Learning Deterministic Finite Automata with Signed Examples: An Investigation into the Role of Entropy in Optimal Model Selection

MSc Thesis *(Afstudeerscriptie)*

written by

John Fergus William Smiles
(born February 6th, 1992 in Exeter, United Kingdom)

under the supervision of **Prof. Dr. Pieter W. Adriaans**, and submitted to the Examinations Board in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam*.

Date of the public defense:	**Members of the Thesis Committee:**
May 18, 2021	Dr. Ekaterina Shutova (Chair)
	Prof. Dr. Pieter W. Adriaans
	Prof. Dr. Khalil Sima'an
	Dr. Ronald de Haan

Institute for Logic, Language and Computation

0.1 Abstract.

The thesis presents an analysis of the hardness results for learning deterministic finite automata from signed data examples. Focusing on the conditions which allow for learning to take place, we investigate a notion of a "fair" sample that would typically permit learning. Despite demonstrating the need for such a sample in overcoming the hardness of the learning problem, we also detail a major problem with the size of the sample that is required to be considered fair.

The major contribution of the thesis is to develop our understanding of the role of entropy in optimal model selection and learning; developing connections to aspects of cognition and inference. We provide a sufficient background introduction to data compression, and an analysis of the Minimum Description Length (MDL) principle for grammar induction, with the purpose of preparing the reader to understand these relationships. We also present novel results which explicate deeper problems with the process of DFA model selection.

The contribution is specifically motivated by a similar result for the class of partial and total recursive functions [Adriaans (2020)]: we analyse data sets which have multiple optimal models under MDL, each of which share little to no mutual information. These so-called "polysemantic" data sets are shown to exist abundantly in the context of the much weaker class of regular languages. Their existence poses severe difficulties to any algorithmic methods for reasoning inductively from low quality data samples; there seems to be no way of avoiding mutually irreconcilable interpretations of certain samples. The focus on polysemy brings some interesting considerations about learning to the fore, which we relate back to the central concerns of the thesis.

0.2 Acknowledgements.

Thanks go out to everyone who supported me on this project: my partner, my friends, family, and the community at the ILLC.

To my supervisor, Professor Pieter Adriaans, for the boundless stream of ideas that helped me shape my own interests in a challenging field that was quite new to me.

To my academic mentor, Professor Dick de Jongh, for his support during my studies at the ILLC.

To Tanja Kassenaar, for looking out for my well-being and for all the help dealing with Brexit-related troubles, which would have proven a far greater distraction otherwise.

To the Whitestone Church House Educational Foundation, for funding my studies after I was delayed graduating due to the Covid-19 pandemic.

Special thanks to my folks, for listening to my struggles and striving to grapple with my ideas. And, lastly for all the beer and whisky while I was writing the manuscript in Devon.

Contents

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

Learning Deterministic Finite Automata with Signed Examples: An Investigation into the Role of Entropy in Optimal Model Selection

MSc Logic Thesis at the Universiteit van Amsterdam

Fergus Smiles April 27, 2021

1 Introduction.

Suppose an infinite set of strings (sequences of symbols) of finite length. This could be, for instance, the set of even numbers, or the set of all finite combinations of letters from the Roman alphabet. Given some finite sampling of this set, how hard would it be, computationally speaking, to infer the structure of the infinite set? That is, how hard would it be to recover (all, or most of) the infinitely many strings that are members of the set: viz. defined by some rule for membership (e.g. the property of *evenness* from our first example)? Moreover, how would this problem change if one were given, in addition, a finite sample of strings labeled to be not a member of the infinite set? And, if it is possible to effectively learn an infinite set from a finite sample, what are the conditions that are typically required in order for learning to take place? This thesis concerns itself with the last question in particular.

In the context of 'proper' regular languages, viz. infinite sets of finite strings which are modelled by finite-state automata, there is a rich history concerning the former questions: it is possible to infer a grammar for a regular language by running a state-merging algorithm which attempts to derive a minimal, consistent, deterministic finite automaton (DFA), from a finite collection of finite strings, each labeled as to whether the DFA should accept or reject. Research into the inferribility of regular grammars provides an analysis of the bedrock of such methods, as the comparatively simple class of languages they generate (regular languages) occupy the lowest rung in the Chomsky computational hier-

archy [1]: regular ⊂ context-free ⊂ context-sensitive ⊂ recursively-enumerable, corresponding to finite, pushdown, and linear-bounded automata, and Turing machines, respectively. Thus, what is at stake is the achievability of methods for inferring grammars from finite samples itself. However, the problem for arbitrary DFA turns out to be *at least* as hard as the hardest problems solvable in nondeterministic polynomial time (NP-hard).

Our object of study is the following general learning algorithm, formally defined in §2, (definition 2.17): using a prefix-tree acceptor (PTA) that represents exactly the positive (accepting) data examples, merge states of the PTA according to a pre-selected heuristic, such as code-length minimisation, in order to develop a hypothesis for the data which generalises the sample. The goal is for the learner to identify a hypothesis that will match the label of the hidden target (membership in the target regular language) on all (or most) examples from the same distribution. The hypothesis developed is a deterministic finite automaton (a model for a regular language), defined as follows:

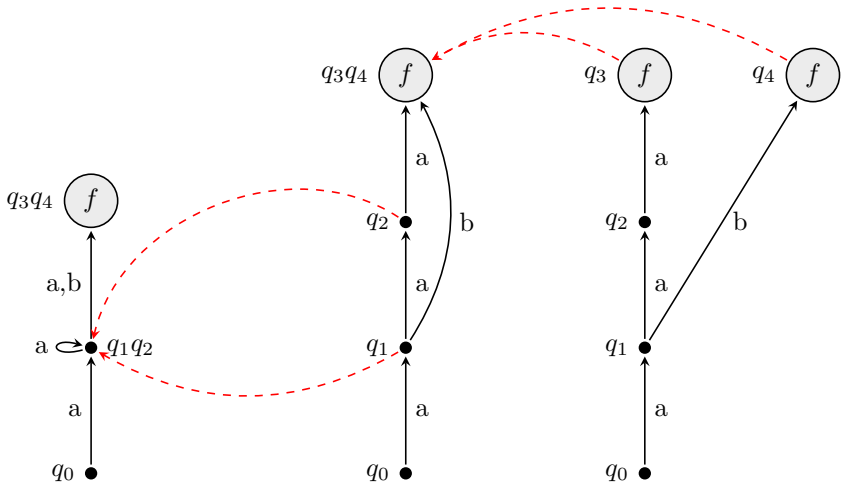
Definition 1.1 (Deterministic finite automaton) *A* deterministic finite-state automaton, *M, is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:*

$$\begin{aligned} Q &\quad \text{is a finite set of states } q \in Q \\ \Sigma &\quad \text{is a finite alphabet of symbols } a \in \Sigma \\ \delta : Q \times \Sigma \mapsto Q &\quad \text{is the } \textit{transition} \text{ or } \textit{step} \text{ function.} \\ q_0 \in Q &\quad \text{is the initial state} \\ F \subseteq Q &\quad \text{is the set of final states } f \in F \end{aligned}$$

See §2, (definitions 2.1; 2.2) for the full definition of deterministism and nondeterministism for finite automata. Terminology is given in §A.1.

Definition 1.2 (Input word) *Starting at the initial state, the automaton reads a finite string of symbols, $w = a_1, a_2, \ldots, a_n$, where $a_i \in \Sigma$, which is called an* input word, *$w \in \Sigma^*$. By reading a symbol a_i, the automaton* transitions *to a new state $q_j \in \delta(q_{j-1}, a_i)$ and* accepts *the word w if for each a_i there is an accessible state and for the final symbol a_n the accessible state is also a final state.*

As an example of the kind of state-merging algorithm we are concerned with, suppose we have a DFA. The set of strings which can be modelled with this DFA is called the *language* of the DFA (see §2, definitions 2.4, 2.5, and 2.6). Call this language, L_T, the target language. Next suppose we have a sample of positive examples $D_+ = \{aaa, ab\}$ such that $D_+ \subseteq L_T$. In other words, we can be sure that the target automaton, when reading either of the sequences '*aaa*' or '*ab*', will successfully transition from the initial state to some final state and halt its computation afterwards. The given information can be captured by constructing the PTA. The sample is then generalised by merging states of the PTA as follows:



The digraph on the far right is the PTA, and represents D_+ exactly. The centre digraph is formed by the merge of states q_3 and q_4. The resulting automaton is equivalent to the language of the PTA and accepts the finite language $L_1 = \{aaa, ab\}$. The merge of q_1 and q_2, however, yields a nondeterministic finite automaton (NFA), that accepts the *infinite* language $L_2 = \{a^+\{a, b\}\}$. That is the language of *at least* one *'a'* followed by an *'a'* or *'b'*. Now that we have a grammar for an infinite language, the question is, for some target language, L_T, do we have $\forall w\big(w \in L_2 \iff w \in L_T\big)$ (i.e. equivalence)? Otherwise, can one say, for *most* w, with high probability $w \in L_2 \iff w \in L_T$?

It is often the case, however, that nondeterminism is controlled for under state merging. This is because the minimal NFA is not unique. It is for this reason that the problem of NFA minimisation is hard (PSPACE-complete) [2]. Focusing on DFA thus simplifies the search problem. Nevertheless, the efficacy of these general learning models for DFA, leaves much to be desired; there are problems with over-generalisation and over-fitting. Furthermore, as mentioned above, the computational complexity of the underlying combinatorial optimisation problem is intractable. An exhaustive search to find the smallest DFA consistent with an arbitrary data sample is NP-hard [3][4], and it is NP-hard to approximate a target DFA of size OPT by a DFA of size within OPT^k for any constant factor k unless P=NP [5]. Phrased as a decision problem, it is NP-complete to decide, given $k \in \mathbb{N}$ and two finite sets A and B of words, whether there exists a DFA with at most k states which accepts every string in A and none of the strings in B [6]. Furthermore, the cryptographic hardness of learning DFA functions from signed data examples under the Probably Approximately Correct (PAC) learning model is a reduction from the problem of inverting the RSA cryptosystem, since one can encode the RSA function into a DFA [7]. This constitutes good evidence that not even the average-case DFA will be easily PAC learnable.

The evidence against PAC learning begs the question as to whether a good choice of learning model can alleviate the difficulty of learning DFA that is suggested by the worst-case analysis. Indeed, many of the positive results for learning formal languages have been achieved by additionally hypothesising constraints on the distribution of examples in the data sample, see [8][9]. For DFA in particular, Dennis (2001) showed that DFA are Probably Exactly Correct (PEC) learnable with simple positive examples [10]. However, this algorithm is equipped with access to a representation of the target language, something we wish to avoid. Examples which are simple according to this representation are weighted likelier. The information can then be used to lessen the difficulty of the learning problem.

Suffice it to say, the problem of inferring a minimally consistent DFA is embedded in a web of related problems of quite some depth. However, the problem is also manifestly interesting from an over-arching computational and philosophical stand-point as it penetrates deep into the heart of issues relating to the foundation of inductive and abductive reasoning; how can we be sure our abductive predictions are good and our inductive inferences valid if we are working with insufficient (noisy, incomplete or inconsistent) information?

Currently, there is no general theory for accurately inferring an arbitrary optimal model under conditions of noisy and/or incomplete information. Computationally speaking, there is good reason to believe that a general theory is not feasible, as even in simple contexts, such as learning DFA, the problem is computationally hard. Considering the conditions which allow for learning to take place may shed light on the feasibility of such a theory. Indeed, we require a deeper understanding of the sort of samples that typically permit learning. This thesis will develop and analyse exactly this notion and its relation to entropy.

In particular, we investigate samples generated by random walks under maximum entropy assumptions: assuming the given sample data is the result of m random walks on the underlying digraph of the automaton, we call the data sample *fair* with respect to the target automaton if and only if after m random walks, every possible sub-path has been "seen" by the random walkers. In other words, if we have a data set which almost surely traces out the complete transition structure of some unknown digraph, underlying a DFA, then we can call the respective sample *fair* with respect to the DFA in the sense that there are no hidden constructions. Thus, we can be sure that a fair sample contains the necessary structural information to reconstruct the DFA. However, in some cases, there are problems with the sample size necessary to be considered fair.

Worse still, there are situations where learning is in principle not possible even with fair data; the data may be "polysemantic". That is, there may be multiple optimal models for the same data set which share little to no mutual information. The existence of polysemantic data sets in very simple situations, proven in this thesis, can be used to demonstrate that they must occur in abundance. The reason for this, we argue, is that there is a kind of subjectivism within the

very heart of optimal model selection that defies resolution, and this is related to entropy in the following sense: optimal compression is sometimes disjoint or irreconcilable, making it incredibly hard to differentiate between equally parsimonious models.

Finding patterns and regularities in data is a natural process of structuring data in order to fit a hypothesis or model. This is in essence how learning and data compression relate to one another, and part of the reason why the problem of optimal model selection is computationally hard: it requires at least the power of nondeterministic polynomial-time computation, where a correct solution can be "guessed" non-deterministically, analogous to the "creative leaps" of human problem-solving. Ensuring the sample is fair, in a sense, removes the guess-work. However, by doing so one introduces problems with the size of the sample necessary to remove guess-work. By paying attention to the kind of samples necessary to recover the correct hypothesis, a better appreciation of the conditions which typically permit learning is achieved. Even if this is a somewhat Pyrrhic victory, seeing as the computational pitfalls surrounding DFA induction are quite severe, the hope is that the approach of this thesis homes in on quintessential features necessary for learning, irrespective of the context.

1.1 Philosophical and Computational Roots.

The problem outlined above is fundamentally related to a problem in the History of Philosophy called the Essential Problem of Empiricism [11]: namely, how one can utilise a finite empirical perspective to form conclusions that go beyond the finitude of one's subjective experience. This boils down to a question of whether universal principles are ever truly learnable, or knowable, from experience, if all of what one can know *a posteriori* is gathered by the senses, which are liable to err, and which provide insufficient information about reality.

In other words, if, by assumption, empirical data does not permit the learning of universals, then for the radical empiricist who does not believe in any form of immediate access to an immaterial world, conclusions about that which is beyond one's experiential data, are, by their nature, only attainable via a process of abduction, or, as Hume argued, from the inductive fallacy coupled with the fact that causal events are "constantly conjoined"[12]. However, there is something of a paradox in that empirical investigation does sometimes reveal truth about the nature of reality; in some cases, one succeeds in learning a universal principle. What is different about these successful situations is not metaphysical. On the contrary, empirical investigation leads to knowledge when there is the right kind of data at hand. More specifically, the data ought to be consistent with the phenomenon and sufficient to predict further observations accurately. The focus from the perspective of Philosophy, however, has been

on the limits of knowledge; Philosophy has sought to find a firm bedrock of unassailable knowledge, something not granted by empirical investigation. The exact conditions which permit learning and the acquisition of knowledge to take place have not been properly addressed being outside of the remit of classical Philosophy.

The situation for machine learners is in many ways also held to account by Philosophy; the exact difficulties regarding *a posteriori* knowledge recur in the context of unsupervised machine learning, where a learning algorithm must generate a model for some data which it builds via an algorithmic version of induction or abduction. Supervised machine learners, on the contrary, can query whether a particular string of data belongs to the target model or not. This kind of algorithm is described as a machine with access to an oracle. Using oracles, or some other kind of instructive feedback, one can more explicitly train an algorithm to learn something. It is primarily due to this ability that supervised learning algorithms are considered computationally 'easier' than unsupervised ones. Learning algorithms which discover optimal models without such access are arguably closer to 'real' cognition. Our focus is on unsupervised learning.

The question is: how can an algorithm, using only a learning set made up of finite data examples—that is to say, partial information—build a model or hypothesis that is sufficiently generalisable for unseen data, whilst also achieving optimal codification for the data? We introduced the notion that this is a rather complex feat to accomplish even in quite simple contexts such as the area of finite automata identification. One explanation of the computational hardness of optimal model selection may be that *identifying* the optimal model for an arbitrary data set is intuitively just as hard as *checking* whether a given model is optimal; in a sense we have to check every possible model in order to decide whether the given model is optimal. As a rule of thumb: problems where it is just as hard to merely check whether a given solution is correct, as it is to actually find a solution, fall into the category of exponential time algorithms, (EXPTIME). In general, if one has no information to indicate the superior likelihood of any model, the difficulty of solving and checking will be identical.

EXPTIME problems are intractable due to the fact that it would take a brute-force algorithm exponential time to even check a correct solution. One example would be computing the perfect strategy for a game of $(n \times n)$-Chess [13]; checking whether the strategy is optimal requires one to check all possible moves and the repercussions of each move, which intuitively seems equally as hard as actually computing which move is optimal. This is different to the situation one finds in Sudoku, where checking whether a given solution is correct can be done quickly without much effort, whereas actually solving the problem may be slow and require complex computation. This difference in computational complexity between checking and solving is the hall-mark of nondeterministic polynomial-time (NP) problems. Note: NP-hardness means *at least* as hard as the hardest problems in NP.

Interestingly, for the case of model selection in human cognitive processing, there seems to be an analogous correspondence between the perceived cognitive complexity of the problem, and the quality and size of the data sample. For instance, let us assume we seek an optimal model to explain the apparent movement of the Sun. Given insufficient data, e.g. the experiential data from observations on Earth with the naked eye, there is no way to rule out swathes of possible models; all one can do is to check and compare all of them—an exponential process. And indeed, before the Copernican model was generally accepted, there were solar models of varying complexity, some more or less equally parsimonious, and most of which also succeeded to "save the phenomenon". That is, to remain consistent with the data at hand. The canonical example is the Ptolemaic model *versus* the Copernican model, both of which are able to explain the "wandering" motions of the Planets. It is worth noting that, although, for instance, Aristarchus correctly conjectured that our planetary system was heliocentric, the Ptolemaic model with epicycles accounted for more of the data. Thus a true model can sometimes be ruled out under measures of parsimony. This is intriguing as it indicates that insufficient data in itself does not preclude learning, rather data is sufficient or insufficient only with respect to a model; it was not until additional information provided by Galileo, that one could finally begin to rule out geocentric models, lessening the space of possible models for the model selection problem.

Relating this back to the machine learning problem, the concerns with *computational* complexity are themselves inherently relatable to the conditions which permit learning to happen, since they provide asymptotic bounds on the difficulty of problems as they scale. It is known by the deterministic time hierarchy theorem, that the class of polynomial-time algorithms, P, is not equal to EXPTIME; there are problems in EXPTIME that simply cannot be solved with a fast algorithm [14]. P is the class of decision problems which can be solved quickly (in deterministic polynomial-time). However, for NP problems this is less clear, and depends on the P=?NP problem. Unless the world is an inherently different place than initial investigation reveals to be the case, P$\neq$NP. Thus it is safe to conjecture that DFA induction problems will be NP-hard in the worst case. What remains to be seen is whether learning DFA is hard on an average-case analysis. If the kind of data one requires is, on average, easy to aquire, learning the average DFA might be easier than originally supposed.

To sum: focusing on the conditions which allow learning to occur provides an alternative philosophical perspective to questions about epistemology and ontology. That which is learned is not necessarily *true* or *real*, but functions exactly the same; false opinions, or irrealities are nevertheless sometimes equally justified by the data. The role of entropy in optimal model selection provides the central theme for this discussion. What is it about the nature of certain data sets that allow them to be captured by a hypothesis? What allows one to identify what is noise from what is not? To describe this we need a notion of sample data that is *fair*. Roughly speaking, we need to be certain that

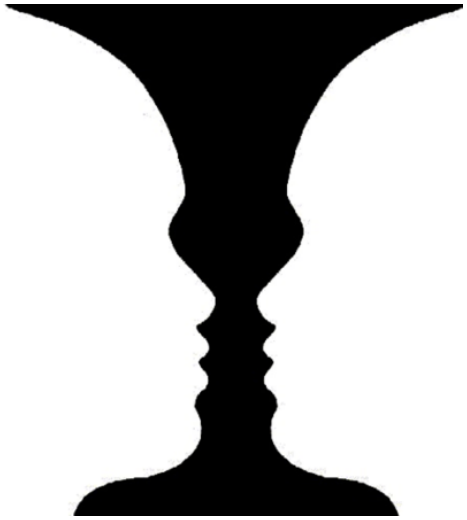


Figure 1: Gestalt vase or faces. See [15].

the sample is truly indicative of the target model that we hope the algorithm identifies. For the technical side of the matter, working out the conditions which permit learning DFA to take place will inform us as to whether the average-case complexity ought to be as hard as is indicated by the worst-case analysis. However, there are some problems with optional model selection which seem to crop up irrespective of the quality of the data samples. That is, problems which arise due to the very process of selecting optimal models. Polysemanticism is one such example. These issues seem to transcend the machine world and apply to aspects of cognition and inference more generally.

1.2 Offshoots in Psychology.

The problem of polysemanticism in optimal model selection, is, in fact, well-known in the field of Psychology. However, it goes under a different guise: namely, the notion of multistability in the theory of Gestalt Psychology. To put it succinctly, a multistable gestalt image is one in which there is an irresolvable conflict that defies the mind's ability to neatly work the information into a single interpretation as to what it is seeing. Figure 1, above, is the canonical example. The gestalt theory of perception is relevant to the concerns of the thesis in the following way: Gestalt Psychology emphasises the prime significance of perceiving patterns, rather than the individual components of an image. The state of one's mind when perceiving Figure 1 demonstrates this fact, as it is the choice of pattern on a backdrop which forces the image to coalesce into a coherent object.

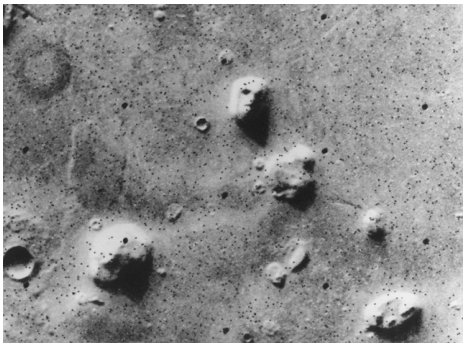


Figure 2: Pareidolia of a face in the Cydonia region of Mars. See [18].

However, the image is multistable, meaning both stable images seem to compete, causing the mind to flip between valid interpretations. In terms of optimal model selection and entropy, this is an example of our mind's failure to compress the data of an image into a single coherent meaning. There are two optimal ways to model the perceptual data; as a black vase on a background of white, or as two white faces silhouetted on a black background. In other words, the data set is polysemantic. These connections between Gestalt psychology and data compression have been studied extensively in Adriaans (2020) [16]. The phenomenon was also known to Wittgenstein, who used the polysemantic image of a "duck-rabbit" to argue for the distinction between seeing *that* and seeing *as* [17]. The ambiguous image can be seen as either a duck or as a rabbit. This suggests a distinction between simply reporting what one sees and providing an interpretation of what one sees. Wittgenstein asks "what is different: my impression? my point of view?——Can I say? I describe the alteration like a perception; quite as if the object had altered before my eyes" [17]. From the perspective of data compression, to say that one 'sees the image *as* a rabbit', is to imply that a cognitive operation has interpreted (compressed) the perceptual data to resemble a rabbit (rather than a duck).

Another related phenomena is that of *pareidolia*: the tendency to incorrectly perceive meaningful objects (often faces) in natural patterns, such as cloud formations, or in inanimate objects (see Fig 2). This is also related to the problem of polysemanticism in optimal model selection in the sense that, if the data are insufficient, there may be enough localised order in the data set to build an optimal model that generalises the data into an entirely incorrect model. This hypothesis about the data may compete with the true model when there is already a lot of randomness or noise present in the true model, as is the case with phenomena such as clouds or waves, or in low-resolution images: the meaningful pattern is simpler to grasp than the random or noisy one. Of course, as we have argued, these problems can sometimes be overcome when the

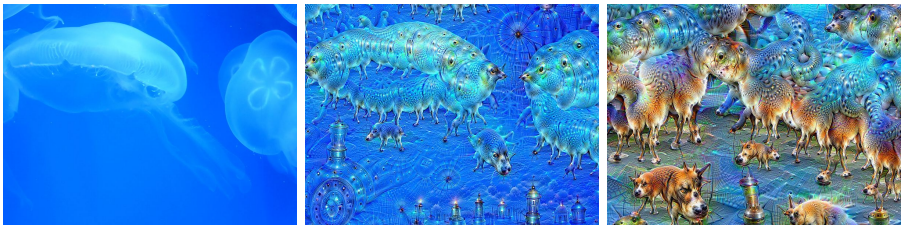


Figure 3: DeepDream pareidolia by Wiki user MartaThoma. See [20].

quality of the data is improved. And indeed, for the Face on Mars [Fig 2], this was exactly the case: in 2007, the Mars Reconnaissance Orbiter provided high-definition pictures of the the Cydonia region. The face was nowhere to be seen. As Adriaans puts it, "pareidolic phenomena can be explained on the basis of Cilibrasi's Normalised Compression Distance (NCD) when a random structured (scale free) data set (e.g. a cloud, a stain on a wall) can be compressed by a data set already known to the beholder (a face, the head of a dog etc.) [19].

So much for cases of pareidolia in human cognitive model selection; for machine learners it is far more apparent how the problem of polysemanticism might be seen to underlie pareidolic pattern recognition: for instance, DeepDream is a deep-learning programme which allows the over-processing of images by training an algorithm to detect a particular image in a data set, and then running the algorithm in reverse so as to generate dream-like enhanced images on unseen data sets. As an example, figure 3 (above) shows zero, ten and fifty iterations of an algorithm trained to detect dogs applied to a new image. What is interesting is that, eventually, after enough iterations, structures develop which could be called dog-like, despite there never having been the image of a dog in the original data set. What is going on here is akin to the cognitive model selection which allows humans to perceive faces in clouds; localised areas of order are generalised in such a way that resembles the target model already known to the beholder.

Relating the psychology of perception and the problem of optimal model selection to learning formal languages, which is the concern of this thesis, it was shown in Adriaans (2020) that for the class of partial and total recursive functions, there exist data sets which have multiple optimal models, which share little to no mutual information [16]. This thesis will show that a similar result also holds for the much weaker class of regular languages. In particular, there are close to behaviourally equivalent simulations, which share little to no algorithmic mutual information. Behaviourally equivalent simulations between finite automata are essentially the regular language counterparts of emulations between Turing machines (TM) for the class of recursively enumerable languages.

Crucially, however, there is not a one-to-one parallel between every feature of a language in each level of the Chomsky Hierarchy. In fact, the result in Adriaans

(2020) is dependent on features of universal Turing machines (UTM) that do not arise in the context of finite automata. In particular, finite automata are equivalent to read-only right moving Turing machines [21], in that both computational models generate exactly the regular languages. Such Turing machines lack a work tape; that is to say they only have constant (finite) memory. As a consequence they cannot be used as universal Turing machines, as lacking even a linear amount of space entails that they are unable to emulate other Turing machines. Since the proof of Adriaans (2020) relies on UTM emulation to prove the existence of such "polysemantic" data sets, the same or a similar argument cannot hold in the context of regular languages.

However, we were able to construct polysemantic data sets, for which there are multiple equally parsimonious models under minimal code length, each of which generate a disjoint language. We also show that there can be no simpler model than these. In the thesis the focus on poylsemanticism acts as a case-study for the way in which the quality and nature of the data condition the possibilities for learning. Moreover, polysemanticism exemplifies how entropy is in effect the key player in what makes learning straightforward, or fiendishly roundabout: it is exactly whenever the entropy of the data given a model is invariant over multiple models, that we see polysemanticism arise.

1.3 Structure and Goals of the Thesis.

We first provide necessary preliminaries (section 2): chiefly, rudimentary Automata Theory (2.1) and an introduction to Grammar Induction (2.2). More specifically, we provide an analysis of the concept of state merging on DFA, and its computational hardness. We specify a state-merging operation for finite automata which corresponds to a partitioning on the set of states of the automata; the expressivity of the automata can then be assessed via partition refinements.

We then go on to analyse the relationship between optimal model selection and entropy in the context of DFA learning (section 3). We provide background and context to the theories underlying data compression algorithms: specifically, Information Theory, including Kolmogorov Complexity (3.1) and the Minimum Description Length (MDL) principle (3.3.1). It is important to see how these theories provide a mathematical foundation to the theory of inductive inference, and, hence, how data compression is related to learning (3.2). We then investigate random walk sampling strategies and use them to develop notions of fair samples that would typically permit learning (3.3).

Next, we construct examples of data sets that are in a certain sense *truly* unlearnable by compression algorithms, namely polysemantic data sets (section 4). The goal is to show that any generic compression algorithm will have trouble finding optimal models under MDL, or other principles that emphasise parsimony, for certain data sets. This constitutes an answer to the question of

whether a state-merging processes will yield multiple models which are optimal for the same data set, whilst sharing little to no mutual information. The answer is positive, a similar result to the case of partial and total recursive functions holds.

Finally, we provide some philosophical analysis about optimal model selection and compression; and how all these relate to patterns in nature, learning and Empiricism (section 5).

A brief conclusion, summaries, and a review of future research directions are also given (section 6).

An appendix is attached for additional material that could not be fitted into the main text.

2 Preliminaries.

In this section we collect together the necessary definitions to understand the main structures we investigate in the thesis. See the appendix for terminology.

2.1 Automata.

Definition 2.1 (Deterministic) *If an automaton can transition to one and only one state, for a given current state and an input symbol, then it is a* deterministic finite automaton (DFA).

Definition 2.2 (Nondeterministic) *An automaton that, after reading an input symbol, may transition to any of a number of states, as given by its transition* relation *(as opposed to a transition* function*), is called a* nondeterministic automaton (NFA). *In this case δ is a relation, and we say that M is* nondeterministic, *and write $\Delta \subseteq Q \times \Sigma \times Q$ for the relation.*

Definition 2.3 (Labeled transition system) *A finite automaton without a given initial state, and with no given accepting states is a finite state* labeled transition system (LTS) *sometimes called a* semi-automaton. *We write $T = (S, \Lambda, \rightarrow)$ where S is a set of states, Λ is a set of labels and $\rightarrow \subseteq S \times \Lambda \times S$ is a set of labelled transitions.*

Definition 2.4 (Recognised language) *The language $L \subseteq \Sigma^*$* recognized *by an automaton is the set of all the words that are accepted by the automaton. The language* recognised *by an automaton M is written $L(M)$.*

Definition 2.5 (Regular language) *A language recognised by an NFA/DFA is* regular.

Definition 2.6 (Infinite language) *We call a language L* infinite *when $|L| = \infty$.*

Definition 2.7 (Language equivalence) *Two automata M_1 and M_2 are* language-equivalent *if and only if they generate the same language: i.e. $L(M_1) = L(M_2)$.*

Definition 2.8 (Finite trace) *Given a run of w, on an automaton M, a* finite trace *is a finite path in the directed digraph underlying M, starting from the initial state, whose labels on the edges trace out the word w. Note: w need not be a word accepted by M to be in M's set of traces, written $trace(M)$.*

Definition 2.9 (Trace equivalence) *Two LTS T_1 and T_2 are* trace-equivalent *if and only if they generate the same set of traces: i.e. $trace(T_1) = trace(T_2)$.*

We follow [22] with the following two definitions, and prove the unproven lemma.

Definition 2.10 (Finite partition) *A* partition π *of a set* X *is a set of nonempty subsets of* X *such that every element* $x \in X$ *is in exactly one of these subsets.* $B(x, \pi) \subseteq X$ *indicates the subset of the partition* π *of which* x *is an element.*

Definition 2.11 (Quotient automaton) *Let* $M = (Q, \Sigma, \delta, q_0, F)$ *be a deterministic finite automaton, the* quotient automaton $A/\pi = (Q_\pi, \Sigma, \delta_\pi, B(q_0, \pi), F_\pi)$ *derived from* M *on the basis of a partition* π *of* Q *is defined as follows*:

(1) $Q_\pi = Q/\pi = \{B(q, \pi) \ ; \ q \in Q\}$

(2) $F_\pi = \{B \in Q_\pi \ ; \ B \cap F \neq \emptyset\}$

(3) $\delta_\pi : (Q_\pi \times \Sigma) \to 2^{Q_\pi} \ ; \ \forall B, B' \in Q_\pi, \ \textit{and} \ \ \forall a \in \Sigma$
$B' \in \delta_\pi(B, a) \iff \exists q, q' \in Q, q \in B, q' \in B' \ \textit{and} \ q' \in \delta(q, a).$

States of Q *that are partitioned into the same equivalence class* B*, are* merged.

Lemma 2.12. *If an automaton* A/π_j *is derived from an automaton* A/π_i *by means of a partition then* $L(A/\pi_i) \subseteq L(A/\pi_j)$.

Proof. Follows immediately from definition 2.11: $\forall a \in \Sigma$ and $q, q' \in Q/\pi_i$ with $q' \in \delta(q, a)$, and some partition π_j of Q/π_i such that $q \in B$, $q' \in B'$, we get $B' \in \delta'(B, a)$. Thus, every transition relation is preserved in the quotient automaton, and since $F/\pi_j = \{B \in Q/\pi_j \mid B \cap F/\pi_i \neq \emptyset\}$, then every accepting word in A/π_i has a path, and is also an accepting word in A/π_j as desired.

□

We base the following two definitions on what is essentially found in [23].

Definition 2.13 (Simulation) *The relation* $\sim$ *which satisfies the following is considered to be a simulation* R_{sim} *between labeled transition systems* T, T'*:*

$$q \sim q' \iff \text{If } q \xrightarrow{a} r, \text{ then } q' \xrightarrow{a} r' \text{ for some } r' \text{ such that } (r, r') \in R_{sim}$$

For automata M, M'*, there are the added properties:*

$$q_0 \sim q_0'$$
$$q \sim q' \implies \Big(q \in F_M \implies q' \in F_{M'}\Big).$$

Definition 2.14 (Simularity) *Two states q, q' are similar, written $q \sim q'$ if and only if there exists a simulation R_{sim} between them, and $(q, q') \in R_{sim}$.*

$$\sim := \bigcup \{R_{sim} \mid R_{sim} \text{ is a simulation}\}$$

Corollary 2.15 *As a consequence of lemma 2.13 and the above definitions, the quotient automaton M' derived from M via some partition*, simulates *M.*

Definition 2.16 (Signed data) *A sample of the set of words accepted by some unseen automaton M produces a positively* signed data set *D_+, whereas a sample of the set of non-accepting words of M, taken from the co-automaton of M, produces a* negatively signed data set *D_-.*

Definition 2.17 (State-merging DFA induction algorithms) *A State-merging DFA induction algorithm starts from an initial automaton called a prefix tree acceptor $PTA(D_+)$. The PTA is the largest trimmed DFA accepting exactly D_+ The algorithm then merges states in the PTA relative to a partition π of the state set of the original automaton. States belonging to the same block of π are merged in the resulting quotient automaton. Any accepting path in the PTA is also an accepting path in PTA/π.* This clear definition is found in [24].

Theorem 2.18 *State-space partitioning on DFA does not always preserve determinism.*

Proof. Suppose a DFA A with $q_1 \in \delta(q_0, a)$ and $q_2 \in \delta(q_1, a)$ for some $a \in \Sigma$ where $uaav \in L(A)$ for some $u, v \in \Sigma^*$. The merge of q_0, $q_1 \in B_1$, and $q_2 \in B_2$ under some partition π implies $B_1 \in \delta(B_1, a)$ and $B_2 \in \delta(B_1, a)$ by part (3) of the definition of quotient automaton. The resulting automaton A' is now nondeterministic.

□

Theorem 2.19 *Partitioning does not always preserve language equivalence.*

Proof. We had for the constructed DFA A, $uaav \in L(A)$, where $a \in \Sigma$, and $u, v \in \Sigma^*$. Further suppose $uaaav \notin L(A)$. However, the DFA A' accepts strings $ua^*av \in L(A')$, where a^* is a finite number of iterations of a. Clearly, therefore $L(A) \neq L(A')$. Thus partitioning can easily break language equivalence.

□

2.2 Grammar Induction.

The problem of identifying an unknown deterministic finite automaton (DFA) from signed (positive D_+ and negative D_-) data examples is central to the study of Grammar Induction. The task is to inductively or abductively infer which DFA generated the data purely in virtue of the labeled sample. If one can be certain that the data are indicative of the model then the inference is an induction from the specific data to a possibly true generalisation. However if they are not indicative of the target language, then the inference is abductive and one can only make likely guesses as to the correct structure. This may occur when the data is an incomplete or corrupted observation of the target language.

Essentially, an automaton may be employed as a finite representation of a formal language. The rules which yield the formal language are called its grammar. Thus, the task of effectively identifying a grammar from strings of data, amounts to syntactic or structural pattern recognition. In turn, this corresponds to the problem of selecting an automaton which represents the formal language.

However, not only is it required that a learning algorithm succeed in identifying a consistent DFA which stands as a theory–that is to say, a hypothesis–for some data sample, it is also normally required that the algorithm select an optimal (viz. minimal) theory for the data encoded in the model. Indeed, if we were only concerned with consistency, *ceteris paribus*, then the universal automaton would suffice for all data sets. Thus, although learning algorithms for grammar induction tend to prefer automata with lower descriptive complexity, that offer some form of data compression by reducing the state space by means of state merging, clustering, or partition-refinement, the smallest model is *not* necessarily the best. In fact, we require not only to have a minimal description of the model, but to minimise the description of the model *and* the data when encoded with the model.

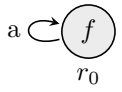
2.2.1 State Merging.

There are two fundamental functions for state merging algorithms for DFA grammar induction that all such algorithms share: namely, (1) the actual function of merging states, and (2) an initial function which constructs the prefix tree acceptor (PTA). The tree automaton accepts exactly D_+, after which the algorithm will successively merge states to induce a generalised language. The PTA learns examples from D_+ by generating a state for each prefix from the examples and thereby constructing the smallest DFA A, which is a tree for which $L(A) = D_+$ holds. The key difference between the respective algorithms is only therefore (1), i.e. the order in which the states are considered for merging [24].

Sometimes, however, as we saw in the previous subsection, merges do not pre-

serve determinism. Recall, that the minimal *NFA* for some language L is not necessarily unique. Hence, we may wish to identify only DFA in order to sidestep this issue. However, transforming an NFA into a DFA may increase the set of states exponentially: if the NFA had n states, the translation into a DFA may require up to 2^n states to encode the same regular language [25]. But, as a corollary to this, it shows that NFA do not actually gain any additional computational power (in terms of computability) from non-determinism. Indeed, if every language that can be recognised by an NFA can also be recognised by a DFA, and vice versa (trivially, every DFA is an NFA), then they define the same class of languages. In sum, for the sake of simplicity we may require that non-determinism is also controlled for: that way, supposing we discover the correct regular language, we can be sure to find a unique optimal model in terms of a DFA by minimising the regular language, see [26].

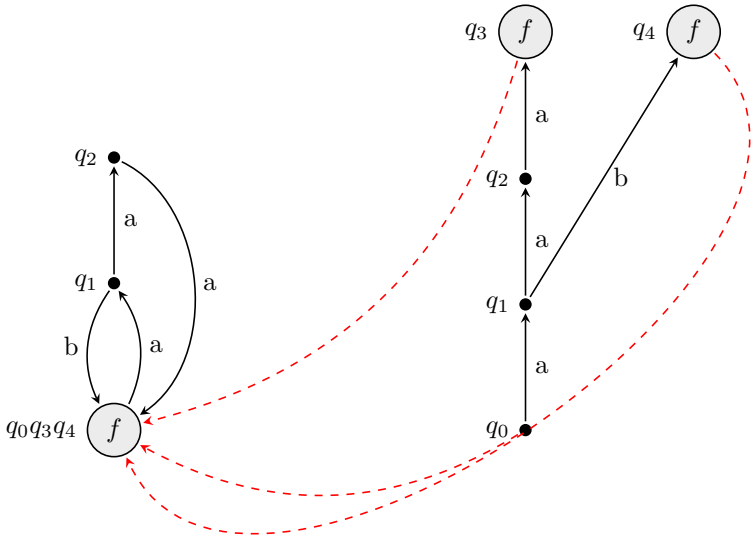
However, perhaps more crucially, if we do not include negative data examples in our learning set, then over-generalisation will pose a problem. This is because the minimum consistent DFA/NFA for purely positive data examples is simply the universal automaton:



Where $a \in \Sigma$ is any transition for any symbol in any finite alphabet. Clearly the universal automaton can accept any word from any regular language. However, the universal automaton will not be optimal in most cases as it does not capture any structure at all: it accepts all possible strings over any alphabet and is not able to discern any of the grammatical rules for any of these languages. Put differently, the universal automaton *simulates* any automaton, but not every automaton may simulate the universal automaton.

Generalisation is controlled by the negative sample D_- to prevent merging states which are not compatible. Merging incompatible states would lead to an inconsistent automaton: in other words, a DFA which accepts at least one negative string from D_-. It is for this reason that DFA induction with only positive examples is more difficult, as there is no way to rule out inconsistent automata.

Suppose we add negative examples D_- to the data set from the introduction, yielding a new data set $D = \{aaa_+, ab_+, b_-, aab_-\}$. The same PTA is generated, below right. In this case, merging, for example q_0 and q_1 would create the transition $q_4 \in \delta(\{q_0, q_1\}, b)$. This automaton would accept the string b, generating an automaton inconsistent with D. The universal automaton would also be inconsistent with the data, as $b \in L(A_U)$ where A_U is the universal automaton. The following state merge would be valid and consistent with D, however.



Let the quotient automaton resulting from this merge be B, the hypothesis that $L(B) = L_T$ is tempting as it is consistent with the positive and negative data examples D, and merging any of the other states results in an inconsistent automaton. But how can we be sure B is optimal for D, and can we be sure that the process of merging states could not have yielded a smaller consistent DFA? These questions, as we shall, see catapult us right back into the heart of philosophical and computational questions regarding the problems of Empiricism, the problem of inductive inference, optimal data compression, entropy and computational complexity.

2.2.2 Partitions and Complexity.

From a more general combinatorial point of view, ultimately all state merging algorithms at some point actually utilise a function for merging states together, reducing the size of the state space by $k \geq 1$ many states. Hence, given some automaton with n states, we can say without loss of generality, that there are a finite number of partitions of n, and hence, a finite number of quotient automata that can be constructed from the original representation of the data. Furthermore, the number of possible partitions decreases as $n \to 0$. Bell numbers, denoted B_n, where n is an integer greater or equal to 0, count the number of possible partitions on a set of n elements. Alternatively, since each parti-

tion denotes an equivalence relation and vice versa, Bell numbers count possible equivalence relations on a set. The first few Bell numbers are $1, 1, 2, 5, 15, 52$. Hence, for a set of 5 elements, like the state space for the PTA of the data set $\{aaa, ab\}$, there are 52 possible partitions, i.e. 52 possible quotient automata to check for optimality. Clearly Bell numbers grow exponentially quickly: as the size of the set increases, the number of possible ways to partition it grows exponentially in the size of the set. Hence any brute force algorithm which has to check all possible partitions for a set of size n will be in EXP. Bad news.

What this means from a computational complexity point of view, is that assessing all the possible models, the optimality of which we have yet to ascertain, will simply be too great for even the most powerful computers when we consider the asymptotic growth of Bell numbers. It is exactly the sheer number of possibilities in optimal model selection that makes DFA identification, and the related issue of DFA induction such hard problems. We may, nevertheless, try to restrict our attention to incremental merging of states, where the number of partitions that need to be checked are far fewer. This at first seems like it might do better than brute-force state-merging algorithms, because it means we can limit the search for optimal compressions of n-state automata to partitions of size $|(n-1)|$. In other words, automata generated by merging just two states. Mathematically, the number of these partitions can be given by the Stirling number $S(n, n-1) = \binom{n}{2} = \frac{1}{2}n(n-1)$, which grow asymptotically slower than the Bell numbers. The aim would be to incrementally compress the automaton, with the goal of approaching an optimal automaton. Unfortunately, however, since partitions form a partial order, there will be some quotient automata that are only reachable via a certain sequence of state merges that might be overlooked because the incremental compression algorithm cannot "see" ahead to the automata reached by all possible sequences of state merges.

Thus, in searching for optimal models, there is a concern that a poor choice of initial states for merging can lead to a model that could have been further optimised had the choice of initial states been different; incremental merging algorithms again require some sophisticated kind of guiding principle. Indeed, the requirement is necessary since, as mentioned above, searching through all possible partitions would require exponential computational time and space. What has become clear is this "guidance" when selecting states for possible merges, is not going to yield perfectly optimal automata, as in that case we would have already found the fast algorithm we are searching for.

However, algorithmic methods may be able to yield approximate optimal minimisation, perhaps retaining other desirable properties. If, for example, determinism is required, then controls are needed to prevent the algorithm yielding an NFA. Likewise, if one requires the derived automaton to be in some way minimal or parsimonious, then code-length minimisation can be the guiding heuristic. Previous investigations have led, along these lines, to the development of algorithms such as blue-fringe [27], which was one of the first to provide

reasonable guidance for the algorithm (using evidence) while merging states. For some other interesting approaches, see [28][29].

However, without surety about the sample, one cannot rule out the possibility that the derived model is a failure. Thus, we must investigate the conditions which permit learning to occur by focusing on fundamental principles of information and randomness.

3 The Relationship Between Entropy and Optimal Model Selection for DFA Induction.

Before making the role of entropy in optimal model selection precise, we first introduce the background notions: data compression, entropy, descriptive complexity and inductive inference. We then go on to show how these notions can be used to specify ways to select optimal models in the context of DFA induction.

3.1 A Brief Introduction to Data Compression: Kolmogorov Complexity and Shannon Entropy.

The introductory remarks of this dissertation make a lot of mention of "optimal codification" and "optimal models" for some data set. Thus we have introduced some notion of data compression. The current section will explain what we mean by compression in a mathematical sense. The main source of reference for this section is [30], but some of the definitions (3.1 - 3.7) are taken from lecture notes by Yfke Dulek and Christian Schaffner, see [31].

There are two main theories which attempt to provide a precise notion of optimal encoding and compression and both are relevant to the present work. The first, Shannon's Information Theory, tackles this notion by providing a definitive answer: the ultimate data compression–alternatively, the optimal codification for the data–is the probabilistic (Shannon) entropy, H. Informally, Shannon entropy is a measure of the average surprisal of a random variable. Or, put differently, it is the average number of bits required to describe a random variable using yes/no questions. However, the concern of Shannon was limited to what is essentially an engineering problem in the field of Communication Theory. Optimal codification, in terms of Shannon's theory, is fundamentally concerned with the ultimate transmission rate of communication, which Shannon worked out to be the channel capacity, C. Thus, optimal data compression is seen as a way of maximising the amount of information in a source to be sent over possibly noisy channels.

(Prefix-free) Kolmogorov Complexity, K, on the contrary, measures the optimal codification of objects in and of themselves, rather than the optimal codification of the data source. It is a descriptive complexity measure that is defined as the length of the shortest binary computer programme required for computing an object—viz. a string in a formal language—on a universal Turing machine that halts after printing. If no such input sequence exists the prefix Kolmogorov Complexity is undefined. Typically, one talks about prefix-free Kolmogorov Complexity rather than plain Kolmogorov complexity, since it is not always possible to determine where one string stops and another begins, without using prefix-free languages, where the programmes are assumed to be self-delimiting.

Both Shannon entropy and Kolmogorov complexity share common ground, in that K is approximately equal to H if the sequence is drawn at random from a distribution that has entropy H [30]. The current section will introduce both Shannon Information Theory and Kolmogorov Complexity more formally than has been done already, in order to provide background for the discussion on Grammar Induction. Later in the thesis we will apply concepts laid out here.

3.1.1 Information Theory: Shannon Entropy.

Definition 3.1 (Probabiliy Space) *A* probability triple $(\Omega, \mathcal{F}, P)$ *is a mathematical structure that models a random process. Also called a* probability space, *it consists of three elements:*

1. Ω is the sample space or set of all possible outcomes.
2. $\mathcal{F} \subseteq 2^{\Omega}$ is the event space, a subset of the power-set of the sample space. We can assume this to be a σ-algebra where an event is a set of outcomes in the sample space.
3. $P : \mathcal{F} \to [0, 1]$ is a probability function, which assigns each event in the event space a probability, which is a number between 0 and 1.

Definition 3.2 (Random Variable) *A* random variable *is a measurable function $X : \Omega \to E$, from a set of possible outcomes Ω to a measurable space E. For our purposes it will suffice to assume a real-valued function, $E = \mathbb{R}$.*

Shannon entropy is a measure of uncertainity contained in a random variable. This is often more intuitively explained by stating that entropy is the amount of yes/no questions one would have to ask, on average, to distinguish between the elements of X. Formally, it is captured as the average surprisal of a random variable, where the surprisal value for some probabilistic event, $\mathcal{A}$, that occurs with probability $P[\mathcal{A}]$, is $\log \frac{1}{P[\mathcal{A}]}$. The logarithm function captures the sense in which, the rarer the event is, the more surprised one would be for it to occur; small probabilities therefore yield high surprisal values and vice versa.

Definition 3.3 (Entropy) *Let X be a random variable with image $\mathcal{X}$. The Shannon entropy $H(X)$ of X is defined as:*

$$H(X) := \mathbb{E}\left[\log \frac{1}{P_X(x)}\right] = \sum_{x \in \mathcal{X}} P_X(x) \cdot \log \frac{1}{P_X(x)} = -\sum_{x \in X} P_X(x) \cdot \log P_X(x),$$

with the convention that the log *function represents the* binary *logarithm* $\log_2$. *And the second convention that, for $x \in \mathcal{X}$ with $P_X(x) = 0$, the corresponding argument in the summation is* 0.

Entropy, as shown by Shannon, arises as a bound for the (optimal) minimum code length of symbol codes.

Definition 3.4 (Minimal Code Length) *The minimal code length of a source P_X is defined as:*

$$\ell_{min}(P_X) := min_{C \in \mathfrak{C}} \ell_C(P_X),$$

where C is a code in some class of codes, $\mathfrak{C}$. And $\ell_C(P_X)$ is the average string length of codewords in C.

We give without proof the following the well-known theorem:

Theorem 3.5 (Shannon's Source-Coding Theorem) *The (optimal) minimum code length $\ell_{min}(P_X)$ for any source P_X is within the following bounds*:

$$H(X) \leq \ell_{min}(P_X) \leq H(X) + 1$$

We draw attention to this theorem simply to show how entropy relates, in the probabilistic setting, to compression of a data source P_X. The theorem tells us that in order to optimally encode any probabilistic data source, one requires on average an amount of bits that is within one bit of the entropy of the source $H(X)$. However, the proof relies heavily on Kraft's inequality and prefix-free codes. Exploring why this is so will shed light on the relationship between probabilistic entropy and optimal code length.

Definition 3.6 (Prefix-Free Code) *A binary symbol code $C : \mathcal{X} \to \{0,1\}^*$ is* prefix-free, *alternatively* instantaneous *, if for all $x, x' \in \mathcal{X}$ with $x \neq x'$, $C(x)$ is not a prefix of $C(x')$.*

We give without proof the following:

Theorem 3.7 (Kraft's Inequality) *There exists a prefix-free code with image $\mathcal{C} = \{c_1, \ldots, c_m\}$ and codeword lengths $\ell_i := \ell(c_i)$ if and only if*

$$\sum_{i=1}^{m} 2^{-\ell_i} \leq 1.$$

The importance of this inequality cannot be overstated. It turns up in many of the contexts we are dealing with. Essentially, it shows us that the length of an optimal prefix-free code coincides with the length of an optimal uniquely decodeable code, where the latter is a binary symbol code $C : \mathcal{X} \to \{0,1\}^*$ where C^* is also injective. Indeed, the same inequality holds for such codes; this variant for uniquely decodeable codes is called McMillian's inequality [32]. Thus for every uniquely decodable code, there exists a prefix code with the

same length distribution. Moreover, these lengths are essentially given by the Shannon entropy.

Due to the restriction that the sum of the fractional lengths of valid codewords is less than or equal to one, the Kraft-McMillian inequality behaves a lot like a probability mass function. Indeed, one can think of the inequality as specifying a resource to be divided up between codewords for symbols; the shorter the codeword the more expensive it is [33]. Mathematically speaking, Kraft's inequality is a semi-measure.

Definition 3.8 (Semi-measure) *A semi-measure on a measure space* (X, σ)*, (where* σ *is a* σ*-algebra), is a function* $\lambda : X \to [0, 1]$ *satisfying:*

1. $\lambda(\sigma) \leq 1$,
2. $\lambda(\bigcup_{i \in I} A_i) \geq \sum_{i \in I} \lambda(A_i)$ for any collection $\{A_i\}_{i \in I}$ of pairwise disjoint elements of X.

That is to say, that unlike a (plain) measure, the full event σ need have a measure of *at most* 1, rather than *exactly* 1. And the union of disjoint events may have a larger measure than the sum of the parts, hence semi-measures are not necessarily additive.

These features of the Kraft-McMillian inequality are essentially what relates the notion of optimal compression to entropy. Indeed, since the probability of a rare event occurring is low, the codewords we use to describe them ought to be longer. This way we save the shortest "expensive" codewords for the most common symbols. As mentioned above, the notion of entropy as the average surprisal of an event succeeds to quantify the information in random variables. We now see that the achievable optimal compression length of codes is essentially the Shannon entropy (with 1-bit of overhead), precisely because the *a priori* amount of information contained in a random variable is given by the entropy. Under the lens of coding theory, if codeword lengths are distributed according to the probability of their occurrence, we capture that information optimally.

Implicitly, we have married together two notions here; that the simplicity of a process is somehow related to the likelihood distribution of its symbols; more common symbols should be given the simplest codes. Thus, a process with uniformly distributed likelihood over its symbols, will appear to give rise to more random or complex code. This is because, intuitively, there is a difference between an ordered string—one that we can see patterns in—and the vast majority of strings, which seem random. In terms of code, if nearly all the probability density is over one symbol, the code will appear very simple, only consisting of a single simple codeword for the most part. The difference we are drawing attention to is this: it would seem natural that an ordered string was generated

by some simple deterministic process, algorithmically speaking, whereas a random string is more likely generated by a complex, possibly non-deterministic or random, sequence. Indeed, it seems that the *most likely* explanation would be that an ordered string of data came from a simpler process than a complex one. For example, it would appear highly unlikely to our intuitions that a sequence of ten trials of a fair die landed on a six each time; we would expect that a uniformly random die would produce sequences of outcomes that appear far more random than this example. The simpler explanation—that the sequence comes from a biased die, that has been weighted to land on a six—appears likelier.

These insights eventually led Solomonoff [34] to the formal definition of a universal *a priori* notion of probability, which he called the Algorithmic "Solomonoff" Probability (AP). This assigns to objects an *a priori* probability that is universal in the sense that it is defined with respect to universal Turing machines, the properties of which are true irrespective of the particular universe we happen to inhabit. Since the AP is essentially a prior (probability) distribution it has clear theoretical applications in many areas. In particular in the field of inductive inference theory. See [35] for further in depth discussion: we repeat what is shown there, that Solomonoff's prior probability is in fact also a semi-measure.

Definition 3.9 (Algorithmic Probability) *Let a computable process that produces a string x be defined as a programme p that when executed on a universal Turing machine $\mathcal{U}$ produces the string x as output. Note p is itself a binary string. We define the discrete universal* a priori *probability, $m(x)$, to be the probability that the output of a universal prefix Turing machine $\mathcal{U}$ is x.*

$$m(x) := \sum_{p:\ \mathcal{U}(p)=x} 2^{-\ell(p)},$$

where the sum is over all halting programmes p for which $\mathcal{U}$ outputs the string x.

AP is a semi-measure: since U is a prefix universal Turing machine, we sum over a prefix-free set, viz. the set of valid programmes forms a prefix-free set. Hence, by the Kraft-McMillian inequality, we have that $\sum_{p:\ \mathcal{U}(p)=x} 2^{-\ell(p)} \leq 1$.

With this insight we are nearly in a position to describe inductive inference problems from a sufficiently general point of view, but first we introduce an inherently related concept: it was shown in Levin (1974) [36], that the AP was related to Kolmogorov Complexity: $-\log m(x) = K(x) + O(1)$, where K is the prefix-free Kolmogorov complexity. As mentioned above, K is a measure of the length of the shortest binary programme for computing some string with a UTM that halts after printing. We now introduce this deeper notion (algorithmic entropy) more formally, and explain how the concept helps us in providing an analysis of data compression, and the relationship of entropy and data compression to inductive inference problems, which is the main concern of this thesis. As it turns out, Kolmogorov Complexity is in fact equivalent

to Solomonoff's algorithmic probability. In this sense, the fundamental limit of data compression—an object's descriptive complexity—is equivalent to some notion of universal likelihood.

3.1.2 Information Theory: Kolmogorov Complexity.

Recalling the definition of Shannon entropy, an object X is a random variable drawn with respect to a probability distribution P_X. The descriptive complexity of the event $X = x$ is given by Shannon's source coding theorem; we require at most $\lceil \log \frac{1}{P_X(x)} \rceil$ bits of Shannon code to describe such an event, provided X is random. In this sense, the individual amount of information in an object x is given by $\log \frac{1}{P_X(x)}$, where the weighted average for $x \in \mathcal{X}$ yields $H(X)$. Thus, the descriptive complexity of an object depends on a probability distribution, and is related to the entropy of the distribution. Kolmogorov decouples the notion of descriptive complexity from probability distributions and instead describes it algorithmically: the descriptive complexity of an object is the length of the shortest binary programme that describes the object. The shortest computer code will be uniformly good for any probability distribution, and in this sense will act as a universal code [30]. It is for this reason that Kolmogorov complexity is considered more fundamental than the Shannon entropy, since the expected length of the shortest binary programme for a random variable is approximately equal to its entropy [37].

Definition 3.10 (Kolmogorov Complexity) *The* Kolmogorov Complexity $K_{\mathcal{U}}(x)$ *of a string x with respect to a universal prefix-free Turing machine $\mathcal{U}$ is defined as*

$$K_{\mathcal{U}}(x) := min_{p:\ \mathcal{U}(p)=x}\ \ell(p),$$

which is the minimum length over all programmes that print x and halt.

Note that the requirement here that p be a halting programme means that Kolmogorov complexity—and also Solomonoff probability for that matter—is uncomputable. The reason for this is due to the undecidability of the question of whether an arbitrary programme will halt or not on a given input. Indeed, Turing (1937) showed that what has come to be known as the halting problem is undecidable over Turing machines [38]. The halting set made of programmes p such that p halts on input x, is thereby uncomputable by any Turing machine. Nevertheless, Kolmogorov complexity can be approximated; in practical situations it is enough to find a very short, but not necessarily minimal algorithm as a guide to the Kolmogorov complexity of the problem. Moreover, Kolmogorov complexity provides us with the apparatus to think about universal codes and inductive inference. In fact, Kolmogorov complexity explicitly formulates the notion that the simplest explanation is best. That is essentially the principle of Occam's Razor, well known to Philosophy.

By way of example, the Kolmogorov complexity of the 100-bit string $x = 0\overbrace{\ldots}^{\times 100}0$ is at least a lot smaller than 100 bits, as the programme: $p_1 = PRINT(0 \times 100)$ is smaller than the programme $p_2 = PRINT(x)$. This is because the sequence is intuitively not at all random. Hence, it has lower descriptive complexity. In other words, it has low algorithmic entropy; the most likely explanation is that this sequence was generated by a simple process. Contrast this sequence with the binary representation of the golden ratio, φ. Here we see the descriptive complexity seems to increase somewhat. Indeed, φ is the most irrational of the irrational numbers. Yet this process is still simple; we already know of a shorter programme (for computing the decimal expansion) which outputs the string, namely the arithmetic algorithm $\frac{1+\sqrt{5}}{2}$. In a truly random sequence, however, each of the 2^n possible sequences of n-bits is equally probable. In this case, the descriptive complexity is the same length as the string itself, as we can do no amount of compression to the string, seeing as no shorter description exists.

Definition 3.11 (Conditional Kolmogorov Complexity) *The* conditional prefix Kolmogorov complexity *of x given y is*

$$K(x|y) = min_p\{\ell(p) \mid \mathcal{U}(\langle p, y\rangle) = x, p \in \{0,1\}^*\}$$

Note: by this definition, we can also understand the non-conditional prefix Kolmogorov complexity to be $K(x) = K(x|\epsilon)$ since $\mathcal{U}(\langle p, \epsilon\rangle) = \mathcal{U}(p)$, that is to say that it is equal to the conditional Kolmogorov complexity with empty input.

We are now in a position to address the role of data compression, algorithmic probability/Kolmogorov Complexity and probabilistic entropy in the theory of inductive inference and optimal model selection. From this sufficiently general vantage point we can then assess particular compression techniques in the context of finite automata. However, first we present the final piece of the puzzle.

3.2 Inductive Inference.

As mentioned in the introduction, the fundamental problem of Empiricism is that it is at root a philosophy which disparages sources of knowledge beyond experience; conjectures, or speculations are considered not well founded, whereas innate ideas, and abstract deductions from what can be garnered *a priori*, are thought to be somewhat less interesting, or trivial, as opposed to that which can be discovered through observation and experiment. This poses something of a self-contradictory paradox within empiricist philosophy, since the conclusions one makes about observations and experiments cannot themselves be founded in experience, at least *prima facie*.

From a naive point of view, Empiricism must posit some extra apparatus in order to explain why theories are true, and in what way this truth is assured by the observations and data that support said theories. Otherwise, the problem

of inductive inference would seem to falsify empirical knowledge. Exactly what is it that allows one to make an inductive generalisation about observations? Answering this question satisfactorily would lead us far into debates in the Philosophy of Science, which for space reasons we cannot attempt. However, what can be said briefly is that the pragmatic notion to focus on heuristics and other criteria (such as confidence) for hypothesis testing, circumvents the actual need for theories to be *'true'*. If we specify some theory or model which we believe to be true, and test it against the observed data, then we can learn from a new observation and update our beliefs in such a way that is consistent with our observations. Probability Theory itself focuses on finding approximate answers. Indeed, Empiricism can avoid many of the philosophical pitfalls by accepting that the kind of knowledge attained by finitely experiencing beings is, by its nature, approximate. Given enough data, the possibility of a false inductive inference shrinks to an inconsequential number.

One such method for updating one's beliefs is Bayes' Law [39], which states that given two events A and B, the conditional probability of A given that B is known, is proportional to the conditional probability of B given A:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Under the frequentist interpretation of probability, this rule is perhaps hard to parse, as the conditional probability $P(A|B)$ is usually defined as $\frac{P(A \cap B)}{P(B)}$, where $P(A \cap B)$ is the probability that both events A and B occur. One can interpret this as a restriction of the sample space to events in which B occurs. However, when probabilities are interpreted in the Bayesian sense—that is to say, where we interpret probabilities as reflecting the (subjective) degree of belief with respect to events, rather than the relative frequency of an event—Bayes' Law can be seen to constitute a learning algorithm.

Thus we interpret $P(M)$ as the initial probability of some theory, i.e. our initial degree of belief in the theory prior to observing any evidence for the theory. Whereas, $P(M|D)$ is the probability of the theory given the observation of some data, which is the probability of M after accounting for the evidence D. Thus, the conditional probability $P(M|D)$ is the posterior probability of the theory after being updated with evidence. Finally, the probability $P(D)$ is the universal probability of the data occurring, that is, the probability of seeing D given all possible models. One can derive a measure of the amount of information in the theory that explains the data, by simply computing the difference $P(M) - P(M|D)$, a formalisation of the difference between the prior probability and the posterior probability.

The reader may have already noticed the similarity in this kind of reasoning to the motivations underlying Solomonoff's algorithmic probability, and equivalently, Kolmogorov's notion of algorithmic complexity. Indeed, both Bayes' Law and Kolmogorov complexity seem to formalise, or provide a basis, for the the-

ory of inductive inference. However, this connection is deeper than one might imagine. The first thing to recall is that the amount of information required to encode an event according to Shannon's probabilistic theory of information is given by the log function. Indeed, it is worth noting that in a very real sense, the logarithmic function is the only suitable measure of information since it is the only function satisfying continuity and additivity for independent events. Hence, we arrive at the following definition for quantifying information:

Definition 3.12 (Shannon Information Content) *The amount of information in the event $X = x$ for some random variable X with probability distribution P_X is*

$$I_S(x) = \log \frac{1}{P_X(x)} = -\log P_X(x)$$

Definition 3.13 (Conditional Shannon Information Content) *The amount of information in the event $X = x$ given that $Y = y$, for random variables X, Y with probability distributions P_X, P_Y is*

$$I_S(x|y) = \log \frac{1}{P_{X|Y}(x|y)} = -\log P_{X|Y}(x|y)$$

This immediately allows us to precisely quantify the amount of Shannon information gained from a Bayesian update as $-\log P(M) + \log P(M|D)$.

However, we know from Levin (1974) that Solomonoff's universal probability is related to Kolmogorov complexity by the equation $-\log m(x) = K(x) + O(1)$. Thus replacing the information-theoretic description of Bayes' Law for an unspecificed probability measure, with Solomonoff's universal probability yields the following equation for the algorithmic probability version of Bayes' Law:

$$-\log m(x) + \log m(x|y) = K(x) - K(x|y) + O(1).$$

Intriguingly this is exactly the definition of mutual algorithmic information.

Definition 3.14 (Kolmogorov's Mutual Algorithmic Information) *The information in y about x is defined as*

$$I(y : x) = K(x) - K(x|y^*)$$

where y^* denotes the first (in a standard enumeration order) shortest prefix programme that generates y and then halts [37][40].

Thus, we have arrived at the result that Bayes' Law is actually a very foundational notion indeed; it is equivalent to Kolmogorov's algorithmic mutual information (see [40] for further discussion). Moreover, this connection relates both Kolmogorov complexity (algorithmic entropy) and thereby compression to

what is essentially an axiomatic foundation of inductive inference, *ergo* also to learning. Indeed, since Bayes' Law gives a kind of algorithm for updating one's beliefs, it ends up relating algorithmic notions of learning with algorithmic data compression (entropy).

This ultimately relates back to Empiricism in the following sense: if we cannot ever find a logically valid reason to infer a conclusion inductively, we can still find an algorithm with a sensible heuristic for making inductive inferences. In fact, we have already mentioned one such heuristic, namely: Occam's Razor. As we have already pointed out the connection of Kolmogorov complexity and this heuristic, the next natural question is how well can we formulate the exact notion of Occam's Razor in this information-theoretic context? The answer is, using the Minimum Descriptive Length (MDL) principle, we can formulate Occam's Razor rather precisely.

Definition 3.15 (The Minimum Descriptive Length Principle) *The best theory to explain the data is the one which minimises the sum of the length in bits of the description of the theory, and the length in bits of the data when encoded according to the theory* [22].

Clearly, therefore, the MDL principle is related to Occam's Razor, which puts forward the idea that the simplest explanation, among equivalent descriptions is preferable to the more complicated explanations. For Occam, this meant that theories which posit fewer assumptions are 'better'; that is to say, that the theory ought to be parsimonious. MDL, by contrast, chooses the simplest description among various explanations of the data. However, the justification that the simpler the description, the more likely it is to be responsible for the data we see, is backed up by Kolmogorov Complexity and the notion of algorithmic probability.

Indeed, the MDL principle is equivalent to the Maximum *A Posteriori Hypothesis* (MAP), which is itself an *argmax* function over models, with Bayes' Law as input. Thus, Occam's Razor, Bayes' Law, algorithmic proability and Kolmogorov complexity all come neatly together in the notion of MDL as two-part code optimisation. Ultimately, we have improved upon the position from which we started this section; Shannon code offers a one-part code optimisation, with MDL we can select not just the shortest descriptions but also take into account the data encoded with the help of the theory.

We now provide the derivation of the equivalence between MAP and MDL [22][41].

Let $M \in \mathcal{M}$ be a model in a class of models, and let D be a data set. The prior probability of a model or hypothesis is $P(M)$. This is the likelihood we ascribe to the model before any observations have been made. $P(D)$ is the probability of the data, defined with respect to all models which are consistent with the data.

The posterior probability is $P(M|D)$, that is the probability we ascribe to the model given the data we have observed. The maximum *a posteriori* probability is the *argmax* function of this probability.

$$
\begin{aligned}
M_{MAP} &\equiv argmax_{M\in\mathcal{M}} P(M|D) && (1)\\
&= argmax_{M\in\mathcal{M}} \left(\frac{P(D|M)\cdot P(M)}{P(D)} \right) && (2)\\
&\equiv argmax_{M\in\mathcal{M}} \Big(P(D|M)\cdot P(M) \Big) && (3)\\
&\equiv argmax_{M\in\mathcal{M}} \log \Big(P(D|M)\cdot P(M) \Big) && (4)\\
&\equiv argmax_{M\in\mathcal{M}} \Big(\log P(D|M) + \log P(M) \Big) && (5)\\
&\equiv argmin_{M\in\mathcal{M}} \Big(-\log P(M) - \log P(D|M) \Big) && (6)
\end{aligned}
$$

(2) Follows by Bayes' Law. Since D is constant over $M \in \mathcal{M}$ we get (3). However, since log is always increasing, we can take the logarithm of argmax without loss of generality (4). Finally by the laws of logarithms, and argmax/argmin we get (5) and (6). In terms of Shannon code, the derived MAP says that the minimal (optimal) amount of bits required to encode the model is $-\log P(M)$, and the optimal length in bits to encode the model given the data is $-\log P(D|M)$. We call the former, the *model code*, and the latter the *data-to-model code*. The MAP we have derived to be the minimum $M \in \mathcal{M}$ of the summed lengths of the model and data-to-model code. But this is exactly the definition of MDL. Hence $M_{MDL} \equiv M_{MAP}$.

Of course, the argmax equation is reliant on the class of models, which we have yet to specify. If we were to specify $\mathcal{M}$ as consisting of an enumeration of all self-delimiting programmes for a preselected arbitrary universal Turing machine $\mathcal{U}$, then we would be dealing with one of the most general classes. Under this interpretation of $\mathcal{M}$, the length of the optimal two-part code is given by the Kolmogorov complexity [22]. We get via Levin's coding theorem:

$$
\begin{aligned}
M_{MDL} &= argmin_{M\in\mathcal{M}} \Big(-\log P(M) - \log P(D|M) \Big)\\
&= argmin_{M\in\mathcal{M}} \Big(K(M) + K(D|M) \Big)
\end{aligned}
$$

In our context, however, we seek to deal with finite automata. Hence $M \in \mathcal{M}$ will specify an automaton in the class of all deterministic finite automata. From now on, we will interpret $\mathcal{M}$ in this sense. Seeing as, in this context, programmes are in general not self-delimiting, it will not be appropriate to concatenate programmes at will. Thus in our case the Shannon code length seems the most appropriate measure of MDL. We merely state the relation to Kolmogorov

complexity for the reader to appreciate the sense in which the work on finite automata identification that is to come, should yet be understood in terms of data compression, algorithmic probability and inductive inference.

Keeping in mind what we have seen about compression, optimal code length, heuristics such as Occam's Razor and MDL, the notion of algorithmic learning in the form of Bayes' Law and Kolmogorov mutual algorithmic information, we now apply this background of theoretical knowledge to the particular context of finite automata induction, in particular grammar induction for finite automata. The goal in the rest of the thesis will be to probe the following questions:

1. Is there a precise relationship between entropy and optimal model selection (in the context of DFA induction)? See section 3.3.
2. Is it possible, in the context of finite automata to find multiple optimal models for the same data set—possibly giving the data a gestalt quality? See section 4.
3. Can bisimulations and other behavioural equivalences aid us in selecting optimal models, and what is the relation of (bi)simulation to entropy and optimal model selection? See sections 6.1.1, 6.2.2 and appendix.

We start with the first of these.

3.3 Learning DFA Using Entropy.

3.3.1 MDL for Grammar Induction.

We have made a point to show how the optimal length of codes are related to learning heuristics; the marriage of which yields the principle of minimum description length. We also mentioned in the preliminaries that state-merging algorithms need some form of guiding principle in order to get around issues of computational complexity for average-case DFA learning. MDL as a guiding principle has had varying success in the field of DFA induction. However, in probing the relationship between entropy and optimal model selection, it is the natural place to start.

The general idea is to measure the descriptive complexity of some automaton (the size of the state-space, number of transitions, size of the alphabet) along with the descriptive complexity of encoding a set of sample words using the automaton as a model. It will not always be the case that the smallest automaton is recognized as the most promising, since the complexity of the process of parsing strings in the sample is also quantified.

Following [42], let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA all of whose non-final states have outgoing edges. The number of bits required to encode the path traversed in order to parse a word w can be assessed by the function ch given below. We associate with each state $q \in Q$ the value $t_q = \sum_{a \in \Sigma} |\delta(q, a)|$ if $q \notin F$. Whereas, if $q \in F$ then $t_q = 1 + \sum_{a \in \Sigma} |\delta(q, a)|$, since one more choice is available, namely the choice to accept. We are now in a position to define $ch(q, w)$. For the empty word we have $ch(q, \epsilon) = \log(t_q)$ if $q \in F$; otherwise $ch(q, \epsilon) = \infty$. For $w = au (a \in \Sigma, u \in \Sigma^*)$, the function depends on the recursive definition: $ch(q, w) = \log(t_q) + min_{r \in \delta(q,a)} ch(r, u)$ and $c(q, w) = \infty$ if $\delta(q, a) = \emptyset$. Given a sample D_+ and a DFA A, we can now measure the MDL score sc of A:

$$sc(A, D_+) = |Q| + ||\delta||(2 \log |Q| + \log |\Sigma|) + \sum_{w \in D_+} ch(q_0, w)$$

where $||\delta||$ is the number of transitions of A. Note, see Wieczorek (2016) for details on this choice of MDL two-part code optimisation for DFA induction.

Recall that in our example from the introduction $D_+ = \{aaa, ab\}$. Note that for the PTA we constructed, $t_{q_0} = t_{q_2} = t_{q_3} = t_{q_4} = 1$, whereas $t_{q_1} = 2$. Then the MDL score for the PTA is $sc(PTA, D_+) = 5 + 4(2 \log 5 + \log 2) + 2 \log 2 = 29.58$. For the first compression $sc(L_1(DFA), D_+) = 4 + 4(2 \log 4 + \log 2) + 2 \log 2 = 26$. For the NFA we have $sc(L_2(NFA), D_+) = 3 + 4(2 \log 3 + \log 2) + 3 \log 3 \approx 24.4$. Whereas for the B automaton, consistent with the data set of both positive and negative data, we have $sc(B, D) = 3 + 4(2 \log 3 + \log 2) + 4 \log 2 + 2 \log 3 \approx 26.85$. Thus from the MDL perspective we ought to prefer the NFA, even if this automaton is not consistent with the negative data examples.

Since the MDL principle works like a form of Occam's Razor, we should prefer the parsimonious theory; the one which takes the fewest bits to describe. But this may not be an entirely satisfying answer. If we are certain that the sample data are fair, then it may be possible to rule out some of these models as possible generators of the data. But if we do not have this assurance, we are left with possible (likely) guesses only. The patterns we think we recognise might be simply noise. In the cases where we are not sure of the fairness of the data, polysemanticism seems to pose a big problem. We will deal with the particular issue of polysemanticism in section 4. First we need to be clear about the kind of data that might constitute a *fair* sample.

3.3.2 Fair Samples from Simple Random Walks on DFA.

Throughout this thesis we have not yet made an assumption about how the data examples were sampled. However, this question has a bearing on the nature of the learning problem. We want to limit abductive "guess-work" to the best of our abilities, in order to lessen the chance that human bias creeps into model selection. One way of going about this would be to assume the principle of

maximum entropy. The principle essentially states that the probability distribution which best represents the current state of knowledge is the one with the largest entropy [43]. In this sense we are maximally uncertain about the data. And because of this, the principle assumes the least number of assumptions. Moreover, it can be seen as another application of Occam's Razor.

In the context of our DFA induction problem, the principle of maximum entropy applies to the sampling process which grants us the positive and negative data examples. To achieve maximum entropy, one may assume that the data were generated by a random walk (stochastic process) on the DFA. The random walk would start at the initial state of the unseen DFA and begin to move around the digraph. To ensure maximum uncertainty about the language defined by this unknown DFA, let the process evolve to the next state by selecting uniformly at random from the possible transitions.

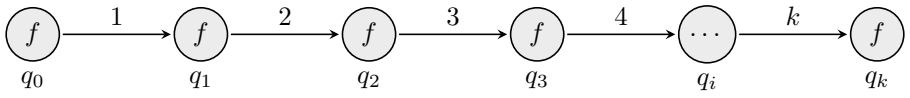
To define a simple random walk on a DFA, we need to pay attention to the underlying digraph of the DFA. For $M = (Q, \Sigma, \delta, q_0, F)$ we can define a directed graph $G = (V, E)$ where $Q = V$ and $\langle q_i, q_j \rangle \in E \iff q_j \in \delta(q_i, a)$ for some $a \in \Sigma$. From this we can study properties of the underlying graph. For instance, we may wish to consider whether the graph is reducible or periodic. We can now state the random walk formally:

Definition 3.16 (Simple random walk) *A discrete-time* simple random walk *on a graph is a sequence of random variables $X_i = X_1, X_2, \ldots$ with $i \in \mathbb{N}$. If the walk is of length k on a finite graph G with a root q_0, the* random walk *is a stochastic process with random variables $X_1, X_2, \ldots, X_k$ such that $X_1 = q_0$ and X_{i+1} is a vertex chosen uniformly at random from the neighbors of X_i.*

We can use this definition of a simple random walk to think about how the sample data might be generated under the principle of maximum entropy. We want to ensure the sample is truly indicative of the target model.

Definition 3.17 (Basic fair sample) *A* basic fair sample *of a language generated by an unknown DFA M is the set of words for which each possible sub path in the DFA has been visited at least once by the m random walks on the underlying digraph of M.*

It is clear from this definition that even for finite languages it could take a number of runs exponential in the number of states to generate such a sample. To see this consider the following linear automaton, M:



Note that $|\Sigma| = k$ and $|L(M)| = k + 1$ since every state is accepting. A random

walk on the underlying digraph of M has two options: halt or transition to the next state. Thus, taking a basic fair sample of this automaton under maximum entropy could require 2^k random walks, since the probability that a word of length n is visited is 2^{-n} [44]. Despite this fact, it seems that given a basic fair sample of this automaton, it would be in principle possible to recover the structure. However, recalling what was mentioned in the introduction: it is NP-hard to approximate a target DFA of size OPT by a DFA of size within OPT^k for any constant factor k unless P=NP [5]. Thus even with a basic fair sample there may be automata that are unlearnable under maxium entropy assumptions. For further discussion of these insights (for context-free languages) see [45].

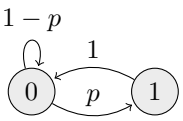
3.3.3 Maximal Entropy Random Walks and Intrinsic Markov Shifts.

In the previous subsection we dealt with simple random walks which evolve with uniform probabilities over choices. However, we will now demonstrate that this, in general, does not maximise the entropy rate of a stochastic process. For that we require a uniform distribution over path lengths of the same length. In this sense one captures the intrinsic characteristics of the underlying process, and thereby its absolute entropy. We investigate this topological notion of entropy as a way to demonstrate that a simple random walk strategy is not the best way to generate a fair sample. Intuitively, in order to generate fair samples, transitions to more topologically complex parts of the automaton should have higher probabilities. But this will not, in general, maximise the entropy of random walk, viz. random walks cannot generate fair samples without breaking the principle of maximum entropy.

For example, take the following stationary stochastic process, representable as a two-state Markov chain with transition matrix:

$$A = \begin{bmatrix} 1-p & p \\ 1 & 0 \end{bmatrix}$$

This matrix yields the following two-state graph:



We can describe a discrete stochastic process $V_1, V_2 \ldots$, as a path moving through the graph, where V_i describes the state of the process at a given time step. Thus, a typical sequence may look like the binary string 0010101, where

the digits refer to the sequence of states visited by the process. We can talk about the probability of reaching the next state, given the current state by analysing the probability matrix. For example, $P_{V_{n+1}|V_n}(0|1)$ is the probability that the next state in the sequence will be 0, given that $V_n = 1$. In this case, we can see by the matrix and by the graph, that $P_{V_{n+1}|V_n}(0|1) = 1$.

Clearly we have

$$P_{V_{n+1}|V_n}(i|j) = P_{V_2|V_1}(i|j) \quad \forall i, j \in \{0, 1\}$$

Hence we can conclude that the stochastic process is a time-invariant Markov process. If this Markov process were evolving under a maximum entropy distribution over choices, the transition probability p would be 0.5. However, this is in fact not the probability which maximises the entropy rate of this process. We give without proof the following:

Proof 3.18 *The process defined by the transition matrix A has entropy rate equal to $\log_2 \varphi$ where $\varphi = \frac{1+\sqrt{5}}{2}$ is the golden ratio* . See [46].

To see why this is so, we have to analyse the topological properties of the Markov chain given by A, divorced from probabilities. In other words, we must consider each sequence of the same length to be equally likely. In this sense we capture the intrinsic (topological) entropy of the system.

Definition 3.19 (Language) *The collection $\mathcal{L}(X) = \{w \in X \mid \ell(w) \neq \infty\}$ where $\ell(w)$ is the length of the string w, is called the* language *of X.*

Definition 3.20 (Word Complexity) *The function $p : \mathbb{N} \to \mathbb{N}$ defined by*

$$p(n) = \#\{w \in \mathcal{L}(X) \mid \ell(w) = n\}$$

where $\ell(w)$ is the length of the string w, is called the word-complexity *of X.*

Definition 3.21 (Topological Entropy) *The* topological entropy *of a subshift is*

$$h_{top}(X, \sigma) = \lim \sup_{n\to\infty} \frac{1}{n} \log p(n)$$

The previous three definitions are found in [47].

Let X be the set of allowable sequences of the Markov process defined by transition matrix A. Intuitively, the process produces all possible sequences of 0's and 1's that do not contain consecutive 1's for different sequence lengths n. The infinite set X, which contains all these possible sequences is a subshift of finite type, known as the *golden mean shift*. As we can see from the table below, to produce sequences of length n, one can construct $p(n-1)$ sequences by adding 0 in front of every $(n-1)$-long sequence because 0 can precede 0 or 1. Another $p(n-2)$ sequences of length n are produced by putting 1 in front of all

n	possible sequences	$p(n)$
1	0, 1	2
2	00, 01, 10	3
3	000,001,010,100,101	5
4	0000,0001,0010,0100,0101,1000,1001,1010	8
...	...	...

$(n-1)$-long sequences that start with 0, since 1 can only occur before 0's, and we have by definition $p(n-2)$ sequences of length $(n-1)$ that start with an 0. Therefore, the linear recurrence that we were looking for is

$$p(n) = p(n-1) + p(n-2)$$

This recurrence relation defines the Fibonacci numbers. To calculate the topological entropy of the subshift we make use of its closed form expression.

$$\begin{aligned}
h_{top}(X,\sigma) &= \lim \sup_{n\to\infty} \frac{1}{n} \log p(n) \\
&= \lim \sup_{n\to\infty} \frac{1}{n} \log \left(\frac{\varphi^n - (1-\varphi)^n}{\sqrt{5}} \right) \\
&= \lim \sup_{n\to\infty} \frac{1}{n} \big(\log(\varphi^n - (1-\varphi)^n) - \log \sqrt{5} \big) \\
&= \lim \sup_{n\to\infty} \frac{1}{n} \big(\log(\varphi^n - (1-\varphi)^n) - \underbrace{\lim \sup_{n\to\infty} \frac{1}{n} \log \sqrt{5}}_{=0} \\
&= \lim \sup_{n\to\infty} \frac{1}{n} \left(\log \left(\varphi^n \left(1 - \frac{(1-\varphi)^n}{\varphi^n} \right) \right) \right) \\
&= \lim \sup_{n\to\infty} \frac{1}{n} \left(\log \varphi^n + \log \left((1 - \frac{(1-\varphi)^n}{\varphi^n} \right) \right) \\
&= \lim \sup_{n\to\infty} \frac{1}{n} (\log \varphi^n) + \underbrace{\lim \sup_{n\to\infty} \frac{1}{n} \left(\log \left((1 - \frac{(1-\varphi)^n}{\varphi^n} \right) \right)}_{=0} \\
&= \lim \sup_{n\to\infty} (\log \varphi) \\
&= \log \varphi \\
&= \log \left(\frac{1+\sqrt{5}}{2} \right) \\
&= 0.6942
\end{aligned}$$

As we can see, the absolute topological entropy is equal to the maximum entropy rate for the stationary stochastic process defined by A. This is because maximal entropy random walks are essentially identical to topological shifts, in that equal probability is given to sequences of the same length. The uniform distribution

over choices, which would yield a probability $p = 0.5$ would lower the chance of choosing a '0' after a '0' thereby increasing the chance of reading a '1' for which there is no information gain. Hence, using simple random walks to sample this process will not be a good strategy for recovering the language.

Thus a significantly different picture emerges when we consider the global uniform distributions over paths of the same length, as opposed to the local notion of maximum entropy over choices. For the problematic linear automaton from the previous section, there is only one path of length n, for each $n \leq k$. Thus $p(n)$ is a constant and does not grow. In trying to compute the (absolute) topological entropy for this automaton we get $\limsup_{n\to\infty} \frac{1}{n}(\log 1) = 0$. In other words, there is no topological entropy (or complexity) in this system. Having zero topological entropy would make sense from a dynamical systems perspective since we are always certain where every walker will end up: there is only ever one transition to the next state, so the walker will end up deadlocked in the final state on all trials. However, on the contrary, this analysis shows that a fair sample generated from a number of random walks on the linear automaton is not going to tell us anything about its topological structure, unless the sample is a complete picture of the finite language itself. In other words, we need to be given every accepting string in order to learn the language as there is no way to tell on the basis of (e.g.) $w \in L$ and $wav \in L$ (for finite words w, v), that $wa \in L$.

Ultimately it seems that a fair sample, though seemingly required in order to be assured that learning will take place, occasionally may require the sample to be identical to the language one seeks to learn. That is to say, that sometimes in order to contain the necessary structural information of the automaton, an extremely large sample (possibly containing the entire language) is needed. It would appear that attention to the distribution of the data examples in this way has not lessened the difficulty of the problem significantly. However, idealised situations may be amenable to learning with data sampled under maximum entropy assumptions. Some interesting avenues for future research were developed during the writing of this thesis, for those see §6.2.

In the next section we consider the problem of polysemanticism in optimal model selection for DFA induction. We first examine the situation of polysemanticism occurring without fair data and go on to explore the possibility of fair data aiding the solution of the problem.

4 Learning Without Fair Data.

We now turn to question (2), posed at the end of section 3.2, namely: whether in the context of finite automata, it is possible to find multiple optimal models for the same data set. This question turns out to be inherently related to the amount of entropy in the model and in the data. Indeed, this phenomenon occurs when one tries to decide between models based on some principle of parsimony. We here utilise the minimal description length principle, introduced in section 3.3.1.

4.1 Polysemanticism.

It was mentioned in the introduction that, for the class of partial and total recursive functions, there can be data sets for which there are multiple optimal models [Adriaans (2020)]. That is, data sets which under the MDL perspective of two-part code optimisation give rise to two or more distinct optimal hypotheses for the data. The concept is essentially the computational version of the "gestalt effect". Adriaans calls the phenomenon polysemy; data which exhibit polysemy are called polysemantic. Since the class of partial and total recursive functions are type 0 languages in the Chomsky hierarchy, it would be interesting to see if the same or similar result would hold for the regular languages. Since regular languages are far simpler, they can in some way be seen as more fundamental. Indeed, regular languages have only a constant amount of memory. Thus, if one were to discover polysemantic data sets for regular languages, it would imply a further barrier against learning in some of the simplest contexts. That is to say, even in rudimentary forms of computations, the correct interpretation of data can possibly be indeterminable.

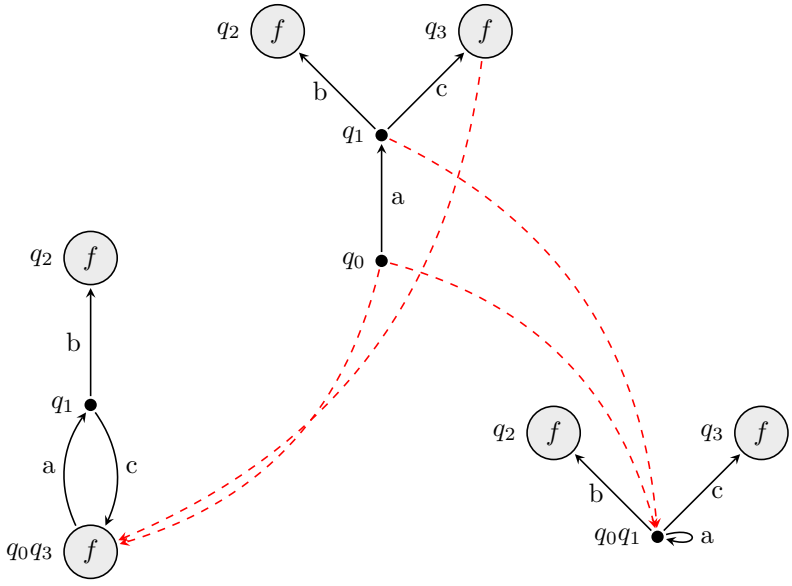
Such data sets will be compressible in mutually exclusive ways, much like the famous duck-rabbit image of Wittgenstein. This is in some sense related to paraconsistency: seeing the image as a duck implies that it cannot be seen as a rabbit and vice versa, yet the image is somehow both a duck and rabbit at the same time. These competing interpretations cannot be reconciled since to be duck means to be not rabbit and to be rabbit means to be not duck. For polysemantic sets, there would be no way to tell from the data which model is the 'true' one. The amount of mutual information shared by the competing optimal models is minimal or zero: to assert rabbit is to reject duck.

Definition 4.1 (Polysemantic data set) *A data set D is* polysemantic *if there exist multiple optimal codifications $M_i \in \mathcal{M}$ for D such that the mutual information between any two M_i is small, or zero, i.e. $I(M_1 : M_2)$ is near to or equal to D.*

With this definition, we try to capture the sense that D is polysemantic ex-

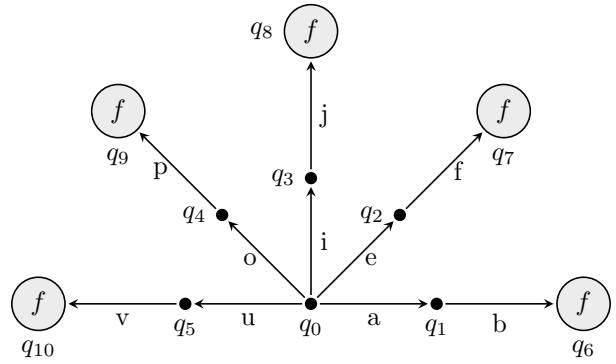
actly because in compressing D to form M_i we become unable to compress it to form M_j. The requirement that the mutual information between two models is minimised captures the sense of polysemanticism exemplified by Wittgenstein's duck-rabbit. Indeed, if there was significant mutual information between interpretations 'duck' and 'rabbit', it would seem to imply a way to fully compress the duck-rabbit into a single coherent image; it would cease to be gestalt. It is precisely the switching roles of uncompressed ad-hoc data (possible noise), and interpreted structural data—which are complements of one another—that gives rise to polysemanticism.

We give the following example as a motivation for the kind of data set we are talking about, take $D_+ = \{ab, ac\}$. We will show how mutually exclusive compression can lead to disjoint languages.



The left hand automaton accepts the language $L_1 = (ac)^* + (ab)$. The right hand automaton accepts the language $L_2 = (a^*b) + (a^*c)$. Clearly these languages are not fully disjoint as they each must contain at least the data D_+ by lemma 2.12. However, since the state space has been partitioned in an entirely irreconcilable way, we have $(L_1 \cap L_2) \setminus D_+ = \emptyset$.

We can assess the MDL scores for these models with respect to the data. Both have the same number of states, final states, transitions and alphabet. Thus the model code is of the same descriptive complexity. However, the data-to-model code is slightly different: $5 \log 2$ bits versus $4 \log 3$ bits, for the left hand and right hand automata, respectively: a difference of roughly 1.34 bits. This is almost too close to call using the MDL perspective. However, this example



does not yet show that there are two competing optimal models for the data set, since the universal automaton is still a stronger contender. In the following we will construct a data set which clearly does have polysemantic properties, and prove the following theorem.

Theorem 4.2 *Regular languages are polysemantic under MDL.*

In principle, just one counterexample is sufficient to disprove the claim that regular languages are *not* polysemantic under MDL. This would then imply, under the assumption that languages are regular, that there are cases where regular languages are also polysemantic.

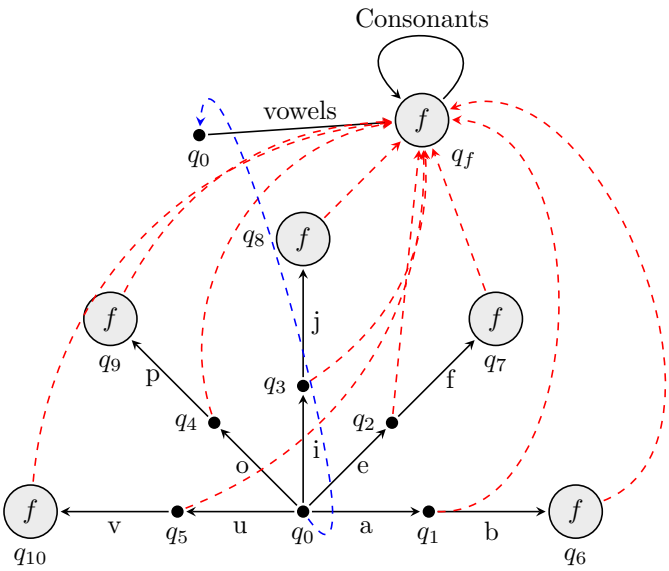
Thus we first prove the following lemma:

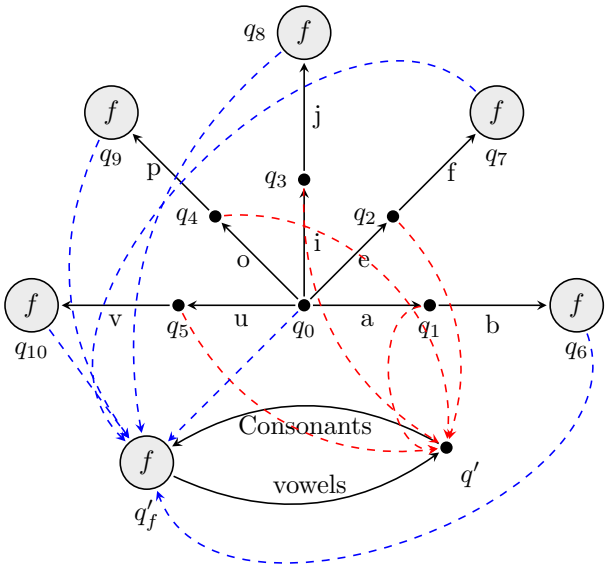
Lemma 4.3 *There exist data samples from regular languages which give rise to multiple optimal models under MDL.*

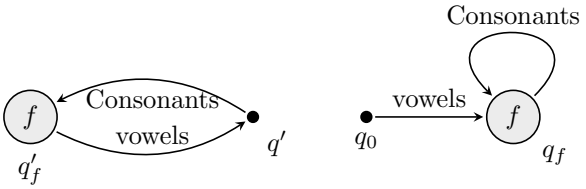
Proof. Suppose we have a data set made up from positive *and* negative examples (in order to control for overgeneralisation). Let $D^0_+ = \{ab_+, ef_+, ij_+, op_+, uv_+\}$ and $D^0_- = \{b_-, f_-, j_-, p_-, v_-\}$. With $D^0 = D^0_+ \cup D^0_-$. The PTA can be formed as usual:

The data seem to have the pattern "vowel followed by consonant", and the rule "never consonant without a vowel prefix." There are two natural choices for regular languages which might describe this: $L_1 = (Vowel) \circ (Consonant)^*$, or, $L_2 = [(Vowel) \circ (Consonant)]^*$. DFA models for these languages can be found by merging states of the PTA.

The merge of $q_i \in B(q_f, \pi)\ \forall i : 0 < i \leq 10$, and $q_0 \in B(q_0, \pi)$ yields the DFA representing L_1. This merge is depicted below top. The merge of $q_0, q_n \in B(q'_f, \pi')\ \forall n : 5 < n \leq 10$, and $q_n \in B(q', \pi') \forall n : 0 < n \leq 5$ yields the DFA representing L_2. This merge is depicted below bottom.


Consonants
vowels
q_0
q_f
q_8
q_9
q_7
q_3
q_4
q_2
q_{10}
q_5
q_1
q_6
j
p
f
i
o
e
v
u
a
b


q_8
q_9
q_7
q_3
q_4
q_2
q_{10}
q_5
q_0
q_1
q_6
j
p
f
i
o
e
v
u
a
b
Consonants
vowels
q'
q'_f



We now have two, 2-state DFA which are consistent with D^0.

A_2 on the left, and A_1 on the right. Note that $L(A_1) \cap L(A_2) = D^0$, there are no other words in the intersection. Indeed, in A_1 a consonant can follow a consonant which is not possible in A_2, whereas in A_2 a vowel can follow a consonant, which is not possible in A_1. We now show that A_1 and A_2 are equally parsimonious optimal descriptions of D^0 despite being disjoint.

Recall the MDL score sc of A:

$$sc(A, D_+) = |Q| + ||\delta||(2\log|Q| + \log|\Sigma|) + \sum_{w \in D_+} ch(q_0, w)$$

where $||\delta||$ is the number of transitions of A.

We associate with each state $q \in Q$ the value $t_q = \sum_{a\in\Sigma} |\delta(q, a)|$ if $q \notin F$. Whereas, if $q \in F$ then $t_q = 1 + \sum_{a\in\Sigma} |\delta(q, a)|$, since the choice to accept is available. We use the recursive definition of $ch(q, w)$ to describe the *data-to-model* code. For the empty word we have $ch(q, \epsilon) = \log(t_q)$ if $q \in F$; otherwise $ch(q, \epsilon) = \infty$. For $w = au (a \in \Sigma, u \in \Sigma^*)$, the function depends on the recursive definition: $ch(q, w) = \log(t_q) + min_{r\in\delta(q,a)} ch(r, u)$ and $c(q, w) = \infty$ if $\delta(q, a) = \emptyset$.

We have $t_{q'_f} = t_{q_f} = 6$ and $t_{q_0} = t_{q'} = 5$. However, since q'_f and q_f are the final states of L_2 and L_1 respectively, and likewise q' and q_0 are the non-final states, it is clear that the cost of encoding D_+ on either A_1 or A_2 will be the same, as encoding D_+ according to either model nullifies what might be saved by encoding in the other model. Since the number of states, transitions and alphabet are also identical for A_1 and A_2, we can be sure $sc(A_1, D_+) = sc(A_2, D_+)$. This is, incidentally, 93 bits rounded to the nearest whole bit.

Furthermore, we can be sure that there can be no smaller model, since the universal automaton is inconsistent with D^0_-. There is of course a 3-state automaton generating a *finite* language equivalent to the PTA which has slightly lower descriptive complexity (91 bits). This is formed by essentially 'collapsing' the fan shaped PTA into a chain of three states. However, this language can be ruled out as it offers no possibility for generalisation. Any other merge will result in an inconsistent automaton. Thus D^0 is polysemantic under MDL.

□

Corollary 4.4 *Regular languages are polysemantic under MDL.*

Proof. We have shown, assuming the sampled language is regular, there can be data sets which give rise to multiple, distinct optimal models and these models share little to no mutual information. D^0 is one such data set. The existence of this data set falsifies the negation of theorem 4.2. From this we conclude that regular languages are polysemantic under MDL.

□

We now seek to generalise the above result by showing that there are, in fact, infinitely many regular languages which are polysemantic under MDL. To do so, we presently go on to define a subclass of regular languages that generalises the essential features of the counterexample. This new class of regular languages captures the structural properties of the kind of regular language we have shown to be polysemantic under MDL. However, there may yet be other examples of polysemantic regular languages outside of this subclass.

We start with several important definitions:

Definition 4.5 (Concatenation) *For regular languages A and B and strings a and b, define the* concatenation *of A and B as $A \circ B = \{ab \mid a \in A \text{ and } b \in B\}$.*

Definition 4.6 (Alteration) *For regular languages A and B and strings c, we define the* alteration *of A and B as $A \cup B = \{c \mid c \in A \text{ or } c \in B\}$.*

Definition 4.7 (Kleene star) *For a regular language A, and strings $a_i \in A$, we define the* Kleene star *of A to be $A^* = \{a_1 a_2 \dots a_k \mid k \geq 0 \text{ and each } a_i \in A\}$*

The above three definitions essentially define the closure properties of regular languages; viz. combining regular languages with Kleene star, concatenation and alteration yield new regular languages. We say, regular languages are *closed* under the aforementioned operations.

Definition 4.8 (Regular expressions) *Regular languages are described by* regular expressions, *which are constructed from the letters of the alphabet, the empty set, and Boolean operators: (concatenation, alteration and Kleene star).*

We have implicitly been using regular expressions throughout the thesis. We state the definition here for use in the proof that is to come. Regular expressions are usually said to *generate* regular languages, whereas deterministic finite automata are said to *accept* regular languages.

One can also talk about sub-classes of regular languages. For instance, languages generated by regular expressions which have been built up from operations that are more restrictive than those used for regular languages. For example:

Definition 4.9 (Alteration-free regular languages) *We call the sub-class of regular languages which are described by regular expressions constructed from the letters of alphabet, the empty set and Boolean operators including concatenation and Kleene star, but excluding alteration, the* alteration-free *regular languages.*

We can define the alteration-free regular languages recursively, as follows:

1. ϵ is an alteration-free regular expression which indicates the language of the empty string.
2. E is an alteration-free regular expression denoting the empty language $L(E) = \{\}$.
3. a is an alteration-free regular expression denoting the language $L = \{a\}$.
4. If A is an alteration-free regular expression denoting the language $L(A)$ and B is an alteration-free regular expression denoting the language $L(B)$, then:
   (a) $A \circ B$ is an alteration-free regular expression corresponding to the language $L(A) \circ L(B)$ where $L(A \circ B) = L(A) \circ L(B)$.
   (b) A^* is an alteration-free regular expression corresponding to the language $L(A^*)$ where $L(A^*) = (L(A))^*$.

Note: technically, the Kleene closure L^* on L is defined as $\bigcup_{i=0}^{\infty} L^i$, which clearly involves some notion of set union in the definition. However, since our recursive definition of alteration-free regular languages disallows the construction of regular expressions of the form $(a \cup b)$, the Kleene closure of an alteration-free regular set simply becomes shorthand for the (potentially infinite) concatenation of some string with itself. That is, since every *finite* alteration-free regular language is simply a language consisting of a single finite string.

Definition 4.10 (Star-height) *The* star height *of a regular expression A over a finite alphabet Σ is inductively defined as follows:*

$$\begin{aligned} h(\emptyset) = h(\epsilon) = h(a) &= 0 \\ h(A \circ B) = h(A \cup B) &= max\Big(h(A), h(B)\Big) \\ h(A^*) &= h(A) + 1 \end{aligned}$$

The star-height $h(A)$ of a regular language A is defined as the minimum star-height among all regular expressions representing A. See [48]

Definition 4.11 (Star-free regular languages) *We call the sub-class of regular languages which are described by regular expressions constructed from the letters of alphabet, the empty set and Boolean operators including concatenation, complementation and alteration, but excluding Kleene star, the* star-free *regular languages* [49].

If complementation is defined so as not to increase star-height ($h(A^c) = h(A)$), then it is clear to see that star-free languages are those with a generalised star-height of 0.

Definition 4.12 (Confusable regular languages) *We call the following sub-class of regular languages* confusable:

$$\{\{L \mid L \text{ is regular}\} \setminus \{L \mid L \text{ is star-free}\}\} \cap \{L \mid L \text{ is alteration-free}\}$$

This is the set of languages which are infinite in size, and which are described by regular expressions constructed from the letters of the alphabet, the empty set, and Boolean operators: concatenation and Kleene star. Such languages also have a star-height of at least 1.

The syntax of confusable regular expressions therefore has a reduced alphabet $S = \{a, b, \ldots, \circ, (,), ^*\}$, and a more restrictive set of grammatical rules. The grammar is the same as ordinary regular languages, with the added rule that well formed confusable regular expressions have at least one sub-string where Kleene star has been applied. Note that $ab = a \circ b$ and $a^* = (a)^*$ are shorthand expressions.

Examples of such languages can be given with regular expresions such as:

$$L = (abc)^* \text{ of star-height 1}$$
$$L = a^*bca(da)^* \text{ of star-height 1}$$
$$L = (de(fe)^*)^* \text{ of star-height 2}$$

Lemma 4.13. *For every confusable language L defined by a regular expression with # alphabet symbols $|\Sigma| \geq 2$, there exists a finite word of the form $w = a_1a_2 \ldots a_n$ with $a_i \in \Sigma$ and a data set $D = \{w\}$, which will be polysemantic under MDL.*

Proof. Consider the regular expression for some confusable language, L_1 using an alphabet of size $|\Sigma| \geq 2$. This will be a string R of symbols, $R = s_1s_2 \ldots s_m$, with $s_i \in S$ where S is the set of syntax symbols used in the regular expression. By definition there exists at least one sub-string of R of the form $(s_j \ldots s_k)^*$ corresponding to a Kleene star operation of star-height 1, on some part of the regular expression.

Suppose we parse R and print the symbol s_i whenever s_i is a symbol from the

alphabet $a \in \Sigma$. This generates a finite string of concatenated alphabet symbols $a_1 a_2 \dots a_n$. Call this string w. It is clear to see that $w \in L_1$. For any string v, $(v)^1 = v$. Thus, parsing the string and printing every alphabet symbol will in effect generate a concatenation of symbols capturing the 1^{st} iteration of each and every Kleene star operation in R.

Now take the sub-string of R, the symbols $s_j \dots s_k$ are by definition a regular expression for some concatenated alphabet symbols, which correspond to a sub-string in w. There are two cases:

1. $\neg\forall s : s_j = s_{j+2} = \dots = s_k$ i.e. there is at least one alphabet symbol inside the brackets which is not identical to the others.
2. $s_j = s_{j+2} = \dots = s_k$ i.e. there is only one kind of alphabet symbol inside the brackets.

For the first case: we can move the position of the brackets: let s_{j-1} exchange positions with s_j. Or, alternatively, let s_k jump over s_{k+1} and s_{k+2}, so that the order is now $s_{k+1} s_{k+2} s_k$.

In each of these operations, we have by definition new regular expressions R', R'' for distinct confusable regular languages, as the substring repeated by the Kleene star operation has been changed. For example, a string of the form $\dots (ab)^* \dots$ might become $\dots ab^* \dots$ or $\dots a^* b \dots$ under these operations.

For the second case, since the regular expression uses an alphabet of at least two symbols, we can be sure there is at least one different alphabet symbol in R. This time we can expand rather than shrink the scope of the bracketed expression. Expanding the scope maximally such that s_{j-1} is now the initial symbol, and s_{k+1}, s_{k+2} is the final segment, will yield a regular expression for a new language R''' by ensuring that at least one other alphabet symbol is now under the scope of the Kleene star operation. Note that doing this may increase the star-height of the regular expression. For example $\dots (aa)^* \dots$ would become $(\dots aa \dots)^*$.

However, if we parse R', R'' or R''' under the same algorithm as before: print the symbol s_i whenever s_i is a symbol from the alphabet $a \in \Sigma$, then we recover w. Thus $w \in L_2 = R'$, $w \in L_3 = R''$, $w \in L_4 = R'''$ but clearly $R \neq R' \neq R'' \neq R'''$ since regular expressions can be unambiguously parsed from the left (or right).

This implies that the data set $D = \{w\}$ is polysemantic under MDL, since the models R, R', R'', R''' differ only at some repeatable sub-string in w. Thus $D = \{w\}$ will be polysemantic under MDL as there is no information in w about which part of the string is repeated by the Kleene star operation. This means it will take approximately the same amount of information to encode D according to multiple confusable languages.

If, for example, the star-height of the target language is guessed to be $h(M) = 1$,

for some target automaton $M = (Q, \Sigma, q_0, \delta, F)$: then if $|w| = n$, there will be confusable languages L_i, with $i \in I$ for some index set, and corresponding automata over the same alphabet with $|Q_i| = n-1$, $||\delta_i|| = n$, and data-to-model code which varies by $\log 2$ i.e. the cost of encoding the final state whenever it is not deadlocked. Such automata are assumed to contain only a single loop.

In other words, there would exist (at least two) models A_i, A_j and a data set $D = \{w\}$ such that $D \subseteq L_1(A_i)$ and $D \subseteq L_2(A_j)$ and $sc(A_i, D) \approx sc(A_j, D)$.

□

Assuming star-height $h(A) > 1$ makes the effect even more extreme: the data set $\{w\}$ would provide proportionally less information about nested repeating patterns from languages of increasing star-height. For instance, consider the language $L = (a(a(ab)^*b)^*b)^*$ and the data set $D = \{aaabbb\}$. The generalisation of the data set to, for example, $L' = a^*b^*$ would seem far more appropriate in terms of MDL, since it is a much simpler language. But $L'' = aaab^*$ is an even simpler language with a very close MDL score. Thus, even though the true model might be excluded under MDL in this case, the data set $\{w\}$ is still a polysemantic sample of the confusable regular language L. If, however, the star-height is known (in this case equal to 3), then using the algorithm given above, we can form another language (for instance) $L''' = (a(a(a)^*bb)^*b)^*$ with an identical MDL score when D is given as input for L or L'''.

To sum up, we know that polysemantic data sets always exist under MDL for confusable regular languages. That is, any confusable regular language can be made polysemantic under MDL by at least one insufficient data set consisting of a singleton data point.

Corollary 4.14 *There are infinitely many regular languages that can be made polysemantic under MDL.*

Proof. This is clearly implied by lemma 4.13, so the proof is immediate.

□

Since confusable regular languages are all infinite regular languages without alteration, we actually improved upon the result of lemma 4.3, which provided a counterexample to the claim that regular languages are not polysemantic under MDL. We have now shown that an infinite number of regular languages are polysemantic under MDL, since there are infinitely many confusable regular languages. The basic idea is still simple: take a data set of positive examples of an unknown DFA. One can do evidence-driven state merging on a constructed PTA. By merging states loops are created on the learned DFA, but sometimes

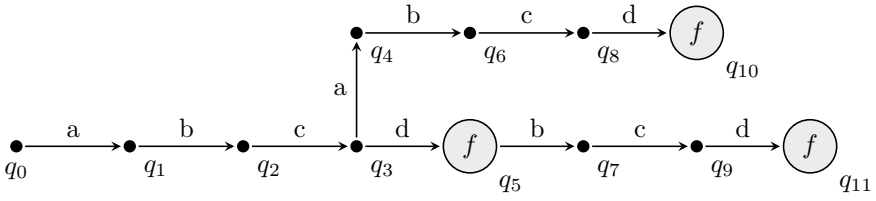
there are mutually exclusive choices the algorithm must take in order to generalises the sample. This creates disjoint hypotheses about the language of the unknown DFA.

Suffice it to say, samples from regular languages can be entirely insufficient for recovering the structure of the target automaton. This is especially the case when the star-height of the target language is large, as this implies a level of recursive complexity in the automaton that cannot be captured by insufficient data. In section 5, we will contextualise the polysemanticism result for confusable languages in relation to patterns in nature and pareidolia. For now, the question arises, does polysemanticism continue to pose a problem as the quality of the data increases? In particular, can fair samples help us escape from this problem?

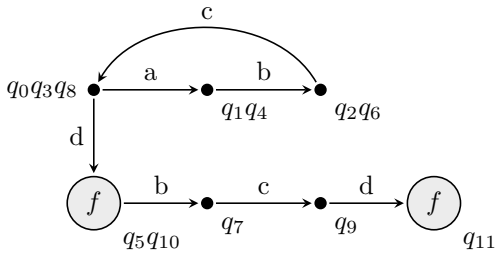
4.1.1 Polysemantic Fair Samples.

Unfortunately, initial investigations point towards the ineffectiveness of fair samples for dealing with polysemanticism. We next construct a model of a polysemantic data set based on fair samples, and show how the problem reoccurs [50].

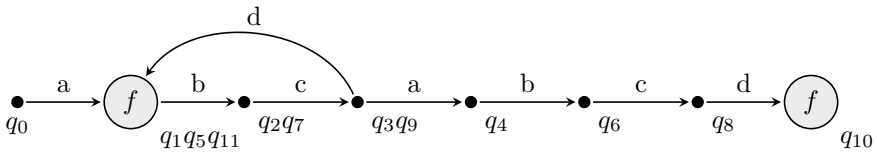
Suppose we have the following data set: $D^0 = \{abcd_+, abcabcd_+, abcdbcd_+\}$. The PTA can be constructed in the usual manner:



There is a sequence of merges which can lead to the following models: first, a model for the regular language $L_1 = \{(abc)^*d + abcdbcd\}$. The string '$abcdbcd$' in this case we interpret as unwanted noise.

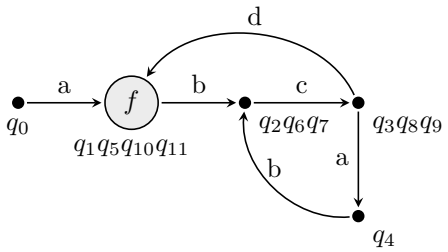


Secondly, the model for the regular language $L_2 = \{a(bcd)^* + abcabcd\}$, where '$abcabcd$' is noise.



It would seem that fair sampling of the target automaton should eventually reveal which of these automata are most parsimonious under MDL. Indeed, there is only one accepting word that is 'noisy' on both examples, whereas there are an infinite amount of words corresponding to the rules $(abc)^*d$ or $a(bcd)^*$. Whilst, it is true that $L(A_1)$ is more parsimonious for these particular examples, there are situations where the sample may come from a confusable automaton such that no amount of sampling will remove the confusion.

Consider $L_3 = a((bca)^*bcd)^*$, given by the following automaton:



It is possible to generate fair samples of this automaton that appear, from the perspective of MDL, to be generated by L_1: for instance, let $(abc)^nd \in D^1$ for $1 \leq n \leq 1000$. The data are almost surely fair for L_3 since the transition structure is captured, but the model of the language $(abc)^*d$ is more parsimonious. Depending on the distributions of examples from L_3, we can generate fair data samples that seem to be without noise, or data sets which appear to have a lot of latent noise. The interpretation of these data sets can lead to polysemanticism as in the case of L_1 and L_2 where in the former, words of the form $a(bcd)^*$ are considered to be noise, while in the latter case words of the form $(abc)^*d$ are considered to be noise. Of course, we can always create longer branches in L_1 or L_2 to represent this noise.

4.2 Read Only Right Moving Turing Machines, DFA, and Regular Languages.

Why is it that regular languages exhibit polysemy? Exactly what kind of notion of computation does one require in order for data to possibly appear polysemantic? In the case of regular languages, we are dealing more or less with very simple parsing algorithms, that seem to behave purely syntactically. They don't seem to exhibit semantic ambiguity until we consider the sampling process.

A regular language is essentially just a set of instructions to move through a graph, we now show how this is identical to the description of a read-only Turing machine that is only allowed to parse its input tape by moving its tape heads around without printing, and can only move the input tape to the right.

Definition 4.6 (Read-only right-moving Turing machine) *A* Read-only right-moving Turing machine (RORMTM), *M, is a 7-tuple $(Q, \Gamma, b, \Sigma, \delta, q_0, F)$ where:*

1. Q is a finite set of states $q \in Q$
2. Γ is a finite alphabet of symbols $a \in \Gamma$
3. b is the blank symbol (the only symbol which may occur infinitely on the work-tape)
4. $\Sigma \subseteq \Gamma \setminus b$
5. $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{R, S\}$ is the *transition* or *step* function which moves the head to the right after reading each symbol at a state or stays put.
6. $q_0 \in Q$ is the initial state
7. $F \subseteq Q$ is the set of final or accepting states $f \in F$

Theorem 25 *Read-only right moving Turing machines accept exactly the regular languages.*

Sketch. Suppose M is a TM with a set of states Q, an alphabet Γ, and transition function $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, S\}$ where S and R determine whether the read-only head stays put or moves to the right. We next define a DFA A with δ_A, a transition function defined as follows:

1. If M is in state q when it first moves rightward onto a symbol $a \in \Gamma$, either M accepts or rejects without ever moving farther to the right. In this case let A transition to an accepting or rejecting state and end the computation.
2. If M falls into a loop and never moves further right, let A reject.

3. Otherwise, let $\delta_A(q, a)$ define the state which M moves to if M moves to the right. This is clearly determined by q and a.

□

Weaker variants of Turing machines, and their relationship to regular languages are discussed in [51], where the crux of this proof sketch are also found.

This model is now equivalent to a DFA. Clearly this TM can never backtrack; it is right-moving. Indeed, nor can it write on the tape; it is read-only. The intuition is that such a Turing machine has only one-way access to its input, much like a DFA. Hence, it is clearly not possible for such a Turing machine to store information, or compute operations on the input tape using more than a constant amount of computational space. These Turing machines are able to parse strings, just like regular languages. If we had a UTM, on the other hand, we could *store* the information of a DFA by first using a read-only right-moving Turing machine to parse the structure of a regular grammar, and then utilise the grammar on the rest of the input tape. However, this is clearly not possible without more than constant memory.

What this demonstrates, however, is that regular languages are unable to emulate one another. Indeed, feeding a read-only right moving Turing machine the instructions of another read-only right moving Turing machine would lead to situations where the one machine trying to emulate the other would be stuck in a loop or rejecting. This would be equivalent to running a DFA with data inconsistent with the regular language it accepts. In fact, a read-only, right moving Turing machine is only able to *parse.* Necessarily, therefore, it cannot do calculations on the data. This is exactly the intuition that prevents DFA from being able to emulate one another, for they too are just graphical representations of ways to parse strings of data. In fact, regular expressions can always be parsed unambiguously from the left (or right). This in itself implies there is no memory function for equivalent constructions.

What this tells us about regular languages is that given perfect data, ultimately all regular languages should be distinguishable from one another. Indeed, a regular language is simply is a set of instructions that tells us how to move around a graph, and so long as the languages are not equivalent to one another, there will be some distinguishing string which can be used to differentiate between languages. Indeed, this was already known by the Myhill-Nerode theorem [52], and used to show the existence of unique minimal DFA for each regular language with a number of states equal to the number of distinct equivalence classes of the language. What this suggests is that the polysemanticism we see is not attributable to the nature of regular languages in and of themselves, but rather to the nature of inferring complete structure from incomplete information; inferring an infinite set from finite data is hard exactly because we cannot be sure

we will ever find a uniquely distinguishing element from finite samples. We will explore what this means, in philosophical terms, in the next section.

5 Patterns in Nature: Encodings and Processes.

Patterns and processes are found everywhere in nature; from the logarithmic spiral in the shell of the nautilus, cyclonic weather, and sprial galaxies; the fractal pattern of coast formations; to atomic rates of decay or fluctuations in the stock-market. But in what sense is reality actually made up of these phenomena (or the processes which underlie them)? And furthermore, how are various examples of the same pattern related? Perhaps a better question to ask is to what extent is the perception of a pattern due to cognitive rather than physical processes? What is clear is that spontaneous pattern formation seems to exhibit mathematical structure. This suggests a link between the structure of patterns and the underlying processes which generate them. But this fact does little to inform us as to whether patterns in nature are simply coincidental (e.g. fluctuations in the stock market may simply be caused by noise), benign (see [53]), or whether they are truly indicative of structure. It is worth stating that here we understand 'pattern' to be an essentially geometrical notion, and hence related to space. 'Process', on the contrary, cannot be divorced from its temporal nature, almost by definition. Thus, by posing a question on the relationship between patterns and processes, by implication we pose a question on the nature of the relationship between space and time.

A more general notion of a pattern would be that of an abstract structure; a formal object defined by a set of rules. Understood this way, the question about how patterns relate to processes becomes a question about the role of abstract structures in the temporal, perceptible world. This is, in essence, the question which occupies most of the discussion in Plato's *Timaeus*: how do general, abstract, mathematical forms underlie the perceptible objects of our everyday life? Suffice it to say, that actual processes seem to unfold according to some kind of rule, e.g. the nautilus' shell grows according to a ratio. Likewise a pattern abstracted from particulars, though timeless, seems to somehow define all the processes which unfold according to it, for instance, the role Fibonacci numbers play in the growth of some spirals and many other examples where growth is the operative concept. It remains to be seen exactly what this duality between mathematical structures (patterns) and processes entails for ontology. However, what has become clear throughout this thesis is that the process underlying a pattern might not be so transparent. One can, for example, look around the world and see Fibonacci numbers in the data, but it is not clear whether or not we are making false generalisations about the system which generated the data. Indeed, if the data are consistent with multiple hypotheses, we cannot ground the justification behind some generalisation in the data alone.

Metaphysicians have argued for the notion that patterns and processes play an underlying role in reality, i.e. part of a substratum of reality [54-58]. However, rarely has the discussion focused on developing theoretical notions to explain how patterns and processes actually relate to one another. This might be due to

the fact that it is rather hard to specify the duality between a timeless syntactic description of an abstract structure (or patterns generally) given by mathematics and logic, and the temporal nature of actual things. However, it is the view of this thesis, that a pattern could be taken to be some distribution of examples (viz. data), whilst a process is the generating system which we are sampling whenever we observe these patterns. That is, namely: that which we attempt to model in Science.

There is a very real sense in which patterns and processes can be understood as kinds of formal languages; patterns can be seen to exhibit a syntax – or set of rules – which specifies some information about how the pattern is generated. An instantaneous result thereof is that simple patterns can be described with less information than more complicated ones, with some patterns behaving so randomly that they achieve a fundamental limit of disorder. Indeed, the notion of a pattern is abstract enough so that there is no sufficient reason to imagine a pattern being instantiated in any perceptible way whatsoever. Processes, on the other hand, are ongoing; they are in a sense, a pattern in time. If a process is observed for a long time, it may exhibit some regularities which can be understood as the exhibition of some pattern. It can then be treated in the aforementioned way: as a language. In this way, one can understand a process as also obeying some kind of syntax or set of rules. Put rather differently, if some process exhibits a pattern, it is equivalent to that process "accepting" the pattern. It is for this reason that patterns and processes seem to be two slides of the same coin.

This way of describing the duality of patterns and processes as languages lends itself particularly well to describing the structure of computation. Indeed, a computation is at once both a set of rules to follow, and an actual process that may or may not terminate. Due to this it is very natural to suppose that computation would be an ideal candidate for explaining the sense in which patterns and processes underlie reality. If the world is in some sense fundamentally computational, then the sense in which patterns and processes underlie reality is clear; processes are the subroutines of some kind of cosmic computation, and patterns presented in those processes reveal the structure of the computation. To paraphrase Prigogine and Stengers [59] here, in describing this duality we search for a new metaphor, one which captures the relationship between stillness and motion, time arrested and time passing.

However, how can one distinguish a pattern of substance from one which is (co)incidental? In fact, this is simply philosophically untenable: as Bohm succinctly puts it "the content of the observed fact cannot coherently be regarded as separate from modes of observation and instrumentation and modes of theoretical understanding" [58]. The messy, unpredictable, imperfect kinds of knowledge that are ascertainable from insufficient data leads to modelling techniques which necessarily permit irreconcilable pictures of reality. Thus, in these contexts in particular, we must be aware of the sense in which, a pattern, however much it

impresses itself on our senses, however much it is suggestive of some underlying process, may be a figment of our minds, or a quirk of the tools and methods which we use to analyse the data.

For the particular case of learning formal languages, in our exploration of polysemantism we have evidenced this kind of problem arising. However, the connection between these idealised formal languages and real instantiated patterns in nature goes deeper than one might imagine: the most striking connection becomes clear when we analyse the relationship between polysemanticism and languages with star-height greater than 0. As we have seen, star-height captures the idea that repeating patterns may be nested and thus recursion can occur at different levels in the pattern. In the context of DFA this implies nested loops within the automaton. In terms of patterns in nature, nested levels of recursion are a fundamental property of fractals and are related to the notion of the scale of a process. The DFA models with nested loops thereby provide a clear, albeit abstract picture of the kind of phenomena that give rise to pareidolia.

More precisely, many random fractal patterns, such as cloud formations are scale-free: we, in fact, mentioned this in the introductory remarks on pareidolia. Indeed, we note that one of the reasons we see images in clouds is the scale-free nature of the underlying physical process: the condensation of water is happening at every scale, from individual atoms to droplets to larger structures and so forth. Thus, in looking at a complex scale-free cloud formation we have the possibility to see patterns at many different scales. If the cloud as a whole does not have a meaningful form there might be smaller formations that can be grouped into meaningful forms. In this sense random fractal forms are overloaded with possible meanings.

This is truly the kind of phenomena we see in confusable regular languages: certain data samples taken from these languages may appear to be generated from simpler languages. The data which do not necessarily fit that model are interpreted as noise. As we have often mentioned, simple meanings are more easily identified, which is part of the reason pareidolic and polysemantic phenomena are possible in data sets that have a lot of latent noise, and/or randomness. However, the more nested the regular language is, the more it is possible to identify patterns at every level, just like scale-free processes in nature. Put this way is it any wonder that the entire subclass of confusable regular languages is polysemantic under MDL? These kinds of formal languages exemplify polysemanticism and paredolia in natural processes.

6 Conclusions.

The work presented in this thesis is a step towards understanding the subtle role of entropy in the process of optimal model selection. From a focused perspective, we have demonstrated for the very simple class of regular languages, that the quality and nature of the data sample conditions the possibilities for learning in the following sense: an insufficient sample, which fails to provide the necessary structural information needed to reconstruct a regular grammar, will, through the lens of minimum description length, potentially give rise to multiple optimal models each of which share little to no mutual information. This problem is attributable to the lack of information contained in the sample, and not to the nature of regular languages themselves. Indeed, it is the inference from the incomplete picture given by an insufficient sample, to a possibly true generalisation, which poses a problem. If (as is often the case in Science) the goal is to find the most parsimonious model, there will be no way to rule out a myriad of inferences to false generalisations; there is no way to distinguish between noise or randomness, and true structural data.

This conclusion that we have arrived at may strike one as obvious. Yet, in fact, we were originally working under the assumption that regular languages could not be polysemantic under MDL. The reason for holding this belief was that it seemed insoluble that such fundamentally simple computations could be mistakable for one another. Ultimately, a DFA is simply an effective procedure for parsing strings; we assumed that a random sampling of one of these procedures would therefore always be determinable, even if there were other "closely" competing hypotheses. In fact, this insight still holds some truth, as the sorts of data sets we have shown to give rise to multiple optimal models are highly unsuitable for drawing conclusions about generalised populations.

Perhaps more interestingly, however, is the fact that these insights provide one with a general picture of the role of entropy in optimal model selection which transcends the specific learning problem we have answered: there are analogous situations in human cognition where polysemanticism seems to arise due to the existence of localised order which is mistakenly taken to be evidence of a model. That is, irrespective of the simplicity of the model and the model selection problem, there is a general relationship of data, sample and population which is not based strictly in statistics or probability, but in the deeper framework of algorithmic learning. Specifically, learning with insufficient information leads to situations where entropy becomes invariant over optimal models. For machine learners this means that without supervision there may never be a way to ensure A.I. make sensible generalisations from insufficient data sets. From the more general view of inference and model selection in cognition, the results of the thesis suggest that the existence of patterns in nature do not necessarily support the reality of an underlying process (a model for the pattern). In fact, such distributions may be benign, or incidental, under measures of parsimony.

6.1 Summaries.

In this section we detail some of the challenges that were overcome in the process of researching this topic. We explain why our approach to the DFA learning problem may seem piecemeal at times, and why we consider the main contribution to be in developing one's understanding of the role of entropy in optimal model selection, as it pertains to the conditions which permit learning to occur.

6.1.1 The Circumstances Which Permit Learning to Take Place.

The thesis attempted to make clear, in the context of DFA identification, the conditions which typically permit the learning of regular languages. Ultimately, the attempt has failed to make these conditions precise. However, the reason for this is that every avenue for learning DFA without queries is problematic. There is, however, a use in acknowledging that an avenue is not worth traveling down.

We have described the computational complexity of the combinatorial problem of selecting an optimal DFA from an arbitrary data sample. It was clear from the outset that looking for a polynomial-time algorithm for selecting optimal DFA from arbitrary data samples was out of the question. Incidentally, finding such an algorithm would have solved the P=?NP problem, a strong indication that an MSc thesis is probably not the suitable place to attempt such a feat.

The original impetus of this thesis was to investigate whether computational equivalence, in particular bisimulation, could offer possible new avenues for research into the problem, (see §A.2). The justification was based on the fact that one would expect an algorithm, whose aim it is to uncover the syntactic structure of a language, to infer an automaton that is in some way equivalent to the target automaton. However, we quickly understood that this would be unfeasible; computational equivalence is far too strict a notion to apply in this context, as state-merging does not preserve equivalence relations. Moreover, seeing as bisimulation is polynomial-time decidable over finite labeled transition systems [60], if there had been a way to utilise bisimulation in this manner, we would have discovered a fast algorithm for DFA identification, and could have concluded P=NP. Nevertheless, as a limit case some insight was attained: bisimulation-invariant grammars for regular languages can be minimised efficiently to a unique optimal DFA *à la* the result shown in [61]. Thus, for perfect samples, where the sample is essentially equivalent to the entire language, a unique optimal (minimal) model is discoverable in polynomial-time.

We then attempted to find some threshold value for the size and/or quality of the samples necessary to permit learning. Under maximum entropy assumptions, the sample might be generated by a number of random walks on the target

automaton. If these random walks were to recover all of the necessary structural information of the target automaton, the resulting data sample would be called *fair* with respect to the target automaton. Whilst in principle fair sampling ought to be possible, there are situations where the size of the sample necessary to recover the automaton is exponentially larger than the size of the target automaton. What this essentially implies is, for arbitrary automata, no threshold value for determining the size of the sample necessary to contain all the relevant information exists.

Finally, we investigated extremely insufficient data samples, to get a sense of the worst-case inferences from insufficient data. For these very simple samples, we discovered that there can be multiple optimal models, each of which share little to no mutual information. These polysemantic data sets are, in essence, the result of being maximally uncertain about the target automaton: that is, polysemanticism arises from uncertainty about the correct model.

What has become clear is that the quality of data needs to be near perfect (infinite, noiseless, uncorrupted) in order to avoid all the problems we have uncovered relating to the learning problem. Whilst, at the other end of the scale, insufficient, incomplete or inconsistent data can not be used to discover parsimonious models, as it is impossible to rule out what is noise from what is structural information. Between these extremes there is no threshold size for a sample set to permit learning. In fact, there is no way, without supervision and without exponentially large sample sets, for an algorithm to uncover the more topologically complex parts of the automaton's transition structure.

6.1.2 The Role of Entropy in Optimal Model Selection.

What does this tell us about the broader picture of entropy in optimal model selection? Ultimately, since entropy is both a measure of the information in objects themselves, and also in distributions over sets of objects, the results of this thesis reinforce the sense in which entropy plays both an epistemological and ontological role in model selection. Epistemological, in the sense that the (probabilistic) entropy of some distribution is a measure of our uncertainty about the average behaviour of individuals over that distribution. Entropy thus measures the epistemological state of affairs, since, if the entropy of a distribution is high, there is more information value gained in each sample of the distribution. However, (algorithmic) entropy measures the information in objects in and of themselves, and for that reason can be considered an ontological measure of information. In other words, the more informationally complex an object may be, the harder it will be find an algorithm to compute it. For that reason such objects may be harder to learn.

The problem of polysemanticism, we have argued, is essentially a problem with a lack of information. However, the reason for this is exactly because there

is a causal relationship between entropy and polysemanticism: data sets are considered to be polysemantic whenever there exist mutually irreconcilable optimal models for the data, under some notion of minimal description lengths, each of which share little to no mutual information. Ergo, whenever the entropy (specifically, the algorithmic mutual information) is invariant over multiple models, we see polysemanticism arise. For that reason polysemanticism is a lack of information. However, whether this lack of information is epistemological or ontological is debatable, since the polysemanticism result can be interpreted under both paradigms. Either the amount of mutual algorithmic information between two objects is invariant, in which case these two objects both seem to be good *a priori* models for the same data, and one is unable to decide which object is the ontologically responsible parent for the data. Or, the amount of mutual probabilistic information between two distributions is invariant, in which case one would be uncertain whether they are drawing examples from the former or the latter distribution, and hence unable to distinguish which one is the true distribution. It may be the case that polysemanticism has mathematical roots.

Mark Kac (1966) once asked "can we hear the shape of a drum?" [62]. The problem is a 2-dimensional generalisation of the question "can we hear the shape of a string". It is possible, in the case of the string, to work out the exact length of a string by sampling the sound of the string, and extracting a fundamental tone via Fourier Analysis. Some additional analysis of the non-fundamental tones—called harmonics—gives one some more information about the kind of stringed instrument which was sampled. However, one cannot, *in principle*, work out the shape of a drum [63]. This is because it has been proven that two distinct shapes can give rise to the same sound waves, and, hence, that there exists some non-surjective 2-dimensional functions which take some shape as an input, and output a sound wave, and which therefore will be indistinguishable from one another on the basis of sonic information alone. This means that whilst the data, viz. the sampled sound waves, would contain some information about the shape of a drum, they don't contain all the information, since you cannot, in principle, distinguish between the shape of some drums on the basis of the sound waves themselves. No amount of sound data will do.

From a cognitive point of view, on the other hand, many instances of pattern recognition and model selection appear polysemantic because of the subjective, solipsistic perspective of consciousness. These are typically instances of pareidolia. However, it is hard to ascribe a purely epistemological barrier to finding a single optimal model for gestalt images, since, in those instances, increasing the quality and size of the given data sample will not affect the polysemantic nature of the model selection problem. They appear, at root, ontologically polysemantic. Thus, we are somewhat stuck in the position of being uncertain about what kind of uncertainty we are uncertain of: either it is the world that is uncertain (or fundamentally gestalt), or it is our methods for apprehending the world that are insufficient (some data are insufficient for some questions). Either way, the answer to this question cannot be found in the data themselves.

6.2 Future Research Directions.

6.2.1 Learning Idealised Regular Languages with Random Walks.

It remains to be proven whether learning the average-case DFA under maximum entropy assumptions is easy. However, previous study of idealised DFA has resulted in some relevant facts concerning random walks and learning: first, the notion of a random DFA can be defined by selecting uniformly at random from the class of functionally complete DFA of state-size n with k outgoing arrows. By analysing the properties of typical elements of this class which hold with high probability, the notion of a typical DFA comes to the fore: typical DFA induce ergodic Markov chains from simple random walks on their underlying digraphs (starting at the initial state) [64]. This is due to the fact that a random k-out digraph $D_{(n,k)}$ of size n, has a unique largest strongly connected component (SCC) that is attractive with high probability [65]. Essentially what this means is that a simple random walk on the digraph of a typical DFA will eventually fall into a closed communicating class which is irreducible by definition, and aperiodic [64]. Hence, ergodic.

A result known as Ornstein's theory states that measure-theoretic entropy completely classifies Bernoulli shifts (generalised Bernoulli processes) up to isomorphism [66]. That is to say, two Bernoulli shifts are isomorphic if and only if they have the same entropy. Since ergodic Markov chains are aperiodic and irreducible by definition, they are also Mixing Markov [67]. This further implies that they are weakly Bernoulli [68], and hence isomorphic to a Bernoulli shift. Thus, there are Bernoulli shifts that are isomorphic to the realisations of ergodic stationary stochastic processes induced by random walks on typical DFA, and Ornstein's theory applies. Thus, there is a connection concerning optimal model selection between certain idealised DFA, random walks, and entropy.

Interestingly, the regular (sub)language generated by a sub-DFA corresponding to the SCC of a typical k-out DFA can be retrieved by decomposing the languages via inverse concatenation. Indeed, regular languages are closed under concatenation. The existence of an SCC in the digraph of a typical DFA implies a closed communicating class such that there are no transitions out of the SCC. This means any accepting word that utilises the SCC can be decomposed into a prefix outside the SCC and a suffix inside the SCC.

This implies that there is a *possibly* infinite language L_1 consisting of paths from the initial state of the typical DFA to states just within the SCC, and an infinite language L_2 consisting of paths starting from exactly those states to final states within the SCC.

Definition 6.1 (Factorial language) *A language* $L \subseteq \Sigma^*$ *is* factorial *if* $\forall x, y, z \in \Sigma^*(xyz \in L \implies y \in L)$.

Factorial languages can be explained intuitively: if all the states of a DFA are accepting, then any sub-word (factor) of an accepting word, is also accepting. Since labelled transition systems have no distinction between accepting and non-accepting states, there is a sense in which LTS describe factorial languages. In the context of typical DFA, if all the states within the SCC were accepting states, the sublanguage L_2 is also factorial. Thus equivalent to a sofic subshift [69]. Interestingly, the properties of random DFA we have been discussing can easily be generalised to LTS, since the results of [64][65] are reliant on the structure of random k-out digraphs of size n. The main difference between automata and transition systems is that the former are models of machines, programmes, languages, etc., while the latter are models of a machine's, programme's or language's behaviour. In this sense, one might argue from these results, that the *behaviour* of a typical DFA eventually becomes equivalent to a sofic subshift.

Clearly in the idealised situation of typical DFA with k-out structure, random walks would be effective for learning: sufficiently long maximal entropy random walks on the underlying graph will eventually fall into the SCC with high probability. Hence, a sample taken from n many of such random walks, will, with high probability, have recovered the structure of the closed communicating class. For the case of typical DFA, we can use the fact that such a language is prolongable and transitive to automatically reconstruct the language from the sample.

However, the conditions for this idealised situation would rarely occur. Indeed, the majority of DFA do not have a k-out structure, and as such typical DFA selected from the class of all DFA doubtfully have a SCC with high probability. The average-case DFA from this bigger class may yet be hard to learn.

6.2.2 Process Equivalence and Entropy.

The original idea of the thesis was to analyse for what notion of bisimulation (if any) is entropy an invariant. See §A.2 for definitions. Bisimulation is a local notion of equivalence (see [70] for logical analysis of bisimulation), whereas entropy (probabilistic, measure-theoretic, topological or algorithmic) is a somewhat more global measure. Models which satisfy some notion of local equivalence are not necessarily equivalent in terms of their global informational structure. Isomorphism, by contrast, is a one-to-one correspondence between points in a model, and thus is more of a likely candidate for preserving the entropic properties of a model's informational structure. Nevertheless searching for a similar result to Ornstein regarding bisimulations is tempting. Initial steps to combine analyses the logical notion of bisimulation with probabilistic processes are taken in [71][72].

Bisimulation is explicitly a form of equivalence that uses local information to verify global structural relationships. This property in particular we sought to

analyse for what it might inform us about optimal model selection. Intuitively, a bisimulation preserves the local information of models, hence if the data were to share the same local information as the model, we ought to be able to find bisimulations between the data and the models to aid in our inferences. Some work has been done to make bisimulation more of a 'relaxed' notion [73] and attempts in related learning problems have had some success using bisimulation, see [74][75]. It might be possible using a tool like mCRL2 [76] to utilise bisimulation for learning DFA in some situations. However, if one were always able to efficiently and effectively draw conclusions about exactly which hypotheses are optimal explanations for arbitrary data sets, it would seem to suggest that total, or near to total information, is contained within samples regardless of the quality of the sample data. This is simply not feasible. Bisimulation can optimise the description of any regular language, but it cannot make guesses about the hidden structure of an automaton from a data set. See §A.3.

However, it would still be interesting, given what is laid out in 6.2.1 to consider bisimulation again. Since the induced sofic subshifts for random LTS will be entropy-invariant under almost-topological conjugacy, there may yet be a similar result for some notion of bisimulation between the LTS (understood to be equivalent to the subshifts).

A Appendix.

A.1 Terminology.

Σ	denotes a finite *alphabet*.
$a, b, c, \ldots$	stand for the *letters* of the alphabet.
Σ^*	denotes the set of finite *words* over Σ.
$u, v, w, \ldots$	stand for finite words.
ϵ	is the empty word.
$\Sigma^+ = \Sigma^* \setminus \epsilon$	is the set of nonempty *words* over Σ.
$L_1, L_2, \ldots$	denote sets of finite words ($*$-languages) $\subseteq \Sigma^*$.
$M_1, M_2, \ldots$	denote automata.
$T_1, T_2, \ldots$	denote labeled transition systems.
$q, r, s, \ldots$	are states of an LTS or finite automaton.
$Q, R, S, \ldots$	are sets of states.

A.2 Definitions of Equivalence Between Computational Processes.

The following nine definitions are essentially given in [23] unless otherwise indicated. We list them here to detail the early work done exploring the relationship between target model and quotient automata. The next section (A.3) provides a rudimentary proof that demonstrates the impossibility of this initial line of research. However, considered alongside the speculative remarks in 6.2.2, there may be some notion of computational equivalence, akin to those listed here, that preserves entropy over certain idealised automata. A clear logical exposition of computational equivalence in terms of bisimulation is outlined in [70].

Definition A.1 (Strong bisimulation) *An equivalence relation that partitions the set of states in such a way that the set of actions that can be executed to reach some class is the same for every two states in a class, induces a* strong bisimulation [72].

Mathematically, the relation $\approx$ which satisfies the following is considered to be a strong bisimulation between labeled transition systems T, T':

$$q \approx q' \iff \forall a \in \Sigma \Big(\Delta_T(q, a) \approx \Delta_{T'}(q', a) \Big).$$

More commonly this is defined like so:

$$q \approx q' \iff \text{If } q \xrightarrow{a} r, \text{ then } q' \xrightarrow{a} r' \text{ for some } r' \text{ such that } (r, r') \in R$$
$$\wedge \text{ If } q' \xrightarrow{a} r', \text{ then } q \xrightarrow{a} r \text{ for some } r \text{ such that } (r, r') \in R.$$

For automata M, M', there are the added properties:

$$q_0 \approx q_0'$$
$$q \approx q' \implies \Big(q \in F_M \iff q' \in F_{M'}\Big).$$

Definition A.2 (Strong bisimularity) *Two states q, q' are strongly bisimilar, written $q \approx q'$ if and only if there exists a strong bisimulation R between them, and $(q, q') \in R$.*

$$\approx := \bigcup \{R \mid R \text{ is a strong bisimulation}\}$$

Definititon A.3 (Weak transition relation) *Let $T = (S, \Lambda, \to)$ be an LTS with an additional label $\tau \notin \Lambda$ for "hidden" or "unobservable" transitions. We define $\overset{a}{\Rightarrow}$ as follows:*

$$\overset{a}{\Rightarrow} = \begin{cases} (\xrightarrow{\tau})^* \circ \xrightarrow{a} \circ (\xrightarrow{\tau})^* & \text{if } a \neq \tau. \\ (\xrightarrow{\tau})^* & \text{if } a = \tau. \end{cases}$$

Where $\circ$ has the usual meaning of concatenation, and $(\cdot)^*$ the usual meaning of iteration. Weak transition relations can also be defined on automata without loss of generality.

Definition A.4 (Weak simulation) *One can define weak simulation by replacing the transition requirement (defined in §2, definition 2.13) with the following:*

$$q \sim q' \iff \text{If } q \xrightarrow{a} r, \text{ then } q' \overset{a}{\Rightarrow} r' \text{ for some } r' \text{ such that } (r, r') \in R_{sim}$$

Definition A.5 (Weak simularity) *Two states q, q' are weakly similar, written $q \sim_w q'$ if and only if there exists a simulation R_{sim_w} between them, and $(q, q') \in R_{sim_w}$.*

$$\sim_w := \bigcup \{R_{sim_w} \mid R_{sim_w} \text{ is a weak simulation}\}$$

Definition A.6 (Weak bisimulation) Mathematically, the relation $\approx_w$ which

satisfies the following is considered to be a weak bisimulation R_w between labeled transition systems T, T':

$$\begin{aligned} q \approx_w q' \iff & \text{ If } q' \xrightarrow{a} r', \text{ then } q \stackrel{a}{\Rightarrow} r \text{ for some } r \text{ such that } (r, r') \in R_w \\ \wedge & \text{ If } q \xrightarrow{a} r, \text{ then } q' \stackrel{a}{\Rightarrow} r' \text{ for some } r' \text{ such that } (r, r') \in R_w. \end{aligned}$$

For automata M, M', there are the added properties:

$$\begin{aligned} & q_0 \approx_w q_0' \\ & q \approx_w q' \implies \Big(q \in F_M \iff q' \in F_{M'} \Big). \end{aligned}$$

Definition A.7 (Weak bisimularity) *Two states q, q' are weakly bisimilar, written $q \approx_w q'$ if and only if there exists a weak bisimulation R_w between them, and $(q, q') \in R_w$.*

$$\approx_w := \bigcup \{ R_w \mid R_w \text{ is a weak bisimulation} \}$$

Definition A.8 (Branching bisimulation) A relation $R_b \subseteq S \times S$ between labeled transition systems T, T' is called a branching bisimulation if it is *symmetric* and satisfies the following property: If $(q, q') \in R_b$ and $q \xrightarrow{a} r$ then:

1. If $a \neq \tau$ then $\exists r', s'$ such that $q' (\xrightarrow{\tau})^* r' \xrightarrow{a} s'$ with $(q, r'), (r, s') \in R_b$.
2. If $a = \tau$ then $(r, q') \in R_b$

For automata M, M', there are the added properties:

$$\begin{aligned} & q_0 \approx_b q_0' \\ & q \approx_b q' \implies \Big(q \in F_M \implies q (\xrightarrow{\tau})^* q' \text{ with } q' \in F_{M'} \Big). \end{aligned}$$

Definition A.9 (Branching bisimularity) *Two states q, q' are branching bisimilar, written $q \approx_b q'$ if and only if there exists a branching bisimulation R_b between them, and $(q, q') \in R_b$.*

$$\approx_b := \bigcup \{ R_b \mid R_b \text{ is a branching bisimulation} \}$$

A.3 Proofs.

Theorem A.3.1 *State merging on finite automata does not preserve strong bisimulation.*

Along the same line as theorem 2.18, of §2. Suppose a DFA A with $q_1 \in \delta(q_0, a)$ and $q_2 \in \delta(q_1, a)$ for some $a \in \Sigma$ where $uaav \in L(A)$ for some $u, v \in \Sigma^*$ where, with some abuse of notation, $a \notin v$ and $a \notin u$. The merge of q_0, $q_1 \in B_1$, and $q_2 \in B_2$ under some partition π implies $B_1 \in \delta(B_1, a)$ and $B_2 \in \delta(B_1, a)$ by part (3) of the definition of quotient automaton. The quotient automaton A' yielded from such a merge is not bisimilar to A: consider the block B_1 and the transition $B_1 \in \delta(B_1, a)$. In order for there to be a bisimulation between A and A' there must be a state(s) bisimilar to B_1. If we assess $q_0 \approx B_1$ and $B_1 \xrightarrow{a} B_1$, we have $q_0 \xrightarrow{a} q_1$. So we must then check whether $(B_1, q_1) \in R$. Again, for the transition $B_1 \xrightarrow{a} B_1$ there is $q_1 \xrightarrow{a} q_2$. However, now we must assess whether $(B_1, q_2) \in R$, but this we know cannot hold, since $q_2 \xrightarrow{a} \top$ is false.

□

In the context of DFA, this theorem is actually already guaranteed by theorem 2.19. This is because determinacy implies the equivalence of strong bisimularity and trace/language-equivalence [77].

For other notions of computational equivalence listed in §A.2 (i.e. those defined using weak transition relations) similar proofs can be used to demonstrate that state-merging can often break equivalence. But it should not really require proof to expose the mismatch of bisimulation and state-merging. Intuitively, "bisimulation-invariant" grammars are especially uninteresting from the perspective of state-merging; inferring such grammars would amount to merely searching through equivalent descriptions. Since such an algorithms would have essentially already fixed the target grammar, optimal minimisation is possible with relative computational ease. However, since the main goal of grammatical inference is to infer a grammar from a sample, having a fixed target grammar, known *a priori*, would be to profoundly miss the wood for the trees.

B Bibliography

[1] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory* (2): 113–124, 1956.

[2] T. Jiang and B. Ravikumar. Minimal NFA Problems are Hard. *SIAM Journal on Computing*, 22: 1117–1141, 1993.

[3] D. Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 3(39): 337–350, 1978.

[4] L. Pitt. Inductive inference, DFAs, and computational complexity. *Lecture Notes in Computer Science*: 18–44, 1989.

[5] L. Pitt and M. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the Association for Computing Machinery,* 40(1): 95–142, 1993.

[6] E. M. Gold. Complexity of automaton identification from given data. *Inform. Contr.*, 37: 302–320, 1978.

[7] M. Kearns and L. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *J. ACM* 41(1): 67–95, 1994.

[8] D. Angluin. Inference of reversible languages. *J. ACM*, 29(3): 741–765, 1982.

[9] Y. Sakakibara. Efficient learning of context-free grammars from positive structural examples. *Information and Computation*, 97:1, 23–60, 1992.

[10] F. Denis. Learning Regular Languages from Simple Positive Examples. *Machine Learning* 44: 37–66, 2001.

[11] A. Benjamin. The Essential Problem of Empiricism. *Philosophy of Science*, 10(1): 13-17, 1943.

[12] D. Hume. *An Enquiry Concerning Human Understanding.* Clarendon Press, Oxford, U.K., edited by Tom L. Beauchamp, 2000.

[13] A. Fraenkel and D. Lichtenstein. Computing a perfect strategy for $n \times n$ chess requires time exponential in n, *J. Combin. Theory Ser. A, 31 (2)*: 199–214, 1981.

[14] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society.* American Mathematical Society: 117: 285–306, 1965.

[15] E. Rubin, *Synsoplevede Figurer*, 1915.

[16] P. Adriaans. A computational theory of meaning, in *Advances in Info-Metrics: Information and Information Processing across Disciplines.* Editors: Min Chen, Michael Dunn, Amos Golan and Aman Ullah, OUP, 2020.

[17] L. Wittgenstein. Part II, §xi in *Philosophical Investigations*. Blackwell Publishing, 1953.

[18] J. Carlotto. Digital Imagery Analysis of Unusual Martian Surface Features *Applied Optics*. 27 (10): 1926–1933, 1988.

[19] P. Adriaans, via private communication.

[20] Wikipedia contributors. DeepDream. *Wikipedia*, 26 Feb. 2021. Web. 10 Mar. 2021.

[21] M. Davis, R. Sigal and E. Weyuker. *Computability, Complexity, and Languages and Logic: Fundamentals of Theoretical Computer Science* (2nd ed.). San Diego: Academic Press, Harcourt, Brace Company, 1994.

[22] P. Adriaans, C. Jacobs. Using MDL for Grammar Induction. In: Sakakibara Y., Kobayashi S., Sato K., Nishino T., Tomita E. (eds) *Grammatical Inference: Algorithms and Applications*. ICGI 2006. *Lecture Notes in Computer Science*, vol 4201. Springer, Berlin, Heidelberg, 2006.

[23] J. Groote and M. Reniers. Chapter Two: Actions, behaviour, equivalence and abstraction. *Modelling and analysis of communicating systems*. MIT Press, 2014.

[24] B. Lambeau, C. Damas and P. Dupont. State-Merging DFA Induction Algorithms with Mandatory Merge Constraints. *ICGI 2008: Grammatical Inference: Algorithms and Applications*: 139-153, 2008.

[25] M. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*. 3 (2): 114–125, 1956.

[26] J. Hopcroft and J. Ullman. Chapter 3, *Introduction to Automata Theory, Languages, and Computation*, Reading, Massachusetts: Addison-Wesley Publishing, 1979.

[27] Lang, K., Pearlmutter, B., Price, R.: Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In: Honavar, V.G., Slutzki, G. (eds.) *ICGI 1998. LNCS (LNAI)*, vol. 1433: 1–12, 1998.

[28] M. Bugalho, A. L. Oliveira. Inference of regular languages using state merging algorithms with search. *Pattern Recognition*, 38(9): 1457–1467, 2005.

[29] Sebban, M., J. Janodet and Frédéric Tantini. "BLUE∗: a Blue-Fringe Procedure for Learning DFA with Noisy Data."

[30] T. Cover & J. Thomas. *Elements of Information Theory* (2nd ed.). John Wiley & Sons, 2006.

[31] Y. Dulek, C. Schaffner, *Lecture Notes*, 2017: accessed from https://homepages.cwi.nl/ schaffne/courses/inftheory/2017/

[32] B. McMillan. Two inequalities implied by unique decipherability, *IEEE Trans. Inf. Theory*, 2 (4): 115–116, 1956.

[33] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, USA, 2002.

[34] R. J. Solomonoff. A formal theory of inductive inference: Parts 1 and 2. *Information and Control*, 7(1): 224-254, 1964.

[35] Marcus Hutter et al. Algorithmic probability. *Scholarpedia*, 2(8):2572, 2009.

[36] L. A. Levin. Laws of Information Conservation (Nongrowth) and Aspects of the Foundation of Probability Theory, *Problems Inform. Transmission*, 10(3): 206–210, 1974.

[37] P. Grünwald & P. Vitányi. Shannon Information and Kolmogorov Complexity. *ArXiv*, cs.IT/0410002, 2004.

[38] Alan Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, Series 2, Volume 42: 230–265, 1937.

[39] C. Grinstead, M. Snell, J. Laurie. *Introduction to probability* (2nd ed.). Providence, RI: American Mathematical Society, 2006.

[40] F. Chedid. Kolmogorov's Algorithmic Mutual Information Is Equivalent to Bayes' Law. *ArXiv* abs/1907.02943: 1-8, 2019.

[41] T. M. Mitchell. *Machine Learning.* McGraw-Hill, New York, 1997.

[42] W. Wieczorek (2016), Grammatical Inference, *Studies in Computational Intelligence*, 673 eBook.

[43] E. T. Jaynes. Information Theory and Statistical Mechanics *Physical Review. Series II. 106 (4)*: 620–630, 1957.

[44] Example due to P. Adriaans, via private communication.

[45] P. Adriaans, Learning Shallow Context-free Languages under Simple Distributions, Algebras, Diagrams, and Decisions in *Language, Logic and Computation, CSLI/CUP* Anne Copestake and Kees Vermeulen (eds.), 2004.

[46] Example taken from a homework exercise for the Information Theory 2019 course at the ILLC. The related calculation of the topological entropy (page 37), appeared in a slightly different form as part of a collaborative effort between Alexandra Lindt, Fergus Smiles.

[47] Henk Bruin *VO Special Topics in Stochastics: Symbolic Dynamic* (2019W.25050.1)

[48] L. C. Eggan. Transition graphs and the star-height of regular events, *Michigan Mathematical Journal*, 10 (4): 385–397, 1963.

[49] M. Schützenberger. On Finite Monoids Having Only Trivial Subgroups. *Information and Control*. 8 (2): 190–194, 1965.

[50] As an example of polysemanticism, L1 and L2 are due to private communication with P. Adriaans. As an example of fair polysemanticism samples L3 is due to the author.

[51] D. Kozen. David Gries, Fred B. Schneider (ed.). *Automata and Computability. Undergraduate Texts in Computer Science* (1 ed.). New York: Springer-Verlag. pp. 124-126, 1977.

[52] A. Nerode. Linear Automaton Transformations, *Proceedings of the AMS, 9*, 1958.

[53] P.W. Adriaans. The philosophy of learning, the cooperative computational universe. Pdf, in *Handbook of Philisophy of Information*, P.W.Adriaans, J.F.A.K. van Benthem (eds.), Elseviers Science Publishers, 2006

[54] D. W. Thompson. *On Growth and Form*. Dover reprint of 1942 2nd ed. (1st ed., 1917), 1992.

[55] G. Bateson. *Mind and Nature: A Necessary Unity (Advances in Systems Theory, Complexity, and the Human Sciences)*. Hampton Press, 1979.

[56] A. Whitehead. *Process and Reality*, (Gifford Lectures 1927–28), New York: Macmillan. Corrected edition, David Ray Griffin Donald W. Sherburne (eds.), New York: The Free Press, 1985.

[57] H. Bergson *Creative Evolution*, [1911] translated by Arthur Mitchell, New York: Dover, 1998.

[58] David Bohm. *Wholeness and the Implicate Order*, Routledge, 1980.

[59] I. Prigogine, I. Stengers. *Order out of Chaos: Man's new dialogue with nature*. Flamingo, 1984.

[60] P. C. Kanellalds and S. A. Smolka. CCS Expressions, Finite State Processes, and Three Problems of Equivalence. *Information and Computation*, volume 86, 1990.

[61] J. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton, *Theory of machines and computations (Proc. Internat. Sympos., Technion, Haifa, 1971)*, New York: Academic Press: 189–196, 1971.

[62] Kac, M. Can One Hear the Shape of a Drum? *American Mathematical Monthly*. 73(4, part 2): 1–23, 1966.

[63] C. Gordon, D. Webb. You can't hear the shape of a drum, *American Scientist*, 84 (January–February): 46–55, 1996.

[64] B. Balle. Ergodicity of Random Walks on Random DFA. *ArXiv*, abs/1311.6830, 2013.

[65] A. Grusho. Limit distributions of certain characteristics of random automaton graphs. *Mathematical Notes*, 1973.

[66] Donald Ornstein. Bernoulli shifts with the same entropy are isomorphic, *Advances in Math.* 4: 337–352, 1970.

[67] K. Petersen. *Ergodic Theory (Cambridge Studies in Advanced Mathematics)*. Cambridge: Cambridge University Press (Ed 1.) p.59, 1983.

[68] N. Friedman and D. Ornstein. On isomorphism of weak Bernoulli transformations. *Advances in Math.*, 5: 365–394, 1970.

[69] N. Jonoska. Sofic shifts with synchronizing presentations. *Theor. Comput. Sci.* 158(1–2): 81–115, 1996.

[70] Blackburn, P., de Rijke M., Venema, Y. *Modal Logic. Cambridge Tracks in Theoretical Computer Science 53* Cambridge University Press, Cambridge 2001.

[71] Larsen, K., & Arne Skou. Bisimulation through probabilistic testing. *Inf. Comput.* 94, 1 (Sept. 1991): 1–28, 1991.

[72] N. Trčka. Strong, weak and branching bisimulation for transition systems and Markov reward chains: a unifying matrix approach. In: *Proc. first workshop on quantitative formal methods: theory and applications*, EPTCS, vol 13: 55–65, 2009.

[73] Katz, M., Hoffmann, J. & Malte Helmert. How to relax a bisimulation? In *Proceedings of the Twenty-Second International Conference on International Conference on Automated Planning and Scheduling* (ICAPS'12). AAAI Press: 101–109, 2012.

[74] Tran, Thanh-Luong Nguyen, Linh Hoang, Thi-Lan-Giao. Bisimulation-based concept learning for information systems in description logics. *Vietnam Journal of Computer Science.*, 2015.

[75] Nissim, Raz Hoffmann, Jörg Helmert, Malte. "Computing Perfect Heuristics in Polynomial Time: On Bisimulation and Merge-and-Shrink Abstraction in Optimal Planning". *IJCAI International Joint Conference on Artificial Intelligence*: 1983-1990, 2011.

[76] Bunte, Olav, J. Groote, J. Keiren, Maurice Laveaux, Thomas Neele, E. Vink, W. Wesselink, Anton Wijs and T. Willemse. "The mCRL2 Toolset for Analysing Concurrent Systems - Improvements in Expressivity and Usability." *TACAS* (2019).

[77] J. Engelfriet. Determinacy → (Observation Equivalence = Trace Equivalence). *Theor. Comput. Sci.* 36: 21-25, 1985.