Undecidability and novelty generation in RNA automata

ADAM J. SVAHN

PhD, B.Sci (Hons)

Supervisor: Mikhail Prokopenko
Associate Supervisor: Joseph Lizier

A thesis submitted in fulfilment of
the requirements for the degree of
Master of Philosophy

School of Computer Science
Faculty of Engineering
Centre for Complex Systems
The University of Sydney
Australia

2023

Abstract

As today, the evolution of the earliest life was an exploration of adaptive forms. However, the earliest life also undertook great leaps in the overall complexity of the molecular dynamic system as a whole. A theoretical framework for this form of evolution has not been resolved. This thesis builds upon the discoveries of early life chemistry and seeks to take the next step towards understanding the organising principles that allowed life to evolve. In computer science novelty generation is often linked to universal computation, as the boundaries of complexity are found at the edge of undecidability where self-referential incomputable statements can be generated.

At the intersection of early life chemistry and computer science, this thesis draws from the dominant RNA-world model and incorporates this into the constructions of automata theory to investigate the computational properties of a system of single-stranded RNA molecules. Limited to the plausible RNA-world operations of ligation and cleavage, RNA automata are constructed of increasing complexity; from the Finite Automaton (RNA-FA) to the Turing machine equivalent 2-stack Pushdown Automaton (RNA-2PDA) and ultimately a universal RNA-UPDA with the capacity to generate undecidability. A path forward from undecidable computation in RNA automata to novelty generation is mapped. The coupled phenotype-environment space is presented as a framework for biological system expansion. The framework draws on the discoveries of Alan Turing and Emil Post on the continual expansion of computational systems overcoming their undecidable boundaries (i.e., Turing's ordinal logics utilising 'Oracle machines' and Post's extensible recursively generated logics). An analogue of these extensions, considered from the perspective of ecological developmental biology (eco-devo), offers a self-referential model of the organism coupled with its environment, which is capable of novelty generation. This thesis concludes by outlining future avenues of research to develop the coupled phenotype-environment space framework as well as identifying biological computational constructions in extant organisms.

Statements

Chapter 3 of this thesis is comprised of an article published as:

Adam J. Svahn and Mikhail Prokopenko (2022). *An Ansatz for Computational Undecidability in RNA Automata,* Artificial Life, 1-28.

The study was conceptualised by myself and Mikhail Prokopenko. I undertook the study, as well as drafting and revision of the manuscript. Mikhail Prokopenko advised the study and contributed to both drafting and revision of the manuscript.

Adam J. Svahn

5. 9. 2022

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Mikhail Prokopenko

5. 9. 2022

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Adam J. Svahn

5. 9. 2022

Acknowledgements

I have been privileged to have had the support of many wonderful people over the course of this endeavour.

Prof. Mikhail Prokopenko has been a fantastic supervisor, mentor and friend. Thank you Mikhail for taking the chance on bringing me over from developmental biology and helping me begin to understand the intricacies of computation. I will treasure all of the time we spent in discussion at the whiteboard.

Thank you to my co-supervisor Assoc. Prof. Joseph Lizier for being supportive throughout. I will miss our lunchtime deep dives into one topic or another. Thanks as well to Dr. Mahendra Piraveenan and Dr. Benjy Marks for useful and supportive review discussions throughout the project.

Thank you to my wonderful wife Rosie, you did everything possible to be the most supportive partner on this journey. To our son Leon who is nearly two years old now, thank you for being an endless delight.

Contents

CHAPTER 1

Introduction

It is a remarkable thing to consider the unbroken chain of life, stretching back to the moment that abiotic molecules began to exhibit the organisation and metabolic processes of an organism. It likely happened just once, and from this moment all living things were built by the living from non-living matter, and the spark of life was transferred from one generation to the next. It is akin to the elusive feeling of gazing at the stars and briefly grasping a sense of the scale of the universe. But what is the spark of life? Is it a specific catalyst? Perhaps a 'living' element? It is, of course, no single thing, but rather an organising principle and a process that is self-sustaining once it has started. Beyond the remarkable fact of its existence, life has passed through great saltations of complexity. Broadly these leaps have been from a loose correspondence between organic molecules, to reproducing systems, to compartmentalised organisms and latterly, multicellularity. The capacity to undergo these transitions suggests there must be a mechanism for a living dynamic system of molecules to increase its own level of complexity. Such a mechanism is non-trivial, and the well-established mechanism of evolution through natural selection, which drives life to explore phenotypes that adapt to its environment, cannot fully account for the complexity leaps that must have occurred through the very earliest stages of life. This sets the motivating question of this thesis: by what organising principles could the molecules present at the origins of life have organised into a system of chemical reactions which was able to expand the boundaries of that system to become more complex?

This thesis will start by recognising the molecular components of early life as defined by the dominant RNA world theory. It will then show how these basic RNA components could be organised theoretically into computational constructions drawn from automata theory,

and show that these constructions exhibit computational dynamics that advance biological functions. It will then show that an RNA computational system is capable of reaching a complexity boundary in the form of undecidable dynamics by embodying the incomputable Liar paradox. The work of Alan Turing and Emil Post on the expansion of a system to overcome an undecidable boundary will be surveyed and incorporated to show how an RNA system may have expanded under pressure from undecidability. This synthesis is brought together in an Ansatz, or proposition, that *i)* RNA automata constructions are capable of reaching Turing machine equivalence, and *ii)* these biological systems are capable of reaching computational undecidability and to expand their own boundaries under pressure to resolve the undecidability.

The research areas that form the foundation for the motivation of this thesis are the RNA world model of the transition from abiotic molecules to biotic life, and the evolution of life as a system which could encode and transmit information. Much of the early work for both of these topics was driven by the discoveries and insights of Carl Woese, a pioneer in the 1960's of genetic sequencing and analysis in microbiology and evolutionary biology. Woese's most far-reaching contribution was to the development of modern phylogeny (the tracing of lineage) through molecular biology and sequencing. Woese, like the rest of the field of microbial genomics in the 1960's, struggled to reconcile the dominant evolutionary model, with its roots in anatomy and taxonomy, to the microbial world in which cells possess no nucleus, contain extra-chromosomal circular plasmid DNA and, most significantly, engage in horizontal gene transfer. By turning his attention to mapping ribosomal RNA, a core component of the ancient system which reads genetic information, Woese built a new evolutionary tree of life which overturned the prevailing two domain model of life (eukaryotes and prokaryotes) to replace it with a three domain model (eukaryotes, bacteria and archaea) that now stands as the dominant evolutionary model. From this new model, Woese gained the insight that the nature of the very earliest evolutions of life must have been different to that which came after (Woese 2004):

> "...the overwhelming amount of novelty needed to bring modern cells into existence ...is the central and most challenging question...This is a kind of

> novelty that we would not encounter in the modern biological era, and it had to have been generated in a kind of way that we have yet to fathom"

Expanding on this, Woese also foresew the saltations, or 'step-changes' that these early evolutionary innovations represented. That is, the first steps of evolution entailed an *expansion* of the dynamic system of the organism, rather than innovation within it (Woese 1965):

> "Evolution during any of these hypothetical stages should be *qualitatively* different from that occurring in any other stage, for the basic cell type would probably differ from one stage to another more drastically than do any of the cell types now extant."

The term that came to represent the capacity for system expansion was 'evolvability' (Wagner and Altenberg 1996; Virgo et al. 2017), and Woese noted that evolvability is best understood in terms of information representation and transmission (Woese and Goldenfeld 2009):

> "...the real problem of the gene did not lie in the here and now. Instead, it lay in the emergence of an incredible and complex mechanism that can extract information (pattern) from the sequence of one type of macromolecule and "express," i.e., store, most of it as the structure (sequence pattern) of another macromolecule of a different type. This process thus gives rise to a new world (space) of macromolecules and possible interactions among them. Moreover, this process of encoding can continue to higher and higher levels of organization, eventually giving rise to cells as we know them..."

Woese wrote passionately that an understanding of organism complexity and evolvability would only be realised with the incorporation of discoveries and methods from computer science, information theory and complex systems (Goldenfeld and Woese 2007; Woese 2004).

At the intersection of these topics we find the field of biological computation, which progresses the goal of utilising the organising principles of computational dynamics to illuminate complex biological phenomena or enable the control of complex biological pathways. An early landmark in biological computation was written by Hofstadter (1980), who emphasised that

the relationship between self-reference and recursive self-representation was important in biology as well as in mind and computation. Once a biological system has been conceptualised in computational terms (Prohaska et al. 2019), the components can be cast into diverse computational frameworks. For example, Ouyang et al. (1997) took a direct approach to solve the NP-complete maximal clique problem by utilising DNA to search for the optimal solution in a massively parallel manner. In a similar vein, and relevent to the RNA constructions of this thesis, Benenson et al. (2003; 2001) demonstrated that the finite automaton drawn from automata theory could be approximated in benchtop DNA interactions. To further understand an existing molecular sub-system, Arnold et al. (2013) surveyed the mechanics of chromatin, the proteins that organise DNA. This work was able to demonstrate that a computational framework built of chromatin protein modifications (a 'chromatin computer'), is possible, greatly expanding the complexity of operations that this seemingly simple system was capable of. Biological computation may also look backward to understand evolutionary transitions; a classical problem in evolutionary models of early life is that of building fidelity into information transfer (gene inheritance) from a noisy starting point. Froese et al. (2018) was inspired by Woese's insight that horizontal transfer may be underrepresented in models of early life, and demonstrated through computational modelling of gene fragment sharing among communities of protocells that such a mechanism could iterate learning and build towards regular and optimal encoding.

Following in the tradition of biological computation, this thesis will seek to illuminate the mechanism which drove increasing complexity of biological systems in early evolution. This thesis will draw on the progress made in the fields of computer science and artificial life, where such a phenomenon is recognised as novelty generation in open-ended evolution (Markose 2004; Kauffman 2016; Markose 2017; Zenil et al. 2016; Abrahão et al. 2019; Adams et al. 2017). In this space it is recognised that the dynamics of expanding system boundaries comes at the point of undecidability and incomputability, where self-referential incomputable statements are formulated (Prokopenko et al. 2019). From the computational perspective it is possible to focus the thesis question to whether self-referential undecidability can be demonstrated for biological systems, which would reveal a general mechanism for novelty generation.

The components used in the constructions of this thesis are restricted to those which can have reasonably been considered to be present at the earliest stage of life. The RNA world hypothesis, first outlined by Woese (1967), puts the single stranded RNA molecule at the centre of the first instances of replication and information preservation that constituted early life. Bringing together the computational principles with the RNA world hypothesis it is possible to formulate a still more focused thesis question of whether formal undecidability can be demonstrated for a system of single-stranded RNA polymers.

The main body of this thesis is an Ansatz. 'Ansatz' derives from the German word 'ansetzen', meaning to fix or place initially. It is intended as an evidenced and reasoned foundation which informs a broader supposition. The supposition then guides further investigation to build a larger body of work which will either confirm or refute the initial supposition. The Ansatz here is built on RNA biology, computational constructions in the form of automata, computational principles of universality, self-reference, undecidability, and the theory of expansion of computational systems beyond boundaries described by Alan Turing and Emil Post. First, a background on RNA world chemistry will be given to inform the constructions of the Ansatz, then the Ansatz will be presented (Svahn and Prokopenko 2023). Finally, avenues for future research to progress the questions posed by the Ansatz will be discussed.

CHAPTER 2

Background

2.1 Joining and splitting in the RNA world

The setting for this thesis is the chemical and environmental milieu at the origin and early evolution of life. Life as we may observe today, with the exception of RNA viruses, utilises DNA as long-term information storage. However, DNA is very unlikely to have been present until long after life had established itself. Therefore, there existed an enigmatic period at the origin of life in which it followed a different set of rules and used a different set of mechanisms than the vertically-inherited evolution via DNA mutation that has dominated since. The RNA world hypothesis (Higgs and Lehman 2015) centres the single stranded RNA molecule as the first biotic molecule. This review will examine the ontogeny and structure of the RNA molecule and its polymerisation into RNA polymer enzymes capable of catalysing chemical reactions. This review will focus on the ontogeny and function of the RNA enzymes that ligate polymers together (RNA ligases) and a specific RNA enzyme that cleaves RNA polymers (RNase P). Throughout, this review will consider how RNA ligases and RNase P have been adapted and iterated upon by life, to inform how they may be utilised for biological computation.

2.1.1 Ribonucleotide structure and abiotic pathways to ribonucleotide formation

The ribonucleotide monomer is a modular construction of a ribose sugar, a nucleobase and a phosphate. The ribose in its linear form consists of a 5 member carbon chain decorated

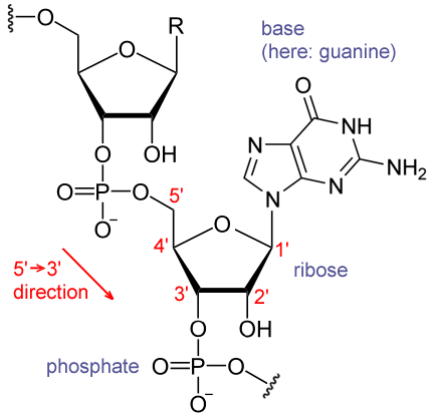


FIGURE 2.1: RNA structure. Two ribose rings are joined by a phosphodiester bond at the 3' and 5' positions. The nucleobase guanine is pictured in the 3' ribonucleotide. Source: wikimedia.commons.org

with hydroxyl groups (-OH) and an aldehyde group (-CHO) in the first carbon position. In the ribonucleotide, the ribose is in a ring form, with the carbons labelled 1' to 5' as shown in Figure 2.1.

The nucleobase that gives the ribonucleotide monomer its identity binds at the 1' carbon position, and the two phosphates that link the RNA into a polymer bind at the 5' and the 3' carbons. For this reason the first monomer of an RNA polymer is referred to as the 5' element and the last monomer as the 3' element. Direction (i.e. top and bottom) is inferred from the process of DNA to RNA transcription, via RNA polymerase, which elongates a new RNA polymer in a 5' to 3' direction. The ribose of each ribonucleotide monomer is identical, whereas the nucleobase bound at the 1' position of the ribose ring may differ. As such, the nucleobase is the information carrying unit of the ribonucleotide. The nucleobases are divided into the purines (adenine and guanine) and the pyrimidines (cytosine and uracil) which form pairwise affinity relationships by hydrogen bonding; adenine to uracil and guanine to cytosine. The phosphodiester bonds are negatively charged, which repel one another and contribute to the RNA helix structure. Importantly, RNA polymers are structurally labile, readily forming complex tertiary structures. These structures create chemical micro-domains that allow the RNA polymer to act as an enzyme, facilitating a chemical reaction. RNA ligases and RNases make use of the hydrolysing aqueous environment to catalyse the forming of a phosphodiester bond (ligation), the breaking of a phosphodiester bond (cleavage), or, recombination of a

polymer by a two-step ligation (splicing). This may be performed by an RNA polymer upon itself (in cis) or upon an opposing polymer seperate to itself (in trans).

Ingenious chemical experimentation and geochemical analysis has resulted in the discovery of pathways for the components of RNA from plausible sources of reagents on an ancient earth, however the field remains far from a unified theory for a pathway to biotic chemistry. Our progress towards understanding how the very first molecules of ribose, nucleobase and phosphate came together is well reviewed (Neveu et al. 2013; Sutherland 2016). With regards to monomer synthesis, the main stumbling blocks have proven to be the 'tar' problem; in which deviations from the proposed reaction pathway result in a dead-end 'tar' substance, and the 'water' problem; in which the hydrolysing aqueous environment, which facilitates the actions of RNA enzymes, also prevents stabilisation of the required complex intermediaries. The 'discontinuous model' is a step towards resolving these problems, proposing plausible micro-environments for each step in the abiotic pathway (Sutherland 2016; Benner et al. 2012). A similar question arises when we consider what kind of environment facilitated the polymersation of riboncleotide monomers into polymer strands. In this vein, template catalysed polymerisation in montmollerite clay (Huang and Ferris 2006) has been explored, as well as nucleating monomers in the eutectic phase of an ice/water mix (Monnard et al. 2003). Similarly, phase changing in a wet/dry cycle promotes ribonucleotide monomer polymerisation (Higgs 2016). A remarkable hypothesis that does away with the need for catalysis has suggested that ribonucleotide monomers in water may spontaneously form stabilising hydrophobic polymer structures through base-stacking between the nucleobases (Cafferty et al. 2013). Indeed, experiments with a variety of canonical nucleobases (A, C,G, U/Thymine) and more exotic non-canonical nucleobases (Xanthine, Di-aminopurine, Isoguanine) demonstrate that such a base-stacking polymerisation is a plausible mechanism in water that may precede phosphodiester bond formation (Kuruvilla et al. 2013).

While the specific pathway of ribonuclotide polymer synthesis remains to be clarified, it is not for a lack of plausible models. Rather, the field is untangling which model best fits the environmental milieu of the ancient earth. As such, the chemical foundation for the RNA

world hypothesis continues to strengthen and it is reasonable to accept the RNA world as a basis for the RNA based computational constructions of the Ansatz.

2.1.2 Enzymatic concatenation of ribonucleotide polymers: ontogeny and *in vitro* synthesis of RNA ligases

From the existence of the monomers and their spontaneous polymerisation, we move on to the enzymatic properties of RNA polymers that can be plausibly considered to have been possible in an ancient RNA world. The first reaction is that of ligation, which is the concatenation of polymers. We saw above that spontaneous polymerisation of RNA monomers is possible, however in modern biotic chemistry, enzymes perform the role of RNA polymerisation by catalysing the formation of the phosphodiester bond. Ligation will be the first of two mechanisms utilised in the biological computational constructions of this thesis.

Ligation by RNA enzymes is a core function in all extant life. For example, intron-rich eukaryotic genes are first transcribed into a pre-mRNA polymer with alternating exons (coding) and introns (non-coding) that are recombined into exon-only mRNA by the 'spliceosome', which is a set of protein-RNA hybrid molecules referred to as snRNPs (small nuclear ribonucleoproteins) (Will and Luhrmann 2011). Most commonly, the spliceosome performs a two-step ligation reaction to cleave out a short RNA polymer and ligate the ends of the exons together. At the core of the snRNP is an snRNA which may confer both binding specificity and catalytic function. The conservation of an RNA core in these hybrid compounds suggests an essential function of the RNA component and in turn informs the hypothesis that an ancient RNA enzyme may have performed ligation in an RNA world. This idea is supported by the existence of group I and II 'self-splicing' (i.e. in cis) introns which encode a ribozyme that is capable of performing the same two-step splicing reaction as the spliceosome, with similar mechanics (Will and Luhrmann 2011). Self-splicing mechanisms represent extant evidence of the potential for independent RNA-based ligation in the RNA-world scenario however no ligases exist that can be traced by conservation back to the last universal common ancestor, LUCA. Evidence for ancient RNA ligases may instead be drawn from the remarkable ability of novel ligase ribozymes to be generated by *in vitro* evolution (Joyce and Szostak 2018).

This has been demonstrated and replicated independently in the form of the class I ligase (Ekland et al. 1995) and the R3C ligase (Rogers and Joyce 2001). Successive designs and further *in vitro* evolution of the R3C ligase culminated in an efficient auto-catalytic set in a cross-catalytic format, in which two enzymes ligated substrates to form the other (Lincoln and Joyce 2009).

A brief discussion of the general computational applications of ligation is continued in the Ansatz of Chapter 3, followed by the demonstration of RNA computational constructions utilising ligation.

2.1.3 Cleavage of ribonucleotide polymers: the LUCA origin of RNase P

RNase P is a ribonucleoprotein that cleaves single-stranded RNA separate from itself and may bind and cleave multiple targets without losing function (Reiter et al. 2010). All kingdoms of life retain the essential RNA core of RNase P, evidence of its ancient origin as an enzyme of LUCA. From this ancestral polymer, the bacteria and a minority of archaea have iterated through RNA secondary and tertiary structure to stabilise the enzyme and increase efficiency (Reiter et al. 2010). Within the RNA polymer, there are 5 distinct conserved regions (labelled CRI-CRV) identified (Chen and Pace 1997). These domains are divided into two modules that illustrate the basic functions required of the enzyme; a C-domain (CRI, IV and V) which contains the catalytic active site, and an S-domain (CRII and III), which confers specificity to the RNase (Torres-Larios et al. 2006). A 263nt minimal variant derived from bacteria containing these conserved RNA regions is sufficient to perform RNA cleavage (Waugh et al. 1989). By contrast, there is no protein sub-unit of RNase P that is similarly conserved, with the bacterial protein sub-unit having been acquired after divergence from the ancestor of archaea and eukarya (Evans et al. 2006).

Inspired by the extensive use of RNaseP throughout the kingdoms of life, there have been synthetic approaches to utilising such a robust, ubiquitous enzyme. These demonstrations illustrate the universality of a mechanism present from LUCA, and its utility in a biological computational construction. Approaching from the perspective of the targets of RNaseP,

mutagenesis study identified an essential role for a specific motif (the 3'-RCCA) in the tRNA targets of bacterial RNaseP cleavage (Wegscheid and Hartmann 2006). This motif is both essential and sufficient for single-stranded RNA polymer cleavage by RNaseP (Derksen et al. 2015). In practice this discovery led to the development of synthesised external guide sequences (EGS), which are RNA polymers bearing the RCCA motif that can bind and draw *any* targeted RNA polymer to the RNase P enzyme, where the unbound region adjacent the motif is cleaved (Guerrier-Takada et al. 1995). Looking to the RNase P enzyme itself, it was discovered that a guide sequence could be ligated to the minimal catalytic unit of RNaseP (Waugh et al. 1989; Derksen et al. 2015). This enables fully *in vitro* cleavage reactions (Yang et al. 2006; Zou et al. 2004). When considered alongside the ligation based encoding discussed above, we can observe that such an RNA enzyme based cleavage mechanism could form a crucial step in realising an RNA-based biological computing construction. A brief discussion of the general computational applications of cleavage is continued in the Ansatz of Chapter 3, followed by the use of cleavage as a mechanism in RNA computational constructions.

2.1.4 A foundation for RNA automata

Taken together, it becomes reasonable to conclude that, if abiotic chemistry in a discontinuous model is indeed capable of sidestepping 'tar' products and producing polymerised ribonucleotides, these RNA polymers are capable of exploring an enormous space of possible ligations and recombinations. From our viewpoint of biological computation, we may credit this ability and investigate how these functions can be built into computational systems. Placing our automata in a specific biochemical context also allows for the enumerating of a foundation of biochemical assumptions to guide the computational constructions. These assumptions are necessary such that the automata follow only one possible pathway of states and transitions (i.e. maintain determinism). We make the assumption that the reactions 'go to completion', meaning that all available reactants are used, i.e. all reactions of a type which are possible *will* occur. Similarly, we assume that the reactions are irreversible. The alternative in which reactions are reversible, i.e. if molecule X can be converted to molecule Y, with a chance to be converted from Y back to X, then some equilibrium will be found with both X and Y forms

present at any given time. We further assume that the reaction profiles for enzymes in our automata do not include 'off-specific' substrates. 'Off-specific' reaction are those in which some molecule that is not the main substrate, but which bears some chemical similarity to the main enzyme substrate, can have a small chance to take part in the reaction. We finally assume that reaction probabilities are not affected by environmental noise. The first two assumptions, that of complete and irreversible reactions, are straightforward and commonly expected in benchtop molecular biology protocols. The subsequent assumptions, that of no 'off-specific' reactions or influence from environmental noise, *can* be achieved under the right conditions, but are unlikely in a natural environment or with increasingly complex reaction systems. For the purpose of the thesis, it was considered important to present automata in a deterministic mode, and to present the constructions in the most precise manner. Further, constructions augmented with mechanisms to compensate for mixed reactants or environmental noise etc. would remain embodying the same computational dynamics. Remaining assumptions specific to the automata designs are detailed in the Ansatz of Chapter 3. These realistic assumptions enabled the construction of a theoretical cascade of precise computational abstractions of simple to complex RNA automata.

2.2 Universality, self-reference and the Liar paradox

In 1900, David Hilbert proposed a list of important challenges to the mathematical community. Second on the list was the question of whether mathematics could be proved consistent (Hilbert 1902). In 1928, Hilbert and Wilhelm Ackermann narrowed the question to one of whether a standard effective procedure could be formulated that could follow the rules of any given logical system of axioms, in order to determine the truth of any statement in that system (Hilbert and Ackermann 1950). This was described as the 'decision problem' or 'Entscheidungsproblem', in Hilbert's native German. This framing was part of a wider movement towards a more mechanistic, procedural approach to mathematics, and it was in 1932, as part of this effort, that Kurt Gödel demonstrated that a formal system of axioms could be represented by a numeral encoding, Gödel numbering, and the theorems of that system could be effectively enumerated (i.e. produced algorithmically) (Van Heijenoort

1967). Much earlier, in 1891, Georg Cantor's Diagonal Argument had shown, for number theory, the existence of infinite sets which cannot be placed in 1:1 correspondence with the natural numbers (i.e. uncountable sets) (Ewald 2005). Gödel numbering and Gödel sentences generalised this concept for all logical systems in the Gödel incompleteness theorems; the first of which states that there will always be a proposition within a consistent formal system that cannot be proved or disproved within that system if the system is at least as powerful as Peano arithmetic. Hilbert's question had produced a fundamental challenge to mathematics, as the incompleteness theorems set up the proposition that mathematics was either complete or consistent, but could not be both. It was in this context, and in response to the Entscheidungs-problem, that Alan Turing demonstrated universal computation. Turing formalised the notion of an effective procedure, or algorithm, into theoretical computational machines commonly known as Turing machines (Turing 1937).

Turing machines are comprised of a tape, with uniform spaces for reading and writing symbols of a defined alphabet to and from the tape by way of a read head. In addition to the defined alphabet of symbols, a Turing machine has a defined set of states and a defined set of rules that contain instruction for how the machine will respond to the combination of the symbol under the read head and the current state. The machine operates in cycles of reading a symbol from the tape, modifying the symbol under the read head, and moving the tape to the left or right, determined by the state-symbol pair found in the transition rules. Importantly, all of the information for performing any given computation is given at initialisation, and the machine can then embody an effective procedure to complete the computation. Turing demonstrated that a Universal Turing Machine (UTM) could be constructed which was capable of reading a description of and then *simulating* any other Turing machine. A UTM represents all computable functions within a given logic. To answer the Entscheidungsproblem, Turing sought a computation that could not be resolved using a UTM. Turing showed that for a given UTM U, that takes descriptions of Turing machines and initial inputs and tries to decide whether they halt, there will always be some combination of TM and input such that U cannot decide whether the computation will halt or not. Importantly, the TM and input combinations that cannot be determined to halt depend on the given logical system such that for any given UTM U that cannot decide whether some combination of TM and inpit halts, there will

be another UTM U' that can decide whether it halts. The Turing machine was therefore a demonstration of consistent enumeration of any system of axioms and theorems, which nevertheless could exhibit undecidability.

Due to Gödel's first incompleteness theorem, we know that for every enumerable logical system, there will be propositions that cannot be proved or disproved within the axioms of that system. Turing demonstrated undecidability by way of a computation that could not be determined to halt within finite time (i.e. the 'halting problem'). An alternative form of undecidability is the creation of a paradox: a logical statement that cannot be resolved without contradiction. The Liar paradox is a demonstration of just such a proposition. In its simplest form, the Liar paradox expressed in language is: 'this sentence is false', which is a self-referential statement which cannot be determined to be true or false without a contradiction arising. Prokopenko et al. (2019) compared equivalent Liar paradox configurations within Gödelian formal logical systems, Turing machines, and dynamical systems in the form of cellular automata (Wolfram 1984; Sutner 2012). For both the Turing machine and cellular automata, an equivalent to the Gödelian sentence was found in the universal 'decider' and 'inverter' computational constructions, which are described in detail in the Ansatz of chapter 3. In brief, a universal decider is a machine which gives a Yes or No answer, by simulating any described machine, as to whether this machine will complete a given computation, and the inverter is a decider with inverted output. The Liar paradox can occur when these decider-inverter constructions are self-referentially run on their own description. Understanding of self-reference in computable functions was significantly progressed by Rogers (1967) and Kleene (1938) in the context of recursive functions. In particular, Kleene's second recursion theoreom has become a widely used tool when demonstrating self-referential dynamics (Moschovakis 2010).

These results are a significant foundation for this thesis, as the Liar paradox represents a computational blueprint for utilising self-reference to generate undecidability in universal computation. These results also show that equivalence can be drawn between paradox in formal logic and computational constructions of dynamical systems. The RNA automata described in this thesis fall into the category of dynamical systems, and as such, if universality

in such a system can be demonstrated, then it can be expected to be capable of embodying the Liar paradox and hence, undecidable computation.

Demonstrating universal computation and the halting problem was not the final word from Alan Turing on this topic. In 1939, two years after the publication of his demonstration of computational undecidability, Alan Turing published a follow-up work on 'Ordinal Logics' (Turing 1939), developed as part of his thesis under Alonso Church. Ordinal logics are an important concept in computability theory, as this concept utilises a key insight of Gödel's incompleteness theorems; that undecidability is defined only in relation to the formal system which produces it. This insight opens a way to demonstrate a sequence of progressively expanded formal systems. Turing was exploring logical statements of the form 'for all' $\forall$ with an effectively decidable predicate $R(x)$, i.e. $\forall x R(x)$, in the universal sense of 'for all, there exists' $\forall\exists$. Ultimately Turing demonstrated that there are statements of the form $\forall x \exists y R(x, y)$ which cannot be computed. In response to this, Turing proposed an 'oracle' that could solve specifically the unsolvable proposition:

> Let us suppose that we are supplied with some unspecified means of solving number-theoretical [i.e., $\forall\exists$] problems; a kind of oracle as it were. ... With the help of the oracle we could form a new kind of machine (call them o-machines), having as one of its fundamental processes that of solving a given number-theoretic problem...

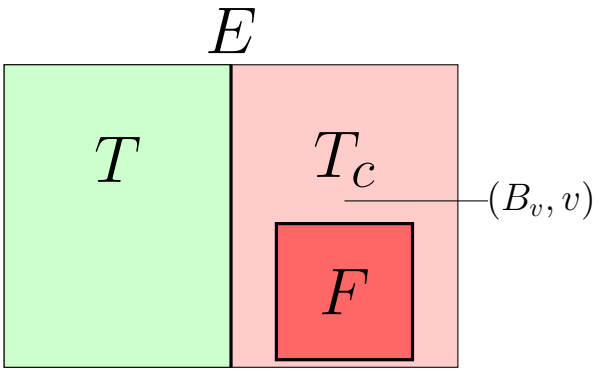
Turing used his o-machines to show that determining whether an oracle machine terminates on any given input is not computable by any oracle machine, and therefore not capable of being resolved by the oracle itself. The conclusion from this insight is that while a specific undecidable computation may be resolved with a specific oracle machine, there will in turn be another configuration that is not resolvable, in an iterative fashion.

In 1944, Emil Post published the influential work 'Recursively Enumerable Sets of Positive Integers and Their Decision Problems' in which he recognised Turing's oracle machines as a precise formulation of recursive unsolvability, or 'relative computability' (Post 1944). Instead of computational automata, Post worked with equivalent recursive functions of positive

integers which generated 'effectively enumerable' sets of positive integers. In the manner of Gödelian logic, Post established that the recursive functions could be represented as 'normal systems' with a defined alphabet and the integers. Normal systems were therefore themselves recursively enumerable. A normal system, i.e. a recursive function represented as symbols in a formal logic, was denoted a Basis, and it follows that the set of Bases is enumerably infinite. A decision problem was then defined as determining whether some positive integer n is a member of the recursive set generated by a Basis B, i.e. whether (B, n) is the true or false statement "n is in the set generated by B". Post demonstrated that the decision problem for all recursively enumerable sets of positive integers is recursively unsolvable.

The demonstration is briefly summarised here as it leads to an insight for an incorporation of an axiom (Post 1944). In the first step, the set of all distinct couples (B, n) is recognised to comprise the set E, with 1:1 correspondence to the positive integers. The subset of E comprising only true statements, T, is found by enumerating each set $B_i(1), B_i(2) \ldots$ and adding the couples which are true statements to T. T is therefore a recursively enumerable subset of E. Now, the set F is defined to be any recursively enumerable subset of false statements in the complement of T, T_c, enumerated in the same manner as T. A contradiction is established from the assumption that all couples (B, n) can be determined to be in T or F. This has similarity to the self-referential diagonalisation argument, typically used to construct equivalents of the Liar paradox in computational frameworks (Prokopenko et al. 2019). A 'diagonal' set S_0 is constructed by checking, for every (B, n) in F, if B is the nth basis in O, the set of Bases. That is, S_0 is the set of positive integers for which (B_n, n) is false. S_0 is recursively enumerable, so it must have a corresponding basis, which is denoted B_v, and a corresponding couple (B_v, v). Out of this self-reference arises a contradiction: suppose (B_v, v) is in F, such that (B_v, v) is false, i.e. v is not in the set generated by B_v, which means that v is not a member of S_0. However, S_0 is the set of all false statements (B_n, n), and if (B_v, v) is false, v must be a member of S_0. The assumption that (B_v, v) is in F must be false. From this it can be concluded that for any recursively enumerable subset $F \in T_c$, there is a couple (B_v, v) which is a member of T_c but not F. As such no recursively generated logic relative to the set of all expressions E is complete and the decision problem as applied to all recursively generated sets of positive integers is recursively unsolvable.

FIGURE 2.2: Illustration of distinct statements (B, n) comprising the set E. The set F of provably false statements is a subset of T_c, the complement of provably true statements T. Post demonstrated that the assumption that all statements can be determined to be in T or F leads to contradiction. Specifically, in the 'diagonal' set S_0, comprising all of the positive integers for which the statement (B_n, n) is false, which is generated by the basis B_v. The assumption that the couple (B_v, v) is in F creates a contradiction. The result is that (B_v, v) must be a member of T_c, but not a member of F.



This takes us up to demonstrating undecidability, but it is the next observation that holds significance for our Ansatz proposition of boundary expansion. Returning to Post's demonstration, if we presume we have a basis which generates F, the recursively enumerable 'diagonal' set S_0 will have a basis, denoted B_k. We can search for B_k in the set of Bases, and directly determine (B_k, k), which resolves the contradiction and extends our logic. In this manner, Post has shown that every formal system is incomplete *and extendable*. Importantly, this demonstration also showed that it is precisely the undecidable proposition that could be added to the system as an axiom.

The progression of understanding from Gödel's incompleteness theorem, to Turing's demonstration of universal computation and computational undecidability, and finally to oracle-machines and Post's 'degrees of unsolvability' traces a thread from recognising incompleteness in formal systems to overcoming incompleteness by extension. This thread has been recognised and described by Markose (Markose 2022; Markose 2017; Markose 2021) as the Gödel-Turing-Post (GTP) framework. Recalling the motivating question of this thesis as a principle behind biological system expansion, the GTP framework provides a way forward if universality and undecidability can first be demonstrated for biological systems.

CHAPTER 3

An Ansatz for computational undecidability in RNA automata

An Ansatz for computational undecidability in RNA automata

Adam J. Svahn*
University of Sydney
Faculty of Engineering
Centre for Complex Systems
Faculty of Medicine and Health
Westmead Clinical School
adam.svahn@sydney.edu.au

Mikhail Prokopenko
University of Sydney
Faculty of Engineering
Centre for Complex Systems
Sydney Institute for Infectious Diseases

Abstract In this ansatz we consider theoretical constructions of RNA polymers into automata, a form of computational structure. The bases for transitions in our automata are plausible RNA enzymes that may perform ligation or cleavage. Limited to these operations, we construct RNA automata of increasing complexity; from the Finite Automaton (RNA-FA) to the Turing machine equivalent 2-stack PDA (RNA-2PDA) and the universal RNA-UPDA. For each automaton we show how the enzymatic reactions match the logical operations of the RNA automaton. A critical theme of the ansatz is the self-reference in RNA automata configurations that exploits the program-data duality but results in computational undecidability. We describe how computational undecidability is exemplified in the self-referential Liar paradox that places a boundary on a logical system, and by construction, any RNA automata. We argue that an expansion of the evolutionary space for RNA-2PDA automata can be interpreted as a hierarchical resolution of computational undecidability by a meta-system (akin to Turing's oracle), in a continual process analogous to Turing's ordinal logics and Post's extensible recursively generated logics. On this basis, we put forward the hypothesis that the resolution of undecidable configurations in RNA automata represent a novelty generation mechanism and propose avenues for future investigation of biological automata.

Keywords
Automata, RNA automata, self-reference, undecidability, Turing's oracle, novelty generation

1 Introduction

Undecidability is an important concept in the theory of computation, where, for certain problems it can be shown that an algorithm always generating a definitive answer is impossible to construct (Gödel, 1931; Turing, 1937b). In other words, it is impossible to always decide that a given computation halts or runs forever. Undecidable dynamics, generated by self-referential relationships, have also been implicated in chaos theory and complexity science (Prokopenko et al., 2019). In this work, we pose the question of whether undecidability can be demonstrated for biological systems.

Specifically, in this ansatz we investigate the computational properties of RNA-based systems. With the minimal RNA-mediated functions of ligation and cleavage (Evans et al., 2006; Jarrous, 2017; Robertson & Joyce, 2014; Will & Luhrmann, 2011) we construct theoretical RNA automata with equivalence to a finite automaton and to push-down automata with one or two stacks, with demonstration of computations achievable within each construction. Importantly, in an RNA

* Corresponding author.

automaton, the ribozymes that constitute the transition rules (the program) and the polymers that serve as symbolic memory (data) are both composed of the same nucleotide substrate. We describe how, from this shared substrate, an RNA-based *universal push-down automaton* (RNA-UPDA) with equivalence to a universal Turing machine (UTM) may simulate any encoded RNA automaton program.

Our main technical objective is to describe an RNA-based computational framework that enables an encoding and decoding relationship that will facilitate the emergence of self-reference. It is useful to distinguish self-replication and self-reference as distinct concepts. For example, remarkable self-replicating mineral crystals that propagate patterns of inhomogeneities from layer to layer and reproduce by random fragmentation (Cairns-Smith, 1966; Schulman et al., 2012) are not self-referential, because the decoding relationship itself is not represented in an encoded form (McMullin & Hasegawa, 2012). On the other hand, computer programs with self-replicating code (e.g., cellular automata) can be fully self-referential by employing *explicit* encoding/decoding mechanisms. Self-reference, unlike self-replication, generates a form of undecidability exemplified by the Liar paradox wherein a self-negating statement is unsolvable within the bounds of the system. In computability theory this is manifested as the halting problem and implicated in studies of novelty generation and open-ended evolution (Abrahão et al., 2019; Adams et al., 2017; Kauffman, 2016; Markose, 2004, 2017; Prokopenko et al., 2019; Zenil et al., 2016). The relation between self-reference and recursive self-representation in a biological context is emphasised by Hofstadter (1980), who highlighted that a biological self-referential system cannot be consistent. In this work we present a series of biologically plausible classes of RNA automata that reach the level at which such inconsistency and the corresponding computational undecidability is ultimately manifested. We then argue that this undecidability is framed only within specific bounds, i.e., within the corresponding formal system, and can be resolved by extending the bounds as a result of interactions between the organism and its environment, that is, as an evolutionary novelty generated by these interactions.

Goldenfeld and Woese (2011, p. 386) in particular focused on self-reference to drive at the question of biological innovation:

> ...what is to us the central aspect of evolution: It is a process that continually expands the space in which it operates through a dynamic that is essentially self-referential.
> Self-reference should be an integral part of a proper understanding of evolution, but it is rarely considered explicitly.

We ultimately hypothesise that an RNA automaton utilising two stacks would be capable of self-reference and so, would lead to the generation of an auto-negating undecidable "Liar paradox," recognising and resolving which would then allow the system to expand its boundaries. We hope to contribute to the early, but already productive investigations of computational principles in biological systems (reviewed in Prohaska et al., 2019), such as the "chromatin computer" (Arnold et al., 2013). The computational approach helps us to understand extant life but also to look back at the origin of life, in particular the investigation of the evolution of evolvability (Virgo et al., 2017). The methods found at this interface of synthetic biology and simulated artificial life represent a promising test bed for construction of computational and *in vitro* models that create self-referential, and paradoxical, scenarios from which the system must "jump out" and break the paradox by abstraction, or meta-level, simulation.

2 Background

An important aim of the ansatz is to develop constructions that were plausible, meaning that the enzymes and reactions required should be already known to exist within the chemistry of single-stranded RNA molecules. In keeping with this constraint, we designed each RNA automaton to include a collection of ligation and cleavage enzymes that together form a system of reaction

profiles. RNA-based ligation and cleavage are well demonstrated, and some background for this is given below. We will also give a brief background of automata theory, definitions of RNA automata components, and a further discussion of physical assumptions for the RNA automata.

2.1 RNA Ligation and Cleavage Enzymes

The ribonucleotide monomer is a modular construction of a ribose sugar, a phosphate, and a nucleobase. The nucleobase is the information carrying unit of the ribonucleotide. The nucleobases are divided into the purines (adenine and guanine) and the pyrimidines (cytosine and uracil), which form pairwise affinity relationships by hydrogen bonding: adenine to uracil and guanine to cytosine. Importantly, RNA polymers are structurally labile, readily forming complex tertiary structures. These structures create chemical micro-domains that allow the RNA polymer to act as an enzyme, facilitating a chemical reaction. RNA ligases (bond forming) and RNases (bond breaking) make use of the hydrolysing aqueous environment to catalyse the forming or the breaking of the phosphodiester bonds that link the ribonucleotide monomers together.

Ligation is a catalysed reaction that forms a bond between RNA polymers, being the joining of two polymers linearly, and is an essential function in all extant life. Remarkably, novel ligase ribozymes can be generated by *in vitro* evolution (Ekland et al., 1995; Joyce & Szostak, 2018). The R3C ligase was evolved out of a library of 10^{14} short RNA polymers (Rogers & Joyce, 2001) and constituted a 74nt RNA polymer enzyme that ligated a target RNA polymer to itself. Importantly, this study demonstrated that a shorter R3C motif of 57nt containing the catalytic site could ligate together two opposing RNA polymers and release the product. This property was exploited to demonstrate that a redesigned R3C ligase could ligate split copies of itself (Paul & Joyce, 2002), starting an autocatalytic replication cycle (Lincoln & Joyce, 2009; Paul & Joyce, 2002).

We may then ask what functional role could ligation perform in a computational RNA system? The product of any ligation is the generation of an RNA polymer that is longer than the components with which the reaction began. Ribozymes perform catalytic roles through the formation of secondary and tertiary structures, and the ribozyme formed from a long polymer may possess a greater propensity to form more complex and stable structures than that of a short polymer. The sequence of RNA nucleotides may also encode information to represent previous visited states of the system, with longer polymers having the potential to encode a longer sequence of symbols. The modular nature of RNA means that ligations may explore a large combinatorial space, limited by the available polymer reactants and the binding properties of the available ligating ribozymes. In combination, it can be hypothesised that ligase reaction cascades may be capable of constructing new ribozymes of increasing complexity, as well as encoding and extending symbolic representations within the system.

If encoding is performed by ligation, then decoding is subserved by cleavage, the splitting apart of an RNA polymer. All kingdoms of life retain a core RNA cleavage enzyme in RNase P. RNase P is a ribonucleoprotein that cleaves opposing single-stranded RNA and may bind and cleave multiple targets without losing function (Reiter et al., 2010). Structural and functional study of RNase P has resulted in a consensus that the RNA component was likely to have been present in the last universal common ancestor (LUCA; Chen & Pace, 1997). A synthetic approach to ligate the minimal catalytic unit of RNase P (Waugh et al., 1989), labelled M1, to a guide sequence (GS), produced an M1GS that performs targeted cleavage (Derksen et al., 2015). Crucially from a computational and synthetic viewpoint, the M1GS approach enables complete *in vitro* reactions. By drawing on the now large library of known RNase P sequences, artificial *in vitro* selection may explore an enormous space of GS targeting (Zou et al., 2004).

2.2 Automata Theory

Automata theory is the study of mathematical models of computation. It is important to recognise that the definition of computation used here goes beyond the design of computing devices to the mathematical formalisation of an algorithm as an effective procedure for performing a calculation

(Turing, 1937a). Full reviews of automata and computational theory are found in the canonical texts by Hopcroft and Ullman (1979) and Sipser (2006).

For our purposes in this ansatz, we briefly establish that each model of computation, or automaton, is defined as an *n-tuple*, meaning it is composed of n distinct components. To illustrate by example, an automaton called a finite state system is a 3-tuple (Q, Σ, δ) where Q is a set of possible internal states of the system and Σ (sigma) is the set of possible distinct inputs to the system. The internal state of the system may change according to a given transition function δ (delta), which maps the current state to a new state dependent on the observed input. This mapping is written as $Q \times \Sigma \rightarrow Q$. We observe the state and consider this to be the output of the system in response to the input. In this conceptualisation, we imagine the system transitioning within a space of all possible configurations of the state, occupying one of these locations at any given time. The state transitions occur in discrete steps, meaning it is always at a single point, never in between points. Looking at our system during its journey through this space of all possible configurations, we may say that the system state is determined by the past inputs, which in turn guides the next transition in response to input. In this ansatz, we will be applying standard constructions from automata theory, imagining how they might be instantiated with RNA polymers.

3 Definitions

RNA enzymes. The steps of the computation, referred to as transitions of the automaton, consist of modular additions and subtractions to the RNA polymer(s) that represent components in the automaton. The reactions allowed are ligation, the joining of RNA polymers, and cleavage, the dissociation of an RNA polymer into parts. It is assumed that a desired ligation or cleavage RNA enzyme is available for any given target RNA polymer(s).

States, symbols, and stacks. The state of the automaton is represented by a designated RNA polymer, termed the *state polymer*. The sequence of this polymer represents the current state of the automaton at any given time during the computation. The input to our RNA automata will be the sequential presentation of designated RNA polymers, termed *symbol polymers*. These polymers are not enzymes, rather they represent symbols in an alphabet defined within the automata. The sequential presentation will be referred to as the *input word*. In the second and third iteration of our RNA automata, we will introduce an extensible memory in the form of a stack for storing symbol polymers. The symbols may be added and subtracted from the stacks by the actions of the ligation and cleavage enzymes in the same manner as for the state polymer. All of the modifications to the state and to the stack(s) are modular operations.

Assumptions. There are multiple possible implementation paths for the automata constructions explored here. Rather than an exploration of experimental design, our aim was to propose constructions that place RNA dynamics within the class of automata. The goal is for the reactions proposed to be the simplest possible reactions, exemplifying an idealised "perfect world" reaction process. In the real world such reliability is possible, but cannot be easily achieved due to environmental noise, off-specific reactions (where a small percentage of reactions occur on targets that resemble the desired substrate), concurrent non-sequential reactions, and reverse reactions amongst other sources of variation. It is therefore probable that a benchtop implementation of these automata, or an example of an automaton in early life, would require more nuanced designs. Importantly, while these more nuanced designs may involve more reaction steps or more components, these would be following the same idealised reaction process, embodying the computational dynamics.

To that end we make the following assumptions:

- The reaction volume is imagined to consist of RNA molecules suspended in an aqueous solution.

- For polymers representing the given alphabets, we assume that it is possible to generate the corresponding sequences with sufficient stability in order to fulfil the role of unique substrates (i.e., non-enzymatic polymers).
- Each RNA polymer enzyme initiates only a single reaction.
- All possible reactions are assumed to go to completion (i.e., the reactants are used up completely).
- The reactions of the transitions do not generate reverse, off-specific reactions, or reactions triggered by environmental noise.
- A mechanism exists that makes input polymers available to the automaton in a sequential manner. This ensures that at the start of each new transition, a single polymer is drawn from the given sequence of input polymers, and made available to the automaton.
- For any given ligation and any given cleavage reactions, there exists an RNA polymer enzyme to initiate this reaction, not conditional on previous reactions. That is, for any RNA polymers a and b, there exists an RNA polymer enzyme x that ligates a to b. Similarly, for any RNA polymer c, there exists an RNA polymer enzyme y that cleaves c into given sub-polymers d and e.
- Between transitions, the reaction volume is in an inert state, prior to the introduction of a new input which marks the start of a new transition.
- Transitions do not require the resolution of "race conditions," where the order of possible reactions at the start of a transition may influence the configuration after the transition. For example, if an input polymer is both modified and placed on a stack, these reactions can occur in either order to produce the same outcome.
- Stack polymers are distinguishable to the automaton as modular units, i.e., there is a signal to indicate the beginning and end of stack polymers. When multiple stacks are utilised, the automaton can distinguish between the stacks.
- The reactions profile of the transitions can proceed without consuming an input polymer from the input stream or without cleaving a stack polymer, or without ligating a polymer to the stack.
- Accept and reject states are designated as specific polymers before automaton construction (further detail below) and are assumed to be distinguishable by an external observer.

4 RNA Automata

An automaton is an abstract construction for performing a computation. We will start with a *finite automaton (FA)* in which only the state polymer is modified in response to the input word. We will then iterate to add one and two stack polymers. At each automaton type, we will first give a theoretical background and notation from automata theory. We then outline the construction of the given RNA automaton and give worked examples of a computation.

4.1 Finite Automata

Background. A FA progresses through sequential transitions, where the state may change in response to the input. The transitions are carried out with reference to a defined set of transition rules for moving between any given state in response to the input. Certain states may be designated to have meaning with respect to the input word, e.g., an Accept or Reject state may be reached and, if halted on, signify a response to the total input.

A FA is defined by a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$. Q is the finite set of states that the automaton may visit. Σ is the alphabet that the input may be drawn from. δ is the transition function, the rules for moving between states, of the form $Q \times \Sigma \rightarrow Q$. q_0 is a designated starting state, where $q_0 \in Q$. F is the set of accept states, where $F \subseteq Q$.

RNA-FA components. The transition function holds the instructions for manipulating the state in response to the current input. Rules within the transition function take the form $(q_i, a) \overset{R}{\rightarrow} (q_j)$, which means that, for automaton R, if the current state is $q_i \in Q$, and the current input is a letter of the alphabet $a \in \Sigma$, then the automaton will change state to $q_j \in Q$. In an RNA-FA, the transition rules are embodied in the reaction profile of RNA enzymes. In the example, when the state polymer has sequence q_i, and the symbol polymer with sequence a is the current input, a ligation or a cleavage reaction occurs to the state polymer such that it is lengthened or truncated to become the sequence q_j.

The computation of the automaton proceeds in a series of steps with defined stages, starting from an inert point either at the initialisation of a new automaton or after the conclusion of a previous transition. First, a symbol polymer from the input is introduced and may be recognised as a pair with the current state polymer sequence q_i. A stage of reactions occur to completion, which may alter the state polymer and thus change the state of the automaton. A final "cleanup" stage resets the reaction volume to an inert state, prior to the introduction of a new input which marks the start of a new step.

RNA-FA notation. RNA enzymes in our automata perform ligation or cleavage reactions, which are denoted by λ (lambda, ligation) and μ (mu, cleavage) respectively. The first term in a recognition pair is the subject of the reaction that will be ligated to or cleaved from. For ligation, the second term in the recognition pair is directly ligated to the first or acts as a catalytic element to facilitate modification of the first term. For cleavage, the second term is a catalytic element.

For ligation in the RNA-FA:

$$\lambda(x,y) : Q \times \Sigma \rightarrow Q \tag{1}$$

i.e., $z = \lambda(x,y)$ where z is the state polymer such that z is the ligation of x with y or catalysed by y.

Similarly, for cleavage in the RNA-FA:

$$\mu(x,y) : Q \times \Sigma \rightarrow Q \tag{2}$$

i.e., $z = \mu(x,y)$ where z is the state polymer such that z is the cleavage of x catalysed by y.

We also define a stasis operation, where the response to an input is to remain in the same state:

$$\kappa(x,y) : Q \times \Sigma \rightarrow Q \tag{3}$$

i.e., $x = \kappa(x,y)$ for all y. The transitions are illustrated in Figure 2 as an accompaniment to the worked example of the RNA-FA.

4.1.1 RNA-FA for $b^*(ab^+)^*$

To perform a computation, we will encode the required alphabet for our automata into unique symbol polymers drawn from the ribonucleotide ACGU alphabet with minimum length determined as required for RNA enzyme activity. We will also encode the unique initial state polymer in the same format. We will then design RNA enzymes that will transition the state polymer through the designated state sequences in the presence of the symbol polymers.

To illustrate, the RNA-FA we are constructing is to determine whether a specific ordering of symbol polymers, the input word, conforms to a pattern. Our RNA-FA will recognise input

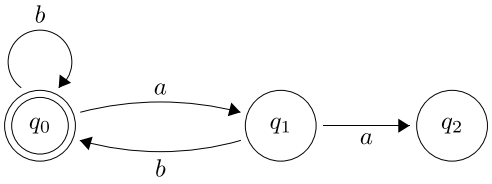


Figure 1. State diagram for the RNA-FA.

sequences of the form $b^*(ab^+)^*$. (See Figure 1.) The * indicates "0 or more of" and the $^+$ operator indicates "at least 1 of." Put together, $b^*(ab^+)^*$ indicates the input polymer may have an arbitrary arrangement of *b*'s, but any instance of *a* must be followed by at least 1 *b*. An empty sequence, or a sequence consisting only of *b*'s should be accepted by this definition. A pair of specific RNA polymers will represent *a* and *b*, forming Σ, from which an ordering of such polymers is chosen as the input word. The reactions cascading from the sequential presentation of the input of the form $b^*(ab^+)^*$ will result in reaching (or remaining in) a sequence of the state polymer designated as the accept state, and any non-conforming input words will reach a reject state. At the exhaustion of input, the sequence of the state polymer determines the acceptance or rejection of the input sequence.

The RNA-FA is represented as a five-tuple, $(Q, \Sigma, \delta, q_0, F)$, where:

Q is the set of states $\{q_0, q_1, q_2\}$ where each q_i is a unique sequence of the state polymer.

Σ is the alphabet $\{a, b\}$ where *a* and *b* are unique symbolic RNA polymers.

F is the set of accept states $\{q_0\}$

The transition function δ is given by the following transitions:

$$\delta = \begin{cases} (q_0, a) = q_1 \equiv \lambda(q_0, a) \\ (q_0, b) = q_0 \equiv \kappa(q_0, b) \\ (q_1, a) = q_2 \equiv \lambda(q_1, a) \\ (q_1, b) = q_0 \equiv \mu(q_1, b) \end{cases}$$

If we take as input *bab*:

1. With state polymer sequence q_0, *b* is a stasis symbol polymer.
2. With state polymer sequence q_0, *a* is recognised and the state polymer is ligated to form the sequence q_1.
3. With state polymer sequence q_1, *b* catalyses cleavage of the state polymer to return to the sequence q_0.
4. At the exhaustion of input, the state polymer has sequence q_0, so the automaton **accepts**.

If we take the input *aba*:

1. With state polymer sequence q_0, *a* is recognised and the state polymer is ligated to form the sequence q_1.
2. With state polymer sequence q_1, *b* catalyses cleavage of the state polymer to return to the sequence q_0.

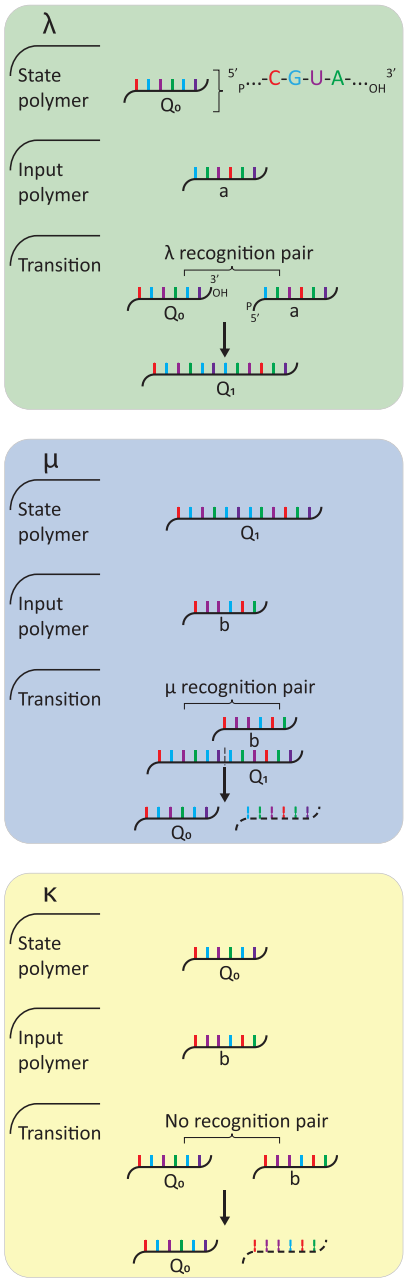


Figure 2. Illustration of λ, μ, and κ transitions. Top: The λ transition is a ligation reaction which creates a 3′5′ phosphodiester linkage between the state polymer and the input polymer to form a single polymer. The sequence of the new polymer corresponds to the state Q_1. Middle: The μ transition is a cleavage reaction which separates the state polymer. The sequence of the truncated state polymer corresponds to the state Q_0. The remaining polymer from cleavage is degraded or washed out prior to the next transition. Bottom: In the κ transition no enzymatic reaction takes place. The state polymer and the input polymer are not recognised as a template and do not catalyse a reaction. The input polymer is degraded or washed out prior to the next transition.

3. With state polymer sequence q_0, a is recognised and the state polymer is ligated to form the sequence q_1.
4. At the exhaustion of input, the state polymer has sequence q_1, so the automaton **rejects**.

RNA-FA computations. We may ask: What kind of computing tasks could such an RNA-FA perform? We may observe that at any given point during the computation, the current sequence of the state polymer describes a trajectory of visited states and inputs encountered. If the input word conforms to an accepted pattern, the RNA-FA will step through an accepting path. Formally, a FA may process the class of *regular languages*. Regular expressions, which describe regular languages, can specify *patterns* used in searching operations (Sipser, 2006). At the molecular scale, biology makes prolific use of regular expressions. In particular, the non-coding subset of the genome contains an enormous variety of patterns, referred to as motifs, which characterise families of genomic elements. For example, the upstream promoter region of a gene can be described as a regular expression such as $G[x]AT[x]AA[x]AT[x]CA$, where $[x]$ represents any of the nucleotides [AGCT] (in this case for the bacterial gene argR; McGuire et al., 2000). Identifying this phenomenon led to significant progress in the practice of scanning and annotating the genome for motifs (Brāzma, Jonassen, Eidhammer, & Gilbert, 1998; Brāzma, Jonassen, Vilo, & Ukkonen, 1998; Das & Dai, 2007). The capability to perform FA computations confers a powerful pattern recognition ability to this simple arrangement of RNA polymers.

A limitation of a FA is that any instance of returning to a previously visited state effectively erases the encoding of the trajectory beyond that state and as such the FA cannot maintain an extensible memory of repeated input. In other words, if a loop exists or the FA may return to some earlier state, there is no way to encode the number of times a loop has been traversed or a given state visited. In the next automaton, we will augment our RNA-automata with a polymer to serve as extensible memory.

4.2 Push-Down Automata

Background. To implement a memory component in our RNA automata, we will introduce RNA polymers with a purely symbolic, informational role. The automaton structure will be a push-down automaton (PDA) which operates along the same principles as the FA with states and transition rules. A PDA is augmented by the addition of a stack which can encode information over the course of the computation in an extensible manner. The stack operates according to a "Last In First Out" principle, in which symbols are prefixed to the top of the stack in a "push," and removed from the top of the stack in a "pop."

A PDA is defined by a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Q, Σ, q_0, and F are defined as above for the FA. With the addition of a stack, we now include Γ (gamma) as a finite set constituting the stack alphabet, and an initialising stack symbol $Z_0 \in \Gamma$. The PDA is defined with the empty symbol ϵ (epsilon). The input is defined as $\Sigma_\epsilon \equiv \Sigma \cup \{\epsilon\}$, in which ϵ may appear in place of input. When ϵ appears in the input, the transition may occur without reading a symbol from the input and without progressing to the next input symbol. Σ_ϵ is required for the full power of a deterministic PDA (Autebert et al., 1997). The stack input and output are defined as $\Gamma_\epsilon \equiv \Gamma \cup \{\epsilon\}$, in which ϵ may appear in place of the top stack symbol. When ϵ appears in the stack input place, the transition may occur without reference to the symbol on the top of the stack. When ϵ appears in the stack output, the transition proceeds without a symbol being placed on top of the stack.

The transition function δ is of the form $Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to Q \times \Gamma_\epsilon$, for example, $(x, y, u) \to (z, w)$, where $x, z \in Q$, $y \in \Sigma_\epsilon$, and $u, w \in \Gamma_\epsilon$. For clarity, we will make use of the instantaneous description notation (Hopcroft & Ullman, 1979). In this notation the automaton has a configuration (i.e., instantaneous description) which is the tuple of the current state, remaining input, and current stack contents. A transition from configuration $\delta(x, yL, uS)$ to configuration (z, L, wS) is indicated by the symbol $\vdash$ so that $(x, yL, uS) \vdash (z, L, wS)$. This means that during the transition from state x to state z, the first input symbol y of the input yL is "consumed," and the top symbol

u of the stack uS is replaced by symbol w, forming the new stack wS. Here $L \in \Sigma^*$ and $S \in \Gamma^*$, with Σ^* and Γ^* being the Kleene star of the input alphabet and stack alphabet respectively, which is the smallest superset containing all possible words derived from symbols in the input or stack alphabets, including the empty word. Incorporating ϵ, in the step-relation $(x, yL, uS) \vdash (z, L, wS)$, y, u and w may be ϵ. To maintain our automaton in a deterministic mode, we establish the rule that if a configuration $\delta(x, y, u)$ containing (z, w) exists, then the configurations $\delta(x, \epsilon, u)$ and $\delta(x, y, \epsilon)$ are empty. Similarly, a configuration $\delta(x, y, u)$ may contain only one of (z, w) or (z, ϵ).

RNA-PDA components. To realise a PDA we need to initialise an RNA polymer to operate as our stack. In overview, such a polymer would be a modular structure, consisting of symbol polymers drawn from the available alphabet Γ. Prefixing of a new polymer to the stack and popping the top polymer from the stack is carried out by the same class of ligating (λ) and cleaving (μ) enzymes as are already in use. Interpreting the ϵ in terms of the RNA-PDA, this means that the reaction profile of the transition does not include the respective ϵ component. For input, this means that the reaction profile is such that the reaction does not consume the input polymer from the input stream. For the stacks, this means that the transition reaction ignores the polymer on the top of the stack, and for the stack output this means that the transition reaction will not result in a polymer being ligated to the stack. In other words, distinct reaction profiles can differentiate between these separate kinds of transitions, instead of employing an explicitly designated polymer ϵ to ensure that the input stream, or the top of the stack, are not consumed, or that the stack contents are not updated. Shifting the design "burden" between reaction profiles and specialised symbols is characteristic of the program-data duality.

Finally, we will introduce special symbol polymers to indicate the end of the stack or of the input word. The stack is initialised with a special end-of-stack symbol polymer denoted by η (eta). All input words, including the empty word, have a special end-of-input symbol polymer in the last position, denoted by ν (nu).

RNA-PDA notation. PDA operation proceeds by two independent modifications of the automaton state and the stack. For simplicity, we use a general notation $\circ$ to represent any of the reaction functions, i.e., ligation, cleavage, and stasis, that is, $\circ \in \{\lambda, \mu, \kappa\}$. Consider a transition δ given by the instantaneous description $(x, yL, uS) \vdash (z, L, wS)$. The state modification produces a new state $z = \circ(x, \circ(y, u))$, given a suitable reaction between polymers y and u resulting in $\circ(y, u)$, and followed by another suitable reaction between polymers x and the intermediate result $\circ(y, u)$. This sequence is ensured by the ansatz assumptions (e.g., each RNA polymer enzyme initiates only a single reaction, all possible reactions are assumed to go to completion, reactions do not generate reverse and off-specific reactions, and are not triggered by environmental noise). The stack modification produces a new stack $wS = \circ(u, \circ(y, x))$, given a suitable reaction between polymers y and x resulting in $\circ(y, x)$ and followed by another suitable reaction between polymers u and the intermediate result $\circ(y, x)$. Again, this sequence is ensured by the ansatz assumptions. These two modifications eliminate the need for a reaction among three reactants: Instead we need only two sequential reactions with two reactants each, using suitable enzymes. Importantly, the choice of reaction type $\circ(a, b)$ between two polymers a and b is determined when the automaton employing this reaction is designed; that is, we do not expect this type to be discovered during the transition as a result of some additional conditions.

RNA-PDA computations. To construct an RNA-PDA, we have added a memory component and augmented our transition rules to read and write from this memory. A stack allows the RNA-PDA to perform cardinality and one-one correspondence tests between distinct components of an input. More generally the stack allows for an extensible counter. The extensibility of the memory is crucial to the power of the RNA-PDA. An RNA-FA could be designed to determine any given *finite* input word incorporating loops; however, each loop would require a distinct subset of

states and therefore come at the cost of a significant expansion in the state complexity. As such, we may observe that expanding from an RNA-FA containing exclusively functional components to include a component which serves a purely informational role, in the form of a stack, represents a significantly simpler path to recognition of more complex patterns containing loops.

In particular, computing with a stack has implications for encoding and decoding with an RNA-PDA. The process of writing and reading with the RNA-PDA can be formally captured within the *deterministic context-free languages* (DCFLs) (Sipser, 2006). The transition rules of an RNA-PDA can embody a deterministic context-free grammar (CFG) (Sipser, 2006). In general, the derivation of a string using a CFG requires intermediate strings which are stored on the PDA stack. Thus, the stack of an RNA-PDA can provide storage for intermediate polymers derived from the input polymer. As such, an RNA-PDA operating as a parser may encode information from the environment into a persistent RNA form, or decode some previously encoded RNA information. A deterministic PDA-based parser has been described (LR; Knuth, 1965) which could be implemented by the RNA-PDA. Additionally, the stack may also embody the computational result of an encoded or parsed string when an encoding/decoding PDA accepts an input.

Encoding and decoding in RNA automata is the first time we can observe a distinction and a relationship between the functional nucleic acid components carrying out the automata operations, and the informational nucleic acid components. Through encoding, a history of states and inputs can be recorded and retrieved through decoding. There is an important limitation in this relationship. In the RNA-PDA this informational representation lacks a reflexive relationship in which the information itself becomes the subject of manipulation. Such a reflexive relationship emerges when there is a cross-reference, an ability to copy and compare within the encoding. Put simply, the automata require an additional space for copying and manipulating a stored encoding in successive transitions. In the next iteration, we will further augment our RNA-PDA with an additional stack to permit this exploration.

4.3 Turing Machines and Two-Stack Push-Down Automata

Background. A Turing machine (TM) is an abstract general-purpose computing device, introduced as a formal model of computation, and intended to capture the entire class of computable functions (i.e., "algorithms") (Turing, 1937b). As a computing device, the TM surpasses the capability of PDAs, being the most powerful computing model. A TM uses a finite set of rules (program) which modify symbols on an infinite tape (data), with the latter distinguished from the stack by being accessible at any location along the tape. The tape is split into discrete cells each capable of holding a single symbol, or of being blank. The TM is conceptualised to have a *read-head* positioned at one cell of the tape, at which it may read and modify the symbol. At the end of each transition, the read-head may move one cell to the left or right along the tape. Formally, a TM is defined as a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where Q, Σ, and Γ are defined as the set of states, the input alphabet, and the tape alphabet respectively. q_0 is the starting state and q_{acc} and q_{rej} are the predetermined accept and reject states. The δ transition function is defined as $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ where L and R represent left or right movement of the read-head. Mechanically, the tape and read-head operation of the TM is a departure from the construction of the RNA automata in this study. For this reason, we will implement a TM-equivalent automaton, the deterministic two-stack PDA (2PDA). As the name suggests, a 2PDA may operate as the PDA above, with the addition of a second stack. Both stacks may be accessed during a transition, including switching a symbol between the stacks. The equivalence of a 2PDA to a TM may be demonstrated by simulation. A 2PDA may simulate a TM by assigning the two stacks to represent the portion of the tape to the left and right of the read-head. Interestingly, information storage on the two-stacks of a PDA is orthogonal to information storage via states, and it can be demonstrated that every TM has an equivalent deterministic, single-state 2PDA (Koslowski, 2013).

Standard constructions of PDAs and 2PDAs include a Σ input source external to the automaton. By contrast, TM construction incorporates the input as a buffered tape, where $\Sigma \in \Gamma$. These are not fundamental differences, and it can be observed that both constructions can be made equivalent

by explicitly designing a TM to buffer an input word from an external source prior to computation, or to buffer the stack of a 2PDA with the input word (Koslowski, 2013). In the RNA-2PDA, we will buffer the stack with the input word prior to computation.

A 2PDA is defined by a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, Z, F)$. We will refer to the two stacks as the *left* (L) and *right* (R) stack, and we will initialise both stacks with Z, where $Z = \Gamma_L^* \times \Gamma_R^*$. As we will be initialising our 2PDA with the input on the L stack, the input alphabet $\Sigma \in \Gamma$. The δ function is of the form $\delta : Q \times \Gamma_\epsilon^L \times \Gamma_\epsilon^R \rightarrow Q \times \Gamma_\epsilon^L \times \Gamma_\epsilon^R$, with ϵ-transitions as above.

RNA-2PDA components. To realise a 2PDA we will need to add a second stack polymer, initialised with a unique end-of-stack symbol. To enable the selective popping from and pushing to each stack, we will assume that, within the automaton, a symbol polymer on the left stack will be distinguishable from a symbol polymer on the right stack.

RNA-2PDA notation. As we now have two stacks, we will re-define the end-of-stack symbols to be end-of-left, ν, and end-of-right, η.

Possessing two stacks that can serve as storage and processing space enables an automaton to perform repeated computation on iterations of intermediate results. A worked example of an RNA-2PDA is given in Appendix 2 for a^{2^n}.

RNA-2PDA computations. The languages computable by a TM or 2PDA are referred to as the *recursively enumerable languages* (Sipser, 2006). The recursion theorem demonstrates that an automaton that can read non-destructively and perform copying and comparison operations, such as a 2PDA, can derive its own description and compute with it (Sipser, 2006). Specifically, if an RNA-2PDA, R, was designed to utilise recursion, then it may encode its own transition rules [R] onto the stacks. The aim may be to self-reproduce, or R may go on to perform any 2PDA computation on [R] and any other input.

To consider this dynamic in our class of RNA-2PDA automata, recall that we considered the single stack of the RNA-PDA to represent a non-reflexive relationship between the functional components and the informational components encoding the history of states and inputs encountered. Such stored information could modify the progression of a subsequent computation but not itself become the subject of manipulation. In the RNA-2PDA, the second space for copying and comparison allows this information to be manipulated. The recursive ability of the two-stack automaton to encode its own transition rules on the stacks means that these symbolic RNA polymers may be copied and modified as with any other polymers on the stacks. The automaton may be represented in both a functional and informational form, with encoding and decoding allowing a reflexive relationship between the representations.

Importantly, the RNA-2PDA may hold an informational representation of any RNA enzyme automaton in the form of an abstracted representation of the internal relationships of the automaton, i.e., as an alphabet and transition rules. Such an encoding contains the information necessary for testing inputs and modifications of the encoding at the level of the encoding. In other words, some suitable automaton may *simulate* the computation of itself or another automaton from such an encoding. In fact, the ability to match and substitute across two stacks enables the RNA-2PDA to simulate *any* automaton through universal computation.

4.4 Universal Computation With an RNA-UPDA

Background. A universal automaton, such as the canonical UTM (Turing, 1937b), may read the description of any automaton and simulate that automaton on some input. There are three key components to universal computation. The first is generating the description of the automaton to be simulated by conversion of its transition rules and inputs into the alphabet of the universal automaton. The resulting description is referred to as an *encoding*. Second, the simulating automaton, in this case our RNA-UPDA, must be capable of manipulating the encoding to faithfully simulate

computation of the encoded automaton. Third, this simulation must be facilitated within a memory layout (on tape(s), stacks, etc.) that accommodates the encoding and the inputs and outputs of the simulated automaton. We will demonstrate an RNA-UPDA encoding strategy and procedures for handling the encoding, and explore how a simulated computation may be accommodated in a specialised RNA-UPDA 3-stack construction.

To show capability of the RNA-UPDA for universal computation, we will focus our attention on the known set of small, size efficient UTMs (Shannon, 1956; Woods & Neary, 2009). One strategy for the implementation of a size efficient UTM is to simulate 2-tag automata. 2-tag automata are a member of the m-tag automata (Minsky, 1962), which compute by modification of a word on a single linear tape. An m-tag automaton always reads the first symbol of the input word, deletes m symbols from the start of the input word, and then appends some symbol(s) to the end of the word. Importantly, any algorithm that can be computed by a TM-equivalent automaton can be computed by a 2-tag automaton (Cocke & Minsky, 1964). Formally, the 2-tag automaton is given by the 2-tuple (S, T) where S is the alphabet of unique symbols that may be read from or appended to the word being computed for $S = \{s_1, s_2, \ldots, s_n, s_{n+1}\}$ in which s_{n+1} is a halt symbol. The transition rules T map the members of S to the finite set of words S^*, which are appended to the input word. A transition rule is of the form $s_i \rightarrow \alpha_i$ for $i = \{1 \ldots n\}$ where $\alpha_i = s_{i_1} s_{i_2} \ldots s_{i_l}$ and in every transition two symbols are to be removed from the start of the word.

We will aim to demonstrate that our RNA-UPDA can simulate any 2-tag automaton on any input. First we will demonstrate an example UPDA alphabet into which the transition rules and input of any target 2-tag automaton may be encoded to be simulated by the RNA-UDPA. Second, we will describe RNA-UPDA functions for matching and copy operations that are required to carry out the simulation. Finally, we will describe an RNA-UPDA with three stacks that may simulate any target 2-tag automaton by manipulating the encoded alphabet. We aim to demonstrate that an RNA-UPDA may simulate any 2-tag automaton by demonstrating that the simulated 2-tag input word during computation and after halting are in concordance with that which would be observed in the target 2-tag automaton.

RNA-UPDA components. To first encode the finite alphabet S of the target 2-tag automaton into the alphabet of the RNA-UPDA, we will encode each symbol of S as complementary pairs of nucleic acid polymers.

Alphabet to symbol polymer encoding. Every $s_i \in S$ is assigned a pair of complementary symbol polymers denoted a_i and $\bar{a}_i$. *Complementary* refers to the property that for all j, the nucleotide at each position a_{i_j} is matched at $\bar{a}_{i_j}$ by the complementary nucleotide to which it preferentially binds (i.e., $C \leftrightarrow G$ and $U \leftrightarrow A$).

The alphabet of the target 2-tag automaton (S) is therefore encoded into the RNA-UPDA alphabet $\mathcal{A} = \{\{a_1, \bar{a}_1\}, \{a_2, \bar{a}_2\}, \ldots, \{a_n, \bar{a}_n\}, \{a_{n+1}, \bar{a}_{n+1}\}\}$, where the symbol polymer a_{n+1} represents a halt symbol.

Input word to input polymer encoding. The input to the RNA-UPDA consists of a word, denoted as K, composed of the a_i members of the encoded pairs of symbol polymers in the RNA-UPDA alphabet $\mathcal{A}$. Put together, the input of the RNA-UPDA will be the modular polymer $a_{K_0} a_{K_1} a_{K_2} \ldots a_{K_n}$.

Transition rule to instruction polymer encoding. The polymers that will serve as instructions for modifying the input, denoted as D, will be composed of the $\bar{a}_i$ members of the encoded pairs of symbol polymers in the RNA-UPDA alphabet $\mathcal{A}$. The special symbol polymer $\triangleright$ will serve to demarcate the start and end of each instruction polymer. It will be a requirement of encoding that every pair $\{a_i, \bar{a}_i\} \in \mathcal{A}$ is associated with exactly one instruction polymer $d_i \in D$

(Rogozhin, 1996). Put together, every $d_i \in D$ for $i = 1 \ldots n$ is of the form $\triangleright \bar{a}_i \bar{\rho}_i$ where $\bar{\rho}_i = \bar{a}_{i_0} \bar{a}_{i_1} \ldots \bar{a}_{i_l}$ for $l \geq 1$.

To initialise the stacks for our RNA-U2PDA, we generate instruction polymers of D and push each to the *instruction* stack, with end-of-stack symbol ν. A special symbol polymer $\triangle$ will be initialised at the top of the instruction stack and will serve to demarcate the boundary of the instruction polymers and the input polymers. For input, we generate the input polymers of K and push each to the *input* stack, with end-of-stack symbol η. We also initialise a third *working* stack, with end-of-stack symbol ω. This stack will take part as a temporary holding space during computation.

$$\text{Instruction} \left\{ \begin{array}{c} \triangle \\ d_0 \\ d_1 \\ \vdots \\ d_n \\ \nu \end{array} \right\} D \quad A^* \left\{ \begin{array}{c} a_{K_0} \\ a_{K_1} \\ \vdots \\ a_{K_m} \\ \eta \end{array} \right\} \text{Input}$$

$$\text{Working} \left\{ \omega \right\}$$

To simulate a 2-tag automaton transition, the RNA-UPDA first engages in *matching* (search, comparison) between the topmost input symbol polymer of the input stack at the start of the transition and the LHS (left-hand side) of the instruction polymers, $\bar{a}_i$ on the instruction stack. After identifying the associated instruction polymer, the RNA-UPDA then engages a *copying* procedure to generate ρ_i from $\bar{\rho}_i$ on the RHS (right-hand side) of the instruction polymer. To enable these functions during the computation, we will utilise the binding between complementary polymers as a targeting mechanism, which is further outlined below.

Matching. The matching function in the RNA-UPDA simulates the transition rule lookup of the 2-tag automaton. When the matching function is invoked, it takes as input a single symbol polymer located on the top of the working stack (denoted a_{match}), identifies an associated instruction polymer on the instruction stack, and initiates the copy function. The matching function consists of a repeating cycle:

- a_{match} is popped from the working stack and allowed to bind to the LHS of the topmost instruction polymer of the instruction stack.
  - If the current instruction polymer d_i is associated with a_{match}, then the complementary binding $a_{match} \leftrightarrow \bar{a}_i$ will serve as the initiating signal for the copy function of the RNA-UPDA and a_{match} is discarded.
  - If no binding occurs, a_{match} is pushed back to the working stack and the topmost instruction polymer is temporarily cycled to the input stack.

Copying. The copy function is the first part of a two-step process within the RNA-UPDA that simulates the 2-tag automaton step of appending a new symbol or word to the input word. The copy function takes as input an instruction polymer and outputs a new polymer to the working stack:

- The input to the copy function is $\bar{\rho}_i$, located on the RHS of d_i. $\bar{\rho}_i$ serves as a template for the process of *template-directed ligation* (Doudna & Szostak, 1998) in which short, random sequence polymers present in the reaction volume, but not encompassed within A, align to $\bar{\rho}_i$ through complementary binding.

- As short polymers align to $\bar{\rho}_i$, these are ligated together by RNA ligases into the complete ρ_i polymer.
- The polymer ρ_i is pushed to the working stack.

RNA-UPDA notation. We will now demonstrate an example RNA-UPDA construction to simulate a 2-tag automaton. We will use a binary encoding, in which C/G corresponds to 1 and A/U corresponds to 0. This encoding will serve to illustrate the relationship of nucleotide encoding to binary representation, and to better illustrate the complementarity mechanism.[1]

Given a 2-tag automaton with alphabet $\{s_1, s_2, s_3, \ldots, s_n, s_{n+1}\}$ encode every $s_i \in S$ as:

$$(a_i, \bar{a}_i) \in A : \begin{cases} a_1 = C & 1 & \bar{a}_1 = G \\ a_2 = CA & 10 & \bar{a}_2 = GU \\ a_3 = ACC & 011 & \bar{a}_3 = UGG \\ a_4 = CAA & 100 & \bar{a}_4 = GUU \\ a_5 = CAC & 101 & \bar{a}_5 = GUG \\ a_6 = CCA & 110 & \bar{a}_6 = GGU \\ a_7 = ACCC & 0111 & \bar{a}_7 = UGGG \\ a_8 = CAAA & 1000 & \bar{a}_8 = GUUU \\ \vdots & \vdots & \vdots \\ a_{n+1} = halt & & \bar{a}_{n+1} = \bar{halt} \end{cases}$$

The transition rules of the simulated 2-tag automaton T will be encoded into the instruction polymers of the RNA-UPDA D. For example, let the 2-tag automaton transition rule t_1 be $s_1 \rightarrow s_6 s_7$. The encoded instruction polymer d_1 will be:

$$\underbrace{\triangleright}_{marker}\ \overbrace{G}^{\bar{a}_1}\ \underbrace{\sqcup}_{spacer}\ \overbrace{GGU\ \sqcup\ UGGG}^{\bar{\rho}_i}\ \underbrace{\triangleright}_{marker}$$

The input word of the simulated 2-tag automaton will be encoded into the symbol polymers of the RNA-UPDA that make up the input polymer K. For example, let the 2-tag automaton input word be $s_1 s_3 s_5$. The encoded input polymer K will be:

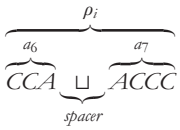
$$\overbrace{C}^{a_1}\ \underbrace{\sqcup}_{spacer}\ \overbrace{ACC}^{a_3}\ \underbrace{\sqcup}_{spacer}\ \overbrace{CAC}^{a_5}$$

1 For an instantiation, we assume that it is possible to generate sequences with no enzymatic activity within the RNA-UPDA and a maximum threshold for similarity that minimises off-target binding. For example, an encoding of each $s_i \in S$ to a random sequence (ACGU)* with lengths >18 nucleotides and $<70\%$ identity between any two sequences would be in keeping with accepted oligomer design to maximise specificity. Within these bounds there remains a very large space of unique sequences.

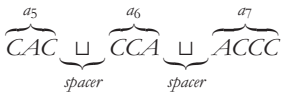
We will now demonstrate a transition of the simulated 2-tag automaton which updates the input word. The first symbol polymer of K will be popped and utilised in the matching function, and the second symbol polymer of K will be discarded.

Input to matching

In our example 2-tag automaton, the transition rule t_1 is the associated rule for the symbol s_1. The RHS of t_1 is α_1, which is the symbol or word that will be appended to the input word to complete the transition. In the RNA-UPDA, the RHS of d_1 is $\bar{\rho}_1$ which is not the direct equivalent of α_1. There is an extra step in the RNA-UPDA, in which $\bar{\rho}_1$ is the template for the template-directed ligation construction of ρ_1 during the copy function of the RNA-UPDA. ρ_1, the output of the copy function, is the equivalent of α_1:



To complete the transition, ρ_i is appended to K, via the working stack.



This completes the simulation of the 2-tag transition $s_1 \rightarrow s_6 s_7$.

RNA-UPDA procedure for simulating a 2-tag automaton transition. For each simulated transition of the 2-tag automaton, the RNA-UPDA will progress through the sequence of transitions below to modify the encoded input word. At each new cycle, $a_{K_{2j}}$ will represent the symbol polymer at the top of the input stack for $j \geq 0$ where j is the number of cycles completed. In overview, the RNA-UPDA computation proceeds as:

1. If the symbol $a_{K_{2j}} = halt$ then the RNA-UPDA halts and the input stack constitutes the output of the RNA-UPDA. Otherwise:
2. $a_{K_{2j}}$ is popped from the input stack and placed on the working stack. The symbol polymer $a_{K_{2j+1}}$ is popped and discarded. This completes the 2-tag step of removing the first two symbols of the input word.
3. The matching function is initiated with $a_{K_{2j}}$ as input at the top of the working stack. The matching function cycles through the instruction stack until the instruction polymer d_i associated with $a_{K_{2j}}$ is at the top. The matching function initiates the copy function. This completes the 2-tag step of matching the first symbol of the input word to the associated transition rule.
4. The copy function is initiated with the associated d_i at the top of the instruction stack. The copy function constructs ρ_i, which is pushed to the working stack.
5. The input and instruction stacks cycle, sequentially popping the top symbol polymer of the input stack and pushing to the instruction stack until the end of the stack symbol polymer η is reached.

6. When η is the top symbol polymer of the input stack, ρ_i is popped from the working stack and pushed to the input stack. Steps 4, 5, and 6 complete the 2-tag step of appending α_i to the input word.
7. The input and instruction stacks cycle in the reverse order, sequentially popping the topmost symbol polymer of the instruction stack and pushing back to the input stack. When $\triangle$, the symbol polymer which demarcates the boundary of D and K, is at the top of the instruction stack the RNA-UPDA has completed one transition of the 2-tag automaton and reset for the next simulated transition.

The RNA-UPDA always starts a new simulated 2-tag transition by popping from the start of the encoded input word and appending ρ_i to the last position of the input stack. If the instruction and input stacks are conceptualised as the left and right portions of a single linear structure, the region containing the encoded input word residing between $\triangle$ to the end-of-stack symbol η is at all times concurrent with the input word of the 2-tag automaton being simulated. At the end of step 7, the input stack, excluding the end-of-stack symbol, is concurrent with the input word of the simulated 2-tag automaton. Hence, this simulates the target 2-tag automaton on target input, and therefore, shows universality.

It is instructive to consider which components of the RNA-UPDA above are serving the role of symbolic "data," and which components are performing an instructive role to guide the progress of the computation as "program." From this perspective the role of nucleotide polymers may cycle between representing data as members of K on the input stack and representing program as members of D on the instruction stack. RNA represents a natural substrate for such *program-data duality*, which we will examine further as a component of undecidability.

RNA-UPDA computations. In the above demonstration, universality was found by enacting a process of encoding the rules and input of automata into data for a simulating computation. An important consequence is observed if we recognise that the transition rules of the RNA-UPDA may be accessed by recursion that was introduced with the RNA-2PDA. These rules may be passed through the same encoding process as any other automaton. The encoded rules of the RNA-UPDA may then serve as the data input for a simulation of its own computation. Encoding an automaton into the data it computes generates an instance of *self-reference*. Such self-reference is a mechanism to generate the Liar paradox at the heart of undecidability, which we now turn our attention to.

5 Undecidability

The class of 2PDA automata are capable of generating undecidable statements, which can be exemplified by logical paradoxes like the Liar paradox, and leading to the halting problem (Turing, 1937b). To unpack the role of self-reference in generating undecidability in RNA automata, we will adapt the examples of Sipser (2006) and the Liar paradox constructions of Prokopenko et al. (2019).

In the above UPDA, the automaton was constructed such that the output of the computation was the input stack after halting. For any 2PDA or equivalent, including the UPDA, there is a 2PDA that accepts or rejects the input. The input for an accepting automaton is formulated such that the question is answered by the accept or reject output. Suppose an accepting RNA-UPDA, U, into which we pass the encoding of some automaton R. With R we will also pass w, the input word for which R accepts or rejects. U will then accept or reject if R would accept or reject. We may write $U = \{[R, w] \mid w \in \Sigma^*\}$. The [ ] notation indicates an encoding into a word of Σ, and we assume that special characters exist such that the encodings of the transition rules and the input word are distinguishable as such to U.

This relationship of U to R is:

$$U([R, w]) \begin{cases} \text{Accept if R accepts w} \\ \text{Reject if R rejects w} \\ \text{Run forever if R runs forever on w} \end{cases}$$

Observe that we can encode R and pass this as the input word, [R, [R]]. In this case, R may accept or reject the word encoding itself or run forever. This relationship is not necessarily paradoxical; for that we will need a special automaton called a *universal decider*.

The proposed universal decider, D, is similar to U above with the additional ability to reject when R would run forever on w. The impossibility of the universal decider D is demonstrated in the paradox that such an automaton generates. The relationship of D to R is:

$$D([R, w]) \begin{cases} \text{Accept if R accepts w} \\ \text{Reject if R rejects w or runs forever} \end{cases}$$

Now suppose a *contrarian* decider, I, which checks and inverts the relationship of D to R when the input word is [R]:

$$I([R]) \begin{cases} \text{Accept if R rejects [R] or runs forever} \\ \text{Reject if R accepts [R]} \end{cases}$$

The contrarian decider has introduced negation to the dynamics. If we now introduce self-reference to negation, we create an auto-negating paradox:

$$I([I]) \begin{cases} \text{Accept if I rejects [I] or runs forever} \\ \text{Reject if I accepts [I]} \end{cases}$$

We can write this as $I([I])$ will accept only when $I([I])$ rejects.

This is a paradox pertinent to all computational frameworks capable of universal computation. In the preceding sections we have shown the theoretical construction of RNA automata that are capable of universal computation with self-referential dynamics. We may observe that RNA-automata demonstrate the key criteria of systems capable of demonstrating undecidable dynamics (Prokopenko et al., 2019): RNA automata demonstrate *program-data duality* as discussed above, *access to an infinite medium* through a renewing supply of short RNA polymers and *negation* through the ability to encode accept and reject representations which may be flipped.

6 Discussion

This ansatz set out to probe the question of whether formal undecidability could be embodied in biological components. To do this we explored configurations of RNA polymers constructed into arrangements termed automata that compute functions on input. We aimed to formally express the RNA-mediated functions of ligation and cleavage in terms that aid in exploring automata construction. Within this framework, we surveyed a progression of RNA automata commencing with the purely functional construction of the RNA-FA, in which the automaton consists of only the RNA state polymer and the RNA enzyme polymers that carry out the transitions. An RNA-PDA was constructed by the addition of a Last-In-First-Out stack. The stack expands the automaton with a structure for storing RNA symbol polymers which may represent transient memory within the automaton. An RNA-2PDA was constructed by addition of a second stack. The RNA-2PDA

is equivalent to a TM and can recognise the recursively enumerable languages. Automata in this class, including the RNA-2PDA, are able to reflexively encode a description of their program into data and to compute with and instantiate this encoding. Here we first encounter clear program-data duality, turning a description of an automaton M into some data $[M]$. From the foundation of a 2PDA, an RNA-UPDA was explored that could achieve universal computation, that is, the capacity to simulate any other automaton in an encoded form. Universality enables such a system to explore the greatest possible solution landscape; but it comes with the price of undecidable dynamics, e.g., when a universal automaton self-referentially runs on its own encoding. In other words, it becomes possible for such a system to generate computational undecidability, the outcomes of which may not be determined within the system itself. We have seen an example of such paradoxical *self-negating* computation, constructed in an analogy with the Liar paradox, which offers no possible resolution within its own set of rules.

Our ansatz is in two parts:

1. RNA automata can be constructed that embody computational models, up to Turing machine equivalence.
2. At sufficient complexity (analogous to universal computation), RNA automata may generate self-reference and hence, computational undecidability. Continual resolution of computational undecidability represents a pathway to progressively expand the boundaries and complexity of the automata, i.e., innovating.

We have addressed the first part of the ansatz above. To address the second part of the ansatz, we pose a question that the prospect of undecidable biological computation raises. Since the paradox requires a perspective outside the system from which to observe and invert the output, where is the space in which the paradox may arise?

To answer the question we must ask if there exists a larger meta-system, encompassing the computational undecidability, which may play the role of universal decider and inverter. Such a system also contains the spark for removing the ceiling on biological complexity. This is because a key concept in undecidability as stated here is the lack of resolution for an automaton *within its own set of rules*. Importantly, an undecidable problem is framed within a given formal system, and once the system is appropriately extended, the problem in point becomes decidable—at the cost of generating other undecidable problems that inevitably arise in the extended system.

A well-known analogy of a meta-system that resolves computational undecidability *at a given level* is an oracle machine, which supersedes a TM (Turing, 1939), being capable of deciding an outcome that could not be decidable by a UTM such as the RNA-UPDA. An oracle is some entity that is not itself a machine and that provides to a TM some information from outside its own bounds. Turing gave a mechanical description of the interface of an oracle and a TM of an o-machine to be a configuration of the o-machine in which the next state depends on feedback from the corresponding oracle (Turing, 1939).

In context of sequential innovations, the o-machine concept was utilised by Penrose (1994), who defined the class of o-machines that may overcome the undecidable halting problem as the first-order o-machines. An o-machine comprises a TM and an oracle that is able to compute the values of a function that may not be computable. This combination, i.e., a TM and an oracle, has an expanded computational capacity relative to the TM alone. For example, a first-order o-machine comprises an oracle that can determine the value of the corresponding TM halting function. In turn there exist second-order o-machines with oracles capable of deciding halting states of the first-order o-machines, and so on, generalised in the concept of α-order machines. Such a chain consecutively expands the boundaries of lower-order systems, by introducing a pointed *innovation* (supplied by the corresponding oracle) in the form of a new description (i.e., axiom) added in the higher-order system. Importantly, the innovation can be provided to the system at the $(\alpha - 1)$-order as a form

of feedback, reacting to which extends the system's boundary to construct an α-order system (discussed in the next section).

An o-machine, a Turing machine with an oracle, is analogous to augmenting the original logical system with a new, independent, axiom. From this basis, a continual, step by step process may follow in which the bounds of any individual logical system may be overcome and the system continually expanded, as suggested by Turing (1939, p. 161) in his introduction on systems of ordinal logics (where a logic would now be described as a formal system):

> The well-known theorem of Gödel (1931) shows that every system of logic is in a certain sense incomplete, but at the same time it indicates means whereby from a system L of logic a more complete system L′ may be obtained. . . . A logic L_ω may then be constructed in which the provable theorems are the totality of theorems provable with the help of the logics L, L_1, L_2,....

A continual, step by step process of expanding system boundaries in an attempt to "reconcile" a paradox, is a recurrent motif in studies of formal systems (Abrahão, 2017; Chaitin, 1987, 2012; Sayama, 2008). An influential early result was established by Post from the perspective of the recursively enumerable sets, by stating that while no recursively generated logic is complete, *every recursively generated logic may be extended* (Post, 1944). In doing so, Post showed that the complement set of the set of true propositions is not recursively enumerable; that is, the sets of propositions that can be "guaranteed" to be true, T, and false, F, do not exhaust the set of all propositions. The proposition which was shown to be outside of either of these two sets, i.e., an "undecidable" proposition, was constructed in a self-referential way, by recursively enumerating false propositions and identifying the set S_0 of corresponding positive integers. The incompleteness is shown by constructing the proposition describing the set S_0 itself: This proposition cannot be false (not in F) but has to remain outside of set T. It is precisely the addition of this proposition to the set F, making a new set F', that constitutes the expansion of the logic (i.e., innovation), and so a sequence of such expansions/innovations may be developed.

6.1 An α-Order o-Machine in Biological Automata

In a search for a meta-system to inform the biological automaton, the niche of the biological system is an obvious candidate. The interdependence of the niche and the genome is captured in the concept of the "reactive genome" (Gilbert, 2003), here characterised by Griffiths and Stotz (2013, p. 227):

> The regulatory architecture of the genome reaches outside the genome itself, outside the cell, and outside the organism . . . Many of the factors involved in genome regulation are highly context-sensitive, which allows them to relay environmental information to a reactive genome which has evolved to let environmental input play an instructive role on the determination of phenotypes.

If we recognise that the universal biological automaton is operating in an environmental niche, then the *coupled phenotype-environment space* can be considered an analogy of a meta-system. Here, by an *environment* we mean a set of conditions that may range from simple environmental variables, like temperature and humidity, to more complex holistic niche conditions. Since it encompasses the automaton, the coupled phenotype-environment space can operate as a first-order o-machine, in which the oracle provides input to the automaton and resolves computational undecidability. We assume that the meta-system would essentially be performing meta-simulation of the universal automaton, by observing and inverting the output of the universal automaton U running on its own encoding $U[U]$. That is, the coupled phenotype-environment space may operate as the inverter $I[I]$. The detection of the paradox therefore occurs outside the bounds of the automaton.

Here we propose the question of delineating the nature of self-referential computational undecidability, generated by the $(\alpha - 1)$-order o-machine of the coupled phenotype-environment space. In other words, by what mechanism may a self-referential biological automaton generate a computational undecidability, and by what form and channel is the corresponding oracle feedback transmitted?

In order to complete the expansion to the first-order system, the lower-order system (e.g., automaton) needs to receive a signal from the meta-level. While, as above, the form of such a signal is an open question, we can presume that the signal carries the information about the detected contradiction, initiating a *generic* self-editing response.[2] The key element is an extension of the automaton's self-description with a new axiom so that the extended genotype better fits the niche. The resolution is implemented within the bounds of the extended, first-order system, which thus makes an evolutionary step by absorbing the innovation. This relationship recalls the tangled hierarchies of biological chemistry explored by Hofstadter (1980), which are extended here to incorporate continual expansion in the $(\alpha - 1)$-order o-machine.

A first-order system, of course, will have its own computational undecidability. For example, questions about co-evolution of the biological automaton and its environmental niche may not be decidable within their first-order system, leading to some contradictions. However, a second-order system expanded with a more complex environmental context, i.e., second-order o-machine, will be able to resolve the ensuing contradictions by providing contextual co-evolutionary feedback and generating further innovations at the level comprising co-evolving components.

To conclude, we highlight an insight from the exploration of "Life Is Physics" by Goldenfeld and Woese (2011, p. 389):

> These rules themselves need to evolve, but how? We need an additional set of rules describing the evolution of the original rules. But this upper level of rules itself needs to evolve. Thus, we end up with an infinite hierarchy, an inevitable reflection of the fact that the dynamic we are seeking is inherently self-referential.

This also emphasises self-reference in biological computation. We argue that such self-reference inevitably generates undecidable dynamics, and hope that the questions raised by this ansatz will help to progress this thread of enquiry.

Acknowledgments

The authors are grateful to Daniel Polani, Nihat Ay, Peter Stadler, Sheri Markose, Nathaniel Virgo, Felipe Abrahão, Joseph Lizier, Michael Harré, and Stuart Kauffman for helpful discussions of this cross-disciplinary topic. The authors were supported through the Australian Research Council grant DP200103005.

References

Abrahão, F. S. (2017). The "Paradox" of computability and a recursive relative version of the busy beaver function. In M. Burgin & C. S. Calude (Eds.), *Information and complexity* (pp. 3–15). World Scientific Publishing Co. https://doi.org/10.1142/9789813109032_0001

Abrahão, F. S., Wehmuth, K., & Ziviani, A. (2019). Algorithmic networks: Central time to trigger expected emergent open-endedness. *Theoretical Computer Science*, *785*, 83–116. https://doi.org/10.1016/j.tcs.2019.03.008

Adams, A., Zenil, H., Davies, P. C. W., & Walker, S. I. (2017). Formal definitions of unbounded evolution and innovation reveal universal mechanisms for open-ended evolution in dynamical systems. *Scientific Reports*, *7*, Article 997. https://doi.org/10.1038/s41598-017-00810-8

Arnold, C., Stadler, P. F., & Prohaska, S. J. (2013). Chromatin computation: Epigenetic inheritance as a pattern reconstruction problem. *Journal of Theoretical Biology*, *336*, 61–74.

2 If the response were not generic, then some information about the contradiction would have to be known beforehand.

Autebert, J.-M., Berstel, J., & Boasson, L. (1997). Context-free languages and pushdown automata. In G. Rozenberg & A. Salomaa (Eds.), *Handbook of formal languages* (pp. 111–174). Springer.

Brāzma, A., Jonassen, I., Eidhammer, I., & Gilbert, D. (1998). Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology*, *5*(2), 279–305. https://doi.org/10.1089/cmb.1998.5.279

Brāzma, A., Jonassen, I., Vilo, J., & Ukkonen, E. (1998). Predicting gene regulatory elements in silico on a genomic scale. *Genome Research*, *8*(11), 1202–1215.

Cairns-Smith, A. G. (1966). The origin of life and the nature of the primitive gene. *Journal of Theoretical Biology*, *10*(1), 53–88.

Chaitin, G. J. (1987). *Algorithmic information theory*. Cambridge University Press.

Chaitin, G. J. (2012). Life as evolving software. In H. Zenil (Ed.), *A computable universe: Understanding and exploring nature as computation* (pp. 277–302). World Scientific Publishing Co.

Chen, J.-L., & Pace, N. R. (1997). Identification of the universally conserved core of ribonuclease P RNA. *RNA*, *3*(6), 557–560.

Cocke, J., & Minsky, M. (1964). Universality of tag systems with $P = 2$. *Journal of the ACM (JACM)*, *11*(1), 15–20. https://doi.org/10.1145/321203.321206

Das, M. K., & Dai, H.-K. (2007). A survey of DNA motif finding algorithms. *BMC Bioinformatics*, *8*(S7), Article S21. https://doi.org/10.1186/1471-2105-8-S7-S21

Derksen, M., Mertens, V., & Pruijn, G. J. M. (2015). RNase P-Mediated sequence-specific cleavage of RNA by engineered external guide sequences. *Biomolecules*, *5*(4), 3029–3050. https://doi.org/10.3390/biom5043029

Doudna, J. A., & Szostak, J. W. (1998). RNA-catalysed synthesis of complementary-strand RNA. *Nature*, *339*(6225), 519–522. https://doi.org/10.1038/339519a0

Ekland, E. H., Szostak, J. W., & Bartel, D. P. (1995). Structurally complex and highly active RNA ligases derived from random RNA sequences. *Science*, *269*(5222), 364–370. https://doi.org/10.1126/science.7618102

Evans, D., Marquez, S. M., & Pace, N. R. (2006). RNase P: Interface of the RNA and protein worlds. *Trends in Biochemical Sciences*, *31*(6), 333–341. https://doi.org/10.1016/j.tibs.2006.04.007

Gilbert, S. F. (2003). The reactive genome. In G. B. Müller & S. A. Newman (Eds.), *Origination of organismal form: Beyond the gene in developmental and evolutionary biology* (pp. 87–101). MIT Press.

Gödel, K. (1931). Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. I [On formally undecidable theorems of Principia Mathematica and related systems. I]. *Monatshefte für Mathematik und Physik*, *38*, 173–198. https://doi.org/10.1007/BF01700692

Goldenfeld, N., & Woese, C. (2011). Life is physics: Evolution as a collective phenomenon far from equilibrium. *Annual Review of Condensed Matter Physics*, *2*(1), 375–399. https://doi.org/10.1146/annurev-conmatphys-062910-140509

Griffiths, P., & Stotz, K. (2013). *Genetics and philosophy: An introduction*. Cambridge University Press.

Hofstadter, D. R. (1980). *Gödel, Escher, Bach: An eternal golden braid*. Penguin.

Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to automata theory, languages, and computation*. Addison-Wesley.

Jarrous, N. (2017). Roles of RNase P and its subunits. *Trends in Genetics*, *33*(9), 594–603. https://doi.org/10.1016/j.tig.2017.06.006

Joyce, G. F., & Szostak, J. W. (2018). Protocells and RNA Self-Replication. *Cold Spring Harbor Perspectives in Biology*, *10*(9), Article a034801. https://doi.org/10.1101/cshperspect.a034801

Kauffman, S. A. (2016). *Humanity in a creative universe*. Oxford University Press.

Knuth, D. E. (1965). On the translation of languages from left to right. *Information and Control*, *8*(6), 607–639. https://doi.org/10.1016/S0019-9958(65)90426-2

Koslowski, J. (2013). Deterministic single-state 2PDAs are Turing-complete. *Instituto deficiencias de la computación, TU Braunschweig, Alemania*.

Lincoln, T. A., & Joyce, G. F. (2009). Self-sustained replication of an RNA enzyme. *Science*, *323*(5918), 1229–1232. https://doi.org/10.1126/science.1167856

Markose, S. M. (2004). Novelty in complex adaptive systems (CAS) dynamics: A computational theory of actor innovation. *Physica A: Statistical Mechanics and Its Applications*, *344*(1), 41–49. https://doi.org/10.1016/j.physa.2004.06.085

Markose, S. M. (2017). Complex type 4 structure changing dynamics of digital agents: Nash equilibria of a game with arms race in innovations. *Journal of Dynamics and Games*, *4*(3), 255–284. https://doi.org/10.3934/jdg.2017015

McGuire, A. M., Hughes, J. D., & Church, G. M. (2000). Conservation of DNA regulatory motifs and discovery of new motifs in microbial genomes. *Genome Research*, *10*(6), 744–757. https://doi.org/10.1101/gr.10.6.744

McMullin, B., & Hasegawa, T. (2012, April 10–13). *Von Neumann redux: Revisiting the self-referential logic of machine reproduction using the Avida world* [Paper presentation]. In EMCSR 2012: European meeting on systems and cybernetics research, Vienna, Austria. Available at https://doras.dcu.ie/16918/

Minsky, M. L. (1962). Size and structure of universal Turing machines using tag systems. In J. C. E. Dekker (Ed.), *Proceedings of the fifth symposium in pure mathematics of the American Mathematical Society* (pp. 229–238). AMS.

Paul, N., & Joyce, G. F. (2002). A self-replicating ligase ribozyme. *Proceedings of the National Academy of Sciences*, *99*(20), 12733–12740. https://doi.org/10.1073/pnas.202471099

Penrose, R. (1994). *Shadows of the mind: A search for the missing science of consciousness*. Oxford University Press.

Post, E. L. (1944). Recursively enumerable sets of positive integers and their decision problems. *Bulletin of the American Mathematical Society*, *50*(5), 284–316.

Prohaska, S. J., Stadler, P. F., & Laubichler, M. (2019). How and what does a biological 7 system compute. In D. H. Wolpert, C. Kempes, P. F. Stadler, & J. A. Grochow (Eds.), *The energetics of computing in life and machines* (pp. 119–139). SFI Press.

Prokopenko, M., Harré, M., Lizier, J., Boschetti, F., Peppas, P., & Kauffman, S. (2019). Self-referential basis of undecidable dynamics: From the Liar paradox and the halting problem to the edge of chaos. *Physics of Life Reviews*, *31*, 134–156. https://doi.org/10.1016/j.plrev.2018.12.003

Reiter, N. J., Osterman, A., Torres-Larios, A., Swinger, K. K., Pan, T., & Mondragón, A. (2010). Structure of a bacterial ribonuclease P holoenzyme in complex with tRNA. *Nature*, 468(7325), 784–789. https://doi.org/10.1038/nature09516

Robertson, M. P., & Joyce, G. F. (2014). Highly efficient self-replicating RNA enzymes. *Chemistry & Biology*, *21*(2), 238–245. https://doi.org/10.1016/j.chembiol.2013.12.004

Rogers, J., & Joyce, G. F. (2001). The effect of cytidine on the structure and function of an RNA ligase ribozyme. *RNA*, *7*(3), 395–404. https://doi.org/10.1017/S135583820100228X

Rogozhin, Y. (1996). Small universal Turing machines. *Theoretical Computer Science*, *168*(2), 215–240. https://doi.org/10.1016/S0304-3975(96)00077-1

Sayama, H. (2008). Construction theory, self-replication, and the halting problem. *Complexity*, *13*(5), 16–22.

Schulman, R., Yurke, B., & Winfree, E. (2012). Robust self-replication of combinatorial information via crystal growth and scission. *Proceedings of the National Academy of Sciences*, *109*(17), 6405–6410. https://doi.org/10.1073/pnas.1117813109

Shannon, C. E. (1956). A universal Turing machine with two internal states. *Automata Studies*, *34*, 157–165.

Sipser, M. (2006). *Introduction to the Theory of Computation* (2nd ed.). Thomson Course Technology.

Turing, A. M. (1937a). Computability and λ-Definability. *The Journal of Symbolic Logic*, *2*(4), 153–163. https://doi.org/10.2307/2268280

Turing, A. M. (1937b). On computable numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, *s2-42*(1), 230–265. https://doi.org/10.1112/plms/s2-42.1.230

Turing, A. M. (1939). Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, *s2–45*(1), 161–228. https://doi.org/10.1112/plms/s2-45.1.161

Virgo, N., Agmon, E., & Fernando, C. (2017). Lineage selection leads to evolvability at large population sizes. In *Proceedings of ECAL 2017: The 14th European conference on Artificial Life* (pp. 420–427). MIT Press.

Waugh, D. S., Green, C. J., & Pace, N. R. (1989). The design and catalytic properties of a simplified ribonuclease P RNA. *Science*, *244*(4912), 1569–1571. https://doi.org/10.1126/science.2472671

Will, C. L., & Luhrmann, R. (2011). Spliceosome structure and function. *Cold Spring Harbor Perspectives in Biology*, *3*(7), Article a003707. https://doi.org/10.1101/cshperspect.a003707

Woods, D., & Neary, T. (2009). The complexity of small universal Turing machines: A survey. *Theoretical Computer Science*, *410*(4), 443–450. https://doi.org/10.1016/j.tcs.2008.09.051

Zenil, H., Hernandez-Quiroz, F., & Hernandez-Quiroz, S. (2016). The limits of decidable states on open-ended evolution and emergence. In *ALIFE 2016: Proceedings of the fifteenth international conference on the synthesis and simulation of living systems* (pp. 200–207). MIT Press. https://doi.org/10.1162/978-0-262-33936-0-ch039

Zou, H., Lee, J., Kilani, A. F., Kim, K., Trang, P., Kim, J., & Liu, F. (2004). Engineered RNase P ribozymes increase their cleavage activities and efficacies in inhibiting viral gene expression in cells by enhancing the rate of cleavage and binding of the target mRNA. *Journal of Biological Chemistry*, *279*(31), 32063–32070. https://doi.org/10.1074/jbc.M403059200

Appendices

Appendix 1: RNA-PDA for $a^n b^n$

To illustrate the use of an extensible memory encoding, the RNA-PDA we are constructing is designed to recognise input sequences of the form $a^n b^n$ where $n \geq 0$, in which the polymer must consist of an arbitrary number of *a*'s followed by an equal number of *b*'s. (See Figure A1.) The automata must be able to encode the number of instances of *a*, and then compare this to the number of instances of *b*. Under this definition, we will accept an empty input, and reject a single *a* or *b*. We will construct the RNA-PDA such that the automaton will halt at the end of the transition in which the end-of-input symbol ν is read. If the automaton is in an accept state at this point the input is considered to be accepted. The automaton does not halt immediately upon reaching an empty configuration, rather, explicit reject states may be reached from which no further input or stack symbol will result in a change of transition or stack operation.

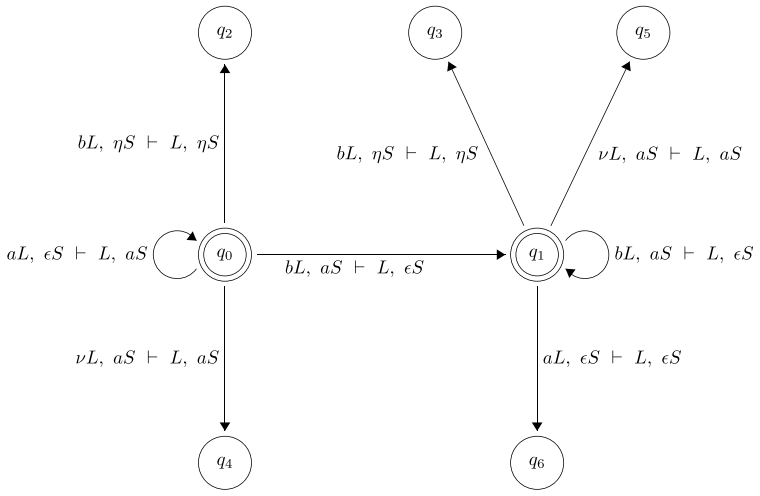


Figure A1. State diagram for the RNA-PDA.

$\mathcal{Q} = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$ where each q_i is a unique sequence of the state polymer.
$\Sigma = \{a, b, \nu\} \cup \{\epsilon\}$ where a and b are symbol polymers and ν is a special symbol polymer indicating the end of input.
$\Gamma = \{a, \eta\} \cup \{\epsilon\}$ where a is a symbol polymer and η is a special symbol polymer indicating the bottom of the stack.
$Z_0 = \{\eta\}$.
$F = \{q_0, q_1\}$.

The transition function δ induces the following step-relations:

$$\delta = \begin{cases} (q_0, aL, \epsilon S) \vdash (q_0, L, aS) & \text{where } q_0 = \kappa(q_0, \kappa(a, \epsilon)), \\ & \text{pushing a to the stack with } \lambda(\epsilon S, \kappa(a, q_0)) \\ (q_0, bL, \eta S) \vdash (q_2, L, \eta S) & \text{where } q_2 = \lambda(q_0, \lambda(b, \eta)), \\ & \text{leaving the stack unchanged with } \kappa(\eta S, \kappa(b, q_0)) \\ (q_0, \nu L, aS) \vdash (q_4, L, aS) & \text{where } q_4 = \lambda(q_0, \lambda(\nu, a)), \\ & \text{leaving the stack unchanged with } \kappa(aS, \kappa(\nu, q_0)) \\ (q_0, bL, aS) \vdash (q_1, L, \epsilon S) & \text{where } q_1 = \lambda(q_0, (\kappa(b, a)), \\ & \text{popping from the stack with } \mu(aS, \kappa(b, q_0)) \\ (q_1, bL, aS) \vdash (q_1, L, \epsilon S) & \text{where } q_1 = \kappa(q_1, (\kappa(b, a)), \\ & \text{popping from the stack with } \mu(aS, \kappa(b, q_1)) \\ (q_1, bL, \eta S) \vdash (q_3, L, \eta S) & \text{where } q_3 = \lambda(q_1, \lambda(b, \eta)), \\ & \text{leaving the stack unchanged with } \kappa(\eta S, \kappa(b, q_1)) \\ (q_1, \nu L, aS) \vdash (q_5, L, aS) & \text{where } q_5 = \lambda(q_1, \lambda(\nu, a)), \\ & \text{leaving the stack unchanged with } \kappa(aS, \kappa(\nu, q_1)) \\ (q_1, aL, \epsilon S) \vdash (q_6, L, \epsilon S) & \text{where } q_6 = \lambda(q_1, \kappa(a, \epsilon)), \\ & \text{leaving the stack unchanged with } \kappa(\epsilon S, \kappa(a, q_1)) \end{cases}$$

If we take as input ν (end of input symbol only):

1. $(q_0, \nu L, \epsilon S)$ is empty, so no transition of state or stack occurs.
2. At the exhaustion of input the state polymer has sequence q_0 so the automaton **accepts**.

If we take as input $aabb\nu$:

1. $(q_0, aL, \epsilon S)$ results in the state polymer sequence remaining q_0 and the symbol polymer a being placed on the stack, without reading the top of the stack. Stack is $a\eta$.
2. $(q_0, aL, \epsilon S)$ as for step 1. Stack is $aa\eta$.
3. (q_0, bL, aS) results in ligation of the state polymer to sequence q_1 and the reading and popping of an a from the stack. Stack is $a\eta$.
4. (q_1, bL, aS) results in the state polymer sequence remaining q_1 and the reading and popping of an a from the stack. Stack in η.
5. $(q_1, \nu L, \epsilon S)$ is empty, so no transition of state or stack occurs.
6. At the exhaustion of input the state polymer has sequence q_1 so the automaton **accepts**.

If we take as input *abbv*:

1. $(q_0, aL, \epsilon S)$ results in the state polymer sequence remaining q_0 and the symbol c being placed on the stack, without reading the top of the stack. Stack is $a\eta$.
2. (q_0, bL, aS) results in ligation of the state polymer to q_1 and the reading and popping of an a from the stack. Stack is η.
3. $(q_1, bL, \eta S)$ results in ligation of the state polymer to q_3. No stack operation occurs.
4. $(q_3, \nu L, \epsilon S)$ is empty, so no transition of state or stack occurs.
5. At the exhaustion of input the state polymer has sequence q_3 so the automaton **rejects**.

If we take as input *abav*:

1. $(q_0, aL, \epsilon S)$ results in the state polymer sequence remaining q_0 and the symbol c being placed on the stack, without reading the top of the stack. Stack is $a\eta$.
2. (q_0, bL, aS) results in the ligation of the state polymer to q_1 and the reading and popping of an a from the stack. Stack is η.
3. $(q_1, aL, \epsilon S)$ results in the ligation of the state polymer to q_6. No stack operation occurs.
4. $(q_6, \nu L, \epsilon S)$ is empty, so no transition of state or stack occurs.
5. At the exhaustion of input the state polymer has sequence q_6 so the automaton **rejects**.

Appendix 2: RNA-2PDA for a^{2^n}

The RNA-2PDA we are constructing will be to recognise the language a^{2^n} consisting of sequences of a in powers of 2 (See Figure A2). The RNA-2PDA must recursively divide the input word by 2,

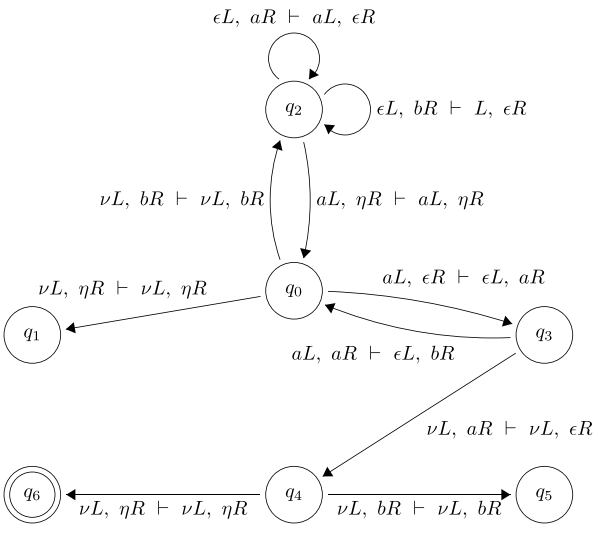


Figure A2. State diagram for the RNA-2PDA.

recognising when this results in an odd number to reject or in a single remaining symbol polymer. Under this definition, we will reject the empty input ν and accept a single a. Any language that matches the form a^{2^n} will result in an accept state.

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$ where each q_i is a unique sequence of the state polymer.
$\Sigma = \{a\}$ where a is a symbol polymer and $\Sigma \in \Gamma$ as the input is initialised to the L tape.
$\Gamma = \{a, b, \eta, \nu\}$ where a and b are symbol polymers and ν and η are special symbol polymers indicating the bottom (3′-end) of the L and R stack respectively.
$F = \{q_6\}$.

The transition function δ is given by the following transitions:

$$\delta = \begin{cases} (q_0, \nu L, \eta R) \vdash (q_1, \nu L, \eta R) & \text{where } q_1 = \lambda(q_0, \lambda(\nu L, \eta R)), \\ & \text{leaving the stacks unchanged with } \kappa(\eta R, \kappa(\nu L, q_0)) \\ (q_0, \nu L, bR) \vdash (q_2, \nu L, bR) & \text{where } q_2 = \lambda(q_0, \lambda(\nu L, bR)), \\ & \text{leaving the stacks unchanged with } \kappa(\nu L, \kappa(bR, q_0)) \\ (q_0, aL, \epsilon R) \vdash (q_3, \epsilon L, aR) & \text{where } q_3 = \lambda(q_0, \kappa(aL, \epsilon R)), \\ & \text{popping L and pushing a to R with } \lambda(\epsilon R, \mu(aL, q_0)) \\ (q_2, \epsilon L, aR) \vdash (q_2, aL, \epsilon R) & \text{where } q_2 = \kappa(q_2, \kappa(\epsilon L, aR)), \\ & \text{popping R and pushing a to L with } \lambda(\epsilon L, \mu(aR, q_2)) \\ (q_2, \epsilon L, bR) \vdash (q_2, L, \epsilon R) & \text{where } q_2 = \kappa(q_2, \kappa(\epsilon L, bR)), \\ & \text{popping R with } \kappa(\epsilon L, \mu(bR, q_2)) \\ (q_2, aL, \eta R) \vdash (q_0, aL, \eta R) & \text{where } q_0 = \mu(q_2, \lambda(aL, \eta R)), \\ & \text{leaving the stacks unchanged with } \kappa(\epsilon L, \kappa(\eta R, q_2)) \\ (q_3, aL, aR) \vdash (q_0, \epsilon L, bR) & \text{where } q_0 = \mu(q_3, \lambda(aL, aR)), \\ & \text{popping L and pushing b to R with } \lambda(aR, \mu(aL, q_3)) \\ (q_3, \nu L, aR) \vdash (q_4, \nu L, \epsilon R) & \text{where } q_4 = \lambda(q_3, \kappa(\nu L, aR)), \\ & \text{popping R with } \kappa(\nu L, \mu(aR, q_3)) \\ (q_4, \nu L, bR) \vdash (q_5, \nu L, bR) & \text{where } q_5 = \lambda(q_4, \lambda(\nu L, bR)), \\ & \text{leaving the stacks unchanged with } \kappa(\nu L, \kappa(bR, q_4)) \\ (q_4, \nu L, \eta R) \vdash (q_6, \nu L, \eta R) & \text{where } q_6 = \lambda(q_4, \lambda(\nu L, \eta R)), \\ & \text{leaving the stacks unchanged with } \kappa(\nu L, \kappa(\eta R, q_4)) \end{cases}$$

If we take as input a: The stacks are initialised as $L = a\nu$ and $R = \eta$.

1. $(q_0, aL, \epsilon R)$ results in ligation of the state polymer to q_3 and a being popped from L and pushed to R such that $L = \nu$, $R = a\eta$.
2. $(q_3, \nu L, aR)$ results in ligation of the state polymer to q_4 and a being popped from R such that $L = \nu$, $R = \eta$.
3. $(q_4, \nu L, \eta R)$ results in ligation of the state polymer to q_6 with no change of the stacks.
4. There are no transitions possible from this configuration. The state polymer has sequence q_6 so the automaton **accepts.**

If we take as input a^2: The stacks are initialised as $L = aa\nu$ and $R = \eta$.

1. $(q_0, aL, \epsilon R)$ results in ligation of the state polymer to q_3 and a being popped from L and pushed to R such that $L = a\nu$, $R = a\eta$.
2. (q_3, aL, aR) results in cleavage of the state polymer to q_0, a being popped from L and b being pushed to R such that $L = \nu$ and $R = ba\eta$.
3. $(q_0, \nu L, bR)$ results in ligation of the state polymer to q_2 with no change to the stacks.
4. $(q_2, \epsilon L, bR)$ results in the state polymer sequence remaining q_2, b being popped from R such that $L = \nu$ and $R = a\eta$.
5. $(q_2, \epsilon L, aR)$ results in the state polymer sequence remaining q_2, with a being popped from R and pushed to L such that $L = a\nu$ and $R = \eta$.
6. $(q_2, aL, \eta R)$ results in cleavage of the stack polymer to q_0 with no change to the stacks.
7. The automaton now proceeds with input a^1, as above. As such, the automaton **accepts.**

If we take as input a^3: The stacks are initialised as $L = aaa\nu$ and $R = \eta$.

1. $(q_0, aL, \epsilon R)$ results in ligation of the state polymer to q_3 and a being popped from L and pushed to R such that $L = aa\nu$, $R = a\eta$.
2. (q_3, aL, aR) results in cleavage of the state polymer to q_0, a being popped from L and b being pushed to R such that $L = a\nu$ and $R = ba\eta$.
3. $(q_0, aL, \epsilon R)$ results in ligation of the state polymer to q_3 and a being popped from L and pushed to R such that $L = \nu$ and $R = aba\eta$.
4. $(q_3, \nu L, aR)$ results in ligation of the state polymer to q_4 and a being popped from R such that $L = \nu$, $R = ba\eta$.
5. $(q_4, \nu L, bR)$ results in ligation of the state polymer to q_5 with no change of stacks.
6. There are no transitions possible from this configuration. The state polymer has sequence q_5 so the automaton **rejects.**

For $n > 0$, a^{2^n} reduces to the computation of a.

CHAPTER 4

Summary and future directions

4.1 Summary

Life as a phenomenon likely arose just once, approximately 3.7 billion years ago on an ancient earth. We know that life is self-sustaining, self-replicating and that it has the capacity to evolve. To explain these functions we can look through the eyes of a molecular biologist through which life is a self-sustaining cascade of molecular interactions which passes energy through chains of reactions that grow the organism, sustain and repair it, as well as fuelling the creation of new organisms. We can maintain this molecular level perspective to conceptualise the origin of life as an arrangement of molecules in a particular way in a particular environment that enabled a simple metabolic cascade to begin. When we consider the capacity of life to encode and transmit information, the molecular perspective can inform about the mechanisms of encoding and decoding. It can also account for evolution as a mechanism driven by mutations of the encoded, inherited representation of the organism (i.e. the genotype) which blindly adapts the decoded phenotype to its environment through successive generations. However, in his essay 'A New Biology for a New Century', the pioneer of genetics and evolution Carl Woese, whose work motivated this thesis, argued that the giant leap in the molecular understanding of life which occurred in the 20th century outpaced the conceptual understanding of the organising principles of life as a dynamic system (2004):

> "Organisms are resilient patterns in a turbulent flow —— patterns in an energy flow. A simple flow metaphor, of course, fails to capture much of what the organism is. None of our representations of organism capture it in its entirety. But the flow metaphor does begin to show us the

> organism's (and biology's) essence. And it is becoming increasingly clear that to understand living systems in any deep sense, we must come to see them not materialistically, as machines, but as (stable) complex, dynamic organization."

This thesis has aimed to contribute to the understanding of the organisation of life as a dynamic system. Specifically this thesis was concerned with the organisation of the single-stranded RNA molecules, that are considered to have comprised the earliest life, into a dynamic system which would be capable of generating increasing complexity by expanding its own boundaries. Automata constructions of RNA polymers were developed and expanded, ultimately resulting in an RNA-UPDA that could achieve universal computation, which answers in the positive the question of whether formal undecidability could be demonstrated for single-stranded RNA polymers. Universality enables the greatest possible solution landscape; but comes with the price for such a system to generate computational undecidability which cannot be resolved within the system itself. The Ansatz which emerged from this effort ultimately proposes that it is the resolution of computational undecidability which can drive novelty generation in biological dynamic systems. This proposition is significant as a theoretical framework for the complexity 'saltations' which occurred during the evolution of early life.

4.2 Progressing the first statement

The first statement of the Ansatz is a proposition that RNA based automata are possible with computational complexity equivalence to a TM, and in the Ansatz it is demonstrated that theoretical RNA automata are plausible. A natural follow up would be to realise these RNA automata as a bench-top lab demonstration in the tradition of the field of biological computation and *in-vitro* evolution. An alternative, intriguing and potentially fruitful avenue would be to explore defining and cataloguing the computational boundaries of biological organisms in the tradition of taxonomy. In other words, if we take the approach that input-processing-output dynamics apply to biological organisms and the sub-components of organisms, then computational dynamics can help to classify and understand the boundaries of biological

processes as the sets and boundaries of the computational automata which they embody. Such a taxonomy would commence with the simplest organisms that fully parasitise the machinery of more complex organisms. The smallest of these are the sub-viral agents, such as the enigmatic viroids, which are short, single-stranded RNA loops (250-400 nucleotides) which co-opt the cellular machinery of their hosts, the flowering plants (Hadidi 2017). From there a taxonomy could progress through the small, unencapsulated viruses and towards more complex viruses which carry an increasing coding content for their own molecular machinery. We may well find that a taxonomy of automata is not in 1:1 correspondence with the taxonomy of organisms. Indeed, it is more likely that complex organisms are a community of automata, and biological automata will be defined as pathways and sub-systems shared between complex organisms. In this vein, such a taxonomy would likely aid in understanding early life, in which biology has long struggled with the concept of organism where there is not a physical boundary to demarcate individuals. Promisingly, a recent examination of this very problem made significant headway in a description of early organisms as a persistent temporal representation in a loosely interconnected biochemical conglomerate (Krakauer et al. 2020), and would serve as an excellent framework for a computational taxonomy of organisms.

4.3 Progressing the second statement

The first statement of the Ansatz is concerned with the possibility of RNA automata that reach universal computation. From this basis the second statement of the Ansatz is concerned with the implications of biological universal computation. At its heart this second proposition states that the organism and environment form a coupled phenotype-environment space that can realise the dynamic of the oracle machine, in which the undecidable boundaries of a logical system can be expanded by incorporation of a new axiom. The notion of computability and undecidability we examine is within the Gödel-Turing-Post framework (Markose 2021; Markose 2017; Markose 2022). Here we will briefly re-examine the underpinning of system expansion through the oracle machine computational constructions of Alan Turing, and the work of Emil Post on effectively enumerable sets, with a view to future research along this framework in light of the Ansatz.

As reviewed in Chapter 2, Alan Turing proposed the 'ordinal logics': a progressive expansion of formal logical systems to incorporate undecidable statements. In the framework of Turing machines, the oracle machine is a Turing machine in which a specific undecidable statement is resolved by input from an 'oracle' situated outside the machine. There have since been several proposed forms of an oracle as applied to a traditional tape based TM. Such constructions propose a TM-like oracle with a tape and read-head that can receive input from the associated TM in an ASK state, process it in the manner of a TM and return an answer if the problem posed is within its rule set (Melkebeek 2000). More simply the oracle may return only a YES or NO when queried if a given input matches the information on its tape, e.g. the TM queries whether there are x symbols of a on its tape, and the oracle simply answers YES or NO if x is a match (Rogers 1967). All of the proposed oracle implementations are equivalent, albeit with differing levels of computational complexity. To answer the second statement of the Ansatz, we therefore seek an oracle machine equivalence in biological automata. The environment can be seen as a dynamic system operating outside of the bounds of the dynamic system of the organism. We propose that by establishing a relationship within the phenotype-environment space, the organism may query the environment which functions analogously to an oracle machine. By incorporating the feedback from the environment, the organism may expand its set of axioms, much in the manner of an oracle machine incorporating an answer from its associated oracle.

Within the world of molecular biology and development, ecological developmental biology (eco-devo) is a field that is concerned with exactly this phenotype-environment space, being the study of how development of the organism is shaped by interactions within the ecological niche of the organism (Gilbert and Epel 2009). Eco-devo therefore takes a particular focus on the molecular mechanisms of perception of the environment, i.e. the molecular pathways which query the environmental system, and gene regulation that controls the responses to the perception, i.e. the pathways which incorporate the environmental feedback into the representation within the organism. Griffiths and Stotz (2013) make the point that this study of the interdependence of the genome and the environment during development was overshadowed in a century-long absolutist debate on nature vs. nurture. They argue that querying the environment is an integral part of gene regulation:

> "Factors outside the gene not only activate, they differentially select and they create biological information. The basis of biological specificity is distributed between coding sequences, regulatory machinery, and the broader developmental niche. Many of the factors involved in genome regulation are highly context-sensitive, which allows them to relay environmental information to a reactive genome which has evolved to let environmental inputs play an instructive role in the determination of phenotypes."

Given the parallel between our conception of the coupled phenotype-environment space as an organising principle toward real biological innovation and the understanding of environmentally driven development in eco-devo, a natural first step would be to incorporate the principles of eco-devo into a computational framework, and incorporate a formalism for eco-devo into our biological automata. This may have the additional benefit of unifying the molecular, mechanistic insights of eco-devo into a computational framework.

We turn then to the concept of incorporating a new axiom. As reviewed in Chapter 2, Emil Post developed the idea of 'degrees of unsolvability' (Post 1944) in response to Turing's ordinal logics. Post examined recursive functions of positive integers, which generated the 'effectively enumerable' sets of positive integers. Post discovered a diagonal set from which he could precisely enumerate a contradictory statement. This statement could be directly resolved and incorporated in to the logical system as as axiom. Post demonstrated that every formal system is incomplete and extendable, including an effectively enumerable method of system expansion. The next, non-trivial, steps will be to investigate a mechanism for incorporating new axioms within the coupled phenotype-environment space. This will effectively be a formalisation of the capture of environmental information into the genotype, with an effective method for determining the contradictory, paradoxical statement within the biological dynamic system at the point of undecidability.

4.4 Conclusion

In order to drive at the question of how the system of interacting RNA components of early life could expand the boundaries of that system to become more complex, this thesis has explored the question of whether single-stranded RNA can theoretically embody computational automata constructions. A framework for representing RNA interactions was developed. It was shown theoretically that RNA may embody universal computation and, as such, may be capable of reaching self-referential undecidable dynamics, i.e. the Liar paradox. An Ansatz and a way forward was mapped out to discover how system expansion at the edge of undecidability may be realised through the Gödel-Turing-Post framework. Ultimately, it is hoped that this thesis has created a foundation for progress towards discovery of novelties generated at the intersection of RNA chemistry, automata theory and eco-devo evolution.

Bibliography

Abrahão, F. S., Wehmuth, K. and Ziviani, A. (2019). 'Algorithmic Networks: Central Time to Trigger Expected Emergent Open-Endedness'. In: *Theoretical Computer Science* 785, pp. 83–116.

Adams, A., Zenil, H., Davies, P. C. W. and Walker, S. I. (2017). 'Formal Definitions of Unbounded Evolution and Innovation Reveal Universal Mechanisms for Open-Ended Evolution in Dynamical Systems'. In: *Scientific Reports* 7, Article 997.

Arnold, C., Stadler, P. F. and Prohaska, S. J. (2013). 'Chromatin Computation: Epigenetic Inheritance as a Pattern Reconstruction Problem'. In: *Journal of Theoretical Biology* 336, pp. 61–74.

Benenson, Y., Adar, R., Paz-Elizur, T., Livneh, Z. and Shapiro, E. (2003). 'DNA Molecule Provides a Computing Machine with Both Data and Fuel'. In: *Proceedings of the National Academy of Sciences* 100.5, pp. 2191–2196.

Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z. and Shapiro, E. (2001). 'Programmable and Autonomous Computing Machine Made of Biomolecules'. In: *Nature* 414.6862, pp. 430–434.

Benner, S. A., Kim, H.-J. and Carrigan, M. A. (2012). 'Asphalt, Water, and the Prebiotic Synthesis of Ribose, Ribonucleosides, and RNA'. In: *Accounts of chemical research* 45.12, pp. 2025–2034.

Cafferty, B. J., Gállego, I., Chen, M. C., Farley, K. I., Eritja, R. and Hud, N. V. (2013). 'Efficient Self-Assembly in Water of Long Noncovalent Polymers by Nucleobase Analogues'. In: *Journal of the American Chemical Society* 135.7, pp. 2447–2450.

Chen, J.-L. and Pace, N. R. (1997). 'Identification of the Universally Conserved Core of Ribonuclease P RNA.' In: *RNA* 3.6, pp. 557–560.

Derksen, M., Mertens, V. and Pruijn, G. J. M. (2015). 'RNase P-Mediated Sequence-Specific Cleavage of RNA by Engineered External Guide Sequences.' In: *Biomolecules* 5.4, pp. 3029–3050.

Ekland, E. H., Szostak, J. W. and Bartel, D. P. (1995). 'Structurally Complex and Highly Active RNA Ligases Derived from Random RNA Sequences'. In: *Science* 269.5222, pp. 364–370.

Evans, D., Marquez, S. M. and Pace, N. R. (2006). 'RNase P: Interface of the RNA and Protein Worlds'. In: *Trends in Biochemical Sciences* 31.6, pp. 333–341.

Ewald, W. B. (2005). *From Kant to Hilbert a Source Book in the Foundations of Mathematics. Volume 2*. Oxford Science Publications. Oxford: Clarendon Press. ISBN: 1-282-36565-7.

Froese, T., Campos, J. I., Fujishima, K., Kiga, D. and Virgo, N. (2018). 'Horizontal Transfer of Code Fragments between Protocells Can Explain the Origins of the Genetic Code without Vertical Descent'. In: *Scientific Reports* 8, Article 3532.

Gilbert, S. F. and Epel, D. (2009). *Ecological Developmental Biology : Integrating Epigenetics, Medicine, and Evolution*. Sunderland, Mass: Sinauer Associates. ISBN: 978-0-87893-299-3.

Goldenfeld, N. and Woese, C. (2007). 'Biology's next Revolution'. In: *Nature* 445, p. 369.

Griffiths, P. and Stotz, K. (2013). *Genetics and Philosophy: An Introduction*. Cambridge Introductions to Philosophy and Biology. New York: Cambridge University Press. ISBN: 1-107-35726-8.

Guerrier-Takada, C., Li, Y. and Altman, S. (1995). 'Artificial Regulation of Gene Expression in Escherichia Coli by RNase P'. In: *Proceedings of the National Academy of Sciences* 92.24, pp. 11115–11119.

Hadidi, A. (2017). *Viroids and Satellites*. London, United Kingdom: Academic Press. ISBN: 978-0-12-801702-9.

Higgs, P. G. (2016). 'The Effect of Limited Diffusion and Wet–Dry Cycling on Reversible Polymerization Reactions: Implications for Prebiotic Synthesis of Nucleic Acids'. In: *Life* 6.2, Article 24.

Higgs, P. G. and Lehman, N. (2015). 'The RNA World: Molecular Cooperation at the Origins of Life'. In: *Nature Reviews Genetics* 16.1, pp. 7–17.

Hilbert, D. (1902). ‘Mathematical Problems’. In: *Bulletin of the American Mathematical Society* 8, pp. 437–479.

Hilbert, D. and Ackermann, W. (1950). *Principles of Mathematical Logic*. New York: Chelsea Publishing Co. ISBN: 0-8284-0069-5.

Hofstadter, D. R. (1980). *Gödel, Escher, Bach: An Eternal Golden Braid*. Harmondsworth: Penguin. ISBN: 0-1400-5579-7.

Huang, W. and Ferris, J. P. (2006). ‘One-Step, Regioselective Synthesis of up to 50-Mers of RNA Oligomers by Montmorillonite Catalysis’. In: *Journal of the American Chemical Society* 128.27, pp. 8914–8919.

Joyce, G. F. and Szostak, J. W. (2018). ‘Protocells and RNA Self-Replication’. In: *Cold Spring Harbor Perspectives in Biology* 10.9, Article a034801.

Kauffman, S. (2016). *Humanity in a Creative Universe*. New York, NY, USA: Oxford University Press. ISBN: 0-19-939045-2.

Kleene, S. C. (1938). ‘On notation for ordinal numbers’. In: *The Journal of Symbolic Logic* 3.4, pp. 150–155.

Krakauer, D., Bertschinger, N., Olbrich, E., Flack, J. C. and Ay, N. (2020). ‘The Information Theory of Individuality’. In: *Theory in Biosciences* 139.2, pp. 209–223.

Lincoln, T. A. and Joyce, G. F. (2009). ‘Self-Sustained Replication of an RNA Enzyme’. In: *Science* 323.5918, pp. 1229–1232.

Markose, S. M. (2004). ‘Novelty in Complex Adaptive Systems (CAS) Dynamics: A Computational Theory of Actor Innovation’. In: *Physica A: Statistical Mechanics and its Applications* 344.1, pp. 41–49.

— (2017). ‘Complex Type 4 Structure Changing Dynamics of Digital Agents: Nash Equilibria of a Game with Arms Race in Innovations’. In: *Journal of Dynamics and Games* 4.3, pp. 255–284.

— (2021). ‘Genomic Intelligence as Über Bio-Cybersecurity: The Gödel Sentence in Immuno-Cognitive Systems’. In: *Entropy* 23.4, Article 405.

— (2022). ‘Complexification of Eukaryote Phenotype: Adaptive Immuno-Cognitive Systems as Unique Gödelian Blockchain Distributed Ledger’. In: *Biosystems* 220, Article 104718.

Melkebeek, D. van. (2000). *Randomness and Completeness in Computational Complexity*. 1st ed. 2000. Lecture Notes in Computer Science, 1950. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 3-540-44545-5.

Monnard, P.-A., Kanavarioti, A. and Deamer, D. W. (2003). 'Eutectic Phase Polymerization of Activated Ribonucleotide Mixtures Yields Quasi-Equimolar Incorporation of Purine and Pyrimidine Nucleobases'. In: *Journal of the American Chemical Society* 125.45, pp. 13734–13740.

Moschovakis, Y. (2010). 'Kleene's Amazing Second Recursion Theorem'. In: *Bulletin of Symbolic Logic* 16.2, pp. 189–239.

Neveu, M., Kim, H.-J. and Benner, S. A. (2013). 'The "Strong" RNA World Hypothesis: Fifty Years Old'. In: *Astrobiology* 13.4, pp. 391–403.

Ouyang, Q., Kaplan, P. D., Liu, S. and Libchaber, A. (1997). 'DNA Solution of the Maximal Clique Problem'. In: *Science* 278.5337, pp. 446–449.

Post, E. L. (1944). 'Recursively Enumerable Sets of Positive Integers and Their Decision Problems'. In: *Bulletin of the American Mathematical Society* 50.5, pp. 284–316.

Prohaska, S. J., Stadler, P. F. and Laubichler, M. (2019). 'How and What Does a Biological System Compute'. In: *The Energetics of Computing in Life and Machines*. Santa Fe, New Mexico: SFI Press, pp. 119–139.

Prokopenko, M., Harré, M., Lizier, J., Boschetti, F., Peppas, P. and Kauffman, S. (2019). 'Self-Referential Basis of Undecidable Dynamics: From the Liar Paradox and the Halting Problem to the Edge of Chaos'. In: *Physics of Life Reviews* 31, pp. 134–156.

Reiter, N. J., Osterman, A., Torres-Larios, A., Swinger, K. K., Pan, T. and Mondragón, A. (2010). 'Structure of a Bacterial Ribonuclease P Holoenzyme in Complex with tRNA'. In: *Nature* 468.7325, pp. 784–9.

Rogers, H. (1967). *Theory of Recursive Functions and Effective Computability.* McGraw-Hill Series in Higher Mathematics. New York: McGraw-Hill. ISBN: 0-262-68052-1.

Rogers, J. and Joyce, G. F. (2001). 'The Effect of Cytidine on the Structure and Function of an RNA Ligase Ribozyme'. In: *Rna* 7.3, pp. 395–404.

Sutherland, J. D. (2016). 'The Origin of Life—Out of the Blue'. In: *Angewandte Chemie International Edition* 55.1, pp. 104–121.

Sutner, K. (2012). ‘Computational Classification of Cellular Automata’. In: *International Journal of General Systems* 41.6, pp. 595–607.

Svahn, A. J. and Prokopenko, M. (2023). ‘An Ansatz for computational undecidability in RNA automata’. In: *Artificial Life* 29.2.

Torres-Larios, A., Swinger, K. K., Pan, T. and Mondragón, A. (2006). ‘Structure of Ribonuclease P—a Universal Ribozyme’. In: *Current opinion in structural biology* 16.3, pp. 327–335.

Turing, A. M. (1937). ‘On Computable Numbers, with an Application to the Entscheidungsproblem’. In: *Proceedings of the London Mathematical Society* s2-42.1, pp. 173–198.

— (1939). ‘Systems of Logic Based on Ordinals’. In: *Proceedings of the London Mathematical Society* s2-45.1, pp. 161–228.

Van Heijenoort, J. (1967). *From Frege to Gödel; a Source Book in Mathematical Logic, 1879-1931*. Source Books in the History of the Sciences. Cambridge, Mass: Harvard University Press. ISBN: 0-674-32450-1.

Virgo, N., Agmon, E. and Fernando, C. (2017). ‘Lineage Selection Leads to Evolvability at Large Population Sizes’. In: *Artificial Life Conference Proceedings 14*, pp. 420–427.

Wagner, G. P. and Altenberg, L. (1996). ‘Perspective: Complex Adaptations and the Evolution of Evolvability’. In: *Evolution* 50.3, pp. 967–976.

Waugh, D. S., Green, C. J. and Pace, N. R. (1989). ‘The Design and Catalytic Properties of a Simplified Ribonuclease P RNA’. In: *Science* 244.4912, pp. 1569–1571.

Wegscheid, B. and Hartmann, R. K. (2006). ‘The Precursor tRNA 3’-CCA Interaction with Escherichia Coli RNase P RNA Is Essential for Catalysis by RNase P in Vivo’. In: *Rna* 12.12, pp. 2135–2148.

Will, C. L. and Luhrmann, R. (2011). ‘Spliceosome Structure and Function’. In: *Cold Spring Harbor Perspectives in Biology* 3.7, Article a003707.

Woese, C. R. (1965). ‘On the Evolution of the Genetic Code’. In: *Proceedings of the National Academy of Sciences* 54.6, pp. 1546–1552.

Woese, C. R. and Goldenfeld, N. (2009). ‘How the Microbial World Saved Evolution from the Scylla of Molecular Biology and the Charybdis of the Modern Synthesis’. In: *Microbiology and Molecular Biology Reviews* 73.1, pp. 14–21.

Woese, C. R. (1967). *The Genetic Code: The Molecular Basis for Genetic Expression*. New York: Harper & Row. ISBN: 0-06-047176-X.

— (2004). 'A New Biology for a New Century'. In: *Microbiology and Molecular Biology Reviews* 68.2, pp. 173–186.

Wolfram, S. (1984). 'Computation Theory of Cellular Automata'. In: *Communications in Mathematical Physics* 96.1, pp. 15–57.

Yang, Y.-H., Li, H., Zhou, T., Kim, K. and Liu, F. (2006). 'Engineered External Guide Sequences Are Highly Effective in Inducing RNase P for Inhibition of Gene Expression and Replication of Human Cytomegalovirus'. In: *Nucleic acids research* 34.2, pp. 575–583.

Zenil, H., Hernández-Quiroz, F. and Hernández-Quiroz, S. (2016). 'The Limits of Decidable States on Open-Ended Evolution and Emergence'. In: *Artificial Life Conference Proceedings 13*. MIT Press, pp. 200–207.

Zou, H., Lee, J., Kilani, A. F., Kim, K., Trang, P., Kim, J. and Liu, F. (2004). 'Engineered RNase P Ribozymes Increase Their Cleavage Activities and Efficacies in Inhibiting Viral Gene Expression in Cells by Enhancing the Rate of Cleavage and Binding of the Target mRNA'. In: *Journal of Biological Chemistry* 279.31, pp. 32063–32070.