KTH Computer Science
and Communication

Automated Inference of Excitable Cell Models as Hybrid Automata

RASMUS ANSIN, RANSIN@KTH.SE
DIDRIK LUNDBERG, DIDRIKL@KTH.SE

Bachelor's Thesis, SA104X
Royal Institute of Technology
School of Computer Science and Communication
Supervisor: Karl Meinke
Examiner: Mårten Olsson

Abstract

In this paper, we explore from an experimental point of view the possibilities and limitations of the new HYCGE learning algorithm for hybrid automata. As an example of a practical application, we study the algorithm's performance on learning the behaviour of the action potential in excitable cells, specifically the Hodgkin-Huxley model of a squid giant axon, the Luo-Rudy model of a guinea pig ventricular cell, and the Entcheva model of a neonatal rat ventricular cell. The validity and accuracy of the algorithm is also visualized through graphical means.

Sammanfattning

I denna uppsats undersöker vi från en experimentell synvinkel möjligheter och begränsningar i den nya inlärningsalgoritmen HYCGE för hybridautomater. Som ett exempel på en praktisk tillämpning, studerar vi algoritmens förmåga att lära sig aktionspotentialens beteende i retbara celler, specifikt Hodgkin-Huxleymodellen av en bläckfisks jätteaxon, Luo-Rudymodellen av en ventrikulär cell i marsvin, och Entchevas modell av en ventrikulär cell i nyfödd råtta. Giltigheten och noggrannheten hos algoritmen visualiseras även genom grafiska medel.

Contents

Chapter 1

Introduction

Systems theory is the science of self-regulating systems. Although the definitions of the terms had not yet been made, applied systems theory or cybernetics could be seen in mechanical and electrical engineering already during the early 20th century. While the Baltic German JAKOB VON UEXKÜLL provided an early perspective on systems theory from a biologist's point of view in his *Theoretische Biologie* from 1920, it was not until the American mathematician NORBERT WIENER wrote the seminal *Cybernetics or Control and Communication in the Animal and the Machine* in 1948 that systems theory was moved from a philosophical level of discussion and formalized in a mathematical language.

During the late 20th century, the increasing prevalence of computers has made computational learning of systems an interesting and viable field of study. Here, it is crucial which representation of the system you choose. For computational reasons, a hybrid automaton representation is useful, since this representation will not have a problem with transitions between different system behaviours - fitting a set of differential equations to a system with typical discontinuous behaviour can be very resource-consuming. In addition, hybrid automata would also more suited for model checking, due to their shared characteristics with finite state automata. As a byproduct, the discretization of fundamentally different system behaviours also creates a natural qualitative understanding of the system.

In this paper, we will discuss automated inference of cyber-physical systems by a new algorithm. More specifically, we will study models of excitable cell behaviour. This could lead to very interesting applications in the future. Primarily, automated learning of cell behaviour would likely drastically reduce the time it takes to construct these models through manual measurements. Worthy of note here is that for example muscular cells throughout the heart do not all behave the same, but come in many different varieties, and creating new, specific, models for these is very much an active field today [4]. It has also been shown [5] that learning cell models as hybrid automata would make the simulation of large arrays of cells much less computationally resource-demanding.

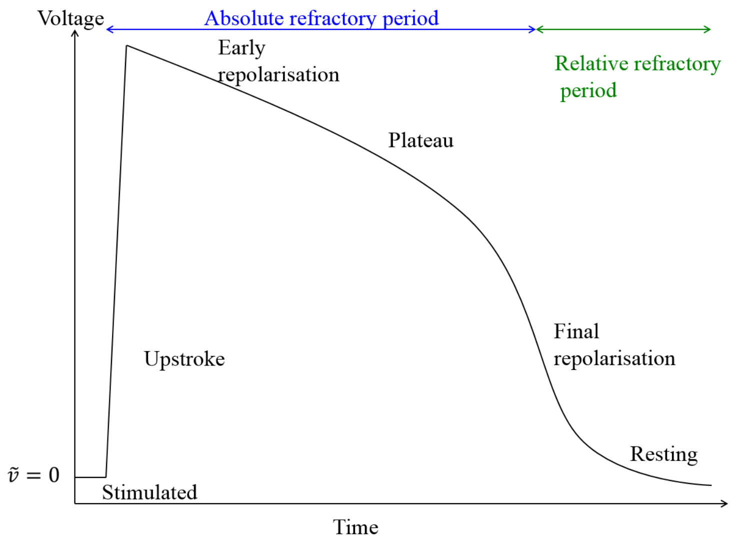

Figure 1.1. A graph of the voltage in an action potential over time

1.1 Excitable Cells

Excitable cells can be found in the vast majority of multicellular organisms and take their name from being able to amplify and propagate changes in transmembrane potential to adjacent cells. The event where an excitable cell does this is known as an *action potential* or AP, and it can be triggered in several ways over different types of transcellular connections known as synapses. In a chemical synapse, neurotransmitters are sent from the end of a neuron to another cell over the synaptic cleft of intercellular space, opening the ion channels of the postsynaptic cell. In electrical synapses, adjacent cells are connected directly via hemichannels between their respective cytoplasms, allowing ions to freely pass between the two respective cells. Various types of voltage-gated ion channels then open and close in distinct phases depending on the transmembrane potential, shaping the AP. The entire process is illustrated in Figure 1.1. Depending on the cell in question, the resting voltage lies between minus 80 to minus 60 millivolts, rises about 100-150 millivolts above that. The whole AP, including full repolarization, takes place over a few hundred milliseconds.

Examples of excitable cells in humans include neurons, endocrine cells and muscle cells of all types. Their action potentials can differ a lot from each other, due to type of synapse, resting potential and types of ion channels, only to name a few factors. In effect, excitable cells control all movement, thinking, and secretion of hormones into the bloodstream. To understand how they work together and map them throughout the body is therefore of vital importance to the medical science.

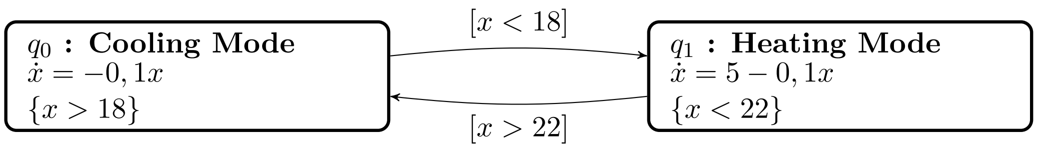
$$\begin{tikzcd}
\boxed{\begin{array}{l} q_0 \textbf{ : Cooling Mode} \\ \dot{x} = -0,1x \\ \{x > 18\} \end{array}} \arrow[r, bend left, "{[x < 18]}"] & \boxed{\begin{array}{l} q_1 \textbf{ : Heating Mode} \\ \dot{x} = 5 - 0,1x \\ \{x < 22\} \end{array}} \arrow[l, bend left, "{[x > 22]}"]
\end{tikzcd}$$

Figure 1.2. A hybrid automaton representation of a simple thermostat

1.2 Hybrid Automata

A hybrid automaton is a representation of a system which combines the logic of states and transitions of a state machine with the continuous modification of variables of ordinary differential equations. During the latter part of the 20th century, hybrid automaton representations have been used to understand and construct embedded systems. It is a powerful tool for modeling systems with both digital and analog parts. Hearkening back to the original cyberneticists who had backgrounds in biology, an interesting current application apart from embedded systems is systems biology - the study of the emergent systemic properties of molecular biology.

In Figure 1.2 we show an example of a hybrid automaton. The different *states* of the hybrid automaton are visualized as boxes. The arrows from one box to another signify *transitions* between the different states, which take place when the *transition conditions* inside the brackets are met, in which case the system changes state. When the hybrid automaton is in a certain state, the *variables* are manipulated in various ways (here, they change over time in a manner governed by differential equations). The transition conditions are inequalities, but they might be any logical statement.

Please note that in literature you might encounter hybrid automata with transitions going from a state to itself. We have here simplified the notation, and written equations governing the behaviour of the hybrid automaton while not transitioning to another state inside the state itself.

In our experiments, when a hybrid automaton is "running", you in one running cycle first increment time by an arbitrary amount, alter the variables as described by the current state, values of the variables and increased time, then check if any transition conditions are met, and finally change state accordingly. The role of time as described above is of course intrinsic to our specific models, and not generally to hybrid automata taken as a whole.

1.3 The Learning Algorithm

The learning algorithm we use, HYCGE, was originally formulated by Karl Meinke and implemented in Python by Fei Niu during 2012 [3]. Consequentially, the hybrid automata we discuss will be a deterministic hybrid Mealy automata, also implemented in Python and tested with the original code. For a complete formal definition of HYCGE, we refer to their unpublished paper. The following is meant to be a rudimentary overview which enables a reader unfamiliar with HYCGE to understand our results and conclusions.

In order to test the learning algorithm, we give a certain input sequence, k real input values and n values of registers, to a system we have full knowledge of, the so-called *system under learning* or *SUL* which the algorithm will treat as a black box. When we get an output sequence of n new register values, we will hand both sequences over to the learning algorithm in order that it might learn the behaviour of the SUL.

The learning algorithm will then have to learn three qualities of this system:

Discrete behaviour The logical transition conditions between the different states. These can be viewed as subsets of $\mathbb{R}^{n\cdot(l+1)}$, where n is the number of registers and l is the number of different symbolic inputs. Note that we assume that only one symbolic input can be true at a time.

Continuous behaviour The functions $f : \mathbb{R}^{n+k} \to \mathbb{R}^n$, one for each edge (note that this includes the edges from a state to itself) of the finite directed graph described below.

Structure A finite directed graph, where the vertices represents the states and the edges the transitions of the hypothetical hybrid automaton representation of the SUL.

It is clear that you will have to learn these two first in parallel. To approximate f, $k + n$-variate polynomials of degree d are used. In our experiments, $d = 2$. In order to approximate the transition conditions, $(n \cdot l)$-spheres are used. Polynomials and spheres are in the language of a mathematician suitably "well-behaved" for computational purposes. Inside a sphere S, we hypothesize that the polynomial tied to S approximates the behaviour of the SUL in our black box within an approximation error ϵ. Different states are thus primarily distinguished by having different d-degree polynomial approximations. Note that this means that parts of the SUL which can not easily be approximated by d-degree polynomials will be fractured into many different spheres.

Key to the algorithm, and to learning the structure of the SUL, is the insertion of these spheres and polynomials as labels of nodes in a *higher-order prefix tree*, which in contrast to the ordinary prefix trees we are used to has polynomials and spheres as labels. The final objective of obtaining an actual hybrid automaton from this higher-order prefix tree is then accomplished through a process called *folding*. This enables us to learn loops - two nodes in the tree are considered equivalent if they yield epsilon-equal output sequences for the same input sequence. In similar, but much more complicated, fashion subtrees are considered equivalent and folded by the same criterion. This is accomplished by checking for *consistency* - if two spheres cover an overlapping space, we compare their polynomials. Because of

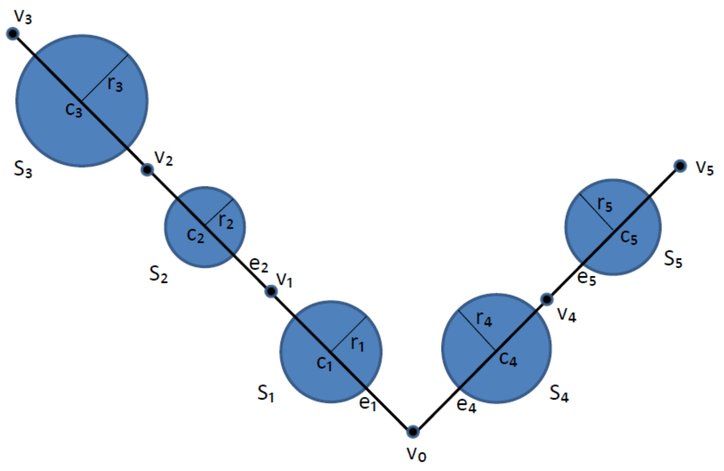

Figure 1.3. A graphical representation of a simple higher-order prefix tree before folding

the assumption that the automaton is deterministic, we know that two different polynomials can not govern the system behaviour of the same space.

If that is the case, the nodes are merged and we add a loop rule to the tree.

For example, if we were to fold the two branches of the higher-order prefix tree in Figure 1.3, we would start checking the spheres for intersection. If spheres have non-empty intersection, they might belong to the same state, so we would check if the polynomials agree with each other. If they are epsilon-equal in the intersection, we have likely detected that they belong to the same state. However, if those spheres are for example S_2 and S_4 in Figure 1.3, we must then check if the aforementioned polynomial is consistent with the *reachable* parts of S_3 and S_5, which means, the parts of S_3 and S_5 which can be reached with the polynomial describing S_2 and S_4. To achieve this, the algorithm uses a heuristic method which we will not describe in further detail here.

If we suppose all subtrees have been checked for, we finally simply fold the ends - in this case if no subtrees were consistent with each other v_3 onto v_2 and v_4 onto v_5. Then, the tree functionally describes a hybrid automaton.

We also utilize the active query generation strategies for HYGCE, as proposed by Karl Meinke and Fei Niu.

Chapter 2

Descriptions of Specific Hybrid Automata

Apart from some general hybrid automata we have used in our research which have no foundation in the physical world, we have studied the performance of the algorithm on several established models of excitable cell behaviour. Note that first two models were originally formulated as a system of nonlinear differential equations only - the hybrid automaton representations and the numerical values for the paramenters were made in a 2005 paper [5] by Pei Ye, Emilia Entcheva, Radu Grosu and Scott Smolka. A table with these parameter values is reproduced in Appendix A.2, to facilitate eventual experiments of the reader. It is also worth noting that we started with randomized initial values adding together through $v = v_{\mathrm{Na}} - v_{\mathrm{K}} + v_{\mathrm{Ca}}$ to a transmembrane voltage v around zero but lower than V_R, zero volt being defined as the resting voltage in the models. Please note that this definition is just a convention - the real resting voltages are described in Section 1.1. This was as good an educated guess as we could do regarding the initial states.

2.1 The Hodgkin-Huxley Squid Giant Axon Model

The Hodkin-Huxley model illustrated in Figure 2.1, or the "HH" model for short, originated in 1952 when the British biophysicists and Nobel laureates Sir Alan Lloyd Hodgkin and Andrew Huxley created a mathematical model of action potential [1]. This new insight into neuron excitation led them to hypothesize ion channels, proteins which regulate the flow of ions across the cell membrane. These were eventually confirmed decades later. The Hodkin-Huxley model and the logical conclusions which followed from it is regarded as one of the landmark achievements of 20$^{\mathrm{th}}$ century biophysics.

The original differential equations in the Hodgkin-Huxley model included approximations of the two voltage-gated ion channels of sodium and potassium in addition to a "leak" term which is time-independent and linear. Here, the behaviour

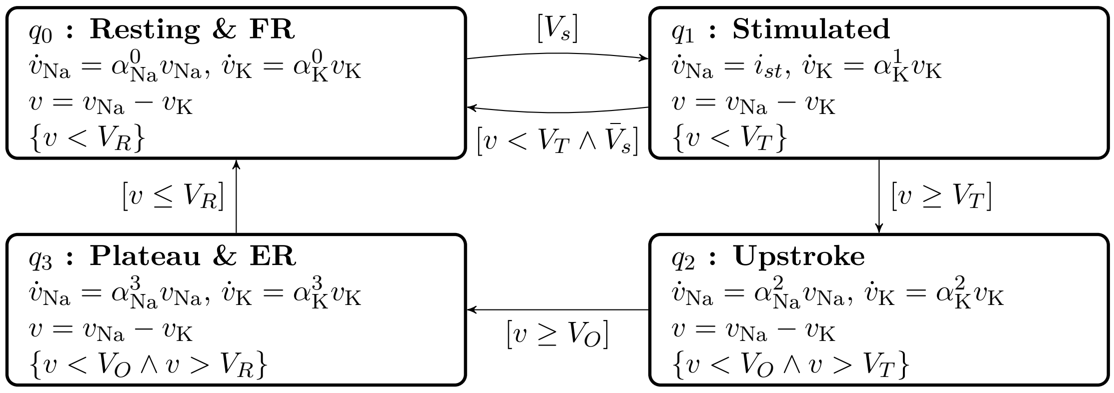
q_0 : Resting & FR
$\dot{v}_{\mathrm{Na}} = \alpha^0_{\mathrm{Na}} v_{\mathrm{Na}}$, $\dot{v}_{\mathrm{K}} = \alpha^0_{\mathrm{K}} v_{\mathrm{K}}$
$v = v_{\mathrm{Na}} - v_{\mathrm{K}}$
$\{v < V_R\}$
$[V_s]$
q_1 : Stimulated
$\dot{v}_{\mathrm{Na}} = i_{st}$, $\dot{v}_{\mathrm{K}} = \alpha^1_{\mathrm{K}} v_{\mathrm{K}}$
$v = v_{\mathrm{Na}} - v_{\mathrm{K}}$
$\{v < V_T\}$
$[v < V_T \wedge \bar{V}_s]$
$[v \le V_R]$
$[v \ge V_T]$
q_3 : Plateau & ER
$\dot{v}_{\mathrm{Na}} = \alpha^3_{\mathrm{Na}} v_{\mathrm{Na}}$, $\dot{v}_{\mathrm{K}} = \alpha^3_{\mathrm{K}} v_{\mathrm{K}}$
$v = v_{\mathrm{Na}} - v_{\mathrm{K}}$
$\{v < V_O \wedge v > V_R\}$
$[v \ge V_O]$
q_2 : Upstroke
$\dot{v}_{\mathrm{Na}} = \alpha^2_{\mathrm{Na}} v_{\mathrm{Na}}$, $\dot{v}_{\mathrm{K}} = \alpha^2_{\mathrm{K}} v_{\mathrm{K}}$
$v = v_{\mathrm{Na}} - v_{\mathrm{K}}$
$\{v < V_O \wedge v > V_T\}$

Figure 2.1. A hybrid automaton representation of the Hodgkin- Huxley squid giant axon model

of these three channels have been summarized in the voltages v_{Na} and v_{K}. V_s is the actual stimulation event, representing saltatory conduction at the nodes of Ranvier or colloquially, the passing on of the nerve signal inside the axon. i_st is the voltage stimulus connected to V_s. $\alpha^0_N a$, α^0_K and so on are simply constants describing the strength of the ion currents. V_T is the threshold voltage value for which the cell will proceed with generating an action potential, in which case we go from state q_1 to q_2. If the stimulation event V_s is too short for the voltage to rise above V_T however, the automaton will simply return to q_1. In q_2, the Na^+ ion channels will quickly change the membrane potential until it exceeds the overshoot voltage V_O at which point the cell will initiate early repolarization or "ER", returning to ordinary membrane potential. When the repolarization voltage V_R is reached, the cell goes back to a restive state and final repolarization or "FR", awaiting the next outside stimulation.

2.2 The Luo-Rudy Guinea Pig Ventricular Cell Model

Another of the models we have studied is the Luo-Rudy model, or the "LRd" model of cardiac myocytes named after Yoram Rudy and Ching-Hsing Luo who defined it in 1994 [2]. A hybrid automaton representation can be seen in Figure 2.2.

In contrast to the HH model, when modeling a cardiac myocyte it is also important to account for the calcium voltage-gated ion channel, v_{Ca} in the representation. Cardiac cells and muscle cells also have a frequency adaption axons lack - a shorter resting phase results in a shorter excited state. From this arises the need for some kind of "memory" in the model, here in the form of v_n. We also define $\theta = v_n/V_R$, meaning that θ is equal to the percentage of the resting threshold voltage when a new excitation event occurs, in other words a measure of the length of the resting phase. In q_3 we also have the function $f(\theta) = 1 + 13\sqrt[6]{\theta}$, adjusting the length of q_3 with respect to θ, accomplishing the goal of frequency adaption.

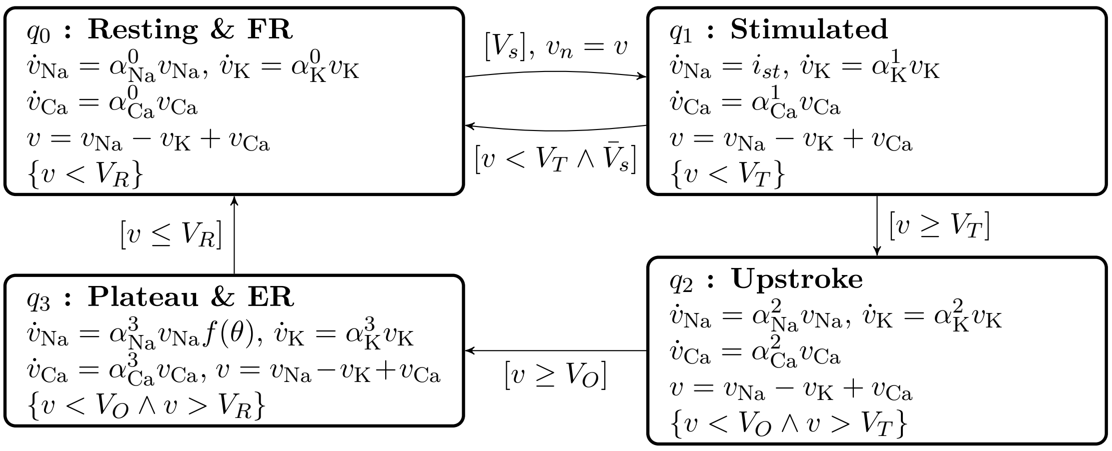
q_0 : **Resting & FR**
$\dot{v}_{\mathrm{Na}} = \alpha^0_{\mathrm{Na}} v_{\mathrm{Na}}$, $\dot{v}_{\mathrm{K}} = \alpha^0_{\mathrm{K}} v_{\mathrm{K}}$
$\dot{v}_{\mathrm{Ca}} = \alpha^0_{\mathrm{Ca}} v_{\mathrm{Ca}}$
$v = v_{\mathrm{Na}} - v_{\mathrm{K}} + v_{\mathrm{Ca}}$
$\{v < V_R\}$
$[V_s]$, $v_n = v$
$[v < V_T \wedge \bar{V}_s]$
q_1 : **Stimulated**
$\dot{v}_{\mathrm{Na}} = i_{st}$, $\dot{v}_{\mathrm{K}} = \alpha^1_{\mathrm{K}} v_{\mathrm{K}}$
$\dot{v}_{\mathrm{Ca}} = \alpha^1_{\mathrm{Ca}} v_{\mathrm{Ca}}$
$v = v_{\mathrm{Na}} - v_{\mathrm{K}} + v_{\mathrm{Ca}}$
$\{v < V_T\}$
$[v \leq V_R]$
$[v \geq V_T]$
q_3 : **Plateau & ER**
$\dot{v}_{\mathrm{Na}} = \alpha^3_{\mathrm{Na}} v_{\mathrm{Na}} f(\theta)$, $\dot{v}_{\mathrm{K}} = \alpha^3_{\mathrm{K}} v_{\mathrm{K}}$
$\dot{v}_{\mathrm{Ca}} = \alpha^3_{\mathrm{Ca}} v_{\mathrm{Ca}}$, $v = v_{\mathrm{Na}} - v_{\mathrm{K}} + v_{\mathrm{Ca}}$
$\{v < V_O \wedge v > V_R\}$
$[v \geq V_O]$
q_2 : **Upstroke**
$\dot{v}_{\mathrm{Na}} = \alpha^2_{\mathrm{Na}} v_{\mathrm{Na}}$, $\dot{v}_{\mathrm{K}} = \alpha^2_{\mathrm{K}} v_{\mathrm{K}}$
$\dot{v}_{\mathrm{Ca}} = \alpha^2_{\mathrm{Ca}} v_{\mathrm{Ca}}$
$v = v_{\mathrm{Na}} - v_{\mathrm{K}} + v_{\mathrm{Ca}}$
$\{v < V_O \wedge v > V_T\}$

Figure 2.2. A hybrid automaton representation of the Luo-Rudy guinea pig ventricular cell model

2.3 The Entcheva Neonatal Rat Ventricular Cell Model

The Entcheva Neonatal Rat Ventricular Cell model, or the "NNR" model for short is the most recent one we will discuss here, originally formulated by EMILIA ENTCHEVA in 2005. It is shown in Figure 2.3. The NNR model is unique in the way that it was first developed as a hybrid state automaton. It is based in part on the LRd model but has the shorter plateau phase characteristic of rats in contrast to most other mammalian species, such as guinea pigs which the LRd model is constructed for and humans for which much of medical science is in the end applied on. Nonetheless, rats have an importance in cardiac electrophysiology since they are often used in other experiments. Here, we define $f(\theta) = 1 + 2\theta$. Also, the voltages in the transition conditions are dependent on θ through $g(V_T) = V_T \cdot (1 + 1,45\sqrt[6]{\theta})$ and $h(V_O) = V_O - 40 \cdot \sqrt{\theta}$.

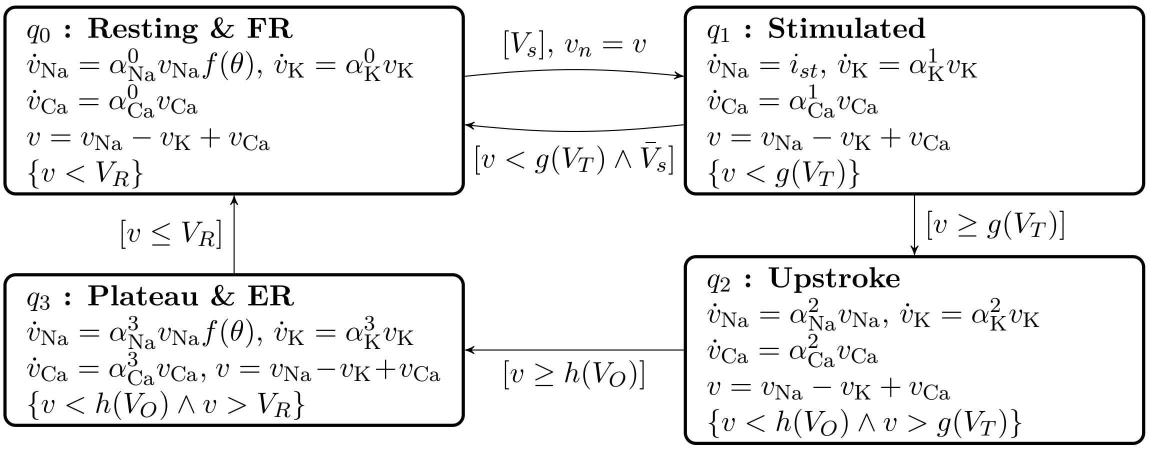
q_0 : **Resting & FR**
$\dot{v}_{\mathrm{Na}} = \alpha^0_{\mathrm{Na}} v_{\mathrm{Na}} f(\theta)$, $\dot{v}_{\mathrm{K}} = \alpha^0_{\mathrm{K}} v_{\mathrm{K}}$
$\dot{v}_{\mathrm{Ca}} = \alpha^0_{\mathrm{Ca}} v_{\mathrm{Ca}}$
$v = v_{\mathrm{Na}} - v_{\mathrm{K}} + v_{\mathrm{Ca}}$
$\{v < V_R\}$
$[V_s]$, $v_n = v$
$[v < g(V_T) \wedge \bar{V}_s]$
q_1 : **Stimulated**
$\dot{v}_{\mathrm{Na}} = i_{st}$, $\dot{v}_{\mathrm{K}} = \alpha^1_{\mathrm{K}} v_{\mathrm{K}}$
$\dot{v}_{\mathrm{Ca}} = \alpha^1_{\mathrm{Ca}} v_{\mathrm{Ca}}$
$v = v_{\mathrm{Na}} - v_{\mathrm{K}} + v_{\mathrm{Ca}}$
$\{v < g(V_T)\}$
$[v \leq V_R]$
$[v \geq g(V_T)]$
q_3 : **Plateau & ER**
$\dot{v}_{\mathrm{Na}} = \alpha^3_{\mathrm{Na}} v_{\mathrm{Na}} f(\theta)$, $\dot{v}_{\mathrm{K}} = \alpha^3_{\mathrm{K}} v_{\mathrm{K}}$
$\dot{v}_{\mathrm{Ca}} = \alpha^3_{\mathrm{Ca}} v_{\mathrm{Ca}}$, $v = v_{\mathrm{Na}} - v_{\mathrm{K}} + v_{\mathrm{Ca}}$
$\{v < h(V_O) \wedge v > V_R\}$
$[v \geq h(V_O)]$
q_2 : **Upstroke**
$\dot{v}_{\mathrm{Na}} = \alpha^2_{\mathrm{Na}} v_{\mathrm{Na}}$, $\dot{v}_{\mathrm{K}} = \alpha^2_{\mathrm{K}} v_{\mathrm{K}}$
$\dot{v}_{\mathrm{Ca}} = \alpha^2_{\mathrm{Ca}} v_{\mathrm{Ca}}$
$v = v_{\mathrm{Na}} - v_{\mathrm{K}} + v_{\mathrm{Ca}}$
$\{v < h(V_O) \wedge v > g(V_T)\}$

Figure 2.3. A hybrid automaton representation of the Entcheva neonatal rat ventricular cell model

Chapter 3

Experimental Results

Our experimental results consist of graphs over completeness and relative accuracy, which are explained below. For the SUL representation of these systems, we have used the forward Euler method to compute the solutions to the ODEs unless otherwise specified.

3.1 Completeness and Accuracy - Benchmarking the Qualitative Performance of the Algorithm

When evaluating the progress of the algorithm, we are interested in two fundamental characteristics. The first of these is how much of the system the learning algorithm has charted. We call this *completeness*. A natural way to measure this would be to see how often a random starting input sequence ends somewhere uncharted by the learning algorithm.

The second characteristic is *accuracy*. We take this to mean how close the predictions of our hypothesis automaton come to the actual SUL. In practice, we measure the relative accuracy, which means the number of already deemed complete input sequences who end within a distance ϵ of the same input sequence fed to the SUL.

3.2 Models of Excitable Behaviour

3.2.1 The Hodgkin-Huxley Squid Giant Axon Model

Studying Figure 3.1 he completeness ratio reaches 55% by 14000 iterations, comparable to the programmable thermostat and the calculator in the very first experiments by Fei Niu. It is relatively stable after 3000 iterations.

We also tried using the midpoint method to compute the solutions of the ODE, which resulted in slower learning per iteration.

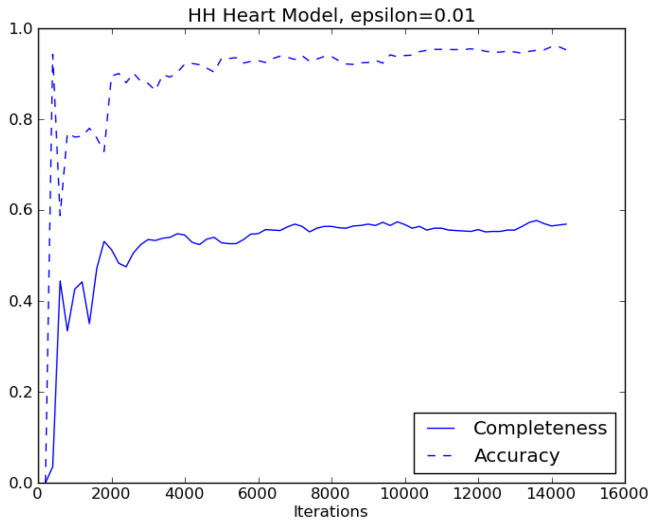

Figure 3.1. HYCGE performance on the Hodgkin-Huxley Squid Giant Axon model

3.2.2 The Luo-Rudy Guinea Pig Ventricular Cell Model

Studying Figure 3.2, we see that at about 27000 iterations, a completeness of 45% is achieved. The learning curve displays significant irregularities in the forms of sudden drops and rises. The calculator in the initial experiments had a similarly slow learning curve.

3.2.3 The Entcheva Neonatal Rat Ventricular Cell Model

Interestingly, Figure 3.3 does not show signs of a significantly slower learning curve - 48% is achieved after 30000 iterations. On the other hand, the relative accuracy show incredibly jumpy signs. That is likely to be caused by the complexity of the model.

3.3 General Performance of the Algorithm

In addition to testing the algorithm on the three aforementioned real models, we "dissected" these and made simpler automata which each captured one characteristic of them, in order to estimate which system behaviours were hard to learn. Figure 3.4, Figure 3.5 and Figure 3.6 depict learning performance on systems with a linear structure and an increasing number of states - 2, 3 and 4, respectively. The first three graphs reach 95% completeness in 60, 400 and 960 iterations, respectively. This would imply that even in the ideal case, an increasing number of states make the learning time increase worse than linearly. In Appendix A.3, we provide the most interesting of these automata.

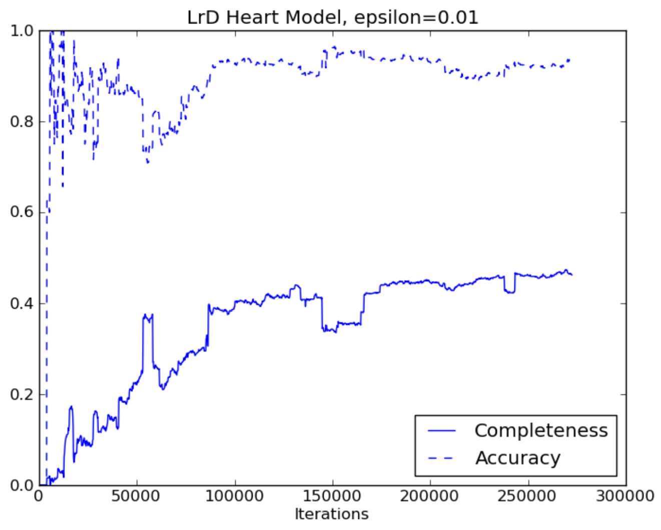

Figure 3.2. HYCGE performance on the Luo-Rudy Guinea Pig Ventricular Cell model

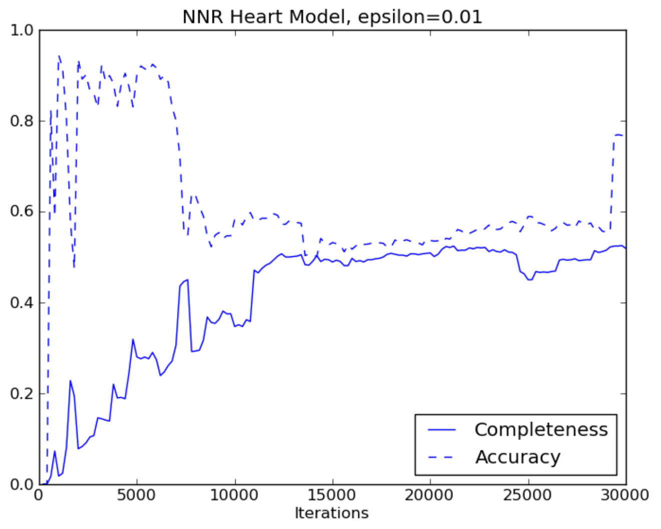

Figure 3.3. HYCGE performance on the Entcheva Neonatal Rat Ventricular Cell model

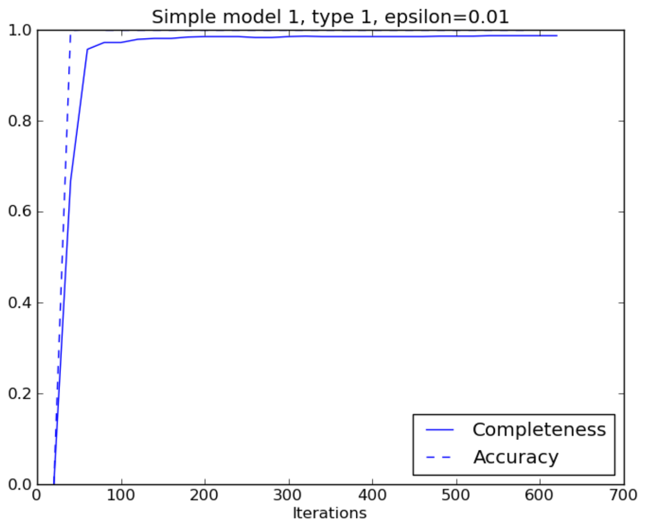

Figure 3.4. HYCGE performance on the simple model of type 1.1

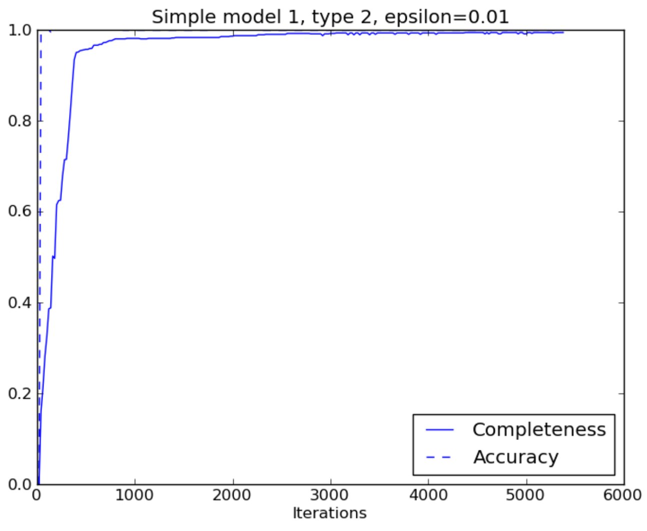

Figure 3.5. HYCGE performance on the simple model of type 1.2

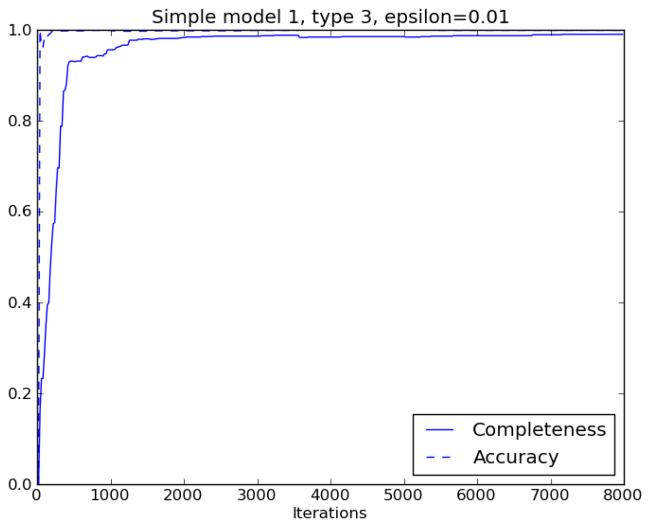

Figure 3.6. HYCGE performance on the simple model of type 1.3

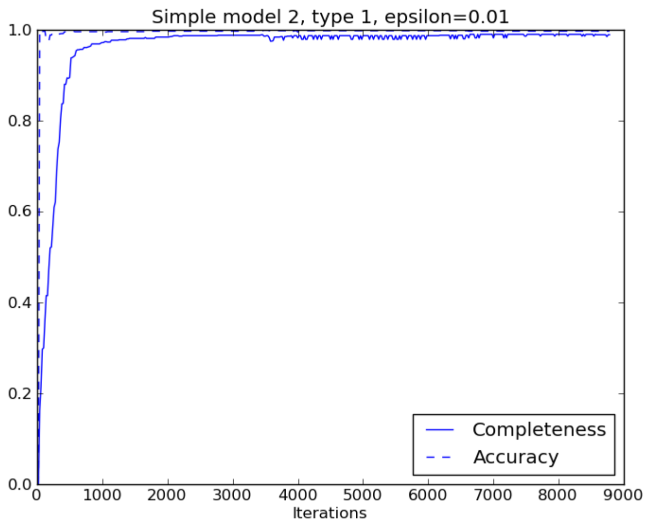

Figure 3.7. HYCGE performance on the simple model of type 2.1

The first three graphs reach 95% completeness in 60, 400 and 960 iterations, respectively. This would imply that even in the ideal case, an increasing number of states make the learning time increase worse than linearly.

On the other hand, as the Figure 3.7 which depicts a simple looping HA, modeling loops is clearly not an issue at all for the algorithm.

We also made two more experimental HAs, one similar to the simple model of type 1.1 albeit with more more complex differential equations in the two different states, and one which is similar to the 2.1 simple model except for that the transitions are solely dependent on V_s, akin to the first transition in our biophysical applications. The first of the was virtually similar to 1.1, but the second proved to be a very interesting anomalous type - the algorithm was not able to start learning it at all. Taken together, these insights might shed some light on which aspects of the algorithm and the Python implementation of it might need further attention and refinement and which do not.

Throughout our experiments, we have found out that how you randomize your initial values for the SUL is crucial in order for the algorithm to learn correctly. Since we do our experiments with a virtual model, we can choose these ourselves. Applying the learning algorithm on real, physical, systems would require you to start with all possible (or rather, non-equal within some distance ϵ) initial values in order to possibly be sure of learning the full system. Depending on what you know about the SUL, this might pose an arduous task, and if you can not adjust the initial values, impossible.

Chapter 4

Conclusions

Seeing how well our results fared compared to the original results, we believe it is fair to think that the completeness ratio would continue to grow for our automata in a similar way as it did in the original experiments. This would imply between 80% and 90% completeness in double that running time, and we are reaching the levels where we are getting usable results.

When you design representations of systems it can be hard to know whether or not your representation is the simplest one possible. It is easy to think that all future applications of learning algorithms like HYCGE lie with learning physical systems. One important application might be to analyze already known models of systems to see if there exists a simpler representation, since HYCGE will always learn the simplest possible model representation of a system in theory, and in practice the simplest one where the ODEs are well approximated by d-degree polynomials. This can however also lead to a backlash - if you try to learn a system which has too complex equations to be approximated well by d-degree polynomials, the result hybrid automaton will have confusingly different states.

In conclusion, we think that the 21$^{\text{st}}$ century interdisciplinary venture of fully charting and simulating the human body could be accelerated by automated learning, and that the HYCGE algorithm is a potent tool for that mission.

Appendix A

Supplementary Tables and Figures

A.1 Performance Reference Sheet

SUL	**Relative accuracy**	**Completeness**	**Training data size**	ϵ
The HH model	0,96	0,55	14000	0,01
The LrD model	0,93	0,45	27000	0,01
The NNR model	0,58	0,48	30000	0,01

Table A.1. HYCGE performance reference sheet

A.2 Parameter Reference Sheet

	HH	**LRd**	**NNR**
V_R	10	20	20
V_T	10	20	30
V_O	83	138	120
i_{st}	30	30	30
α^0_{Na}	-0,98	-0,1	-0,025
α^0_{K}	-0,16	-0,1	-0,07
α^0_{Ca}	N/A	-0,1	-0,2
α^1_{Na}	N/A	N/A	N/A
α^1_{K}	-0,16	-0,1	-0,07
α^1_{Ca}	N/A	-0,1	-0,02
α^2_{Na}	1,4	200	250
α^2_{K}	15	0	200
α^2_{Ca}	N/A	100	125
α^3_{Na}	-0,98	-0,001	-0,025
α^3_{K}	-0,16	0,036	-0,07
α^3_{Ca}	N/A	0,008	-0,2

Table A.2. Parameter reference sheet

A.3 Figures of Simple Experimental Models

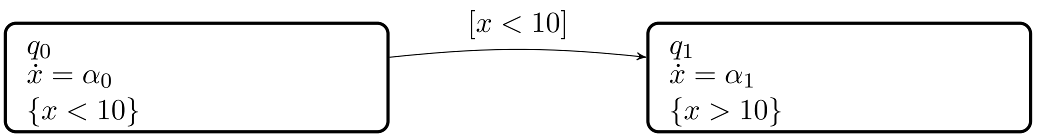

Figure A.1. A hybrid automaton representation of a simple model of type 1.1

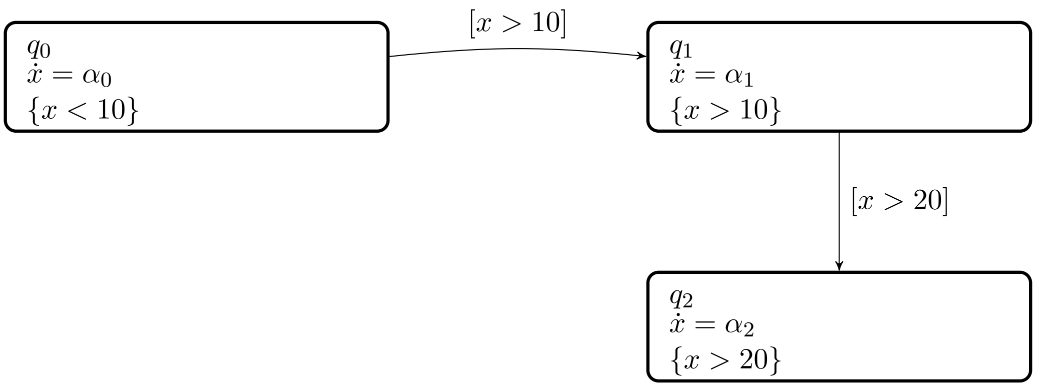

Figure A.2. A hybrid automaton representation of a simple model of type 1.2

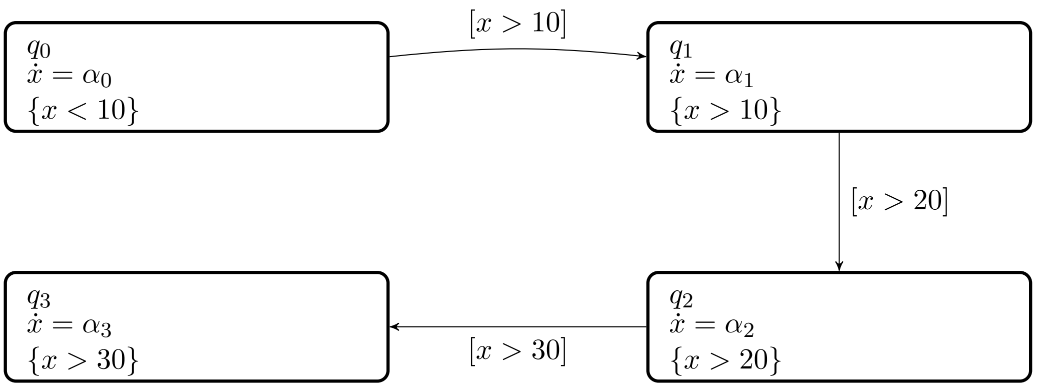

Figure A.3. A hybrid automaton representation of a simple model of type 1.3

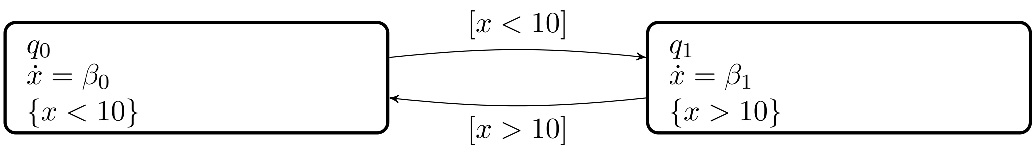

Figure A.4. A hybrid automaton representation of a simple model of type 2.1

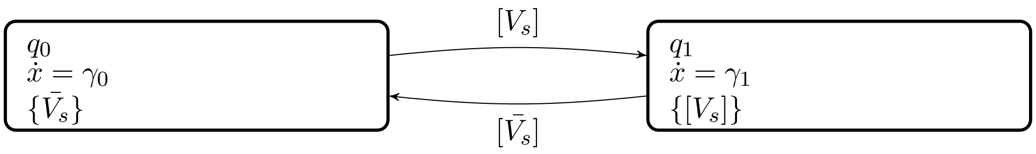

Figure A.5. A hybrid automaton representation of a simple model of type V_s, used for testing performance of learning systems with different symbolic inputs

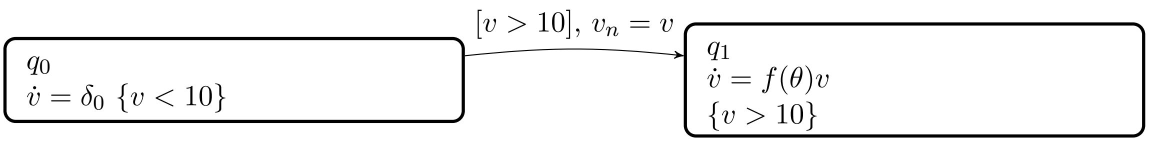

Figure A.6. A hybrid automaton representation of the simple model of type $f(\theta)$, $f(\theta)$ defined as in the NNR model

Bibliography

[1] Alan Lloyd Hodgkin and Andrew Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117:500–544, 1952.

[2] Ching-Hsing Luo and Yoram Rudy. A dynamic model of the cardiac ventricular action potential. i. simulations of ionic currents and concentration changes. *Circulation Research*, 74:1071–1096, 1994.

[3] Karl Meinke and Fei Niu. An active learning algorithm for hybrid mealy automata. Unpublished internal paper, March 2013.

[4] Mathias Wilhelms, Hanne Hettmann, Mary M. Maleckar, Jussi T. Koivumäki, Olaf Dössel, and Gunnar Seemann. Benchmarking electrophysiological models of human atrial myocytes. *Frontiers in Physiology*, 3:487, 2012.

[5] Pei Ye, Emilia Entcheva, Radu Grosu, and Scott A. Smolka. Efficient modeling of excitable cells using hybrid automata. In *In Proceedings of Computational Methods in System Biology*, pages 216–227, 2005.