} \subseteq B \tag{6.5}$$

The child nodes of the root B, i.e. S_1, S_2, NN_{buffer}, are the next higher level of the SBS resolution.

From figure 6.15 it follows that NN_{buffer} has three neurons as children:

$$\{N_{\text{sensA}}, N_{\text{buffer}}, N_{\text{motor}}\} \subseteq NN_{\text{buffer}} \tag{6.6}$$

From the description of the RealNeuron, in §4.4.2 p. 64, the general neuron N_i has a soma (S), an axon (A) and bulb ends (Υs). According to definition 22 p. 66, the sensor neuron does not have an axon. According to definition 24 p. 68, the motor neuron has neither an axon nor any bulb ends. In the instance of the NN_{buffer},

N_{buffer} has no axon. This means that the next higher level of the SBS resolution for equation (6.6) can then be:

$$\{S_{\text{sensA}}, \Upsilon_{\text{sensA}}\} \subset N_{\text{sensA}} \tag{6.7}$$
$$\{S_{\text{buffer}}, \Upsilon_{\text{buffer}}\} \subset N_{\text{buffer}} \tag{6.8}$$
$$\{S_{\text{motor}}\} \subset N_{\text{motor}} \tag{6.9}$$

From the descriptions of the soma, in §4.4.3.1 p. 71, and the bulb end, in §4.4.5 p. 79, the next level of higher resolution of the SBS for equation (6.7) can be

$$\{M_{pS_{\text{sensA}}}, M_{m\text{sensA}}\} \subset S_{\text{sensA}} \tag{6.10}$$
$$\{S_{\Upsilon_{\text{sensA}}}, V_{\Upsilon_{\text{sensA}}}\} \subset \Upsilon_{\text{sensA}} \tag{6.11}$$

where $V_{\Upsilon_{\text{sensA}}}$ is a leaf of the SBS.

Similarly, then, to equations (6.10) and (6.11), the next level of higher resolution of the SBS for equation (6.8) can be

$$\{M_{pS_{\text{buffer}}}, M_{m\text{buffer}}\} \subset S_{\text{buffer}} \tag{6.12}$$
$$\{S_{\Upsilon_{\text{buffer}}}, V_{\Upsilon_{\text{buffer}}}\} \subset \Upsilon_{\text{buffer}} \tag{6.13}$$

where $V_{\Upsilon_{\text{buffer}}}$ is a leaf of the SBS.

Similarly, to equation (6.10), the next level of higher resolution of the SBS for equation (6.9) can be

$$\{M_{pS_{\text{motor}}}, M_{m\text{motor}}\} \subset S_{\text{motor}} \tag{6.14}$$

Similarly, to equation (6.10), the next level of higher resolution of the SBS for equation (6.11) can be

$$\{M_{pS_{\Upsilon_{\text{sensA}}}}, M_{mS_{\Upsilon_{\text{sensA}}}}\} \subset S_{\Upsilon_{\text{sensA}}} \tag{6.15}$$

Similarly, to equations (6.15), the next level of higher resolution of the SBS for equation (6.13) can be

$$\{M_{pS_{\Upsilon_{\text{buffer}}}}, M_{mS_{\Upsilon_{\text{buffer}}}}\} \subset S_{\Upsilon_{\text{buffer}}} \tag{6.16}$$

Equations (6.10), (6.12), (6.14) to (6.16) could be expanded to the highest level of resolution of channels, pumps and receptors. Systems engineering is a pragmatic application of principles. These principles are typical of those described in [77, 81]. An SBS is normally established by a systems engineer to make the complexity of a problem easier to manage and this makes the detailed structure of an SBS an arbitrary choice.

The above SBS structure is based on the theoretical description of the RealNeuron network as described in §4.4 p. 59. A more practical SBS, that can be used for the description of the RealNeuron interconnections, is developed.

Connections between RealNeurons in a network operate through synapses, see §3.3.2 p. 38, i.e. between bulb ends and post-synaptic membranes. It makes sense to define an SBS in which the leaves are bulb ends and post-synaptic membranes. By applying the theory that $A \subset B$ and $B \subset C$ then $A \subset C$ [99] to the SBS defined in equations (6.5) to (6.16), the next level of resolution for equation (6.6) can be defined as

$$\{M_{pS_{\text{sensA}}}, M_{m\text{sensA}}, \Upsilon_{\text{sensA}}\} \subset N_{\text{sensA}} \tag{6.17}$$
$$\{M_{pS_{\text{buffer}}}, M_{m\text{buffer}}, \Upsilon_{\text{buffer}}\} \subset N_{\text{buffer}} \tag{6.18}$$
$$\{M_{pS_{\text{motor}}}, M_{m\text{motor}}\} \subset N_{\text{motor}} \tag{6.19}$$

6.10.1.2 Connections of the BUFFER network

Based on figure 6.15 and the SBS described in equations (6.5), (6.6), (6.17) to (6.19), the following interconnections are identified:

- *Body to Brain* connections, i.e. connections between the sensors in a body and a RealNeuron network:

$$S_1 \rightarrow M_{m S_{\text{sensA}}}$$

- *Brain to Body* connections, i.e. connections between a RealNeuron network and the actuators in a body:

$$M_{m\,\text{motor}} \rightarrow S_2$$

- *Brain to Brain* connections, i.e. the connections between RealNeurons inside a RealNeuron network:

$$\begin{aligned} \Upsilon_{\text{sensA}} &\rightarrow M_{p S_{\text{buffer}}} \\ \Upsilon_{\text{buffer}} &\rightarrow M_{p S_{\text{motor}}} \end{aligned}$$

6.10.1.3 Expected results of the BUFFER function

Table 6.7 shows the expected results if (6.2) is used as input signal to the BUFFER RealNeuron network in figure 6.15. In this case $c = -30\text{mV}$ in (6.4) and the membrane parameters per table 6.1. The reader is referred to the *Buffer RealNeuron Simulation Results* report for the other parameters that is available on the enclosed CD-ROM.

The BUFFER function can be implemented by a perceptron as shown in figure 6.16.

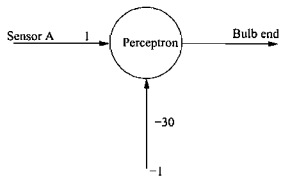


Figure 6.16: A perceptron implementation of the BUFFER function

6.10.1.4 Results of the BUFFER function

The results in figure 6.17 correspond with the expected results in table 6.7. The BUFFER function output is an absolute high value ($a = -67.15\text{mV}$) if the input is equal or above -30mV, and for anything else an absolute low value ($b = -56.95\text{mV}$).

Table 6.7: Expected results of the BUFFER function

Time step	(mV)	Bulb end
0	-30.00	a
1	-17.57	a
2	-5.91	a
3	4.23	a
4	12.22	a
5	17.55	a
6	19.90	a
7	19.11	a
8	15.24	a
9	8.53	a
10	-0.61	a
11	-11.59	a
12	-23.73	a
13	-36.27	b
14	-48.41	b
15	-59.39	b
16	-68.53	b
17	-75.24	b
18	-79.11	b
19	-79.90	b
20	-77.55	b
21	-72.22	b
22	-64.23	b
23	-54.09	b
24	-42.43	b
25	-30.00	a

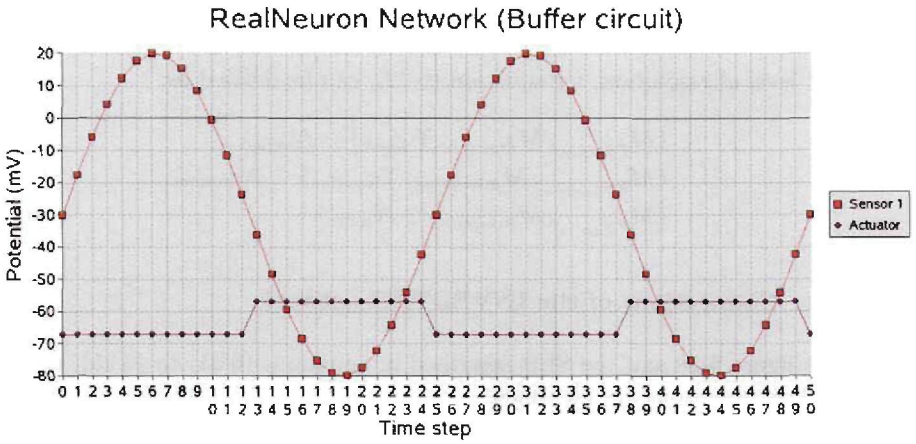


Figure 6.17: Results of the BUFFER RealNeuron network

6.10.2 INVERTER function

The logical INVERTER function of a signal can be defined as follows: if the value of the signal is above a specific threshold, the output is low, otherwise the output is high. This is the inverse of the BUFFER function defined in 6.10.1. Mathematically, this can be formulated as

$$\text{inverter}(x) = \begin{cases} b & \text{if } x \geq c \\ a & \text{otherwise} \end{cases} \tag{6.20}$$

where $|b| < |a|$.

6.10.2.1 SBS of the INVERTER network

The INVERTER RealNeuron network in figure 6.18 can also be described by a system breakdown structure (SBS), by following a similar reasoning to §6.10.1.1.



Figure 6.18: INVERTER RealNeuron network

Let NN_{inverter} represent the INVERTER RealNeuron network. NN_{inverter} is part of the body (B). B is chosen as the root of the SBS and, thus, the lowest level of resolution. B provides input to NN_{inverter} from the sensor. NN_{inverter} provides a motoric signal to the actuator in B. B has the states $\{S_1, S_2\} \in \mathbb{S}$. Mathematically, the root and its children can be expressed as:

$$\{S_1, S_2,\ NN_{\text{inverter}}\} \subseteq B \tag{6.21}$$

The child nodes of the root B, i.e. S_1, S_2, NN_{inverter}, are the next higher level of resolution of the SBS.

From figure 6.18 it follows that NN_{inverter} has three neurons as children:

$$\{N_{\text{sensA}}, N_{\text{inverter}}, N_{\text{motor}}\} \subseteq NN_{\text{inverter}} \tag{6.22}$$

The next level of resolution, for equation (6.22), can be defined as:

$$\{M_{pS_{\text{sensA}}}, M_{m\text{sensA}}, \Upsilon_{\text{sensA}}\} \subset N_{\text{sensA}} \tag{6.23}$$
$$\{M_{pS_{\text{inverter}}}, M_{m\text{inverter}}, \Upsilon_{\text{inverter}}\} \subset N_{\text{inverter}} \tag{6.24}$$
$$\{M_{pS_{\text{motor}}}, M_{m\text{motor}}\} \subset N_{\text{motor}} \tag{6.25}$$

6.10.2.2 Connections of the INVERTER network

Based on figure 6.18 and the SBS described in equations (6.21) to (6.25), the following interconnections are identified:

- *Body to Brain* connections, i.e. connections between the sensors in a body and a RealNeuron network:

$$S_1 \rightarrow M_{mS_{\text{sensA}}}$$

- *Brain to Body* connections, i.e. connections between a RealNeuron network and the actuators in a body:

$$M_{m\,\mathrm{motor}} \rightarrow S_2$$

- *Brain to Brain* connections. i.e. the connections between RealNeurons inside a RealNeuron network:

$$\begin{aligned} \Upsilon_{\mathrm{sensA}} &\rightarrow M_{pS_{\mathrm{inverter}}} \\ \Upsilon_{\mathrm{inverter}} &\rightarrow M_{pS_{\mathrm{motor}}} \end{aligned}$$

6.10.2.3 Expected results of the INVERTER network

Table 6.8 shows the expected results, if (6.2) is used as input signal to the INVERTER RealNeuron network in figure 6.18, with $c = -30\mathrm{mV}$ in equation (6.20) and the membrane parameters per table 6.1. The reader is referred to the *INVERTER RealNeuron Simulation Results* report for the other parameters that is available on the enclosed CD-ROM.

Table 6.8: Expected results of the INVERTER function

Time step	U (mV)	Bulb end
0	-30.00	b
1	-17.57	b
2	-5.91	b
3	4.23	b
4	12.22	b
5	17.55	b
6	19.90	b
7	19.11	b
8	15.24	b
9	8.53	b
10	-0.61	b
11	-11.59	b
12	-23.73	b
13	-36.27	a
14	-48.41	a
15	-59.39	a
16	-68.53	a
17	-75.24	a
18	-79.11	a
19	-79.90	a
20	-77.55	a
21	-72.22	a
22	-64.23	a
23	-54.09	a
24	-42.43	a
25	-30.00	b

The INVERTER function can be implemented by a perceptron as shown in figure 6.19.

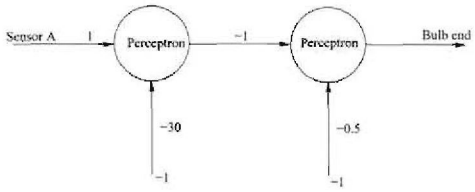


Figure 6.19: A perceptron implementation of the INVERTER function

6.10.2.4 Results of the INVERTER network

The results in figure 6.20 correspond with the expected results in table 6.8. The INVERTER function output is an absolute low value ($b = -56.95\text{mV}$) if the input is equal or above -30mV, and for anything else an absolute high value ($a = -67.15\text{mV}$).

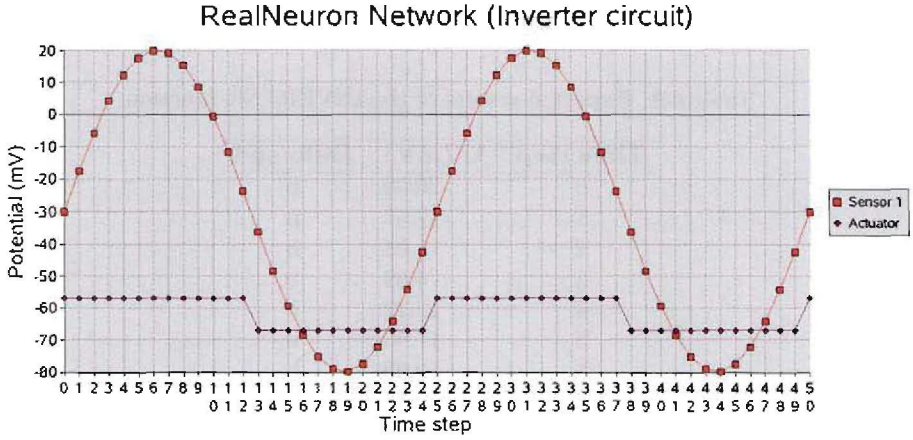


Figure 6.20: Results of the INVERTER RealNeuron network

6.10.3 AND function

The logical AND function between two signals can be defined as follows: if the values of both signals are above a specific threshold, the output is high, otherwise the output is low. Mathematically, this can be formulated as:

$$\text{and}(x, y) = \begin{cases} a & \text{if } x \geq c \text{ and } y \geq c \\ b & \text{otherwise} \end{cases} \tag{6.26}$$

where $a > b$.

6.10.3.1 SBS of the AND network

The AND RealNeuron network in figure 6.21 can also be described by a system breakdown structure (SBS), by following a similar argument to that in §6.10.1.1.

Let NN_{and} represent the AND RealNeuron network. NN_{and} is part of the body (B). B is chosen as the root of the SBS and, thus, the lowest level of resolution.

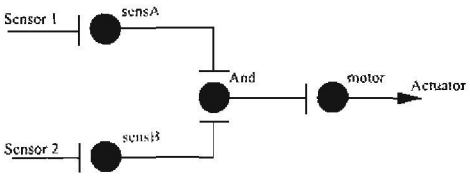


Figure 6.21: AND RealNeuron network

B provides input to NN_{and} from the sensors. NN_{and} provides a motoric signal to the actuator in B. B has the states $\{S_1, S_2\} \in \mathbb{S}$. Mathematically, the root and its children can be expressed as:

$$\{S_1, S_2,\ NN_{\text{and}}\} \subseteq B \tag{6.27}$$

The child nodes of the root B, i.e. S_1, S_2, NN_{and}, are the next higher level of resolution of the SBS.

From figure 6.21 it follows that NN_{and} has four neurons as children:

$$\{N_{\text{sensA}}, N_{\text{sensB}}, N_{\text{and}}, N_{\text{motor}}\} \subseteq NN_{\text{and}} \tag{6.28}$$

The next level of resolution for equation (6.28) can be defined as:

$$\{M_{pS_{\text{sensA}}}, M_{m\text{sensA}}, \Upsilon_{\text{sensA}}\} \subset N_{\text{sensA}} \tag{6.29}$$
$$\{M_{pS_{\text{sensB}}}, M_{m\text{sensA}}, \Upsilon_{\text{sensA}}\} \subset N_{\text{sensB}} \tag{6.30}$$
$$\{M_{pA_{S_{\text{and}}}}, M_{pB_{S_{\text{and}}}}, M_{m\text{and}}, \Upsilon_{\text{and}}\} \subset N_{\text{and}} \tag{6.31}$$
$$\{M_{pS_{\text{motor}}}, M_{m\text{motor}}\} \subset N_{\text{motor}} \tag{6.32}$$

6.10.3.2 Connections of the AND network

Based on figure 6.21 and the SBS described in equations (6.27) to (6.32), the following interconnections are identified:

- *Body to Brain* connections, i.e. connections between the sensors in a body and a RealNeuron network:

$$\begin{aligned} S_1 &\rightarrow M_{mS_{\text{sensA}}} \\ S_2 &\rightarrow M_{mS_{\text{sensB}}} \end{aligned}$$

- *Brain to Body* connections, i.e. connections between a RealNeuron network and the actuators in a body:

$$M_{m\text{motor}} \rightarrow S_2$$

- *Brain to Brain* connections, i.e. the connections between RealNeurons inside a RealNeuron network:

$$\begin{aligned} \Upsilon_{\text{sensA}} &\rightarrow M_{pA_{S_{\text{and}}}} \\ \Upsilon_{\text{sensB}} &\rightarrow M_{pB_{S_{\text{and}}}} \\ \Upsilon_{\text{and}} &\rightarrow M_{pS_{\text{motor}}} \end{aligned}$$

6.10.3.3 Expected results of the AND network

Table 6.9 shows the expected results, if (6.2) and (6.3) are used as input signals to the AND RealNeuron network in figure 6.21, with $c = -30\text{mV}$ in equation (6.26) and the membrane parameters per table 6.1. The reader is referred to the *And RealNeuron Simulation Results* report for the other parameters available on the enclosed CD-ROM.

Table 6.9: Expected results of the AND function

Time step	Sensor 1 (mV)	Sensor 2 (mV)	Bulb end
0	-30.00	20.00	a
1	-17.57	18.43	a
2	-5.91	13.82	a
3	4.23	6.45	a
4	12.22	-3.21	a
5	17.55	-14.55	a
6	19.90	-26.86	a
7	19.11	-39.37	b
8	15.24	-51.29	b
9	8.53	-61.87	b
10	-0.61	-70.45	b
11	-11.59	-76.49	b
12	-23.73	-79.61	b
13	-36.27	-79.61	b
14	-48.41	-76.49	b
15	-59.39	-70.45	b
16	-68.53	-61.87	b
17	-75.24	-51.29	b
18	-79.11	-39.37	b
19	-79.90	-26.86	b
20	-77.55	-14.55	b
21	-72.22	-3.21	b
22	-64.23	6.45	b
23	-54.09	13.82	b
24	-42.43	18.43	b
25	-30.00	20.00	b

The AND function can be implemented by a perceptron as shown in figure 6.22.

6.10.3.4 Results of the AND network

The results in figure 6.23 correspond with the expected results in table 6.9. The AND function output is high ($a = -57.14\text{mV}$) if both inputs are equal or above -30mV, and low ($b = -67.15\text{mV}$) for anything else.

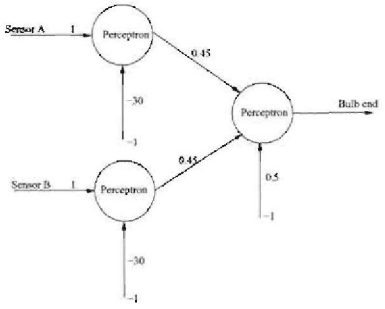


Figure 6.22: A perceptron implementation of the AND function

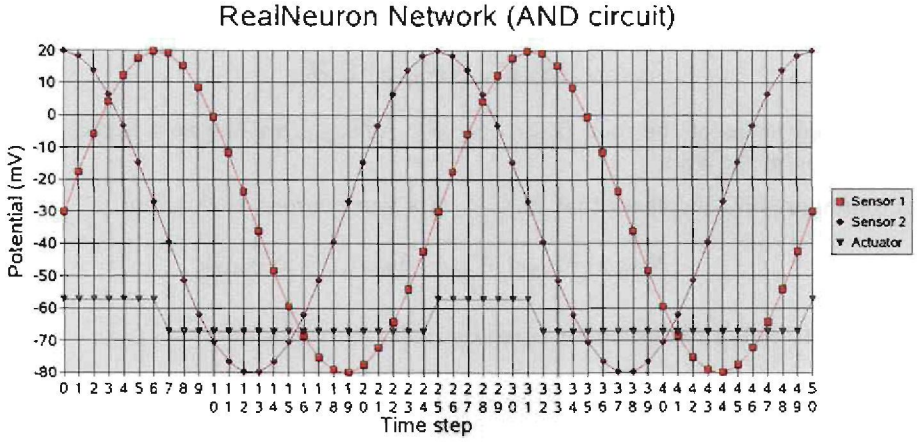


Figure 6.23: Results of the AND RealNeuron network

6.10.4 NAND function

The logical NAND function between two signals can be defined as follows: if the values of both signals are above a specific threshold, the output is high, otherwise the output is low. This is the inverse of the AND function defined in 6.10.3. Mathematically, this can be formulated as:

$$\text{nand}(x, y) = \begin{cases} b & \text{if } x \geq c \text{ nand } y \geq c \\ a & \text{otherwise} \end{cases} \tag{6.33}$$

where $b < a$.

6.10.4.1 SBS of the NAND network

The NAND RealNeuron network in figure 6.24 can also be described by a system breakdown structure (SBS), by following a similar argument to that in §6.10.1.1.

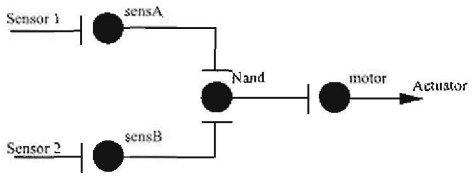


Figure 6.24: NAND RealNeuron network

Let NN_{nand} represent the NAND RealNeuron network. NN_{nand} is part of the body (B). B is chosen as the root of the SBS and, thus, the lowest level of resolution. B provides input to NN_{nand} from the sensors. NN_{nand} provides a motoric signal to the actuator in B. B has the states $\{S_1, S_2\} \in \mathbb{S}$. Mathematically, the root and its children can be expressed as:

$$\{S_1, S_2,\ NN_{\text{nand}}\} \subseteq B \tag{6.34}$$

The child nodes of the root B, i.e. S_1, S_2, NN_{nand}, are the next higher level of resolution of the SBS.

From figure 6.24 it follows that NN_{nand} has four neurons as children:

$$\{N_{\text{sensA}}, N_{\text{sensB}}, N_{\text{nand}}, N_{\text{motor}}\} \subseteq NN_{\text{nand}} \tag{6.35}$$

The next level of resolution for equation (6.35) can be defined as:

$$\{M_{pS_{\text{sensA}}}, M_{m\text{sensA}}, \Upsilon_{\text{sensA}}\} \subset N_{\text{sensA}} \tag{6.36}$$
$$\{M_{pS_{\text{sensB}}}, M_{m\text{sensA}}, \Upsilon_{\text{sensA}}\} \subset N_{\text{sensB}} \tag{6.37}$$
$$\{M_{pA_{S_{\text{nand}}}}, M_{pB_{S_{\text{nand}}}}, M_{m\text{nand}}, \Upsilon_{\text{nand}}\} \subset N_{\text{nand}} \tag{6.38}$$
$$\{M_{pS_{\text{motor}}}, M_{m\text{motor}}\} \subset N_{\text{motor}} \tag{6.39}$$

6.10.4.2 Connections of the NAND network

Based on figure 6.24 and the SBS described in equations (6.34) to (6.39), the following interconnections are identified:

- *Body to Brain* connections, i.e. connections between the sensors in a body and a RealNeuron network:

$$\begin{aligned} S_1 &\rightarrow M_{mS_{\text{sensA}}} \\ S_2 &\rightarrow M_{mS_{\text{sensB}}} \end{aligned}$$

- *Brain to Body* connections, i.e. connections between a RealNeuron network and the actuators in a body:

$$M_{m\,\text{motor}} \rightarrow S_2$$

- *Brain to Brain* connections, i.e. the connections between RealNeurons inside a RealNeuron network:

$$\begin{aligned} \Upsilon_{\text{sensA}} &\rightarrow M_{pA_{S_{\text{nand}}}} \\ \Upsilon_{\text{sensB}} &\rightarrow M_{pB_{S_{\text{nand}}}} \\ \Upsilon_{\text{nand}} &\rightarrow M_{pS_{\text{motor}}} \end{aligned}$$

6.10.4.3 Expected results of the NAND network

Table 6.10 shows the expected results, if (6.2) and (6.3) are used as input signal to the NAND RealNeuron network in figure 6.24, with $c = -30\text{mV}$ in equation (6.33) and the membrane parameters per table 6.1. The reader is referred to the *Nand RealNeuron Simulation Results* report for the other parameters avaliable on the enclosed CD-ROM.

The NAND function can be implemented by a perceptron as shown in Figure 6.25.

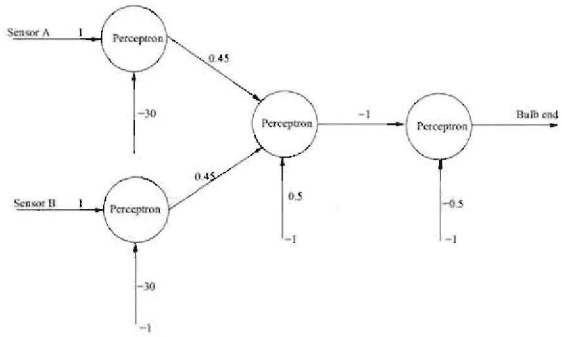


Figure 6.25: A perceptron implementation of the NAND function

6.10.4.4 Results of the NAND network

The results in figure 6.26 correspond with the expected results in table 6.10. The NAND function output is low ($b = -67.15\text{mV}$) if both inputs are equal or above -30mV, and high ($a = -57.14\text{mV}$) for anything else.

Table 6.10: Expected results of the NAND function

Time step	Sensor 1 (mV)	Sensor 2 (mV)	Bulb end
0	-30.00	20.00	b
1	-17.57	18.43	b
2	-5.91	13.82	b
3	4.23	6.45	b
4	12.22	-3.21	b
5	17.55	-14.55	b
6	19.90	-26.86	b
7	19.11	-39.37	a
8	15.24	-51.29	a
9	8.53	-61.87	a
10	-0.61	-70.45	a
11	-11.59	-76.49	a
12	-23.73	-79.61	a
13	-36.27	-79.61	a
14	-48.41	-76.49	a
15	-59.39	-70.45	a
16	-68.53	-61.87	a
17	-75.24	-51.29	a
18	-79.11	-39.37	a
19	-79.90	-26.86	a
20	-77.55	-14.55	a
21	-72.22	-3.21	a
22	-64.23	6.45	a
23	-54.09	13.82	a
24	-42.43	18.43	a
25	-30.00	20.00	b

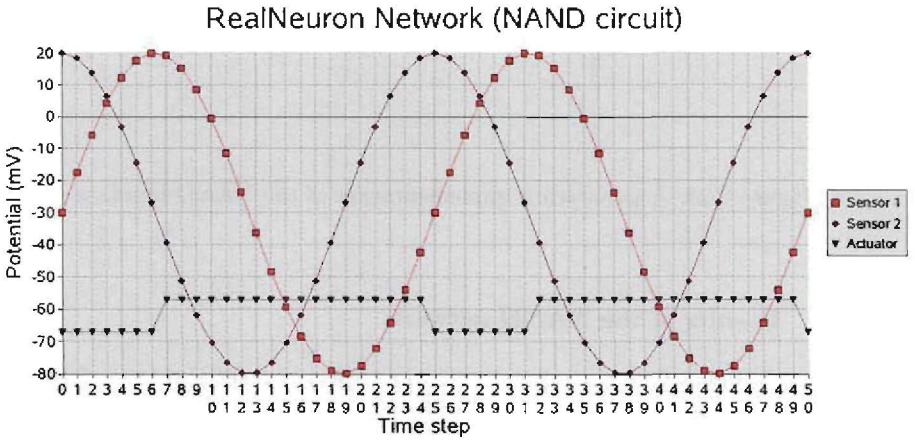


Figure 6.26: Results of the NAND RealNeuron network

6.10.5 OR function

The logical OR function between two signals can be defined as follows: if the values of one of the signals are above a specific threshold, the output is high, otherwise the output is low. Mathematically, this can be formulated as:

$$\mathrm{or}(x, y) = \begin{cases} a & \text{if } x \geq c \text{ or } y \geq c \\ b & \text{otherwise} \end{cases} \tag{6.40}$$

where $a > b$.

6.10.5.1 SBS of the OR network

The OR RealNeuron network in figure 6.27 can also be described by a system breakdown structure (SBS), by following a similar argument to that in §6.10.1.1.

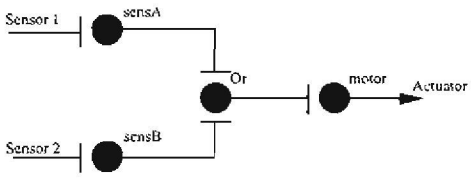


Figure 6.27: OR RealNeuron network

Let NN_{or} represent the OR RealNeuron network. NN_{or} is part of the body (B). B is chosen as the root of the SBS and, thus, the lowest level of resolution. B provides input to NN_{or} from the sensors. NN_{or} provides a motoric signal to the actuator in B. B has the states $\{S_1, S_2\} \in \mathbb{S}$. Mathematically, the root and its children can be expressed as:

$$\{S_1, S_2,\ NN_{\mathrm{or}}\} \subseteq B \tag{6.41}$$

The child nodes of the root B, i.e. S_1, S_2, NN_{or}, are the next higher level of resolution of the SBS.

From figure 6.27 it follows that NN_{or} has four neurons as children:

$$\{N_{\mathrm{sensA}}, N_{\mathrm{sensB}}, N_{\mathrm{or}}, N_{\mathrm{motor}}\} \subseteq NN_{\mathrm{or}} \tag{6.42}$$

The next level of resolution for equation (6.42) can be defined as

$$\{M_{pS_{\mathrm{sensA}}}, M_{m\mathrm{sensA}}, \Upsilon_{\mathrm{sensA}}\} \subset N_{\mathrm{sensA}} \tag{6.43}$$
$$\{M_{pS_{\mathrm{sensB}}}, M_{m\mathrm{sensA}}, \Upsilon_{\mathrm{sensA}}\} \subset N_{\mathrm{sensB}} \tag{6.44}$$
$$\{M_{pA_{S_{\mathrm{or}}}}, M_{pB_{S_{\mathrm{or}}}}, M_{m\mathrm{or}}, \Upsilon_{\mathrm{or}}\} \subset N_{\mathrm{or}} \tag{6.45}$$
$$\{M_{pS_{\mathrm{motor}}}, M_{m\mathrm{motor}}\} \subset N_{\mathrm{motor}} \tag{6.46}$$

6.10.5.2 Connections of the OR network

Based on figure 6.27 and the SBS described in equations (6.41) to (6.46), the following interconnections are identified:

- *Body to Brain* connections, i.e. connections between the sensors in a body and a RealNeuron network:

$$\begin{aligned} S_1 &\rightarrow M_{mS_{\text{sensA}}} \\ S_2 &\rightarrow M_{mS_{\text{sensB}}} \end{aligned}$$

- *Brain to Body* connections, i.e. connections between a RealNeuron network and the actuators in a body:

$$M_{m\,\text{motor}} \rightarrow S_2$$

- *Brain to Brain* connections, i.e. the connections between RealNeurons inside a RealNeuron network:

$$\begin{aligned} \Upsilon_{\text{sensA}} &\rightarrow M_{pA_{S_{or}}} \\ \Upsilon_{\text{sensB}} &\rightarrow M_{pB_{S_{\text{or}}}} \\ \Upsilon_{\text{or}} &\rightarrow M_{pS_{\text{motor}}} \end{aligned}$$

6.10.5.3 Expected results of the OR network

Table 6.11 shows the expected results, if (6.2) and (6.3) are used as input signals to the OR RealNeuron network in figure 6.27, with $c = -30\text{mV}$ in equation (6.40) and the membrane parameters per table 6.1. The reader is referred to the *Or RealNeuron Simulation Results* report for the other parameters available on the enclosed CD-ROM.

The OR function can be implemented by a perceptron as shown in Figure 6.28.

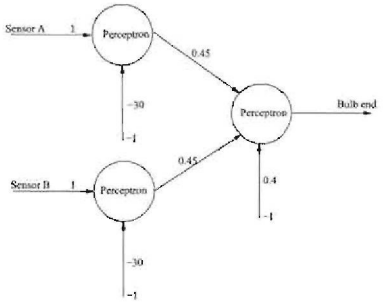


Figure 6.28: A perceptron implementation of the OR function

6.10.5.4 Results of the OR network

The results in figure 6.29 correspond with the expected results in table 6.11. The OR function output is high ($a = -57.14\text{mV}$) if any one of the input signals or both are equal or above -30mV, and low ($b = -67.15\text{mV}$) for anything else.

Table 6.11: Expected results for the OR function

Time step	Sensor 1 (mV)	Sensor 2 (mV)	Bulb end
0	-30.00	20.00	a
1	-17.57	18.43	a
2	-5.91	13.82	a
3	4.23	6.45	a
4	12.22	-3.21	a
5	17.55	-14.55	a
6	19.90	-26.86	a
7	19.11	-39.37	a
8	15.24	-51.29	a
9	8.53	-61.87	a
10	-0.61	-70.45	a
11	-11.59	-76.49	a
12	-23.73	-79.61	a
13	-36.27	-79.61	b
14	-48.41	-76.49	b
15	-59.39	-70.45	b
16	-68.53	-61.87	b
17	-75.24	-51.29	b
18	-79.11	-39.37	b
19	-79.90	-26.86	a
20	-77.55	-14.55	a
21	-72.22	-3.21	a
22	-64.23	6.45	a
23	-54.09	13.82	a
24	-42.43	18.43	a
25	-30.00	20.00	a

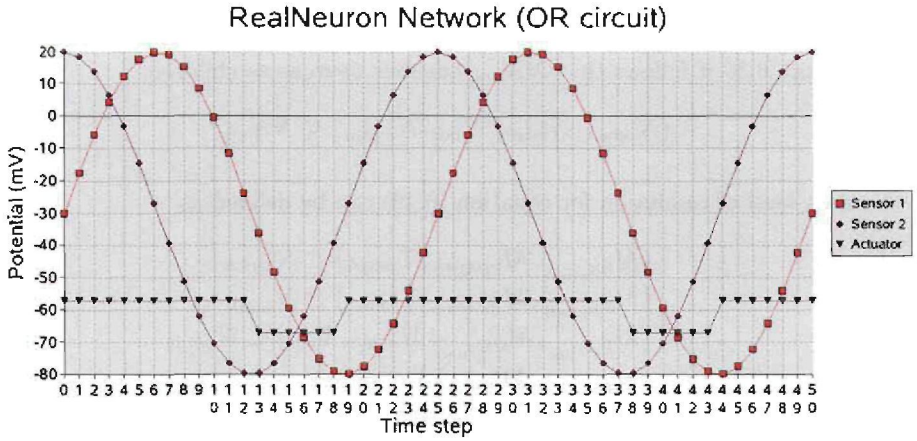


Figure 6.29: Results of the OR RealNeuron network

6.10.6 NOR function

The logical NOR function between two signals can be defined as follows: if the value of one of the signals is above a specific threshold, the output is low, otherwise the output is high. This is the inverse of the OR function defined in 6.10.5. Mathematically, this can be formulated as:

$$\mathrm{nor}(x,y) = \begin{cases} b & \text{if } x > c \text{ nor } y > c \\ a & \text{otherwise} \end{cases} \tag{6.47}$$

where $b < a$.

6.10.6.1 SBS of the NOR network

The NOR RealNeuron network in figure 6.30 can also be described by a system breakdown structure (SBS), by following a similar argument to that in §6.10.1.1.

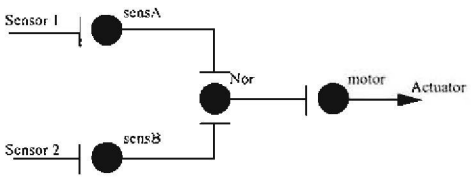


Figure 6.30: NOR RealNeuron network

Let NN_{nor} represent the NOR RealNeuron network. NN_{nor} is part of the body (B). B is chosen as the root of the SBS and, thus, the lowest level of resolution. B provides input to NN_{nor} from the sensors. NN_{nor} provides a motoric signal to the actuator in B. B has the states $\{S_1, S_2\} \in \mathbb{S}$. Mathematically, the root and its children can be expressed as:

$$\{S_1, S_2,\ NN_{\mathrm{nor}}\} \subseteq B \tag{6.48}$$

The child nodes of the root B, i.e. S_1, S_2, NN_{nor}, are the next higher level of resolution of the SBS.

From figure 6.30 it follows that NN_{nor} has four neurons as children:

$$\{N_{\mathrm{sensA}}, N_{\mathrm{sensB}}, N_{\mathrm{nor}}, N_{\mathrm{motor}}\} \subseteq NN_{\mathrm{nor}} \tag{6.49}$$

The next level of resolution for equation (6.49) can be defined as

$$\{M_{pS_{\mathrm{sensA}}}, M_{m\mathrm{sensA}}, \Upsilon_{\mathrm{sensA}}\} \subset N_{\mathrm{sensA}} \tag{6.50}$$
$$\{M_{pS_{\mathrm{sensB}}}, M_{m\mathrm{sensA}}, \Upsilon_{\mathrm{sensA}}\} \subset N_{\mathrm{sensB}} \tag{6.51}$$
$$\{M_{pA_{S_{\mathrm{nor}}}}, M_{pB_{S_{\mathrm{nor}}}}, M_{m\,\mathrm{nor}}, \Upsilon_{\mathrm{nor}}\} \subset N_{\mathrm{nor}} \tag{6.52}$$
$$\{M_{pS_{\mathrm{motor}}}, M_{m\mathrm{motor}}\} \subset N_{\mathrm{motor}} \tag{6.53}$$

6.10.6.2 Connections of the NOR network

Based on figure 6.30 and the SBS described in equations (6.48) to (6.53), the following interconnections are identified:

- *Body to Brain* connections, i.e. connections between the sensors in a body and a RealNeuron network:

$$\begin{aligned} S_1 &\rightarrow M_{mS_{\text{sensA}}} \\ S_2 &\rightarrow M_{mS_{\text{sensB}}} \end{aligned}$$

- *Brain to Body* connections, i.e. connections between a RealNeuron network and the actuators in a body:

$$M_{m\text{motor}} \rightarrow S_2$$

- *Brain to Brain* connections, i.e. the connections between RealNeurons inside a RealNeuron network:

$$\begin{aligned} \Upsilon_{\text{sensA}} &\rightarrow M_{pA_{S_{\text{nor}}}} \\ \Upsilon_{\text{sensB}} &\rightarrow M_{pB_{S_{\text{nor}}}} \\ \Upsilon_{\text{nor}} &\rightarrow M_{pS_{\text{motor}}} \end{aligned}$$

6.10.6.3 Expected results of the NOR network

Table 6.12 shows the expected results, if (6.2) and (6.3) are used as input signal to the NOR RealNeuron network in figure 6.30, with $c = -30\text{mV}$ in equation (6.47) and the membrane parameters per table 6.1. The reader is referred to the *NOR RealNeuron Simulation Results* report for the other parameters available on the enclosed CD-ROM.

The NOR function can be implemented by a perceptron as shown in Figure 6.31.

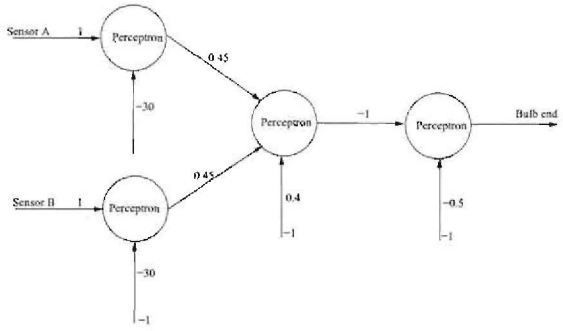


Figure 6.31: A perceptron implementation of the NOR function

6.10.6.4 Results of the NOR network

The results in figure 6.32 correspond with the expected results in table 6.12. The NOR function output is high ($a = -57.14\text{mV}$) if the input signals are both below -30mV, and low ($a = -67.15\text{mV}$) for anything else.

Table 6.12: Expected results for the NOR function

Time step	Sensor 1 (mV)	Sensor 2 (mV)	Bulb end
0	-30.00	20.00	b
1	-17.57	18.43	b
2	-5.91	13.82	b
3	4.23	6.45	b
4	12.22	-3.21	b
5	17.55	-14.55	b
6	19.90	-26.86	b
7	19.11	-39.37	b
8	15.24	-51.29	b
9	8.53	-61.87	b
10	-0.61	-70.45	b
11	-11.59	-76.49	b
12	-23.73	-79.61	b
13	-36.27	-79.61	a
14	-48.41	-76.49	a
15	-59.39	-70.45	a
16	-68.53	-61.87	a
17	-75.24	-51.29	a
18	-79.11	-39.37	a
19	-79.90	-26.86	b
20	-77.55	-14.55	b
21	-72.22	-3.21	b
22	-64.23	6.45	b
23	-54.09	13.82	b
24	-42.43	18.43	b
25	-30.00	20.00	b

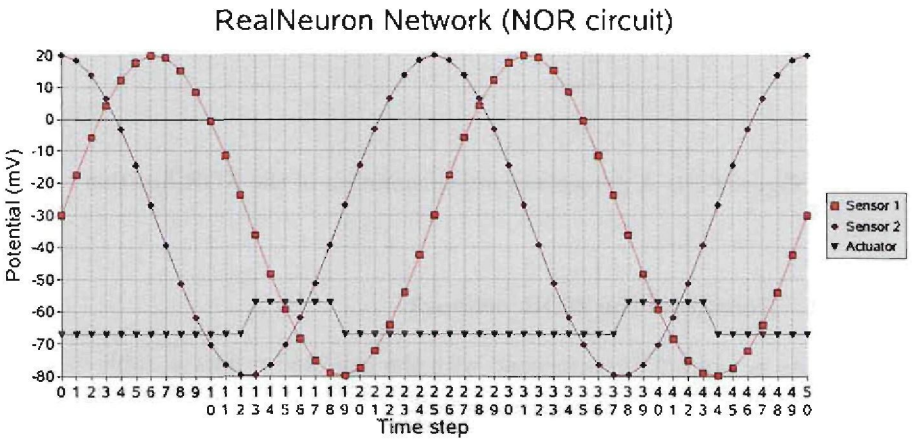


Figure 6.32: Results of the NOR RealNeuron network

6.10.7 XOR function

The logical XOR function between two signals can be defined as follows: if the values of both signals are above or below a specific threshold, the output is low, otherwise the output is high. Mathematically, this can be formulated as:

$$\text{xor}(x,y) = \begin{cases} b & \text{if } x \geq c \text{ and } y \geq c \\ b & \text{if } x \leq c \text{ and } y \leq c \\ a & \text{otherwise} \end{cases} \tag{6.54}$$

where $b < a$.

6.10.7.1 SBS of the XOR network

The XOR RealNeuron network in figure 6.33 can also be described by a system breakdown structure (SBS), by following a similar argument to that in §6.10.1.1.

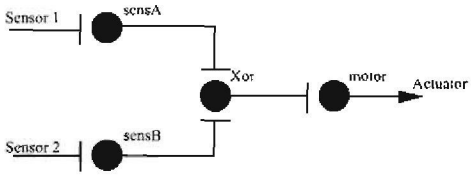


Figure 6.33: XOR RealNeuron network

Let NN_{xor} represent the XOR RealNeuron network. NN_{xor} is part of the body (B). B is chosen as the root of the SBS and, thus, the lowest level of resolution. B provides input to NN_{xor} from the sensors. NN_{xor} provides a motoric signal to the actuator in B. B has the states $\{S_1, S_2\} \in \mathbb{S}$. Mathematically, the root and its children can be expressed as:

$$\{S_1, S_2,\ NN_{\text{xor}}\} \subseteq B \tag{6.55}$$

The child nodes of the root B, i.e. S_1, S_2, NN_{xor}, are the next higher level of resolution of the SBS.

From figure 6.33 it follows that NN_{xor} has four neurons as children:

$$\{N_{\text{sensA}}, N_{\text{sensB}}, N_{\text{xor}}, N_{\text{motor}}\} \subseteq NN_{\text{xor}} \tag{6.56}$$

The next level of resolution for equation (6.56) can be defined as

$$\{M_{pS_{\text{sensA}}}, M_{m\text{sensA}}, \Upsilon_{\text{sensA}}\} \subset N_{\text{sensA}} \tag{6.57}$$
$$\{M_{pS_{\text{sensB}}}, M_{m\text{sensA}}, \Upsilon_{\text{sensA}}\} \subset N_{\text{sensB}} \tag{6.58}$$
$$\{M_{pA_{S_{\text{xor}}}}, M_{pB_{S_{\text{xor}}}}, M_{m\text{xor}}, \Upsilon_{\text{xor}}\} \subset N_{\text{xor}} \tag{6.59}$$
$$\{M_{pS_{\text{motor}}}, M_{m\text{motor}}\} \subset N_{\text{motor}} \tag{6.60}$$

6.10.7.2 Connections of the XOR network

Based on figure 6.33 and the SBS described in equations (6.55) to (6.60), the following interconnections are identified:

- *Body to Brain* connections, i.e. connections between the sensors in a body and a RealNeuron network:

$$\begin{aligned} S_1 &\rightarrow M_{mS_{\text{sensA}}} \\ S_2 &\rightarrow M_{mS_{\text{sensB}}} \end{aligned}$$

- *Brain to Body* connections, i.e. connections between a RealNeuron network and the actuators in a body:

$$M_{m\text{motor}} \rightarrow S_2$$

- *Brain to Brain* connections, i.e. the connections between RealNeurons inside a RealNeuron network:

$$\begin{aligned} \Upsilon_{\text{sensA}} &\rightarrow M_{pA_{S_{\text{xor}}}} \\ \Upsilon_{\text{sensB}} &\rightarrow M_{pB_{S_{\text{xor}}}} \\ \Upsilon_{\text{xor}} &\rightarrow M_{pS_{\text{motor}}} \end{aligned}$$

6.10.7.3 Expected results of the XOR network

Table 6.13 shows the expected results, if (6.2) and (6.3) are used as input signal to the XOR RealNeuron network in figure 6.33, with $c = -30\text{mV}$ in equation (6.54) and the membrane parameters per table 6.1. The reader is referred to the *XOR RealNeuron Simulation Results* report for the other parameters available on the enclosed CD-ROM.

The XOR function can be implemented by a perceptron as shown in Figure 6.34.

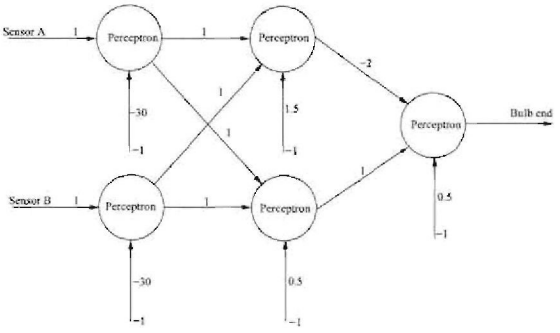


Figure 6.34: A perceptron implementation of the XOR function

6.10.7.4 Results of the XOR network

The results in figure 6.35 correspond with the expected results in table 6.13. The XOR function output is low ($a = -67.15\text{mV}$) if the input signals are equal or above, or both below -30mV, and high ($a = -57.14\text{mV}$) for anything else.

Table 6.13: Expected results for the XOR function

Time step	Sensor 1 (mV)	Sensor 2 (mV)	Bulb end
0	-30.00	20.00	b
1	-17.57	18.43	b
2	-5.91	13.82	b
3	4.23	6.45	b
4	12.22	-3.21	b
5	17.55	-14.55	b
6	19.90	-26.86	b
7	19.11	-39.37	a
8	15.24	-51.29	a
9	8.53	-61.87	a
10	-0.61	-70.45	a
11	-11.59	-76.49	a
12	-23.73	-79.61	a
13	-36.27	-79.61	b
14	-48.41	-76.49	b
15	-59.39	-70.45	b
16	-68.53	-61.87	b
17	-75.24	-51.29	b
18	-79.11	-39.37	b
19	-79.90	-26.86	a
20	-77.55	-14.55	a
21	-72.22	-3.21	a
22	-64.23	6.45	a
23	-54.09	13.82	a
24	-42.43	18.43	a
25	-30.00	20.00	b

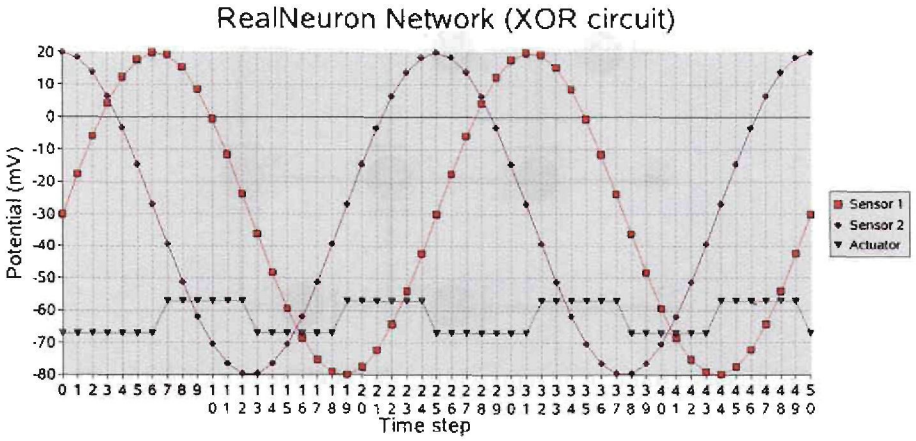


Figure 6.35: Results of the XOR RealNeuron network

6.11 Classical conditioning

The work in this section has been peer reviewed and accepted [49].

This section implements a neural circuit proposed by [110] that can implement classical or Pavlov conditioning. Classical conditioning is one of the forms of associative learning discussed in §2.4.1. Learning is one of the intellifacts discussed in §2.2 on p. 17.

Classical conditioning is best described by the Pavlov experiment [124]. A hungry dog reacts by drooling *(unconditioned reaction, UR)* when powdered meat is blown into its mouth *(unconditioned stimulus, US)*. If the US is given several times just after the ring of a bell *(conditioning stimulus, CS)*, then only the CS will make the dog drool (the now *conditioned reaction, CR)* later.

The association that the animal makes between CS and US is the result of classical conditioning [110].

Thus, it is deduced from [110], that an association is formed, in classical conditioning, between the conditioning stimulus and unconditioned stimulus. The association is the result of the time series pairing (CS, US).

$$\text{associationSet} = \{(CS(t), US(t + \Delta t), t)\}$$

If CS is given for a long time with no US following it, then the association between CS and US weakens.

6.11.1 SBS for the emulation of classical conditioning

Only a few biological neural networks have been identified that underlie learning. The RealNeuron network in figure 6.36 is deduced form the classical conditioning circuit described in [110]. The circuit in [110] is deduced from results of rabbit eye blinking experiments [110].

The classical conditioning RealNeuron network or circuit in figure 6.36 can also be described by a system breakdown structure (SBS) by following a similar argument to that in §6.10.1.1.

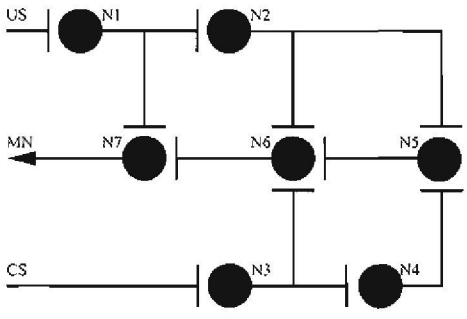


Figure 6.36: RealNeuron circuit representing the blinking of a rabbit's eye, derived from [110].

Let NN_{cc} represent the classical conditioning circuit. NN_{cc} is part of the body (B). B is chosen as the root of the SBS and, thus, the lowest level of resolution. B provides input to NN_{cc} from the sensors. NN_{cc} provides a motoric signal to the

actuator in B. B has the states $\{S_1, S_2\} \in \mathbb{S}$. Mathematically, the root and its children can be expressed as:

$$\{S_1, S_2,\ NN_{cc}\} \subseteq B \tag{6.61}$$

The child nodes of the root B, i.e. S_1, S_2, NN_{cc}, are the next higher level of resolution of the SBS.

From figure 6.36 it follows that NN_{cc} has seven neurons as children:

$$\{N_1, N_2, ..., N_7\} \subseteq NN_{cc} \tag{6.62}$$

The next level of resolution for (6.62) can be defined as

$$\{M_{pS_1}, M_{m1}, \Upsilon_{11}, \Upsilon_{21}\} \subset N_1 \tag{6.63}$$
$$\{M_{pS_2}, M_{m2}, \Upsilon_{12}, \Upsilon_{22}\} \subset N_2 \tag{6.64}$$
$$\{M_{pS_3}, M_{m3}, \Upsilon_{13}, \Upsilon_{23}\} \subset N_3 \tag{6.65}$$
$$\{M_{pS_4}, M_{m4}, \Upsilon_4\} \subset N_4 \tag{6.66}$$
$$\{M_{p1S_5}, M_{p2S_5}, M_{m5}, \Upsilon_5\} \subset N_5 \tag{6.67}$$
$$\{M_{p1S_6}, M_{p2S_6}, M_{p3S_6}, M_{m6}, \Upsilon_6\} \subset N_6 \tag{6.68}$$
$$\{M_{p1S_7}, M_{p2S_7}, M_{m7}\} \subset N_7 \tag{6.69}$$
$$\tag{6.70}$$

6.11.2 Connections for classical conditioning

Based on figure 6.36 and the SBS described in (6.61) to (6.69), the following interconnections are identified:

- *Body to Brain* connections, i.e. connections between the sensors in a body and a RealNeuron network:

$$\begin{aligned} S_1 &\rightarrow M_{mS_1} \\ S_2 &\rightarrow M_{mS_3} \end{aligned}$$

- *Brain to Body* connections, i.e. connections between a RealNeuron network and the actuators in a body:

$$M_{m7} \rightarrow S_2$$

- *Brain to Brain* connections, i.e. the connections between RealNeurons inside a RealNeuron network:

$$\begin{aligned} \Upsilon_{11} &\rightarrow M_{pS_2} \\ \Upsilon_{21} &\rightarrow M_{p2S_7} \\ \Upsilon_{12} &\rightarrow M_{p1S_5} \\ \Upsilon_{22} &\rightarrow M_{p2S_6} \\ \Upsilon_{13} &\rightarrow M_{pS_4} \\ \Upsilon_{23} &\rightarrow M_{p3S_6} \\ \Upsilon_4 &\rightarrow M_{p2S_5} \\ \Upsilon_5 &\rightarrow M_{p1S_6} \\ \Upsilon_6 &\rightarrow M_{p1S_7} \end{aligned}$$

6.11.3 Expected results for classical conditioning

Figure 6.37 shows the deduced logical functions from [110] that underlie the neuron circuit. In figure 6.37, N_5 is a Purkinje cell, see §6.9.3, and N_6 is a nucleus interpositus cell, see §6.9.4.

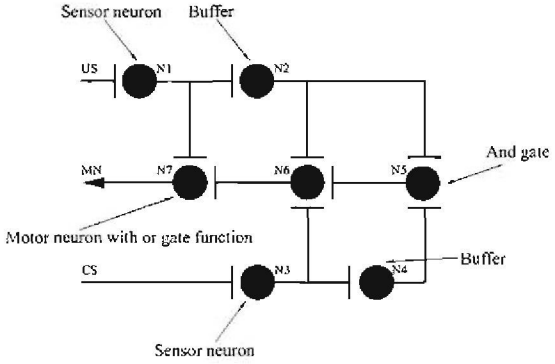


Figure 6.37: Digital logic functions for classical conditioning

It follows from the Purkinje cell description in §6.9.3 that the classical conditioning circuit cannot be built with perceptrons in a feed-forward network. The perceptron implementation of the classical conditioning circuit introduces an unnecessarily complex and unhierarchical structure that can result in the loss of the overview on the functionality.

The digital logic functions that can be used to implement a classical conditioning circuit are summarised in table 6.14.

Table 6.14: Digital logic functions to implement classical conditioning

Neuron	Description	Digital Logic Function	Ref
N1	Sensor	BUFFER	§6.10.1
N2	Buffer	BUFFER	§6.10.1
N3	Sensor	BUFFER	§6.10.1
N4	Buffer	BUFFER	§6.10.1
N5	Purkinje	OR, 2×AND & 3×Mono Stable Vibrator	§6.9.3
N6	Nucleus Interpositus	AND & OR	§6.9.4
N7	Motor	OR	§6.10.5

The qualitative synaptic strengths are deduced from [110]. They are identified in figure 6.38 as well as the "addition" of the nucleus interpositus' post-synaptic membranes.

The time delay on the synapse between N_5 and N_6 is achieved by implementing an LTP function, see §4.3.9 p. 59, in N_5. The implementation of the N_5 description in §6.9.3 p. 119 results in a long-term delay function in the classical conditioning circuit.

The expected behaviour of the classical conditioning circuit is deduced from [110]. It is expected that if only US is stimulated, then MN will react accordingly. If only CS is stimulated, then MN will not give any response.

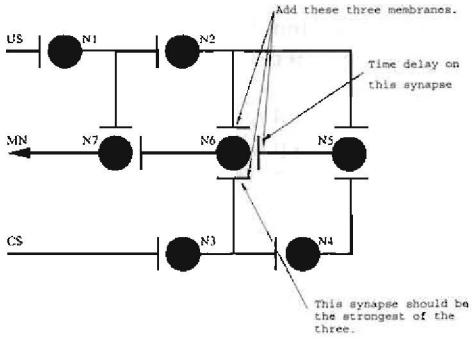


Figure 6.38: Functional identification of N_5 and N_6

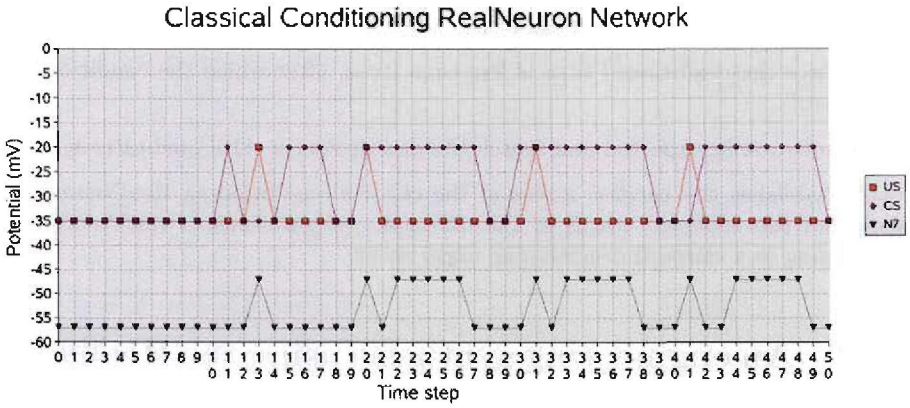


Figure 6.39: Results of classical conditioning RealNeuron network

If CS and US are stimulated together within the two-step time window with which the conditioning circuit is configured, the circuit will be conditioned after one time step of the last input of five time steps. If CS is stimulated during the conditioned state of the circuit, MN will react accordingly.

The classical conditioning RealNeuron network can also be implemented with discrete logic integrated circuits (ICs) or perceptrons. The component descriptions in table 6.14 are integrated into one circuit. Similar limitations are applicable to this implementations as for the Purkinje cell, see §6.9.3.

6.11.4 Results of the classical conditioning network

The results are presented in figure 6.39; N7 is synonymous to MN in figure 6.36.

CS and US are stimulated as follows:

- CS is stimulated at time step 11, and N7 does not respond.
- US is stimulated at time step 13, and N7 responds.
- CS is stimulated from time step 15 to 17, and N7 does not respond.

- CS is stimulated from time step 20 to 27, US is stimulated at time step 20, and N7 responds at time step 20. N7 does not respond at time step 21, and responds again from time step 22 to 26. It does not respond at time step 27.
- CS is stimulated from time step 30 to 38, US is stimulated at time step 31, and N7 responds at time step 31. N7 does not respond at time step 32, and responds again from time step 33 to 37. It does not respond at time step 38.
- CS is stimulated from time step 42 to 49, US is stimulated at time step 41, and N7 responds at time step 41. N7 does not respond at time step 42 and 43, and responds again from time step 44 to 48. It does not respond at time step 49.

The time step after CS or US is a result of the LTP function implementation, see §6.9.3. The two time steps in the last result come about because CS is presented after US during the two-step time window, but the circuit is not yet conditioned. The following time step is a result of the LTP function implementation.

In all cases the conditioned time is five time steps, after which the conditioning is "forgotten".

The results are as expected and match the description in [110] qualitatively.

This establishes the *verified* status of the classical conditioning RealNeuron network and the *validated* status of the different configurations of RealNeurons for functioning in a classical conditioning experiment.

6.12 An engineering synthesis using the RealNeuron

This section describes how individual classical conditioning circuits are combined to implement more complex forms of classical conditioning. This has been peer reviewed and accepted [49].

The requirement is to implement a RealNeuron network that can be used in machines with grades of machine intelligence, §2.1 on p. §15 and §2.2 on p. 17, that requires classical conditioning learning implemented with neural networks[4]. Further, the RealNeuron network must be able to provide the machine with the capability to change learned associations as the environment changes.

To meet the above requirement, an use-case needs to be constructed to help in the synthesis of a solution to meet the above requirement. The poison and food game is proposed as such an use-case.

6.12.1 The poison and food game

The poison and food game describes a concept application, in a controlled environment, for a robot to adapt to a changing poison and food environment. Figure 6.40(a) shows the robot inside the poison and food environment. This concept application is an ideal set-up to demonstrate the potential of the RealNeuron technology in a learning robot.

[4]The reason for this requirement is given in §1.3 p. 4: "People do not have huge databases of formally-encoded knowledge as required in classical artificial intelligence. They capture knowledge in a distributed way through neural networks in the brain." This requirement excludes other artificial intelligence techniques like rule-based techniques, etc.

The robotic platform used is the Lego Mindstorms NXT[5]. This platform is relatively cheap compared to other platforms and is fully programmable. It also has a Bluetooth connectivity option for communication with the platform. The following commercial off-the-shelf (COTS) items are required for this example:

- Laptop or PC (used for programming).
- Lego Mindstorms NXT robot platform.

6.12.1.1 Behavioural description

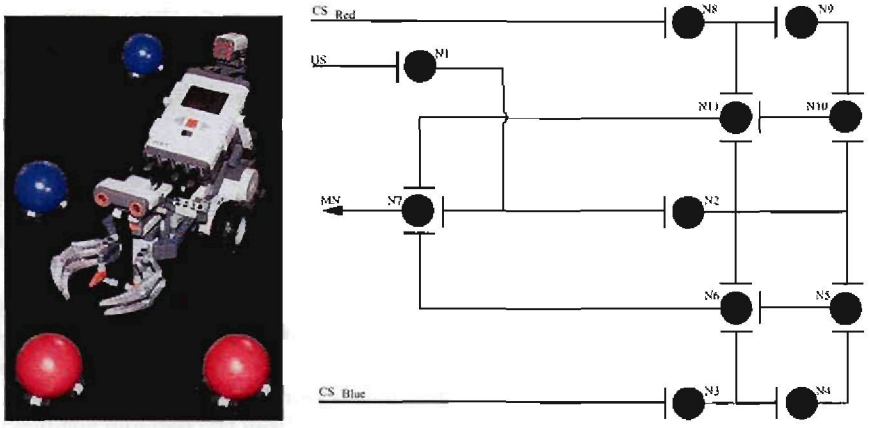
The desired behaviour of the robot is to move in an environment scattered with blue and red balls and stay healthy. The balls can change their status as either poison or food at any time. The poison is only strong enough to make the robot feel sick. The robot discovers by itself when balls are poisonous and when they are food again.

The robot plays around in the environment, avoiding the balls. When it becomes hungry, it switches into food-searching mode. It identifies a ball with an ultrasonic sensor and determines its colour with an optical sensor.

If the robot has no previous knowledge of the effect of eating a specifically coloured ball, it "eats" the ball by picking it up and dropping it again. If the robot experiences discomfort, the association should be made that the colour is poisonous. If the robot picks up the other coloured ball, it is fed.

If the robot has previous knowledge of the effect of eating a specifically coloured ball, it avoids "eating" poisonous balls.

Throughout the game, the state of the poison or food of a ball changes several times at varying arbitrary intervals.



(a) Lego Mindstorms NXT (b) NXT Brain

Figure 6.40: Hardware and brain for the poison and food game

6.12.1.2 Solution with RealNeuron technology

Figure 6.40(b) shows the RealNeuron network required to implement the solution. This solution is the combination of two conditioned learning RealNeuron networks,

[5]http://mindstorms.lego.com

see §6.1.1. The motor neuron N7 in figure 6.40(b) is a NOR function that takes the output of each conditioned learning sub-network as input. The US input is shared between the two conditioned learning sub-networks. Neurons N5 and N10 are configured with a 60s conditioned-state duration.

The Lego Mindstorms NXT robot, in figure 6.40(a), uses the following sensors for this game:

- Optical sensor - used as input to discriminate between blue and red balls.
- Touch sensor - to feel when to close the claw to pick up a ball.
- Ultrasonic sensor - to see where the balls are.
- Sound sensor - to communicate when the status of the balls change.

The robot has the following simulated internal states:

- A simulated energy level that is implemented through a timer that counts down from 20s.
- A poisonous colour status that indicates what colour is currently poisonous. This status is not part of the robot's consciousness.

When the robot boots, it defaults to the poisonous colour status of red, i.e. a red ball is poisonous and a blue ball is food. When the sound sensor senses a clapping sound, the poisonous colour status is reversed, i.e. red changes to blue or blue changes to red.

The robot plays in the environment and avoids the balls by using its ultrasonic sensor. When its energy levels are low, i.e. the energy timer counts down to 0, it switches from avoiding balls to searching for balls to eat.

When the touch sensor is triggered, the robot determines the colour of the ball in front of it. The detected colour is fed into the RealNeuron network as the appropriate CS (CS_{Red} or CS_{Blue}) in figure 6.40(b). If the motor neuron in figure 6.40(b) fires, the ball is eaten by closing the claw of the robot. If the colour is poisonous, CS, together with the US, is sent as input to the RealNeuron network. The network is conditioned with the input. The output of the conditioned network will cause the robot claw to open. The motor neuron will not fire if the colour has been determined as poisonous. The energy timer is reset to 20s and a claw opening command will be sent to the claw. The robot backs off from the ball and the playing routine starts again.

If the robot is not conditioned, it only learns when a poisonous ball is picked up. During the conditioned state, it avoids all poisonous balls. After 60s, the robot loses its conditioned state.

If the poisonous colour status is changed during the 60s of conditioning, e.g. from red to blue at about 23s, the robot avoids both blue and red balls, until the 60s has expired, after which the right food coloured ball is eaten. The conditioned state for the blue ball expires 60s after the status has changed from red to blue, thus, the robot is conditioned 37s after the robot starts eating red balls.

6.12.1.3 Kinematic description

The above description can be reduced to only the stimulation of the RealNeuron network in figure 6.40(b). N7 in figure 6.40(b) is an OR function. The conditioned time duration is 20*s* for N5 and N10, is opposed to the 60*s* proposed for the NXT. This reduction in the conditioned time makes it possible to simulate faster.

If the inputs (US, CS_{Red} and CS_{Blue}) are low, the output (MN) is low. If only the US signal is high, MN is high. If only CS_{Red} or CS_{Blue} is high when the network is unconditioned, MN is low.

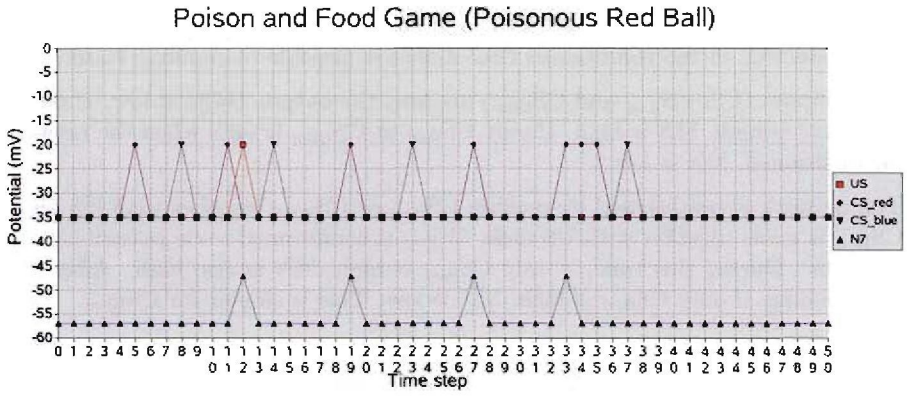
The network is conditioned by making CS_{Red} or CS_{Blue} together with US high. If CS_{Red} is made together with US high, CS_{Red} is conditioned to be poisonous for the next 20s. During this time, if only CS_{Red} is high, MN is also high. After 20s of conditioning, if only CS_{Red} is high, MN is low. This is similar for CS_{Blue}.

6.12.1.4 Results

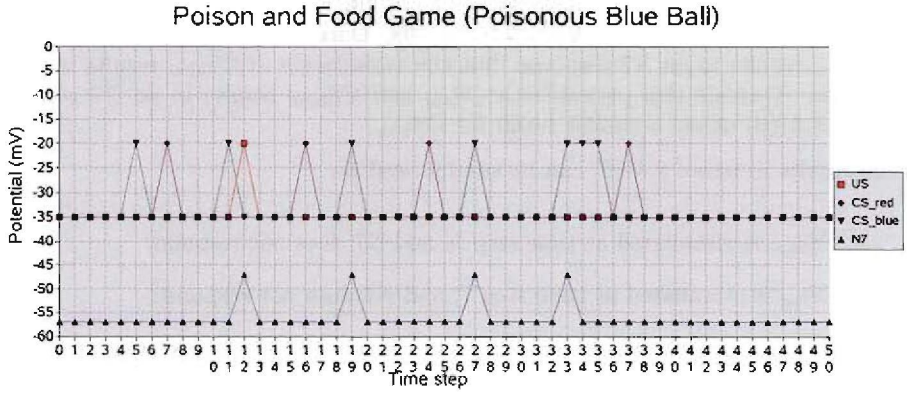
The simulation results of the RealNeuron network in figure 6.40(b) are shown in figure 6.41. Figure 6.41(a) shows when the circuit is conditioned for CS_{red} signals. The circuit is in the conditioned state for 20*s*. During this period, the stimulation of CS_{red} results in an N7 response, but the stimulation of CS_{blue} results in no N7 response. Outside this period both CS_{red} and CS_{blue} results in no N7 response. Figure 6.41(b) shows a similar result for CS_{blue}.

The results in figure 6.41(c) can be summarised as:

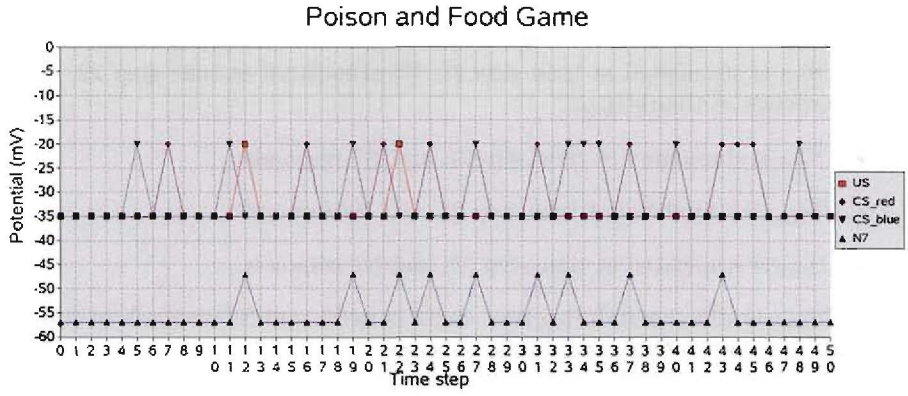
- CS_{blue} is stimulated at time step 5, and N7 does not respond.
- CS_{red} is stimulated at time step 7, and N7 does not respond.
- CS_{blue} is stimulated at time step 11, US stimulated at time step 12, and N7 responds at time step 12.
- CS_{red} is stimulated at time step 16, and N7 does not respond.
- CS_{blue} is stimulated at time step 19, and N7 responds.
- CS_{red} is stimulated at time step 21, US stimulated at time step 22, and N7 responds at time step 22.
- CS_{red} is stimulated at time step 24, and N7 responds.
- CS_{blue} is stimulated at time step 27, and N7 responds.
- CS_{red} is stimulated at time step 31, and N7 responds.
- CS_{blue} is stimulated during time steps 33 to 35, and N7 responds at time step 33. N7 does not respond during time step 34 and 35.
- CS_{red} is stimulated at time step 37, and N7 responds.
- CS_{blue} is stimulated at time step 40, and N7 does not respond.
- CS_{red} is stimulated during time step 43 to 45, and N7 responds at time step 43. N7 does not respond during time step 44 and 45.
- CS_{blue} is stimulated at time step 48, and N7 does not respond.



(a) Only the red ball is poisonous



(b) Only the blue ball is poisonous



(c) Both balls become poisonous at different points of time

Figure 6.41: Results of the Poison and Food Simulation

The RealNeuron network is in a blue conditioned state from time step 14 to 33. It is in a red conditioned state from time step 24 to 43. Thus, the network is:

- Conditioned exclusively blue from time step 14 to 33.
- Conditioned blue and red from time step 24 to 33.
- Conditioned exclusively red from time step 34 to 43.

If the circuit is conditioned:

- Exclusively blue, then CS_{red} stimulation results in no N7 response. During this period, CS_{blue} stimulation results in an N7 response.
- Exclusively red, then CS_{blue} stimulation results in no N7 response. During this period, CS_{red} stimulation results in an N7 response.
- Blue and red, then any CS_{blue} and CS_{blue} stimulation result in an N7 response.

These results confirm the description in §6.12.1.3. The simultaneous conditioning of both colours is also demonstrated as described in §6.12.1.2.

All the learning takes place in one step with an eleven-neuron RealNeuron network. This is in contrast to classical neural networks that need several iterations to learn.

This application establishes the *verified* status for the synthesised classical conditioning RealNeuron network in figure 6.40(b) for meeting the requirement stated in the beginning of §6.12, thus also the grade of machine intelligence requirement which is classical-conditioning learning implemented with neural networks that can change learned associations in a dynamic environment.

6.13 Reflecting on the results

6.13.1 Reflecting on the membranes

The implementation of the main membrane and post-synaptic membrane in §6.7 provided the expected results. The membranes of the RealNeuron perform as described in the neurobiology literature. Most important is the repeatability of the Hodgkin and Huxley results [73, 72], that form part of the work for which they received the Nobel Prize.

The main membrane and post-synaptic membrane can also be used on their own for signal processing by adjusting the parameters of the membrane, e.g. the main membrane can be used for edge detection and the post-synaptic membrane's function can be changed midstream of processing data based on the neurotransmitter that it receives.

The major computation in the RealNeuron takes place at the membrane level. This computation is done by adding or subtracting the three ion species based on input and internal states of the membranes.

6.13.2 Reflecting on the RealNeuron sub-components

The implementation of the input soma and bulb end in §6.8 provided the expected results. The input soma, axon and bulb end are control structures that integrate the results from the different membranes and activate or deactivate different functions inside a RealNeuron based on input and internal states. It also switches signals between the different membranes inside a RealNeuron based on input and internal states. The "rule"-based activation and deactivation of functions and switching of information forms part of the underlying mechanisms of biological intelligence.

6.13.3 Reflecting on the sensor neuron

The implementation of the sensor neuron in §6.9.1 provided the expected results. The sensor neuron is the gate for signals from an environment to a RealNeuron network.

6.13.4 Reflecting on the motor neuron

The implementation of the motor neuron in §6.9.2 provided the expected results. The motor neuron is the gate for signals from a RealNeuron network to an environment.

6.13.5 Reflecting on the Purkinje cell

The implementation of the Purkinje cell in §6.9.3 provided the expected results. The Purkinje cell is one of the key operations that contribute to associative learning through classical conditioning, see [110].

The Purkinje cell contributes to the long term delay inside a classical conditioning neuron circuit. This gives rise to what is generally known as short-term memory. The long term delay is activated when both the conditioning signal (CS) and unconditioning signal (US) are on the input to the Purkinje cell or for the implementations of the RealNeuron in this thesis, when any one of the signals is given when the receptor of the post-synaptic membrane for the other signal is in a block state.

All the implementation examples in this chapter can be implemented with perceptrons in a feed-forward network topology, except for the Purkinje cell. A fairly complex perceptron network with time delay elements and feedback is necessary to implement a Purkinje cell. This complex implementation is not hierarchical and can result in losing the overview on the Purkinje cell's functionality. The complexity of the RealNeuron, handled in a rigorous hierarchical structure derived from neurobiology, provides a better overview on the Purkinje cell's functionality.

The Purkinje cell can also be implemented with discrete logic components. The long term delay achieved for discrete logic components falls far short of what is achievable in the RealNeuron, if multi-vibrators are used. A solution would be to use a digital-counter circuit to achieve the biological time constant.

6.13.6 Reflecting on the nucleus interpositus cell

The implementation of the nucleus interpositus cell in §6.9.3 provided the expected results. The nucleus interpositus cell performs as one of the key operations that contributes to associative learning through classical condition, see [110].

Once the long term delay, implemented through the Purkinje cell, is active, the nucleus interpositus cell will react on the CS without the US necessarily being present during the long term delay period.

6.13.7 Reflecting on the logic functions

The implementations of AND, OR, XOR, NOT, NAND, and NOR logic functions in §6.10 provided the expected results. All the basic logic functions, used for implementing digital computers, can be implemented using the RealNeuron based on biological parameter values. These logic functions are also the building blocks for thought, see discussion in §2.4.2.

6.13.8 Reflecting on classical conditioning

The implementation of classical conditioning in §6.11 provided the expected results. Classical conditioning is one of several methods of learning.

The results reported in the neurobiological literature can be qualitatively repeated by the classical conditioning circuit.

6.13.9 Reflecting on the poison and food game

The synthesis of the RealNeuron network, based on combining classical conditioning circuits, in §6.12 can give a robot the ability to play the poison and food game. The simulations in §6.12 provided the expected results.

6.14 Summary

Several implementations of the RealNeuron have been reported to be successful in this chapter. The simulation reports are available on the enclosed CD-ROM.

The RealNeuron network examples in this chapter can also be implemented with perceptrons and logic discrete components. The implementations for classical conditioning are complex implementations. In the case of perceptron implementations, they are complex and unhierarchical which can result in losing the overview on the Purkinje cell's functionality. The complexity of the RealNeuron, handled in a rigorous hierarchical structure derived from neurobiology, provides a better overview on the Purkinje cell's functionality.

The RealNeurons and RealNeuron networks in this chapter have been implemented by using a standard spreadsheet and the simulation speeds are fast compared to the results reported in [60].

The description given of the RealNeuron in chapter 4, together with the biological parameters given in this chapter, is a descriptive theory, see definition 2.

Chapter 7

Conclusion

> *The real question is not whether machines think but whether men do.*
> — B.F. Skinner, *Contingencies of Reinforcement* (1969) ch. 9

This thesis formulates and evaluates a neuron model based on biological neurons from an engineer's point of view. It is not trying to explain the working of a biological neuron, but to specify a neuron model or RealNeuron for engineering applications that is based on the current knowledge available form various disciplines.

It is an exercise in complex systems modelling and simulation. A systems engineering approach is used to build the RealNeuron, i.e. the model is described (specified) as using multiple resolution levels with configurable modular elements at each resolution level and is then implemented, verified and validated in a bottom-up method. The elements on a specific level of resolution is integrated as a new element on the next lower level of resolution; this is repeated until the system has been integrated.

The RealNeuron is evaluated in various neural network configurations with predictable outcomes. Although other methods, e.g. rule-based methods or perceptrons, can also be used to demonstrate the functions of these configurations, the purpose in this thesis is to show that a RealNeuron network can do that as well.

The specification of the RealNeuron model, together with its evaluation, form a baseline for current use in engineering solutions and future developments.

7.1 Formulation of a neuron model

A neuron model is developed in the engineering context presented in chapter 2.

The structure of the RealNeuron is presented in §3.3. This structure is based on biological neuron building blocks. An intelligent system overview is presented in figure 3.6 showing the different components on different levels of resolution.

A set predicate model of the RealNeuron is completed and the terminology and symbols are standardised in §4.2, based on the structural description in §3.3. The standardised terminology and symbols are applied to the message processing algorithms in §4.3.

A mathematical RealNeuron is used to model the information-processing capabilities of biological neurons as shown in §4.4.

A state space model of the RealNeuron is used to model the information processing capabilities of biological neurons as shown in §4.4. This model makes an explicit

distinction between the input states, internal states, output states and the parameters of the model. The parameters of the model will make it possible, in future, to optimise the configuration of a RealNeuron instance using an optimisation algorithm, e.g. a genetic algorithm. The structure of this model might, at first, seem very complex, but the underlying mechanism is just the addition and subtraction of concentrations based on potential levels and the presence of neurotransmitters. The structures only provide a framework for the information flow inside the neuron. This is in contrast with the Hodgkin-Huxley model, compartmental models and some pulse neuron models that are based on electrical circuit equivalents in which differential equations need to be solved.

The model in §4.4 is also presented in block diagrams. The block diagrams visually show how the signals flow inside each component of the RealNeuron.

The model of the RealNeuron is summarised in table 4.14 which can be used as a description to implement the RealNeuron.

7.2 Evaluation of a neuron model

The RealNeuron evaluation is presented in chapters 5 and 6.

The RealNeuron model in §4.4 is theoretically evaluated in chapter 5. It is shown in §5.1 that a RealNeuron is related to other types of artificial neurons. The reference equation (5.1) compares the RealNeuron implicitly to the Hodgkin-Huxley model. All the work that has been done for first, second and third generation artificial neuron models can be applied to the RealNeuron by using the relationships presented in §5.1.

The RealNeuron relation to the spiking neuron is given in (5.15) to (5.18). Mathematically, this confirms the result that a RealNeuron is a type of pulsing neuron as discussed in chapter 5. The theories for pulsing neurons are also applicable to RealNeuron configurations that fulfil the conditions in (5.15) to (5.18). Based on the discussion about the relation of the RealNeuron to the spiking neuron in chapter 5 and the proofs in [102], it can be postulated that a RealNeuron network, consisting of a finite number of RealNeurons, can simulate a universal Turing machine. This means that, in principle, any problem that is computable [21] could be solved with RealNeuron networks.

Chapter 5 also presents a theoretical evaluation of the model developed in chapter 4, in §5.2, in order to determine the performance of the model in the presence of noise. The performance of a RealNeuron network, in a noisy engineering environment, dependents on the output value of the refractory functions of the sensor neurons in the network.

Chapter 6 presents an evaluation of the performance of a RealNeuron and its subcomponents compared to published subject matter in the neuroscience and electronics engineering literature. Implementation and evaluation of the RealNeuron are done in a bottom-up fashion, i.e. the components on the highest level of resolution of the RealNeuron are first implemented and evaluated and then integrated into the next lower level of resolution. This process is repeated until the relevant level of resolution is reached.

The result in §6.11 is the main verification result.

The engineering synthesis of a RealNeuron network, based on classical conditioning, demonstrates how to implement a RealNeuron network that can be used in machines built to the grade of machine intelligence requirement which is classical-conditioning

learning implemented with neural networks that can change learned associations in a dynamic environment.

This thesis has demonstrated the successful modelling and simulation, using a standard office package's spreadsheet[1], of a complex system in chapter 6.

The RealNeuron is a multi-resolutional and modular model that consists of several components in various layers of abstraction. It is computationally efficient and only adds or subtracts ion concentrations based on the states at the level of a membrane structure.

The synapse between two RealNeurons can be both excitatory or inhibitory and this is determined by the neurotransmitters in the synaptic gap. This is in contrast to the current practice which assumes that a synapse is either excitatory or inhibitory [16, 104, 83]. This makes it possible to implement the long-term potential (LTP) and long-term depression (LTD) functions with binary functions.

With the RealNeuron simple calculations, simulations on personal computers are possible. A spreadsheet is used to simulate 50 steps of a seven-neuron classical conditioning neural circuit in literally the blinking of an eye on a Turion64 1.8GHz notebook computer. (On an older Pentium 4 1.8GHz notebook computer, it took less than 30 seconds to execute.) All the simulated states on the highest level of resolution (i.e. the pumps, channels and metabolic receptors,) the intermediate levels of resolution (i.e. membrane potentials, ion concentrations, action potential at the axon hillock, neurotransmitters in the synapse, etc.,) and the lowest level of resolution (i.e. the conditioned signal (CS), unconditioned signal (US) and conditioned response (CR),) are available on the spreadsheet. Its performance in a noisy environment is dependent on the pump parameters in the main membrane of the sensor neurons.

7.3 Future work

7.3.1 Long-term memory

The classical conditioning RealNeuron network shows that machines, based on the RealNeuron, are capable of short-term learning through conditioning. This learning takes place by executing short-term changes to the RealNeuron's functional configuration.

A RealNeuron network can be put to "sleep", i.e. taken offline, and an optimisation algorithm can be executed in order to strengthen or weaken, or even build new or destroy existing synapses. Optimisation is performed with a cost function and is equivalent to a series of rewards or punishments. For humans the cost function is, in the Maslow sense [106], a function of body needs, security needs, ego needs, social needs, and self-actualisation.

7.3.2 Simpler and closer to biology

More work still needs to be done to reduce the number of calculations in the model and to get the model to resemble biological neurons better.

The number of times that the Nernst and Goldmann equations are calculated should be reduced. This means that the integration function in the soma will change to integrate ionic concentration.

[1] OpenOffice.org Calc was used.

The structure of the axon should be rethought and the calculation of the inverse Nernst and inverse Goldmann equations should be eliminated.

The bulb-end soma needs to be eliminated from the RealNeuron model.

7.3.3 Repeating the Hodgkin-Huxley results

The seminal work of Hodgkin and Huxley [72] provides a good reference to neurobiological experiments on the dynamics of neurons. It also presents a mathematical analysis of these experimental results.

The RealNeuron models more than just the action potential as for the case of the Hodgkin-Huxley model. The dynamics of the main membrane in the RealNeuron is the area that should be used for comparison with the Hodgkin-Huxley model as is done in chapter 5.

An investigation should be done to see how well the main membrane of theRealNeuron can repeat the experimental results in [72]. This will require the adaptation of the pump and channel functions to repeat the Hodgkin-Huxley model's dynamics for the time scale used in [72].

7.4 Final Thought

The simple operations of ion flows can lead to tremendously complex operational structures in the human brain. These ion flows are modelled in the RealNeuron through simple addition and subtraction of concentration under certain conditions that are embedded in a complex structure. Is this perhaps the simplicity that is embedded in complexity that is being searched for?

The question then is in the paraphrased words of Turing [149]: Can machines based on a RealNeuron "think"? In other words, can machines based on a RealNeuron be intelligent?

This thesis has successfully formulated and evaluated a neuron model based on biological neurons. It is a baseline for future developments of the RealNeuron model. This thesis demonstrates that this baseline can be used to synthesise a solution to implement the grade of machine intelligence requirement which is classical-conditioning learning implemented with neural networks that can change learned associations in a dynamic environment.

The engineering answer to the second question above is yes, but only for certain grades of machine intelligence.

This thesis has contributed to an understanding of how the RealNeuron can be applied as a building block for "thinking" and "learning" intellifacts. The Understanding and Planning intellifacts are still lacking. In future, this thesis should be used as a baseline for implementing more intellifacts.

Each day, humans use tools to extend their physical and mental capabilities, machine intelligent systems are tools that extend the mental capabilites of humans. The different grades of machine intelligence should be traceable to the functions of these tools.

Appendix A

Glossary

A.1 Symbology

2^X the power set of X.

AP action potential in mV.

$\mathbb{S}$ states.

φ actuators.

Φ set of actuators.

A axon.

B body.

$\mathbb{S}_B$ local body states.

$\mathbb{S}_\varphi$ effectory state.

$\mathbb{S}_\varpi$ sensory state.

C channel.

$\mathcal{C}$ set of channels.

C_c concentration channel.

C_τ receptor-gated channel.

$\mathcal{C}_R$ set of receptor-gated channels.

C_v voltage-gated channel.

$\mathcal{C}_v$ set of voltage-gated channels.

$\mathcal{E}$ environment.

$\mathbb{S}_\mathcal{E}$ environmental states.

$\mathcal{F}$ function.

$\mathcal{I}$ integration function.

$\mathcal{I}_M$ membrane integration function.

$\mathcal{I}_S$ soma integration function.

$\mathcal{K}$ connections matrix.

μ LTP function.

M membrane.

M_a axonal membrane.

M_m main membrane.

M_{mS} soma main membrane.

$M_{m\Upsilon}$ bulb-end main membrane.

M_p post-synaptic membrane.

$\mathcal{M}_p$ set of post-synaptic membranes.

M_{pA} axon post-synaptic membrane.

M_{pS} soma post-synaptic membrane.

$M_{p\Upsilon}$ bulb-end post-synaptic membrane.

Ψ membrane pore.

N neuron.

N_φ motor neuron.

N_ϖ sensor neuron.

N RealNeuron.

Ξ set of RealNeuron.

Γ Nernst and Goldmann equations.

Γ^{-1} inverse Nernst and inverse Goldmann equations.

P pump.

$\mathcal{P}$ set of pumps.

R metabolic receptor.

$\mathcal{R}$ set of metabolic receptors.

$\mathbb{R}$ real numbers.

Ψ RealNeuron network.

$\mathbb{S}_A$ local axon states.

$\mathbb{S}_C$ local channel states.

$\mathbb{S}_M$ local membrane states.

$\mathbb{S}_N$ local RealNeuron states.

$\mathbb{S}_\Psi$ local membrane pore states.

NN neural network.

$\mathbb{S}_P$ local pump states.

$\mathbb{S}_R$ local metabolic receptor states.

$\mathbb{S}_\Psi$ local RealNeuron network states.

$\mathbb{S}_S$ local soma states.

$\mathbb{S}_\Upsilon$ local bulb-end states.

ϖ sensors.

Π set of sensors.

σ ionic species.

Ψ_{SN} sub-network.

S soma.

S_i input soma.

S_ϖ sensoric soma.

S_Υ bulb-end soma.

Υ bulb end.

$\mathcal{Y}$ set of bulb ends.

τ neurotransmitter.

$\mathcal{T}$ set of neurotransmitters.

V vesicle.

A.2 Abbreviations and Acronymns

CS conditioning stimulus.

DTD Document Type Definition.

EEG electroencephalogram.

GFACS grouping, focusing attention, and combinatorial search.

HTML Hyper Text Markup Language.

IEEE Institute of Electrical and Electronics Engineers, Inc.

IG intelligence grade

ISG intelligent system grade

INM Institut für Neue Medien e.V., Frankfurt am Main, Germany.

LTD long-term depression.

LTP long-term potential.

MRC multi-resolutional control.

mV millivolt.

NASA National Aeronautics and Space Agency.

NIST National Institute of Standards and Technology.

PerMIS Performance Measurements for Intelligent Systems.

PV potential value in mV.

SBS system breakdown structure.

SEP systems engineering process.

SGML Standard General Markup Language.

UML Unified Modeling Language.

US unconditioned stimulus.

XML eXtended Markup Language.

Appendix B

Disclosure of research

B.1 Presentations

- L.D. Erasmus, "An Engineering Approach to Semiotic Representation", (presented by Döben-Henisch), Workshop on Computational Intelligence and Semiotics, Faculdade de Engenharia Elétrica e de Computação UNICAMP Campinas SP, Brasil, 31/05 - 01/06/2000.

- G. Döben-Henisch & L. Erasmus, "REALNEURON - Das biologisch motivierte Neuronmodell von Knowbotic Systems und sein Einsatz im Bereich sprachverarbeitender Systeme", Mainzer KI Kreis, University of Mainz, Mainz Germany, 3/07/2001,
  http://www.mathematik.uni-mainz.de/deutsch/aktuelles/veranstal/informatik/KI/.

- L.D. Erasmus, "The formulation of a mathematical model based on the human brain and its performance evaluation in a noisy environment", Colloquium, School for Electrical, Electronics and Computer Engineering, PU vir CHO, Potchefstroom, South Africa. 28/2/2003.

- J. Hasebrook, L. Erasmus, and G. Döben-Henisch, (Presented by Hasebrook), "Knowledge robots for knowledge workers: Self learning agents connecting information and skills", 6th International Conference on Knowledge based Intelligent Information and Engineering Systems (KES'2002), podere d'Ombriano. Crema, Italy, 16-18 September 2002.

- L.D. Erasmus and C.P. Bodenstein, "RealNeurons, Artificial Intelligence Based on the Mechanisms of Biological Intelligence", AFRICON 04, Gaborone, Botswana, 15-18 September 2004.

- L.D. Erasmus, G. Döben-Henisch, and C.P. Bodenstein, "The RealNeuron", AFRICON 04, Gaborone, Botswana, 15-18 September 2004

- L.D. Erasmus, "Comparing the RealNeuron with Spiking Neurons", AFRICON 07, Windhoek, Namibia, 26-28 September 2007.

- L.D. Erasmus, "RealNeuron Implementation of Classical Conditioning", AFRICON 07, Windhoek, Namibia, 26-28 September 2007.

B.2 Publications

- G. Döben-Henisch, L. D. Erasmus, R. Goldschmitt, and R. Stuckardt, innneuron version 0.2, work in progress, Knowbotic Systems GmbH & Co KG, Frankfurt am Main, Germany, Tech. Rep., 15 July 1999. [36]
- J. Hasebrook, L. Erasmus, and G. Döben-Henisch, "Knowledge robots for knowledge workers: Self learning agents connecting information and skills", in Proc. of the 6th International Conference on Knowledge based Intelligent Information and Engineering Systems (KES'2002), E. Damiani et.al., Ed. Amsterdam, The Netherlands: IOS Press, 16-18 Sept. 2002, pp. 572-581, podere d'Ombriano, Crema, Italy. [65]
- L.D. Erasmus and C.P. Bodenstein, "RealNeurons, Artificial Intelligence Based on the Mechanisms of Biological Intelligence", in Proc. of AFRICON 04, Gaborone, Botswana. pp. 423-428, 15-18 September 2004. [50]
- L.D. Erasmus, G. Döben-Henisch, and C.P. Bodenstein, "The RealNeuron", in Proc. of AFRICON 04, Gaborone, Botswana, pp. 1071-1078, 15-18 September 2004. [51]
- L.D. Erasmus, "Comparing the RealNeuron with Spiking Neurons", in Proc. of AFRICON 07, Windhoek, Namibia, 26-28 September 2007. [47]
- L.D. Erasmus, "RealNeuron Implementation of Classical Conditioning", in Proc. of AFRICON 07, Windhoek, Namibia, 26-28 September 2007. [49]

B.3 Publication in book series

- J. Hasebrook, L. Erasmus, and G. Doeben-Henisch, "Knowledge robots for knowledge workers: Self learning agents connecting information and skills", in Intelligent Agents and their applications, ser. Springer series: Studies in Fuzziness and Soft Computing, L. Jain, Z. Chen, and N. Ichalkaranje, Eds. Heidelberg: Physica Verlag, 2002, vol. 98, ch. 3, pp. 59-79. [64]

B.4 Citations of work in other publications

- INSPEC Updates: Neural Networks, The IEE, Hertfordshire, UK, Issue 2005-006. [76]
- H.A. Maurer, J. P. Hasebrook, Learning Support Systems for Organizational Learning. World Scientific Organizational learning / Computer network resources, 2004
- ACM Digital Library,
  http://portal.acm.org/citation.cfm?id=644436.644441&coll=GUIDE&dl=ACM&CFID=15151515&CFTOKEN=6184618

B.5 Recognition of contributions in other publications

- A. Meystel, "Part I, Measuring performance and intelligence of systems with autonomy, the white paper" , Performance Measurements for Intelligent Systems (PerMIS), NIST, Gaithersburg, Maryland, USA, 2000. [112]

B.6 Future disclosures

- L.D. Erasmus, "Experiences and insights gained from the application of systems engineering principles in the formulation and evaluation of a neuron model based on biological neurons", Submitted for INCOSE South African Chapter Conference, CSIR, Pretoria, South Africa, 28-30 August 2007 (conference cancelled). [48]

Appendix C

XML Description of the RealNeuron

Only a possible Document Template Description (DTD) of the RealNeuron network is given in this section. See the CD-ROM for an example of the classical conditioning (Duckbug) circuit as well as the AND circuit in XML format based on the DTD. These two example circuits have been done at Knowbotic Systems GmbH & CO KG, Frankfurt am Main, Germany, by Döben-Henisch and the parameters are not based on biological parameters.[1]

The DTD presented here is deduced from the Database Relation Definition in [144].

C.1 DTD

```
<!--
Title: RealNeuron Network DTD
Author: Louwrence D Erasmus
Date: 2000/08/24
version: 1.01

Remark: This DTD is composed from the Database Relation Defintion done by
        dr. Roland Stuckardt, revision: from 2.2.00
-->
<!ENTITY % boolean '(true|false|nd) "false"'>
<!-- ********************************************************************* -->
<!ELEMENT rnns (network, info, links)>
<!ELEMENT network (input+, output+, neuron+)>
<!ATTLIST network id NMTOKEN #IMPLIED
                  name ID #REQUIRED
                  write_assigned %boolean;>
<!-- ********************************************************************* -->
<!ELEMENT input   EMPTY>
<!ATTLIST input   id NMTOKEN #IMPLIED
                  name NMTOKEN #REQUIRED
                  link_to_main_membrane NMTOKEN #IMPLIED>
<!-- ********************************************************************* -->
<!ELEMENT output  EMPTY>
<!ATTLIST output  id NMTOKEN #IMPLIED
                  name NMTOKEN #REQUIRED>
<!-- ********************************************************************* -->
<!ELEMENT neuron  (soma, axon?, synapse*)>
<!ATTLIST neuron  id NMTOKEN #IMPLIED
                  name NMTOKEN #REQUIRED>
<!-- ********************************************************************* -->
<!ELEMENT ltp EMPTY>
<!ATTLIST ltp transmitter_substance_id    NMTOKEN #IMPLIED
```

[1] This is in contrast with the parameters in this thesis that have been based on biological parameters found in the neurobiology literature.

```
            emittance_event             %boolean;>
<!-- ********************************************************************* -->
<!ELEMENT soma (main_membrane, postsynaptic_membrane*, ltp?)>
<!ATTLIST soma  id   NMTOKEN #IMPLIED
                name NMTOKEN #REQUIRED>
<!-- ********************************************************************* -->
<!ELEMENT synapse (soma)>
<!ATTLIST synapse  id                       NMTOKEN #IMPLIED
                   name                     NMTOKEN      #REQUIRED
                   potential_threshold      NMTOKEN #REQUIRED
                   transmitter_substance_id NMTOKEN #IMPLIED
                   emittance_event          %boolean;
                   link_to_other_neuron     NMTOKEN #IMPLIED
                   link_to_output           NMTOKEN #IMPLIED>
<!-- ********************************************************************* -->
<!ELEMENT main_membrane (pump*, voltage_channel*, receptor_channel*,
                         default_concentration_channel*,
                         membrane_substance_concentration*)>
<!ATTLIST main_membrane  id              NMTOKEN #IMPLIED
                         name            NMTOKEN      #REQUIRED
                         temperature     NMTOKEN #REQUIRED
                         input_potential NMTOKEN #IMPLIED>
<!-- ********************************************************************* -->
<!ELEMENT postsynaptic_membrane (pump*,voltage_channel*, receptor_channel*,
                                 default_concentration_channel*,
                                 metabolic_receptor*,
                                 membrane_substance_concentration*)>
<!ATTLIST postsynaptic_membrane id  NMTOKEN #IMPLIED
                         name                  NMTOKEN      #REQUIRED
                         temperature           NMTOKEN #REQUIRED
                         LTP_output_set_member %boolean;>
<!-- ********************************************************************* -->
<!ELEMENT axon (axon_membrane*)>
<!ELEMENT axon_membrane (pump*, voltage_channel*, receptor_channel*,
                         default_concentration_channel*,
                         membrane_substance_concentration*)>
<!ATTLIST axon_membrane id              NMTOKEN #IMPLIED
                        serial_number   NMTOKEN #REQUIRED
                        temperature     NMTOKEN #REQUIRED
                        input_potential NMTOKEN #IMPLIED>
<!-- ********************************************************************* -->
<!ELEMENT pump EMPTY>
<!ATTLIST pump  id                NMTOKEN #IMPLIED
                name              NMTOKEN      #REQUIRED
                link_to_substance NMTOKEN #REQUIRED
                capacity          NMTOKEN #REQUIRED
                threshold         NMTOKEN #REQUIRED
                direction         (i|o) "i">
<!-- ********************************************************************* -->
<!ELEMENT voltage_channel EMPTY>
<!ATTLIST voltage_channel  id                 NMTOKEN #IMPLIED
                           name               NMTOKEN #REQUIRED
                           link_to_substance  NMTOKEN #REQUIRED
                           capacity           NMTOKEN #REQUIRED
                           openess            NMTOKEN #REQUIRED
                           threshold_potential NMTOKEN #REQUIRED
                           threshold_direction (p|n) "p">
<!-- ********************************************************************* -->
!ELEMENT receptor_channel EMPTY>
<!ATTLIST receptor_channel id                 NMTOKEN #IMPLIED
                           name               NMTOKEN #REQUIRED
                           link_to_substance  NMTOKEN #REQUIRED
                           capacity           NMTOKEN #REQUIRED
                           openess            NMTOKEN #REQUIRED
                           link_to_molecule   NMTOKEN #REQUIRED
                           blocktime          NMTOKEN #REQUIRED>
<!-- ********************************************************************* -->
<!ELEMENT default_concentration_channel EMPTY>
<!ATTLIST default_concentration_channel
                           id                   NMTOKEN #IMPLIED
                           name                 NMTOKEN #REQUIRED
                           link_to_substance    NMTOKEN #REQUIRED
                           capacity             NMTOKEN #REQUIRED
                           openess              NMTOKEN #REQUIRED
                           default_concentration NMTOKEN #REQUIRED>
<!-- ********************************************************************* -->
<!ELEMENT metabolic_receptor EMPTY>
<!ATTLIST metabolic_receptor
                           id                   NMTOKEN #IMPLIED
                           name                 NMTOKEN #REQUIRED
```

```
                        link_to_molecule      NMTOKEN #REQUIRED
                        time_blocked          NMTOKEN #REQUIRED
                        receptor_event        %boolean;>
<!-- ********************************************************************* -->
<!ELEMENT membrane_substance_concentration EMPTY>
<!ATTLIST membrane_substance_concentration
                        link_to_substance_description NMTOKEN #REQUIRED
                        internal_concentration        NMTOKEN #REQUIRED
                        external_concentration        NMTOKEN #REQUIRED>
<!-- ********************************************************************* -->
<!ELEMENT info (transmitter_substance*, receptor_molecule*, openness_block_time_induced*, membrane_substance*)>
<!-- ..................................................................... -->
<!ELEMENT membrane_substance EMPTY>
<!ATTLIST membrane_substance
                        id NMTOKEN #IMPLIED
                        name NMTOKEN #REQUIRED
                        valence NMTOKEN #REQUIRED
                        gain NMTOKEN #REQUIRED>
<!-- ********************************************************************* -->
<!ELEMENT transmitter_substance EMPTY>
<!ATTLIST transmitter_substance
                        id       NMTOKEN #IMPLIED
                        name     NMTOKEN #REQUIRED>
<!-- ********************************************************************* -->
<!ELEMENT receptor_molecule EMPTY>
<!ATTLIST receptor_molecule
                        id       NMTOKEN #IMPLIED
                        name     NMTOKEN #REQUIRED>
<!-- ********************************************************************* -->
<!ELEMENT openness_block_time_induced EMPTY>
<!ATTLIST openness_block_time_induced
                        link_to_receptor_molecule     NMTOKEN #IMPLIED
                        link_to_transmitter_substance NMTOKEN #IMPLIED
                        induced_openness_degree        NMTOKEN #IMPLIED
                        induced_block_time            NMTOKEN #IMPLIED>
<!-- ********************************************************************* -->
<!ELEMENT links (link*)>
<!ELEMENT link EMPTY>
<!ATTLIST link from CDATA #IMPLIED
               to   CDATA #IMPLIED>
<!-- ********************************************************************* -->
```

Appendix D

Implementations examples

In this appendix, examples are given of the possible implementation of the RealNeuron using the C++ computer programming language.

In §D.1 the C++ code is given, i.e. all the necessary header files and the definition files, for the pump.

In §D.2 the example code is given for possible unit tests that could be used in the verification process of the pump implementation.

D.1 C++ Code

D.1.1 species.h

```
namespace Species
{
   enum Ion {Begin, K = Begin, Na, Cl, End};
};
```

D.1.2 RNNException.h

```
#include <exception>
#include <string>
#include <cxxabi.h>
class RNNException{
   std::string m_message;
public:
   RNNException(const std::string &message): m_message(message){};
   RNNException(const std::string &name, const std::string &message)
   {
     int status;
     char * realname = abi::__cxa_demangle(name.c_str(), 0, 0, &status);
     m_message = message + " in " + realname;
     free(realname);
   };
   virtual const char* what() const throw() { return m_message.c_str(); }
};
```

D.1.3 pores.h

```
#include "species.h"
class Pores{
public:
    Pores ( Species::Ion sigma, double gamma );
    virtual ~Pores(){};
protected:
```

```
    double calculateConcentration(double ci, double o /*, dt*/);
    Species::Ion m_sigma;
    double m_gamma;
private:
    Pores();
};
```

D.1.4 pores.cc

```
#include "pores.h"

Pores::Pores( Species::Ion sigma, double gamma ) : m_sigma(sigma), m_gamma(gamma)
{
}
double Pores::calculateConcentration(double ci, double o /*, dt*/)
{
   return ci + o*m_gamma/* * dt*/;
}
```

D.1.5 pumps.h

```
#include "pores.h"
#include "species.h"
#include <vector>
class Pump : public Pores{
public:
    enum Direction {in, out};
    Pump(
         Species::Ion sigma,
         double gamma, /// ionic flow rate mmol/s
 double theta, /// threshold value mmol/l
 Direction phi /*,
 double t0)*/
    );
    double execute(Species::Ion sigma, double ci/*, double t)*/);
protected:
private:
    double m_theta;
    Direction m_phi;
    /* double m_t0)*/
    Pump ( ); /// Make sure that a default construction is not possible
};
typedef std::vector<Pump *> Pumps;
```

D.1.6 pumps.cc

```
#include "pump.h"
#include "RNNException.h"
#include <typeinfo>
Pump::Pump(Species::Ion sigma, double gamma, double theta, Direction phi /*, double t0)*/)
        : Pores(sigma,gamma), m_theta(theta), m_phi(phi)/*, m_t0(t0))*/
{
}
double Pump::execute(Species::Ion sigma, double ci /*, double t)*/)
{
    if(sigma != m_sigma)
        throw RNNException(typeid(this).name() + std::string(":sigma"));
    //dt = t - m_t0
    double o = (ci < m_theta && m_phi == in) ? 1.0 : ( (ci > m_theta && m_phi == out) ? -1.0 : 0.0 );
    //m_t0 = t;
    return calculateConcentration(ci, o /*, dt*/);
}
```

D.2 C++ Unit Test Code

D.2.1 pumpstest.h

```
#include <cppunit/extensions/HelperMacros.h>
```

```
#include "pump.h"
#include "RNNException.h"
/** Unit test for the pump implementation for the RealNeuron
    @author Louwrence D Erasmus
*/
class PumpTest : public CppUnit::TestFixture{
    CPPUNIT_TEST_SUITE( PumpTest );
    CPPUNIT_TEST( testConstructor );
    CPPUNIT_TEST_EXCEPTION( testExecute1, RNNException );
    CPPUNIT_TEST_EXCEPTION( testExecute2, RNNException );
    CPPUNIT_TEST( testExecute3 );
    CPPUNIT_TEST( testExecute4 );
    CPPUNIT_TEST( testExecute5 );
    CPPUNIT_TEST( exit );
    CPPUNIT_TEST_SUITE_END();
    static bool flag;
    Pump *m_pump_in;
    Pump *m_pump_out;
public:
    void setUp();
    void tearDown();
protected:
    void testConstructor();
    void testExecute1();
    void testExecute2();
    void testExecute3();
    void testExecute4();
    void testExecute5();
    void exit();
};
CPPUNIT_TEST_SUITE_REGISTRATION( PumpTest );
```

D.2.2 pumpstest.cc

```
#include "pumptest.h"
bool PumpTest::flag = false;
void PumpTest::exit()
{
    flag = false;
    delete m_pump_in;
    delete m_pump_out;
    CPPUNIT_ASSERT(true);
}
void PumpTest::setUp()
{
    if(flag)
    {
        m_pump_in = new Pump(Species::Na, 100, 200, Pump::in);
        m_pump_out = new Pump(Species::Na, 100, 200, Pump::out);
    }
}
void PumpTest::tearDown()
{
    if(flag)
    {
        delete m_pump_in;
        delete m_pump_out;
    }
    m_pump_in = 0;
    m_pump_out = 0;
}
void PumpTest::testConstructor()
{
    m_pump_in = 0;
    m_pump_out = 0;
    flag = true;
    // Pump *a = new Pump; -> gives a compilation error
    Pump *a = new Pump(Species::Na, 100, 200, Pump::in);
    CPPUNIT_ASSERT( a!=0 );
    delete a;
}
void PumpTest::testExecute1()
{
    m_pump_in->execute(Species::Cl, 100);
}
void PumpTest::testExecute2()
{
    m_pump_out->execute(Species::K, 100);
```

```
}
void PumpTest::testExecute3()
{
    CPPUNIT_ASSERT_DOUBLES_EQUAL(200, m_pump_in->execute(Species::Na, 100), 1e-16);
    CPPUNIT_ASSERT_DOUBLES_EQUAL(100, m_pump_out->execute(Species::Na, 100), 1e-16);
}
void PumpTest::testExecute4()
{
    CPPUNIT_ASSERT_DOUBLES_EQUAL(500, m_pump_in->execute(Species::Na, 500), 1e-16);
    CPPUNIT_ASSERT_DOUBLES_EQUAL(400, m_pump_out->execute(Species::Na, 500), 1e-16);
}
void PumpTest::testExecute5()
{
    CPPUNIT_ASSERT_DOUBLES_EQUAL(200, m_pump_in->execute(Species::Na, 200), 1e-16);
    CPPUNIT_ASSERT_DOUBLES_EQUAL(200, m_pump_out->execute(Species::Na, 200), 1e-16);
}
```

Appendix E

Motivation for the bulb-end soma

In §3.3, §4.2.3, §4.2.5 and §4.4.3, it is mentioned that no biological equivalent for a bulb-end soma exists. In this appendix, a motivation is given for the legacy behind this and to show that the function of the bulb end is not changed by this structural definition.

This legacy is a result of the modelling being driven from the database model of the RealNeuron, which made the object-orientated data model [144] simpler for implementation in Postgresql.[1] This adds unnecessary structural complexity to the model and should be eliminated in a future revision of the RealNeuron model.

If the description of the biological bulb-end is examined in [13, 41, 125, 137, 158], it is seen that the action potential arrives from the axon at the bulb end. If the potential of the bulb end is above a certain threshold then a vesicle releases a neurotransmitter into the synaptic gap.

A bulb-end membrane has pumps and voltage gated channels that restore the steady state potential of the bulb-end, in order to prevent it from releasing neurotransmitters at following time steps. The parameters describing a bulb-end membrane are the same as those in the structure for a main membrane.

From the descriptions in [110], a bulb end can have post-synaptic membranes. The metabolic receptors in the post-synaptic membranes mean that an LTP function can also be implemented for the bulb end.

If all the subcomponents' parameters of the bulb end are evaluated, then it is seen that the parameters similar to the main membrane structure, the parameters for the post-synaptic membranes and the parameters for the LTP function are the same as for a soma.

Looking from a data model perspective at the parameters, and utilising the potential of Postgresql's ability to use object-orientated data models, it follows that the parameters from the bulb end can be saved in such a soma structure.

The bulb-end soma modelling element does not change the functionality of the bulb end.

[1] see http://www.postgresql.org

Appendix F

Simulation Implementation Notes

This appendix describes how the RealNeuron network simulations have been implemented.

The RealNeuron is a hierarchical modular model and is implemented in a bottom-up fashion. The ion concentration, membrane permeability and temperature parameters are from [28].

F.1 Implementing the calculating parts

The following process has been used:

1. The pumps are implemented with (4.77) and (4.77), and use the following parameters:
   - $\gamma_K = 300$, $\gamma_{Na} = 80$, $\gamma_{Cl} = 65$
   - $\theta_K = 345$, $\theta_{Na} = 80$, $\theta_{Cl} = 65$
   - $\phi_K = in$, $\phi_{Na} = out$, $\phi_{Cl} = out$
2. The voltage-gated channels are implemented with (4.82) and (4.86), and use the following parameters:
   - $\gamma_K = 300$, $\gamma_{Na} = 80$, $\gamma_{Cl} = 65$
   - π_K, π_{Na}, and π_{Cl} are chosen to fire at the desired threshold potential in the specific membrane
   - $\phi_K = minimum$, $\phi_{Na} = maximum$, $\phi_{Cl} = maximum$
   - $[K^+]_e = 10$, $[Na^+]_e = 455$, $[Cl^-]_e = 540$
3. The receptor-gated channels are implemented with (4.82), (4.84) and (4.85), and use the following parameters:
   - $\gamma_K = 300$, $\gamma_{Na} = 80$, $\gamma_{Cl} = 65$
   - κ_K, κ_{Na}, and $\kappa_{Cl} = maximum$ are set to the desired neurotransmitter names
   - t_b is set to the desired block time

- $o_b = 1$
- $[K^+]_e = 10$, $[Na^+]_e = 455$, $[Cl^-]_e = 540$

4. More than one metabolic receptor can be implemented in the same post-synaptic membrane. The metabolic receptors are implemented with (4.90) and (4.91), and use the following parameters:

   - t_b is set to the desired block time
   - τ_R is set to the desired neurotransmitter names

5. The Nernst and Goldmann equations are implemented with (3.7) and use the following parameters:

   - $T = 279.45$
   - $P_K = 1.8$, $P_{NA} = 0.07$ and $P_{Cl} = 0.8$
   - $[K^+]_e = 10$, $[Na^+]_e = 455$, $[Cl^-]_e = 540$

6. More than one vesicle can be implemented in the same bulb end. The vesicle is implemented with (4.67) and uses the following parameters:

   - τ_V is set to the desired neurotransmitter names
   - pi_V is set to the desired threshold potential

7. the rest of the components are implemented as described in chapter 4.

F.2 The nucleus interpositus cell's implementation

The nucleus interpositus cell's post-synaptic membrane, that receives input from N3, is configured in the following way:

1. The pumps are implemented with (4.77) and (4.77), and use the following parameters:

   - $\gamma_K = 3000$, $\gamma_{Na} = 800$, $\gamma_{Cl} = 650$
   - $\theta_K = 345$, $\theta_{Na} = 80$, $\theta_{Cl} = 65$
   - $\phi_K = in$, $\phi_{Na} = out$, $\phi_{Cl} = out$

2. The voltage-gated channels are implemented with (4.82) and (4.86), and use the following parameters:

   - $\gamma_K = 3000$, $\gamma_{Na} = 800$, $\gamma_{Cl} = 605$
   - π_K, π_{Na}, and π_{Cl} are chosen to fire at the desired threshold potential in the specific membrane
   - $\phi_K = minimum$, $\phi_{Na} = maximum$, $\phi_{Cl} = maximum$
   - $[K^+]_e = 10$, $[Na^+]_e = 455$, $[Cl^-]_e = 540$

F.3 The Purkinje cell's implementation

The Purkinje cell is a special cell and holds the key to the classical conditioning implementation. The LTP function inside the neuron gives the Purkinje cell its short-term memory. Two LTP functions are implemented in this cell. The one LTP implements the time window in which the two input signals should appear to remember the association, and the second LTP implements the memory. This is achieved by configuring two receptors at each post-synaptic membrane. The first receptor has a block time t_b equal to the desired time window size. The time window LTP function then emits a specific neurotransmitter in the synaptic gaps to activate the short-term memory. The second receptor has a block time t_b equal to the desired short term memory length and is activated by the specific neurotransmitter emitted by the time window LTP function. This puts the Purkinje cell into a conditioned state.

F.4 General notes

It is important to make sure that the amount of ions that flow into a cell is equal to the amount that flows out of the cell for a specific period. This is the equivalent of the mass balance equation that needs to be adhered to in various disciplines of science and engineering.

Appendix G

CD-ROM Contents

The enclosed CD-ROM contains the following:

```
.
|-- knowboticsystems
|   |-- datenmodell-inmneuron.ps
|   |-- inmneuron-fragen.pdf
|   `-- inmneuron-spezifik.pdf
|-- lde-thesis.pdf
|-- list.txt
|-- papers
|   |-- africon07-lderasmus2.pdf
|   |-- africon07-lderasmus.pdf
|   |-- A model of the Real Neuron - Erasmus.pdf
|   |-- Part I White Paper.pdf
|   |-- Physica-verlag.PDF
|   |-- Real Neurons, Artificial - Erasmus.pdf
|   |-- doeben-henisch_introrealneuron.pdf
|   `-- eos.pdf
|-- presentations
|   |-- 28.2.2003-PUKEEI.pdf
|   |-- africon07-lderasmus-RN-classical-conditioning.pdf
|   |-- africon07-lderasmus-RN-SN-compare.pdf
|   |-- A Model of the RealNeuron.pdf
|   |-- INCOSE.pdf
|   |-- LDE-AFRICON04.pdf
|   |-- Research Proposal for a Ph.D-L.D.Erasmus.pdf
|   |-- mainzer KI.txt
|   `-- rnnpresentation.pdf
|-- simulationReports
|   |-- BulbEnd-biological-sinus.pdf
|   |-- MainMembrane-biological-sinus.pdf
|   |-- MainMembrane.pdf
|   |-- PoisonAndFoodGame-blue-and-red-balls.pdf
|   |-- PostsynapticMembrane.pdf
|   |-- Soma-biological.pdf
|   |-- and2.pdf
|   |-- buffer.pdf
|   |-- duckbug2.pdf
```

```
|   |-- dugbug-xml-description.pdf
|   |-- inverter.pdf
|   |-- motor.pdf
|   |-- nand.pdf
|   |-- nor.pdf
|   |-- nucleusInterpositus.pdf
|   |-- or.pdf
|   |-- purkinje.pdf
|   |-- sensor.pdf
|   |-- spreadsheets
|   |   |-- BulbEnd-biological-sinus.ods
|   |   |-- MainMembrane-biological-sinus.ods
|   |   |-- MainMembrane.ods
|   |   |-- PoisonAndFoodGame-blue-and-red-balls.ods
|   |   |-- PoisonAndFoodGame-blue-ball.ods
|   |   |-- PoisonAndFoodGame-red-ball.ods
|   |   |-- PostsynapticMembrane.ods
|   |   |-- Soma-biological.ods
|   |   |-- and2.ods
|   |   |-- buffer.ods
|   |   |-- duckbug2.ods
|   |   |-- inverter.ods
|   |   |-- motor.ods
|   |   |-- nand.ods
|   |   |-- nor.ods
|   |   |-- nucleusInterpositus.ods
|   |   |-- or.ods
|   |   |-- purkinje.ods
|   |   |-- sensor.ods
|   |   `-- xor.ods
|   `-- xor.pdf
`-- workCitedByOther
    `-- neural_networks.pdf
```

Bibliography

[1] *Proceedings of the 1998 IEEE International Symposium on Intelligent Control Held jointly with the IEEE International Symposium on Computational Intelligence in Robotics and Automation and the Intelligent Systems and Semiotics, A joint Conference on the Science and Technology of Intelligent Systems.* IEEE, Sept.14 – 17 1998.

[2] L. Abbot and T. Kepler, *Model neurons: from Hodgkin-Huxley to Hopfield.* Berlin: Springer, 1990.

[3] J. S. Albus, "4-D/RCS: A reference model architecture for Demo III," U.S. Department of Commerce, Technology Demonstration, National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NISTIR 5994, Mar. 1997.

[4] J. S. Albus and A. Meystel, "Behaviour generation in intelligent systems," U.S. Department of Commerce, Technology Demonstration, National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NISTIR 6083, Oct. 1997.

[5] J. Albus, "A theory of cerebellar function," *Mathematical Biosciences*, vol. 10, pp. 25–61, 1971.

[6] ——, "Outline for a theory of intelligence," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 473–509, 1991.

[7] American Psychological Association, "Award for distinguished scientific contributions: Allan R. Wagner," *American Psychologist*, vol. 54, no. 11, pp. 887–890, Nov. 1999.

[8] W. Balzer, *Empirische Theorien: Modelle, Strukturen, Beispiele.* Braunschweig - Wiesbaden: Fr.Viehweg & Sohn, 1982.

[9] W. Balzer, C. Moulines, and J. Sneed, *An Architectonic for Science.* Dordrecht: The Structuralist Program, 1987.

[10] S. Barnett and R. Cameron, *Introduction to Mathematical Control Theory*, 2nd ed., ser. Oxford Applied Mathematics and Computing Science Series, J. Buxton, R. Churchhouse, and A. Taylor, Eds. Oxford: Oxford University Press, 1985.

[11] B. Blanchard and W. Fabrycky, *Systems Engineering and Analysis*, 3rd ed., ser. Prentice Hall International Series in Industrial and Systems Engineering. Englewood Cliffs, NJ: Prentice Hall, Inc., 1998.

[12] G. Bloch, F. Sirou, V. Eustache, and P. Fatrez, "Neural intelligent control for a steel plant," *IEEE Transactions on Neural Networks*, vol. 8, no. 4, pp. 910–918, July 1997.

[13] F. E. Bloom and A. Larsen, *Brain, Mind and Behaviour*, 2nd ed. New York: An Annenberg/CPB Project, W.H. Freeman and Company, Worth Publishers, 1999, 7th print.

[14] G. Boole, *The Laws of Thought*. Cambridge: MacMillen, 1854.

[15] N. Bourbaki, *Theorie des Ensembles*, ser. Elements de Mathematique. Paris: Hermann, 1970.

[16] J. M. Bower and D. Beeman, *The Book of GENESIS, Exploring Realistic Neural Models with the GEneral NEural SImulation System*, 2nd ed. New York: Springer-Verlag TELOS, 1998.

[17] N. T. Carnevalea, T. B. Woolf, and G. M. Shepherd, "Neuron simulations with SABER," *Journal of Neuroscience Methods*, vol. 33, no. 2-3, pp. 135–148, Aug. 1990.

[18] P. Chauvet and G. Chauvet, "Purkinje local circuits with delays: mathematical conditions of stability for learning and retrieval," *Neural Networks*, vol. 12, pp. 59–77, 1999.

[19] T. Chen and H. Chen, "Universal approximation to non-linear operators by neural networks with arbitrary activation functions and its application to dynamical systems," *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 911–917, 1995.

[20] G. A. Clark, R. D. Hawkins, and W. N. Frost, "How neural are neural networks? A comparison of information processing and storage in artificial and real neural systems," *Journal of Statistical Planning and Inference*, vol. 33, no. 1, pp. 27–65, Oct. 1992.

[21] D. Cohen, *Introduction to Computer Theory*, 2nd ed. New York: John Wiley & Sons, Inc., 1997.

[22] A. D. Coop and G. N. Reeke, "Simulating the temporal evolution of neuronal discharge," *Neurocomputing*, vol. 32-33, pp. 91–96, 2000.

[23] B. J. Copeland. "The Turing Test," *Minds and Machines*, vol. 10, pp. 519–539, 2000.

[24] B. Copeland, *A lecture and two radio broadcasts by Alan Turing*. Oxford University Press, 1999.

[25] F. Crick, *The Astonishing Hypothesis: The Scientific Search for the Soul*. New York: Touchstone, 1994.

[26] R. Damper, R. French, and T. Scutt, "The Hi-NOON neural simulator and its applications," *Microelectronics Reliability*, vol. 41, pp. 2051–2065, 2001.

[27] J. M. Delgado-Garcia and A. Gruart, "Firing activities of identified posterior interpositus nucleus neurons during associative learning in behaving cats," *Brain Research Reviews*, vol. 49, pp. 367–376, 2005.

[28] S. Deutsch and A. Deutsch, *Understanding the Nervous System, An Engineering Approach*. New York: IEEE Press, 1993, IEEE Engineering in Medicine and Biology Society, Sponsor.

[29] K. Diamantaras and S. Kung, *Principal Component Neural Networks, Theory and Applications*. New York: John Wiley & Sons, Inc., 1996.

[30] G. Döben-Henisch, "Taxonomy for INM-Neuron," Intitut für Neue Medien, Frankfurt am Main, Germany, 1998. [Online]. Available: http://www.inm.de/kip/kip.html

[31] ——, "Inverse Nernst and inverse Goldmann," Knowbotic Systems GmbH & Co KG, Frankfurt am Main, Germany, Tech. Rep., Aug.3 1999.

[32] ——, "Processing order of membranes," Knowbotic Systems GmbH & Co KG, Frankfurt am Main, Germany, Tech. Rep., Aug.3 1999.

[33] ——, "Computational neurosemiotics and learning," June 2000, personal notes.

[34] ——, "Computational neurosemiotics: Methodological issues," presented at the Workshop on Computational Intelligence and Semiotics. Campinas, SP, Brazil: Faculdade de Engenharia Elétrica e de Computação, May 2000. [Online]. Available: http://www.dca.fee.unicamp.br/projects/semiotics/TechPages/Workshop2000/index.html

[35] ——, "KS-Theory - Learning: Basic theory," Jan.16 2000, personal notes.

[36] G. Döben-Henisch, L. D. Erasmus, R. Goldschmitt, and R. Stuckardt, "inm-Neuron version 0.2, work in progress," Knowbotic Systems GmbH & Co KG, Frankfurt am Main, Germany, Tech. Rep., July15 1999.

[37] G. Döben-Henisch and S. Höppner, "Defining agency," in 3rd International Conference on Interactive Systems IS-99, Ulyanovsk, Russia, Sept. 1999.

[38] G. Döben-Henisch. Knowbotic Interface Project. Institut fuer Neue Medien. Frankfurt am Main, Germany. [Online]. Available: http://www.inm.de/kip/kip.html

[39] E. O. Doebelin, *Measurement Systems: Applications and Design*, 4th ed., ser. Mechanical Engineering Series. McGraw-Hill, Inc., 1990, International Edition.

[40] J. Dudel, R. Menzel, and R. F. Schmidt, Eds., *Neurowissenschaft Vom Molekül zur Kognition.* Berlin Heidelberg New York: Springer-Verlag, 1996.

[41] ——, *Neurowissenschaft Vom Molekül zur Kognition*, 2nd ed. Berlin Heidelberg New York: Springer-Verlag, 2001.

[42] R. Eberhart, P. Simpson, and R. Dobbins, *Computational Intelligence PC Tools.* Boston San Diego New York: AP Professional, 1996.

[43] U. Eco, Ed., *A Theory of Semiotics.* Bloomington, USA: Indiana University Press, 1979.

[44] H. Eisner, *Essentials of Project and Systems Engineering Management*, 2nd ed. New York: John Wiley & Sons, Inc., 2002.

[45] P. Ellappan and R. Sundararajan, "A simulation study of the electrical model of a biological cell," *Journal of Electrostatics*, vol. 63, pp. 297–307, 2005.

[46] "The brain, brain cells," Enchanted Learning, Mar. 2007. [Online]. Available: http://www.enchantedlearning.com

[47] L. D. Erasmus, "Comparing the RealNeuron with spiking neurons," in *IEEE Africon 2004*, Windhoek, Namibia, Sept.26-28 2007.

[48] ——, "Experiences and insights gained from the application of systems engineering principles in the formulation and evaluation of a neuron model based on biological neurons," submitted for publication in INCOSE South African Chapter Conference, CSIR, Pretoria, South Africa, Aug.28-30 2007.

[49] ——, "RealNeuron implementation of classical conditioning," submitted for publication in IEEE Africon 2007, Windhoek, Namibia, Sept.26-28 2007.

[50] L. D. Erasmus and C. P. Bodenstein, "RealNeurons, artificial intelligence based on the mechanisms of biological intelligence," in *IEEE Africon 2004*, vol. 1. Gaborone, Botswana: IEEE Region 8, SAIEE, BIE and the University of Botswana, Sept.15-17 2004, pp. 423 – 428.

[51] L. D. Erasmus, G. Döben-Henisch, and C. P. Bodenstein, "A model of the RealNeuron," in *IEEE Africon 2004*, vol. 2. Gaborone, Botswana: IEEE Region 8, SAIEE, BIE and the University of Botswana, Sept.15-17 2004, pp. 1071 – 1078.

[52] L. Erasmus, "Numerical solution of the population balance equation," Master's thesis, Department of Mathematics and Applied Mathematics, Potchefstroom University for Christian Higher Education, Potchefstroom, South Africa, 1993.

[53] R. FitzHugh, "Impulses and physiological states in models of nerve membrane," *Biophys.J.*, vol. 1, pp. 445–446, 1961.

[54] T. Floyd, *Digital Fundamentals*, 3rd ed. Columbus, Ohio, USA: Charles E. Merrill Publishing Company, 1986.

[55] D. K. Frederick and C. M. Close, "Block diagrams," in *The Control Handbook*, ser. The Electrical Engineering Handbook Series, W. S. Levin, Ed. Boca Raton, FL: IEEE Press and CRC Press, 1996.

[56] W. Freeman, *How brains make up their minds*, ser. Maps of the Mind. London: Weidenfeld and Nicolson, 1999.

[57] K. Fukushima, "A neural network for visual pattern recognition," *IEEE Computer*, vol. 21, no. 3, pp. 65–75, 1988.

[58] W. Gerstner, "Spiking neurons," in *Pulsed Neural Networks*, W. Dudel and C. Bishop, Eds. Cambridge: MIT Press, 1999, ch. 1, pp. 3 – 53.

[59] J. T. Gillis, "State space," in *The Control Handbook*, ser. The Electrical Engineering Handbook Series, W. S. Levin, Ed. Boca Raton, FL: IEEE Press and CRC Press, 1996.

[60] N. Goddard and N. Hood, "Parallel GENESIS for large scale modeling," in *Computational Neuroscience '96*, J. M. Bower, Ed. New York: Plenum, 1997.

[61] R. Gonçalves, F. Gomide, and R. Gudwin, "Fielded object networks as a framework for computer intelligence," in *Proceedings of the 1998 ISIC/CIRA/ISAS, A Joint Conference on the Science and Technology of Intelligent Systems*, Gaithersburg, MD, USA, Sept. 1998.

[62] A. Guyton, *Function of the human body.* Philadelphia: W.B. Saunders Company, 1974.

[63] S. Hakimian, C. H. Anderson, and W. T. Thach, "A PDF model of populations of Purkinje cells: Non-linear interactions and high variability," *Neurocomputing*, vol. 26-27, pp. 169–175, 1999.

[64] J. Hasebrook, L. Erasmus, and G. Döben-Henisch, "Knowledge robots for knowledge workers: Self learning agents connecting information and skills," in *Intelligent Agents and their applications*, ser. Springer series: Studies in Fuzziness and Soft Computing, L. Jain, Z. Chen, and N. Ichalkaranje, Eds. Heidelberg: Physica-verlag, 2002, vol. 98, ch. 3, pp. 59–79.

[65] ——, "Knowledge robots for knowledge workers: Self learning agents connecting information and skills," in *Proc. of the 6th International Conference on Knowledge-based Intelligent Information and Engineering Systems (KES'2002)*, E. Damiani et.al., Ed. Amsterdam, The Netherlands: IOS Press, Sept.16 – 18 2002, pp. 572–581, Podere d'Ombriano, Crema, Italy.

[66] J. Hasebrook and W. Nathusius, "An expert advisor for vocational guidance," *Journal of Artificial Intelligence in Education*, vol. 8, no. 1, pp. 21–41, 1997.

[67] S. Haykin, *Neural Networks, A Comprehensive Foundation.* New York: IEEE Computer Society Press and IEEE Press, 1994.

[68] ——, *Neural Networks, A Comprehensive Foundation*, 2nd ed. Upper Saddle River, New Jersey: Prentice Hall, 1999.

[69] E. Hilgard, R. Atkinson, and R. Atkinson, *Introduction to Psychology.* Harcourt Brace Jovanovich Inc., 1979.

[70] M. Hines, *NEURON.* Dordrecht: Kluwer Academic Publishers, 1994.

[71] M. Hinesa and N. Carnevale, "Discrete event simulation in the NEURON environment," *Neurocomputing*, vol. 58-60, pp. 1117–1122, 2004.

[72] A. L. Hodgkin and A. F. Huxley, "A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes," *Journal of Physiology*, vol. 117, pp. 500–544, 1952.

[73] A. L. Hodgkin and B. Katz, "The effect of sodium ions on the electrical activity of the giant axon of the squid," *Journal of Physiology*, vol. 108, pp. 37–77, 1949.

[74] P. Husbands, "Evolving robot behaviours with diffusing gas networks," in *EvoRobots*, ser. Lecture Notes in Computer Science, P. Husbands and J.-A. Meyer, Eds., vol. 1468. Springer, 1998, pp. 71–86.

[75] P. Husbands, T. Smith, N. Jakobi, and M. O'Shea, "Better living through chemistry: Evolving GasNets for robot control," *Connect. Sci.*, vol. 10, no. 3-4, pp. 185–210, 1998.

[76] IEE, "Neural networks," The IEE Inspec, INSPEC Subject Updates 2005-006, Feb. 2005.

[77] INCOSE, *Systems Engineering Handbook, A How to Guide For All Engineers, Version 2.0.* INCOSE, July 2000.

[78] *IEEE Standard Glossary of Software Engineering Terminology*, The Institute for Electrical and Electronics Engineers, Inc. Std. 610.12, 1990.

[79] *Industry Implementation of International Standard ISO/IEC 12207:1995, (ISO/IEC12207) Standard for Information Technology - Software life cycle processes*, The Institute for Electrical and Electronics Engineers, Inc. Std. 12207.0, 1996.

[80] *IEEE Guide for Developing System Requirements Specifications*, The Institute for Electrical and Electronics Engineers, Inc. Std. 1220, 1998.

[81] *IEEE Standard for Application and Management of the Systems Engineering Process*, The Institute for Electrical and Electronics Engineers, Inc. Std. 1233, 1998.

[82] *IEEE Standard for Software Verification and Validation Plans*, The Institute for Electrical and Electronics Engineers, Inc. Std. 1012, 1998.

[83] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, pp. 1569– 1572, 2003.

[84] ——, "Which model to use for cortical spiking neurons?" *IEEE Transactions on Neural Networks*, vol. 15, pp. 1063–1070, 2004.

[85] P. Jackson, *Introduction to Expert Systems*. Addison-Wesley, 1990.

[86] P. Johnston, F. Henskens, and W. McGowan, "NEU-MODEL: A multi-level object-oriented dynamic emulation laboratory," *Neurocomputing*, vol. 38-40, pp. 1671–1677, 2001.

[87] J. J. Kim and R. F. Thompson, "Cerebellar circuits and synaptic mechanisms involved in classical eyeblink conditioning," *Trends in Neurosciences*, vol. 20, no. 4, pp. 177–181, Apr. 1997.

[88] W. Kistler, W. Gerstner, and J. van Hemmen, "Reduction of Hodgkin-Huxley equations to a single-variable threshold model," *Neural Computation*, vol. 9, pp. 1015–1045, 1997.

[89] B. Kosko, *Neural Networks and Fuzzy Systems*. Upper Saddle River, New Jersey: Prentice Hall, 1992.

[90] ——, *Neural Networks for Signal Processing*. Upper Saddle River, New Jersey: Prentice Hall, 1992.

[91] S. Kotani, S. Kawahara, and Y. Kirino, "Purkinje cell activity during learning a new timing in classical eyeblink conditioning," *Brain Research*, vol. 994, pp. 193–202, 2003.

[92] ——, "Purkinje cell activity during classical eyeblink conditioning in decerebrate guinea pigs," *Brain Research*, vol. 1068, pp. 70–81, 2006.

[93] B. Kröese and P. van der Smagt, *An Introduction to Neural Networks*, 5th ed. Amsterdam: University of Amsterdam, 1983.

[94] D. Kunkel, "57352c: Excitatory synapse from the central nervous system," Dennis Kunkel Microscopy, Inc., PO Box 2008, Kailua, HI 96734, USA, 2004. [Online]. Available: http://www.denniskunkel.com

[95] ——, "8418b: Neurons from the central nervous system growing on a glial / fibroblast matrix." Dennis Kunkel Microscopy, Inc., PO Box 2008, Kailua, HI 96734, USA, 2004. [Online]. Available: http://www.denniskunkel.com

[96] A. G. Kuo, G. Lee, B. M. McKay, and J. F. Disterhoft, "Enhanced neuronal excitability in rat CA1 pyramidal neurons following trace eyeblinknext term conditioning acquisition is not due to alterations in i_m," *Neurobiology of Learning and Memory*, In Press. Corrected Proof. Available online 20 August 2007.

[97] J. D. Lambert, *Computational Methods in Ordinary Differential Equations.* John Wiley and Sons, 1990.

[98] R. Lippman, "An introduction to computing with neural networks," *IEEE ASSP Magazine*, pp. 4–22, Apr. 1987.

[99] S. Lipshutz, *Schaum's Outline of Theory and Problems of Set Theory and Related Topics*, ser. Schaum's Outline Series. McGraw-Hill Company, 1964.

[100] G. Lucas, "Star Wars Episode I: The Phantom Menace," Lucas Films Ltd., 2001, DVD-ROM Movie.

[101] G. Ludwig, *Die Grundstrukturen einer physikalischen Theorie.* Berlin Heidelberg New York: Springer, 1978.

[102] W. Maass, "Lower bounds for the computational power of networks of spiking neurons," *Neural Computation*, vol. 8, no. 1, pp. 1–40, 1996.

[103] ——, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[104] W. Maass and C. M. Bishop, *Pulsed Neural Networks.* Cambridge: MIT Press, 1999.

[105] D. Marr, "A theory of cerebellar cortex," *Journal of Physiology*, vol. 202, pp. 437–470, 1969.

[106] A. Maslow, *Towards a Psychology of Being.* D. Van Nostrand Company, 1968.

[107] S. R.-L. Masson, G. L. Masson, L. Alvado, S. Saïghi, and J. Tomas, "A neural simulation system based on biologically realistic electronic neurons," *Information Sciences*, vol. 161, pp. 57–69, 2004.

[108] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[109] R. Menzel, *Neuronale Plastizität, Lernen und Gedächtnis.* Berlin Heidelberg New York: Springer-Verlag, 1996, pp. 485 – 518.

[110] ——, *Neuronale Plastizität, Lernen und Gedähtnis*, 2nd ed. Berlin Heidelberg New York: Springer-Verlag, 2001, ch. 23, pp. 487 – 526.

[111] C. Meunier and I. Segev, "Playing the Devil's advocate: Is the Hodgkin-Huxley model useful?" *Trends in Neurosciences*, vol. 25, no. 11, pp. 558–563, Nov. 2002.

[112] A. Meystel, "Part I: Measuring performance and intelligence of systems with autonomy, the white paper," NIST, 2000. [Online]. Available: http://www.nist.gov

[113] ——, "Multiresolutional representation and behavior generation: How do they affect the performance of intelligent systems," NIST, 2001. [Online]. Available: http://www.isd.mel.nist.gov

[114] S. Mojtahedian, D. Kogan, S. Kanzawa, R. Thompson, and D. Lavond, "Dissociaton of conditioned eye and limb responses in the cerebellar interpositus," *Physiology and Behavior*, vol. 91, no. 1, pp. 9–14, May 2007.

[115] A. B. Moore and C. Murphy, "A demonstration of classical conditioning of the human eyeblink to an olfactory stimulus," *Physiology and Behavior*, vol. 66, no. 4, pp. 689–693, 1999.

[116] J. H. Moore, "An analysis of the Turing Test," *Philosophical Studies*, vol. 30, pp. 249–257, 1976.

[117] J. Moore, Ed., *The Turing Test, The elusive standard of artificial intelligence*, ser. Studies in Cognitive Systems. Dordrecht: Kluwer Academic Publishers, 2003.

[118] J. Nagumo, S. Arimoto, and S. Yoshizawa, "An active pulse transmission line simulating nerve axon," *Proc. Institute of Radio Engineers*, vol. 50, pp. 2061–2070, 1962.

[119] N. S. Nise, "Signal-flow graphs," in *The Control Handbook*, ser. The Electrical Engineering Handbook Series, W. S. Levin, Ed. Boca Raton, FL: IEEE Press and CRC Press, 1996.

[120] M. Nørgaard, O. Raven, N. Poulsen, and L. Hansen, *Neural Networks for Modelling and Control of Dynamic Systems*. London: Springer-Verlag, 2000.

[121] B. B. Oswalda, S. A. Maddoxa, and D. A. Powell, "Impairments in trace EB conditioning by knife-cut lesions to the fornix in rabbits: next term reversal by galantamine," *Neurobiology of Learning and Memory*, vol. 88, no. 3, pp. 369–380, October 2007.

[122] X. Pan and B. Livneh, "Neural networks to predict and control pyrometallurgical processes," *Heavy Minerals 1997*, pp. 231–236, 1997.

[123] F. Pasemann, "A simple chaotic neuron," *Physica D*, vol. 104, pp. 205–211, 1997.

[124] I. Pavlov, *Lectures on Conditioned Reflexes*. New York: International Publishers, 1967.

[125] H. Penzlin, *Lehrbuch der Tierphysiologie*, 6th ed. Jena, Germany: Gustav Fischer Verlag, 1996.

[126] B. Porr and F. Worgotter, "Predictive learning in rate-coded neuronal networks: a theoretical approach towards classical conditioning," *Neurocomputing*, vol. 44-46, pp. 585–590, 2002.

[127] J. Rinzel and B. Ermenhout, *Analysis of neural excitability and oscillations*. Cambridge: MIT Press, 1989.

[128] J. L. Rosenberg, *Theory and Problems of College Chemistry*, 5th ed., ser. Schaum's Outline Series. New York: McGraw-Hill Book Company, Inc., 1972.

[129] D. Rumelhart and J. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA.: MIT Press, 1986, vol. 1.

[130] S. M. Sait and H. Youssef, *Iterative Computer Algorithms with Applications in Engineering, Solving Combinatorial Optimization Problems*. Los Altimos, California: IEEE Computer Society Press, 1999.

[131] C. L. R. Santos, P. P. B. de Oliveira, and P. Husbands, "Three case studies of the GasNet model in discrete domains," *Int. J. Neural Syst.*, vol. 11, no. 3, pp. 295–304, 2001.

[132] C. L. R. Santos, C. R. Souza, P. P. B. de Oliveira, and P. Husbands, "Using the GasNet model in discrete domains," in *SBRN*, 2000, pp. 231–236.

[133] A. P. Saygin, I. Cicekli, and V. Akman, "Turing Test: 50 years later," *Minds and Machines*, vol. 10, no. 4, pp. 463–518, 2000.

[134] T. Sebeok, L. Zadeh, A. Meystel, J. Fetzer, M. Danesi, J. Goguen, and J. Albus, "Intelligent systems: The degrees of Mind," Tutorial 1 of the 1998 ISIC/CIRA/ISAS, A Joint Conference on the Science and Technology of Intelligent Systems, Gaithersburg, MD, USA, Sept. 1998.

[135] I. Segev, "Single neurone models: Over simple, complex and reduced," *Trends in Neurosciences*, vol. 15, no. 11, pp. 414–421, Nov. 1992.

[136] M. Shell, "IEEEtran homepage on CTAN," 2002. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/supported/IEEEtran/

[137] G. M. Shepherd, *Neurobiology*, 3rd ed. New York: Oxford University Press, 1996.

[138] J. Sinclair, Ed., *Collins Cobuild Advanced Learner's Dictionary, English Dictionary.* Glasgow: Harper Collins Publishers, 2003.

[139] R. G. Smith, "Partial reinforcement of the classically conditioned eyelid response in the rabbit," *Journal of Neuroscience Methods*, vol. 43, no. 2-3, pp. 83–108, July 1992.

[140] P. R. Solomon, D. Flynn, J. Mirak, M. Brett, N. Coslov, and M. Grocci, "Five-year retention of the classically conditioned eyeblink response in young adult, middle-aged, and older humans," *Psychology and Aging*, vol. 13, no. 2, pp. 186–192, June 1998.

[141] V. Steuber and E. D. Schutter, "Rank order decoding of temporal parallel fibre input patterns in a complex Purkinje cell model," *Neurocomputing*, vol. 44-46, pp. 183–188, 2002.

[142] V. Steuber and D. J. Willshaw, "Adaptive leaky integrator models of cerebellar Purkinje cells can learn the clustering of temporal patterns," *Neurocomputing*, vol. 26-27, pp. 271–276, 1999.

[143] J. Strout, "CONICAL: The computational neuroscience class library," 1996. [Online]. Available: http://cajalnt.ucsd.edu/jstrout/conical/

[144] R. Stuckardt, "Database model for the INM-Neuron Network Simulator (draft)," Knowbotic Systems GmbH & Co KG, Frankfurt am Main, Germany, Tech. Rep., Sept.8 1999, Theory Group.

[145] ——, "Specification of the INM-Neuron, networks of INM-Neurons, and the simulation algorithm. version 1," spezifik-inm-n.ps, Knowbotic Systems GmbH & Co KG, Frankfurt am Main, Germany, Tech. Rep., Aug.19 1999, Theory Group.

[146] F. Suppe, Ed., *The Structure of Scientific Theories*, 2nd ed. Urbana Chicago London: University of Illinois Press, 1979.

[147] E. Thomas and A. R. Wagner, "Partial reinforcement of the classically conditioned eyelid response in the rabbit," *Journal of Comparative and Psychological Psychology*, vol. 58, pp. 157–158, 1964.

[148] J. A. Tracy, J. K. Thompson, D. J. Krupa, and R. F. Thompson, "Evidence of plasticity in the pontocerebellar conditioned stimulus pathway during classical conditioning of the eyeblink response in the rabbit," *Behavioral Neuroscience*, vol. 112, no. 2, pp. 267–285, Apr. 1998.

[149] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, pp. 433–460, 1950.

[150] ——, *Intelligent machinery.* Edinburgh University Press, 1969.

[151] ——, *Lecture to the London Mathematical Society on 20 February 1947.* Cambridge, Mass.: MIT Press, 1986.

[152] ——, *Proposal for Development in the Mathematics Division of an Automatic Computing Engine (ACE).* Cambridge, Mass.: MIT Press, 1986.

[153] M. G. Velarde, V. I. Nekorkin, V. B. Kazantsev, V. I. Makarenko, and R. Llinas, "Modeling inferior olive neuron dynamics," *Neural Networks*, vol. 15, pp. 5–10, 2002.

[154] P. F. Verschure and M. Mintz, "A real-time model of the cerebellar circuitry underlying classical conditioning: A combined simulation and robotics study," *Neurocomputing*, vol. 38-40, pp. 1019–1024, 2001.

[155] G. Vogel and H. Angermann, *dtv-Atlas zur Biologie Tafeln und Texte, Band I*, 17th ed. München, Germany: Deutschen Taschenbuch Verlag Gmbh & Co. KG, Apr. 1982.

[156] ——, *dtv-Atlas zur Biologie Tafeln und Texte, Band II*, 16th ed. München, Germany: Deutschen Taschenbuch Verlag Gmbh & Co. KG, Apr. 1982.

[157] B. Widrow and M. C. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, Sept. 1990.

[158] S. Wolfe, *Cell and Molecular Biology.* Belmont, USA: Wadsworth Publishing Company, 1995.

[159] S. R. y Cajal, "Drawing of Purkinje cells (A) and granule cells (B) from pigeon cerebellum," Drawing, Madrid, Spain, 1899.

[160] S. Zettel, *Fool's War.* New York, NY: Warner Books, Inc., 1997.

[161] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A neural model of timed response learning in the cerebellum," *Neural Networks*, vol. 7, no. 6-7, pp. 1101–1114, 1994.