Geometric Perspectives on Program Synthesis and Semantics

By

Thomas Waring

Supervised by
Daniel Murfet

Submitted In Partial Fulfilment
of the Requirements for the Degree
Master of Science in the
School of Mathematics and Statistics

University of Melbourne

October 2021

Abstract

We develop a formalism to apply Watanabe's Singular Learning Theory to the problem of program synthesis. In our case, a program is a sequence of letters on the tape of a Turing Machine, which we associate to a singularity of an analytic function. The key invariant associated to a singularity in this context is the real log canonical threshold, and we introduce methods to compute this value from an ideal generated by polynomials. We provide a semantic interpretation of these singularities, which distinguishes distinct algorithms by viewing them as the limit of a learning process.

Acknowledgements

Thanks first and always to Daniel Murfet, who is responsible for most of the ideas and much of the presentation. Thanks to early readers and everybody who nodded sagely while I talked about Turing Machines (no Bryn, Turing didn't actually build one). Thank you MK Tandon: your support and example means more than you know.

I live and work on unceded land belonging to the Wurundjeri people of the Kulin nation.

Contents

1 Introduction

The central object of study in theoretical computer science is the *program*, a finite set of instructions which effectively compute a function. There may, however, be many different programs which compute the same function: even for something as elementary as integer multiplication, several different algorithms are known [Ber01]. In many cases, the *ideas* of these algorithms are distinct, and it is far from clear how to capture this semantic difference mathematically. The *syntax*, or lines of code, certainly differs from algorithm to algorithm, but these differences are too numerous to pin down the difference in meaning. Logician Dana Scott addresses this in [Sco77]:

> In giving precise definitions of operational semantics there are always to be made more or less arbitrary choices ... and to a great extent these choices are irrelevant for a true "understanding" of a program. Mathematical semantics tries to avoid these irrelevancies and should be more suitable to the study of such problems as the equivalence of programs.

The contention of this thesis is that in order to understand semantic differences between programs, we ought to examine how they come about. Conventionally, a *construction* of a program is understood to be a step-by-step recipe which assembles a program from simple units [GLT93, §3.3]. Following Turing, we take a different view, in which programs are the limit of a *learning process* that gradually tunes an initially unorganised machine towards an organised result [Tur04]. By working towards an understanding of this learning process, we endeavour to enrich our understanding of programs themselves.

We can formalise the idea of of learning programs, or *program synthesis*, in the language of inductive inference [Hut04, §2.1]. If our system of computation evaluates functions $S \to T$, and we have some set $\mathscr{C}$ of possible programs, then there is a natural evaluation map

$$\mathscr{C} \longrightarrow \mathcal{P}(S \times T),$$

taking each program F to its graph $\Gamma_F = \{(x, Fx) \mid x \in S\}$, as an element of the power set. Program synthesis, or more specifically *Programming by Example*[1] [Lib01], looks to go the other way. Given a set $G \subset S \times T$ of input/output pairs, we search for a program F which matches this specification in the sense that $G \subset \Gamma_F$.

This thesis provides a mathematical framework for understanding programs as the limit of a learning process, using the analytic geometry of Watanabe's Singular Learning Theory (SLT) [Wat09]. In particular, we discover a natural analogue of machine education, as described by Turing in [Tur04]. The main technical result is Theorem 4.24, which links the analytic framework to algebraic geometry. This result demonstrates that, for a natural class of synthesis problems P, which we call *compact*, the learning process is strongly controlled by a polynomial ideal I_P, called the *fibre ideal* [Lin11, §1.5.1], inside the ring of analytic functions on the space of possible programs.

As quoted above, Scott aimed to provide a mathematical interpretation of programs, with the goal of abstracting from the "irrelevancies" of operational semantics, in order to understand

[1] It should be observed that this is not the only form that program synthesis can take; it does not include, for example, synthesis of programs from natural language inputs [GM14, MGA13].

the fundamental properties of a given algorithm. To do this, he axiomatises data types as complete lattices: the ordering, $x \sqsubseteq y$, encodes a refinement of the data x to more specific data y. We discover in Section 5 exactly this structure on the set $\mathcal{I}(M)$ of synthesis problems on a given machine M, which allows us to use Scott's work to provide a semantic interpretation of learning. To extend this to programs themselves, we define a function $[\![-]\!] : \mathscr{C} \to \mathcal{I}(M)$ which embeds codes on a Universal Turing Machine M into the lattice. In particular, given a program $\mathbf{w} \in \mathscr{C}$, we can specify the associated problem $[\![\mathbf{w}]\!] \in \mathcal{I}(M)$ as the directed limit of compact problems (Proposition 5.15), to which we can apply Theorem 4.24.

The lattice structure on $\mathcal{I}(M)$ mirrors the lattice of ideals I_P, and via SLT its geometric content controls the process of learning. Using this, we can find a novel interpretation of the limits that are central to Scott's vision of semantics. If a code $\mathbf{w}$ is approximated by compact synthesis problems

$$P_0 \sqsubseteq P_1 \sqsubseteq P_2 \sqsubseteq \cdots \sqsubseteq [\![\mathbf{w}]\!], \tag{1}$$

then the varying geometry of the ideals I_{P_i} encodes the increasing complexity of the solution (Proposition 4.19 and Remark 5.19).

To understand programs — say to distinguish semantically different programs, or identify equivalent ones — we look for ways of modelling them in areas of mathematics that we understand better, such as geometry. For this to be useful, the semantics must expose information about the original object in a more accessible way. This thesis gives some initial steps in the direction of a geometric semantics of arbitrary programs.

The rest of this section lays out the concepts which we need to make this story work, and in that way serves as an outline for the rest of the thesis. In brief, Sections 2 and 3 explain the necessary background on programs and SLT, and Section 4 uses this to formalise learning as a problem of inductive inference. Section 5 examines the higher level structure of the theory we develop, which will help to interpret the semantic information packaged by the geometry of program synthesis.

1.1 Outline

In Section 2 we lay out the model of computation which underlies our treatment. Clearly, the specifics of program synthesis depend on the model of computation used, as this controls the space of possible programs. Frequently, it is desirable to work in a very restricted model of computation, an extreme example of which is machine learning: there is a sense in which the search for the correct weights on a neural net is a search for a program, just in a very limited language [GSLT+18, §3.2]. Similarly, even if one is trying to generate code in `C` or `Haskell`, it can be productive to limit the space of possible programs to those involving manipulations relevant to the problem at hand [GHS12]. A priori, the approach we take is different. For us, the space of "programs" will be strings on the input tape of a Turing Machine (TM), which, according to the Church-Turing thesis, potentially encompasses any *effectively calculable* function [Tur39, Ros39]. It should be stressed that we are *not* considering synthesis of the Turing Machine itself (as represented by its transition function), but rather the contents of the tape of a fixed TM. Since there exists a *Universal Turing Machine* (UTM) which simulates any given TM via a code on its tape, the former problem reduces to the latter [HMU01, §9.2.3].

This model of computation has a few advantages. Firstly, we choose TMs over more specific (and potentially practical) models of computation, as they are universal. Any problem which is

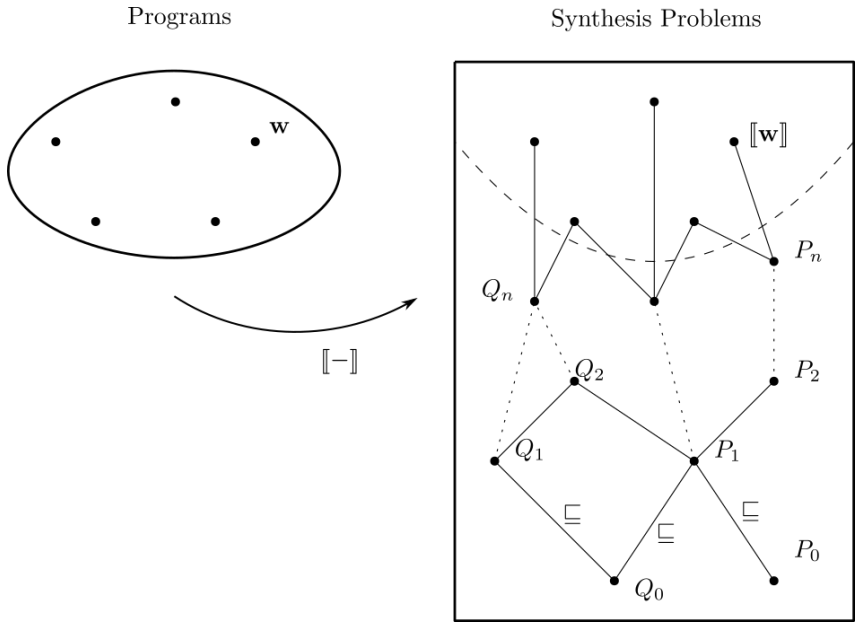

Figure 1: A representation of the map $[\![-]\!] : \mathscr{C} \to \mathcal{I}(M)$, defined in Definition 5.16. The lattice $\sqsubseteq$, on the right, gives structure to the programs, and allows us to approximate them by inference problems, as in (1).

soluble by computation may be solved on a TM [Tur39], so any sufficiently general theoretical result proven or explored in this case extends to more practically interesting languages. This will allow us to discover structure which we claim is *not* dependent on the specific model. Secondly, TMs are simpler to apprehend mathematically than the full complexity of a programming language in the more everyday sense of the term. They are specified by fairly minimal data, and their running can be (relatively) easily simulated by hand. Finally, as contrasted with even simpler models such as the lambda calculus, TMs correspond very closely to our intuitive idea of computation: this, in fact, motivated their introduction, which was preceded by other equivalent definitions [Tur39, p.8]. Using an intuitively comprehensible model will make the logical structure of the resulting theory more clear.

Having settled on Turing Machines, program synthesis looks like a discrete problem. However, with reference to our intuitive understanding of *learning*, it is desirable to allow for small (even infinitesimal) variations in programs. Our solution to this problem is to allow for uncertainty in the configuration of the Turing Machine at a fixed time, following [CMW21]. For example, each square on the tape will hold a *distribution* over the tape alphabet Σ, rather than a definite letter. The details of this extension will be discussed in Section 2.2.

This innovation turns the problem of program synthesis into one of statistical inference: from knowledge of some true distribution $q(x)$, corresponding to the given input-output specification, we seek a matching distribution among some family $p(x|\mathbf{w})$, parametrised by possible programs $\mathbf{w} \in \mathcal{W}$. A key observation is that we cannot assume this statistical model p is *regular*. In particular, the map p from programs $\mathbf{w}$ to distributions $p(x|\mathbf{w})$ will not be injective. Apart from the fact that this is verifiably the case in real-world machine learning problems, it accords well with our intuitions about learning and program synthesis. We expect that there may be

qualitatively different ways of computing the same function, and a theory of program synthesis which does not account for this will not be able to answer important questions about the learning process. The right framework to deal with this fact is Sumio Watanabe's *Singular Learning Theory* (SLT), as laid out in [Wat09]. In Section 3 we will detail the necessary elements of Watanabe's theory, focussing particularly on the role of the *real log canonical threshold* (RLCT), a geometric invariant which controls the learning process over large data sets.

In Section 4, we formulate statistical inference on Turing Machines, and provide examples demonstrating that this is relevant to program synthesis. This will lead us to define the *fibre ideal* of an inference problem, which links program synthesis to algebraic geometry. Given a learning problem, we can associate to a true parameter $\mathbf{w}$ — for which $p(x|\mathbf{w}) = q(x)$ — a local RLCT. These values package information about the geometry of the learning problem around $\mathbf{w}$. In the case of program synthesis, it is from there that we obtain semantic information about programs via geometry. For example, codes $\mathbf{w}$ with smaller local RLCT will be preferred by a Bayesian learning machine, a result which is the smooth analogue of Occam's razor [CMW21, Bal97].

Section 5 defines the lattice of inference problems, and interprets it as a semantics of programs. Using our results about the fibre ideal, we explain the way in which this lattice complements the geometric information we discovered in the previous section. In particular, we interpret directed limits in our lattice as a version of machine education, as described by Turing in [Tur04]. We show in Appendix A that this limiting process enacts a sequence of second-order phase transitions.

Notation	Explanation	Reference
$M = (\Sigma, Q, \delta)$	A Turing Machine M with tape alphabet Σ, states Q and transition function δ.	Definition 2.1
$\Sigma^{U,\square}$	Sequences of elements of Σ, indexed by U and with only $\square$ appearing infinitely often	Definition 2.1
$\mathcal{U}$	A pseudo-Universal Turing Machine. Inputs are on tape squares U, codes on V.	Definition 2.3
ΔZ	The standard simplex over a set Z.	Definition 2.5
$(\Delta\Sigma)^{U,\square}$	Sequences of distributions over Σ, indexed by U and with only $\square$ appearing infinitely often	Definition 2.7
Δstep_M	The smooth relaxation of the step function associated to a TM M.	Definition 2.8
(q, p, φ)	Singular Learning Triple: true distribution $q(x)$, model $p(x\|w)$, prior distribution $\varphi(w)$.	Setup 3.1
$\mathcal{W}_0 \subset \mathcal{W}$	The subset $\mathcal{W}_0$, inside the code space $\mathcal{W}$, of solutions: $p(x\|w) = q(x)$.	Setup 3.1
$D_{\text{KL}}(q \,\|\| \, p)$	The Kullback-Leibler divergence between distributions q, p.	Definition 3.5
$\text{RLCT}_{\mathcal{W}}(f; \varphi)$	The real log canonical threshold of a phase function f with the prior φ.	Definition 3.9
$\text{RLCT}_{\mathcal{W}}(I; \varphi)$	The real log canonical threshold of an ideal I inside the ring of analytic functions $\mathcal{A}_{\mathcal{W}}$ on $\mathcal{W}$, using the prior φ.	Definition 3.18
$\mathcal{S}$	The possible constraints for an inference problem.	Definition 4.1
$\mathscr{C}(V, c)$	The code space over V with allowed codes c.	Definition 4.2
$\mathfrak{c}(P)$	The Kolmogorov complexity of a synthesis problem P.	Definition 4.18
$\mathcal{S}_+(P)$	The support of an inference problem P.	Definition 4.6
$I_P \subset \mathcal{A}_{\mathcal{W}}$	The fibre ideal of an inference problem P, inside the ring of analytic functions on $\mathcal{W}$.	Definition 4.23
$\mathcal{I}(M, V, c)$	The set of inference problems on a Turing Machine M, with fixed code window V and allowed codes c.	Definition 5.1
$\bot, \top$	Overspecified and underspecified inference problems.	Definition 5.1
$P \sqsubseteq P'$	The inference problem P specialises to P'.	Definition 5.2
$P + Q$	The sum of inference problems P and Q.	Definition 5.7
$P \cap Q$	The intersection of inference problems P and Q.	Definition 5.8
$[\![\mathbf{w}]\!]$	The inference problem associated to a code $\mathbf{w} \in \mathscr{C}(V, c)$	Definition 5.16

2 Turing Machines

This section fixes our definitions and conventions for our chosen model of computation: single-tape Turing Machines. Section 2.1 deals with classical Turing Machines, and Section 2.2 outlines the process of smooth relaxation.

2.1 Conventional Turing machines

This section endeavours to be self-contained, but for a detailed treatment of this topic see [HMU01, §8-9], noting small differences in conventions. In particular, [HMU01, §8.3] demonstrates how familiar programming techniques are translated into the language of TMs. Loosely, a Turing Machine is a computer with a one-dimensional tape for memory, and finitely many internal states. Each "tape square" holds one of a finite collection of letters, and the "head" reads one of the tape squares at a time. Depending on the internal state and the square under the head, the TM writes a new letter, and possibly changes the state and/or moves to the left or right. More formally:

Definition 2.1. A (single-tape) *Turing Machine* $M = (\Sigma, Q, \delta)$ is a tuple consisting of a finite set Σ, called the tape alphabet, a finite set Q of states, and a transition function:

$$\delta : \Sigma \times Q \longrightarrow \Sigma \times Q \times \{-1, 0, 1\}.$$

If $\delta(\sigma, q) = (\sigma', q', d)$, then if the machine is in state q and reads the letter σ, it will write σ' to the tape, transition into state q', and move according to d: $d = 1$ indicates "move right", $d = -1$ "move left" and $d = 0$ stay still. We will often write δ_i for the component $\pi_i \circ \delta$.

We assume there is a specified blank symbol $\square \in \Sigma$, which is the only symbol allowed to appear infinitely many times on the tape. Therefore, the state of the machine at a fixed time is specified by a pair (y, q), with $q \in Q$ and:

$$y \in \Sigma^{\mathbb{Z},\square} := \{(y_i)_{i\in\mathbb{Z}} \mid y_i = \square \text{ for all but finitely many } i\}$$

The tape squares are numbered relative to the head: y_0 is the symbol being read, y_1 immediately to the right, and so on. As such, if (y, q) is the state at time t, then after one step of the machine, the new state is:

$$\text{step}\big((y_i)_{i\in\mathbb{Z}}, q\big) = \big((y'_{i+d})_{i\in\mathbb{Z}}, q'\big),$$

where $\delta(y_0, q) = (y'_0, q', d)$, and $y'_i = y_i$ for $i \neq 0$. Among the states Q, we will often fix specified initial and halting states, denoted $\underline{\text{init}}$ and $\underline{\text{halt}}$, and force that:

$$\delta(x, \underline{\text{halt}}) = (x, \underline{\text{halt}}, 0).$$

Example 2.2 (The Shift Machine)**.** The following example, which we will return to throughout, is taken from [CM19, Section 7.2]. The Turing Machine M has alphabet and states:

$$\Sigma = \{\square, A, B, 0, \ldots, 9\}$$
$$Q = \{\underline{\text{init}}, \underline{\text{halt}}, \text{goR}, \text{goLA}, \text{goLB}\},$$

and transition function:

$$\begin{aligned}
\delta(\sigma, \underline{\text{halt}}) &= (\sigma, \underline{\text{halt}}, 0) && \\
\delta(n, \underline{\text{init}}) &= (n-1, \text{goR}, 1) && 0 < n \leq 9 \\
\delta(0, \underline{\text{init}}) &= (0, \underline{\text{halt}}, 0) && \\
\delta(\sigma, \text{goR}) &= (\sigma, \text{goR}, 1) && \sigma \in \{A, B\} \\
\delta(n, \text{goR}) &= (n, \underline{\text{halt}}, 0) && 0 \leq n \leq 9 \\
\delta(\square, \text{goR}) &= (\square, \text{goLA}, -1) && \\
\delta(\sigma, \text{goL}\sigma') &= (\sigma', \text{goL}\sigma, -1) && \sigma, \sigma' \in \{A, B\} \\
\delta(n, \text{goL}\sigma) &= (n-1, \text{goR}, 1) && \sigma \in \{A, B\}, 1 \leq n \leq 9 \\
\delta(0, \text{goL}\sigma) &= (0, \underline{\text{halt}}, 0) && \sigma \in \{A, B\}.
\end{aligned}$$

Informally, initialised to a tape configuration consisting of a counter n, and some string of A's and B's, the machine moves the string left by n squares, padding with A's at the right-hand end. For example, initialised to the following, with the underline indicating head position:

$$\ldots \square \underline{2}\, A\, B\, A\, A\, B\, B\, A\, B \square \ldots$$

the machine will eventually halt in the configuration:

$$\ldots \square \underline{0}\, A\, A\, B\, B\, A\, B\, A\, A \square \ldots$$

This specification leaves some components of δ undefined — for example, if the machine is in state goR, it will never read an integer n under the head. However, we will see in Example 2.11 that the behaviour of the smooth relaxation depends on these choices. We have set the machine to halt in these cases, to ensure that for large enough t the machine is in state $\underline{\text{halt}}$ with probability 1.

The example that allows us to view codes on a TM as a programming language is the following.

Definition 2.3. A *pseudo-Universal Turing Machine* (pseudo-UTM) is the data:

- A *simulation alphabet* Σ and *simulation states* Q, and an ordering on $\Sigma \times Q$;
- a Turing Machine $\mathcal{U} = (\Sigma_{\mathcal{U}}, Q_{\mathcal{U}}, \delta_{\mathcal{U}})$, with $\Sigma \cup Q \cup \{-1, 0, 1\} \subset \Sigma_{\mathcal{U}}$;
- two disjoint subsets $V, U \subset \mathbb{Z}$, called the code and input windows, with $|V| = 3 \cdot |\Sigma| \cdot |Q|$, and an indexing (bijection) $u : U \to \mathbb{Z}$,
- and a *background initialisation* $z \in (\Sigma_{\mathcal{U}})^{U', \square}$, where $U' = \mathbb{Z} \setminus (V \cup U)$.

We define $N = |\Sigma|$ and $M = |Q|$. Of this Turing Machine we require the following. Given a sequence:

$$w \in (\Sigma \times Q \times \{-1, 0, 1\})^{MN} \subset \Sigma^V,$$

of the form:

$$\sigma_1\, q_1\, d_1\, \ldots\, \sigma_{MN}\, q_{MN}\, d_{MN},$$

we use the ordering on $\Sigma \times Q$ to define a function $\delta_w : \Sigma \times Q \to \Sigma \times Q \times \{-1, 0, 1\}$, reading from left to right. The triple (Σ, Q, δ_w) defines a Turing Machine M_w, and if the tape regions V and U are initialised to w and $x \in \Sigma^{\mathbb{Z},\square}$, and U' to z, we require that $\mathcal{U}$ have the same halting behaviour as M_w. That is, if M_w halts on the input x, $\mathcal{U}$ should to, and the tape squares U should contain the halting tape configuration of M_w. Note that the tape configuration of $\mathcal{U}$ is determined by the indexing u: if we have $(x_i)_{i\in\mathbb{Z}} \in \Sigma^{\mathbb{Z},\square}$, then the corresponding sequence in $\Sigma^{U,\square}$ is $(x_{u(j)})_{j\in U}$.

Remark 2.4. We will usually identify the data of a pseudo-UTM with the Turing Machine $\mathcal{U}$. Many constructions of such a machine exist, for example see [CMW21, Appendix E]. Such a machine is not a genuine UTM, as, in the manner described at least, it can only simulate machines with alphabets and states contained in Σ and Q. Briefly, in the general case, a UTM might simulate triples of (M, e_1, e_2), where $M = (\Sigma, Q, \delta)$ is a TM, and e_1 and e_2 are encodings of Σ and Q into the tape alphabet of $\mathcal{U}$. [HMU01, §9.2.3] outlines the construction of such a machine in detail, encoding the i^{th} tape letter of M by i zeros followed by 1. We will not consider this case here, as the behaviour of the smooth relaxation of such a machine is unclear [Xu21, §4].

2.2 Smooth relaxation

As discussed in the introduction, we want to use Turing Machines as a basis for our model of the concept of learning, which means we want to allow for small variations in programs. As such, we need to choose a *smooth relaxation* of the discrete definition in the previous section.

The approach we take follows [CMW21] in using the smooth relaxation of [CM20, CM19]. Each square of the tape will hold a probability distribution over the tape letters, and the internal state will be a distribution over the states Q. This is a natural choice: [GBS+16] generalises the version of this technique appearing in [KAS16] to synthesise programs by gradient descent, and [EG18, §4] describes a version of this for logic programming. This issue is discussed in more detail in Remark 2.10.

Probability distributions will be represented as vectors in Euclidean space, and the following definitions fix our notation for these distributions.

Definition 2.5. Given a set Z, the *standard simplex* over Z is the collection:

$$\Delta Z = \{f : Z \to [0, 1] \mid \sum_{z\in Z} f(z) = 1 \text{ and } f(z) = 0 \text{ for all but finitely many } z\}.$$

When $Z = \{0, \ldots, n\}$, we abbreviate $\Delta Z = \Delta^n$. We embed Z in ΔZ by defining, for an element $z \in Z$, $z(z) = 1$, and $z(z') = 0$ otherwise. ΔZ is a subset of the free vector space $\mathbb{R}Z$ with basis Z.

Definition 2.6. Given a set Z and non-empty subset $Z' \subset Z$, we embed $\Delta Z' \subset \Delta Z$ as:

$$\{f \in \Delta Z \mid z \notin Z' \implies f(z) = 0\}.$$

This is referred to as the Z' *face* of ΔZ. If Z' is finite, the interior of the Z' face is the image of the interior of $\Delta Z'$ in ΔZ. That is, the set:

$$\{f \in \Delta Z \mid \forall z \in Z((f(z) = 0 \wedge z \notin Z') \vee (f(z) > 0))\}.$$

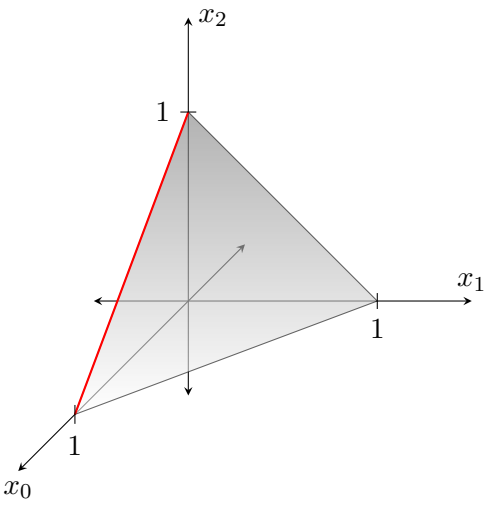

Figure 2: The standard simplex $\Delta\{0,1,2\} = \Delta^2 \subset \mathbb{R}^3$, with the $\{0,2\}$-face marked in red.

The methods of [CM20, CM19] define a particular approach for updating uncertainties in the configuration of a TM, as the machine is run. In particular, samples from the tape squares and states will be conditionally independent (in a sense we shall make precise in Remark 2.9). To emphasise this, we will refer to a configuration of our relaxed TM as the state of belief for a *naive Bayesian observer* (see [CM19, §1]), to differentiate it from a more standard probabilistic approach. The following definitions fix our notation for these states of belief, and define the function which updates them in response to a time step of the machine.

Definition 2.7. At a fixed time, the *state of belief* of a naive Bayesian observer to the running of a Turing Machine $M = (\Sigma, Q, \delta)$, is an element of $(\Delta\Sigma)^{\mathbb{Z},\square} \times \Delta Q$, where we define:

$$(\Delta\Sigma)^{\mathbb{Z},\square} = \{g : \mathbb{Z} \to \Delta\Sigma \mid g(i) = \square \text{ for all but finitely many } i\}.$$

We also define projections, π^{state} and π_a^{tape} onto the state and a^{th} tape squares:

$$\pi^{\text{state}}\left((\mathbf{y}_i)_{i\in\mathbb{Z}}, \mathbf{q}\right) = \mathbf{q}$$
$$\pi_a^{\text{tape}}\left((\mathbf{y}_i)_{i\in\mathbb{Z}}, \mathbf{q}\right) = \mathbf{y}_a.$$

Definition 2.8. The *smooth relaxation of step* for a Turing Machine $M = (\Sigma, Q, \delta)$, is the function:

$$\Delta\text{step}_M : (\Delta\Sigma)^{\mathbb{Z},\square} \times \Delta Q \longrightarrow (\Delta\Sigma)^{\mathbb{Z},\square} \times \Delta Q,$$

defined by $\Delta\text{step}_M\big((\mathbf{y}_i)_{i\in\mathbb{Z}}, \mathbf{q}\big) = \big((\mathbf{y}'_i)_{i\in\mathbb{Z}}, \mathbf{q}'\big)$, and:

$$\mathbf{y}'_i(\sigma) = \sum_{\sigma^w, q^w} \sum_{\sigma^d, q^d} \mathbf{y}_0(\sigma^w)\mathbf{q}(q^w)\mathbf{y}_0(\sigma^d)\mathbf{q}(q^d)\big\{\mathbb{1}[i \neq -d]\mathbf{y}_{i+d}(\sigma) + \mathbb{1}[i = -d]\mathbb{1}[\sigma = \delta_1(\sigma^w, q^w)]\big\} \tag{2}$$

$$\mathbf{q}'(q) = \sum_{\sigma^s, q^s} \mathbf{y}_0(\sigma^s)\mathbf{q}(q^s)\mathbb{1}[q = \delta_2(\sigma^s, q^s)] \tag{3}$$

where in the first formula we set $d = \delta_3(\sigma^d, q^d)$ in each summand.

Remark 2.9. Loosely, the above formulae correspond to the following method for propagating uncertainty through the running of the machine. At each time step, the machine makes three decisions, corresponding to the three components of δ. Corresponding to these decisions, a naive Bayesian observer takes three independent samples each from the distributions $\mathbf{y}_0 \in \Delta\Sigma$ and $\mathbf{q} \in \Delta Q$, and calculates the action of the Turing Machine accordingly. Here, by action, we mean a tuple $(\sigma, q, d) \in \Sigma \times Q \times \{-1, 0, 1\}$, so that the machine writes σ to the square under the head, updates its internal state to q, and moves in direction d. We can express the new distribution as a sum over the outcomes of these samples, weighted by the probability of that sample coming up (using the initial distribution).

More precisely, we have presented the smooth relaxation map Δstep_M as the restriction of a polynomial:

$$\Delta\text{step}_M : (\mathbb{R}^\Sigma)^\mathbb{Z} \times \mathbb{R}^Q \longrightarrow (\mathbb{R}^\Sigma)^\mathbb{Z} \times \mathbb{R}^Q. \tag{4}$$

Define $P = (\Sigma \times Q)^3$ to be the collection of possible outcomes for the samples described above. Examining (2) and (3), we can rewrite our polynomial as:

$$\Delta\text{step}_M = \sum_{\sigma^w, q^w} \sum_{\sigma^s, q^s} \sum_{\sigma^d, q^d} \mathbf{y}_0(\sigma^w)\mathbf{q}(q^w)\mathbf{y}_0(\sigma^s)\mathbf{q}(q^s)\mathbf{y}_0(\sigma^d)\mathbf{q}(q^d) f_p, \tag{5}$$

where $(f_p)_{p \in P}$ is a family of polynomial functions $(\mathbb{R}^\Sigma)^\mathbb{Z} \times \mathbb{R}^Q \longrightarrow (\mathbb{R}^\Sigma)^\mathbb{Z} \times \mathbb{R}^Q$, indexed by the possible samples $p = (\sigma^w, \sigma^s, \sigma^d, q^w, q^s, q^d)$. So far all we have done to the first equation is to add in a sum over σ^s, q^s, and the terms $\mathbf{y}_0(\sigma^s)\mathbf{q}(q^s)$ to each summand (likewise for the second). Each component of f_p is simple: it is either $0, 1$ or $\mathbf{y}_i(\sigma)$ for some $(\sigma, i) \in \Sigma \times \mathbb{Z}$. Specifically, if for $\sigma \in \Sigma$, $i \in \mathbb{Z}$ and $q \in Q$ we write $\pi_{\sigma,i}$ and π_q for the projections out of the codomain of (4), we have the formulae (with $d = \delta_3(\sigma^d, q^d)$):

$$\pi_{\sigma,i} \circ f_p = \begin{cases} 0 & i = -d,\ \sigma \neq \delta_1(\sigma^w, q^w) \\ 1 & i = -d,\ \sigma = \delta_1(\sigma^w, q^w) \\ \mathbf{y}_{i+d}(\sigma) & i \neq -d \end{cases}$$

$$\pi_q \circ f_p = \begin{cases} 0 & q \neq \delta_2(\sigma^s, q^s) \\ 1 & q = \delta_2(\sigma^s, q^s) \end{cases}$$

We can view (5) as a sum over "paths" $p \in P$, where each summand is the probability of executing that path (taking the given sample p), multiplied by a polynomial version of the classical update rules for a TM. The important fact here is that each summand is a monomial, and the same goes for Δstep_M^t (though the summands are not as easily interpreted).

Remark 2.10 (Why this smooth relaxation?)**.** With this intuitive understanding of our formulae in hand, the question remains as to why we should care about the naive Bayesian observer. We will answer this by contrasting the method used here with a more obvious, but less useful, version, which we will refer to as the *standard probability approach*. For further analysis see [CM19, §5].

Speaking loosely, we define our alternate smooth relaxation $\Delta^{\text{std}}\text{step}_M$ by replacing our conditionally independent samples (as in the previous remark), with samples according to the usual rules of Bayesian probability. Letting the possible states of M be $Z := \Sigma^{\mathbb{Z},\square} \times Q$, we define

for $\mathbf{z} \in \Delta Z$:

$$\Delta^{\text{std}}\text{step}_M(\mathbf{z}) = \sum_{z \in Z} \mathbf{z}(z) \cdot \text{step}_M(z).$$

This is more intuitive: the state after one time step is the superposition of all of the classical update rules, weighted by the probability M is in that state. However, this version of smooth relaxation is less well adapted to *learning*, or statistical inference, than the process we have outlined in this section.

Suppose our initial belief is $z = \big((y_i)_{i \in \mathbb{Z}}, q\big) \in Z$, and we want to vary z so there is a small chance that the state is q'. Then we might set $\mathbf{z}' = \big((y_i)_{i \in \mathbb{Z}}, (1-\epsilon) \cdot q + \epsilon \cdot q'\big)$. If we use the standard probabilistic extension to update the belief $\mathbf{z}'$, then after one time step we have:

$$\Delta^{\text{std}}\text{step}_M(\mathbf{z}') = (1-\epsilon) \cdot \text{step}_M\big((y_i)_{i \in \mathbb{Z}}, q\big) + \epsilon \cdot \text{step}_M\big((y_i)_{i \in \mathbb{Z}}, q'\big). \tag{6}$$

As such, the standard probabilistic extension doesn't offer any incremental benefit over computing the evolution of the Turing Machine over all possible states. On the other hand, in the naive approach we take multiple independent samples from the state distribution, so the new state isn't a straightforward superposition. The consequence of this is that it is useful to vary parts of the belief in question, say corresponding to different tape squares, independently.

The perspective of [CM19] validates this intuition. In [CM20] the smooth relaxation is derived by encoding Turing Machines as proofs in Linear Logic, and applying a certain semantics of Linear Logic to these proofs. The details will not be important for us here, but this embedding induces more structure than the smooth relaxation alone. Namely, it makes sense to take algebraic derivatives of proofs, and as such of Turing Machines, using the structure of Differential Linear Logic [Ehr16].

In the situation we have been describing, we will *learn* programs by nudging them slightly towards the behaviour we want. These derivatives correspond, under semantics, to precisely this kind of small variation in the smooth relaxation.

There are two main practical consequences of this. The first is that computation using the standard approach is far more time-intensive — each state in the superposition of (6) must be computed individually. By contrast, the learning process using the naive probabilistic extension is polynomial time [CM19, Proposition 7.15] (specifically, evaluation of the *loss*, corresponding to the KL divergence of Definition 4.9, can be done in polynomial time). Secondly, as in Remark 7.8 of *loc. cit.*, if we treat the tape squares as independent, it makes sense to ask how much a certain "bit" (if $\Sigma = \{0, 1\}$) is used. The naive probabilistic extension assigns a higher degree (in the polynomial) to bits that are used more frequently, which implies that (as described in Sections 7.2 and 7.3 of *loc. cit.*) gradient descent on the loss will change more frequently used bits more quickly. We can interpret this as showing that a learning machine will tend to look for a *minimal* explanation of an observation.

Example 2.11. We can now calculate the polynomials associated to the smooth relaxation of the Shift Machine described in Example 2.2. Suppose an observer to the running of our Shift Machine M is initially uncertain about whether the counter, under the head, is 0 or 2, and whether the first tape square to the right is A or B. This corresponds, say, to the distributions:

$$\begin{aligned} \nu_{\text{counter}} &= (1-h) \cdot 0 + h \cdot 2 \\ \nu_{\text{string}} &= (1-k) \cdot B + k \cdot A \end{aligned} \tag{7}$$

for $(h, k) \in [0, 1]^2$. If the rest of the tape is known for sure, then we can represent the situation as:

$$\ldots\ x_{-1}\ \underline{\nu_{\text{counter}}}\ \nu_{\text{string}}\ x_2\ x_3\ \ldots$$

for some $x_i \in \Sigma$. This corresponds to the state of belief $((\mathbf{y}_i)_{i\in\mathbb{Z}}, \mathbf{q})$, with $\mathbf{q} = 1 \cdot \underline{\text{init}}$ and:

$$\mathbf{y}_i = \begin{cases} \nu_{\text{counter}} & i = 0 \\ \nu_{\text{string}} & i = 1 \\ 1 \cdot x_i & \text{else} \end{cases}$$

Using the smooth relaxation, we can update this state of belief as the machine runs. Let us assume for simplicity that $x_i = \square$ for $i \notin [0, 3]$. The only non-zero summands in (3) are $(\sigma^s, q^s) \in \{(0, \underline{\text{init}}), (2, \underline{\text{init}})\}$, so after one time step the state distribution is:

$$\mathbf{q}(1) = (1 - h) \cdot \underline{\text{halt}} + h \cdot \text{goR}.$$

Corresponding to the same two possibilities, the machine will either write 0 or 1 to the tape, and will stay still or move right. This gives us the following:

$$\begin{aligned}
\mathbf{y}_{-1}(1) &= (1 - h) \cdot \square + h(1 - h) \cdot 0 + h^2 \cdot 1 \\
\mathbf{y}_0(1) &= (1 - h)^2 \cdot 0 + h(1 - h) \cdot 1 + h \cdot \nu_{\text{string}} \\
\mathbf{y}_1(1) &= (1 - h) \cdot \nu_{\text{string}} + h \cdot x_2 \\
\mathbf{y}_2(1) &= (1 - h) \cdot x_2 + h \cdot x_3 \\
\mathbf{y}_3(1) &= (1 - h) \cdot x_3 + h \cdot \square
\end{aligned}$$

We will do one more time step. For $n \in \{0, 1\}$ and $\sigma = \{A, B\}$, we have possible samples:

$$(\sigma, q) \in \{(n, \underline{\text{halt}}), (\sigma, \underline{\text{halt}}), (n, \text{goR}), (\sigma, \text{goR})\}.$$

In all but the last case, the machine will halt, meaning the state distribution is:

$$\mathbf{q}(2) = (1 - h^2) \cdot \underline{\text{halt}} + h^2 \cdot \text{goR}.$$

In all cases, the write symbol is the same as the read symbol, so we get for the tape:

$$\begin{aligned}
\mathbf{y}_{-2}(1) &= (1 - h^3) \cdot \square + h^3(1 - h) \cdot 0 + h^4 \cdot 1 \\
\mathbf{y}_{-1}(1) &= (1 - h)(1 - h^2) \cdot \square + h(1 + 2h)(1 - h)^2 \cdot 0 + h^2(1 + 2h)(1 - h) \cdot 1 + h^3 \cdot \nu_{\text{string}} \\
\mathbf{y}_0(1) &= (1 - h)^2(1 - h^2) \cdot 0 + h(1 - h)(1 - h^2) \cdot 1 + h(1 - h)(1 + 2h) \cdot \nu_{\text{string}} + h^3 \cdot x_2 \\
\mathbf{y}_1(1) &= (1 - h)(1 - h^2) \cdot \nu_{\text{string}} + h(1 + 2h)(1 - h) \cdot x_2 + h^3 \cdot x_3 \\
\mathbf{y}_2(1) &= (1 - h)(1 - h^2) \cdot x_2 + h(1 + 2h)(1 - h) \cdot x_3 + h^3 \cdot \square \\
\mathbf{y}_3(1) &= (1 - h)(1 - h^2) \cdot x_3 + (h + h^2 - h^3) \cdot \square
\end{aligned}$$

We can read this as a superposition of two copies of $\mathbf{y}_i(1)$ shifted according to the move direction $d = (1 - h^2) \cdot 0 + h^2 \cdot 1$, that is $\mathbf{y}_i(2) = (1 - h^2) \cdot \mathbf{y}_i(1) + h^2 \mathbf{y}_{i+1}(1)$. In the case that $i \in \{-1, 0\}$ this relies on the fact that the write symbol is identical to the read symbol.

Remark 2.12. Observe that, in Definition 2.3, we did not place any constraints on the smooth relaxation of $\mathcal{U}$. As described in [Xu21, §3.2], there exists for any choice of Σ and Q a pseudo-UTM which, initialised as described but allowing $\mathbf{x} \in (\Delta\Sigma)^{\mathbb{Z},\square}$, simulates the smooth relaxation of the coded machine. More specifically, there is an increasing sequence $0 = T_0 < T_1 < \dots$ of natural numbers, such that

$$\pi_U^{\text{tape}} \Delta\text{step}_{\mathcal{U}}^{T_n}(w + \mathbf{x}, \underline{\text{init}}) = \Delta\text{step}_M^n(\mathbf{x}, \underline{\text{halt}}).$$

Better, we can allow the sequence w to contain uncertainty. In this case any $\mathbf{w} \in (\Delta\Sigma \times \Delta Q \times \Delta\{-1, 0, 1\})^W$ lets $\mathcal{U}$ simulate the "smooth" Turing Machine with generalised transition function:

$$\delta : \Sigma \times Q \to \Delta\Sigma \times \Delta Q \times \Delta\{-1, 0, 1\},$$

defined in the same way.

3 Singular Learning

In this section we give a brief overview of Sumio Watanabe's theory of singular learning. Section 4 will state the problem of program synthesis on a Turing Machine in the language of statistical learning, in order to apply these techniques. The key takeaways are the *Bayesian posterior* and *Kullback-Leibler divergence*, which determine the model we select, and the free energy, which helps us to estimate the generalisation error.

3.1 Definitions

This discussion follows §1.1-3 of [Wat09].

Setup 3.1. We take as given a probability space $(\Omega, \mathcal{B}, P)$, and $X : \Omega \to \mathcal{X}$ a random variable subject to probability distribution $q(x)dx$. In this note the state space $\mathcal{X}$ will usually be discrete, ie $\mathcal{X} = [k] := \{1, \dots, k\}$, but to match Watanabe we might as well embed $\mathcal{X} \subset \mathbb{R}^N$ as a collection of discrete points. A model $p(x|w)$ is a family of probability distributions $p(x|w)dx$ on Ω, parametrised by codes $w \in \mathcal{W}$ — here we will take $\mathcal{W} \subset \mathbb{R}^d$ compact. We consider also a prior distribution $\varphi(w)dw$ on the space of codes $\mathcal{W}$. The theory of singular learning is centred around such triples (q, p, φ).

We will assume that the collection of *solutions* $\mathcal{W}_0 = \{w \in \mathcal{W} \mid p(x|w)dx = q(x)dx\}$ is non-empty, but in general we will not have access to the true distribution q. As such, we consider a family $D_n = \{X_1, \dots, X_n\}$ of independent samples subject to q, and consider the problem of choosing an estimate $p^*(x)$ of q based on this family.

Watanabe considers two methods of estimation, both of which rely on the following definition. With $\beta = 1$, this is exactly Bayes' rule [Mac19, §2.1-2]: the term $\prod_{i=1}^n p(X_i|w)$ is simply $p(D_n|w)$.

Definition 3.2. The (generalised) *Bayesian posterior* distribution $p^\beta(w|D_n)$ with inverse temperature $\beta > 0$ is:

$$p^\beta(w|D_n) = \frac{1}{Z_n}\varphi(w)\prod_{i=1}^n p(X_i|w)^\beta,$$

where Z_n is a normalisation constant, namely:

$$Z_n = \int dw\ \varphi(w)\prod_{i=1}^n p(X_i|w)^\beta.$$

We refer to Z_n as the *partition function*.

We denote expectation with respect to p^β by $E_w^\beta[-]$. Note that any such expectation is still a random variable, via its dependence on D_n.

The posterior concentrates near $w \in \mathcal{W}$ which assign higher probability to the samples D_n. Using this, we have two main choices for p^*.

Definition 3.3. Suppose given a triple (q, p, φ), and samples D_n. The *Bayes estimation* is:

$$p^*(x) = E_w^\beta\big[p(x|w)\big],$$

and the Gibbs estimation is $p^*(x) = p(x|w^*)$, where w^* is drawn randomly from the distribution $p^\beta(w|D_n)$.

Remark 3.4. It is useful to see how these definitions might play out in practice. For the purpose of intuition, we will roughly outline how *Hamiltonian Monte Carlo* (HMC) might be used to sample from the Bayesian posterior, in order to find an estimate for the true distribution. For a proper exegesis see [BGJM11].

As mentioned, we cannot assume that the true distribution is known. As such, the "student machine" is provided with n samples from q by some oracle, or "teacher machine". The collection D_n defines a distribution $p^\beta(w|D_n)$ on the parameter space $\mathcal{W}$, by Definition 3.2. In order to estimate a true parameter $w \in \mathcal{W}_0$, we sample from $p^\beta(w|D_n)$ using HMC. This process simulates a particle moving around $\mathcal{W}$ according to Hamiltonian dynamics, where the potential energy function $U(w)$ is smaller in areas assigned larger probability by the posterior. This generates a sequence of samples, which will tend to approach areas where $p^\beta(w|D_n)$ is concentrated. This is schematically represented in Figure 3.

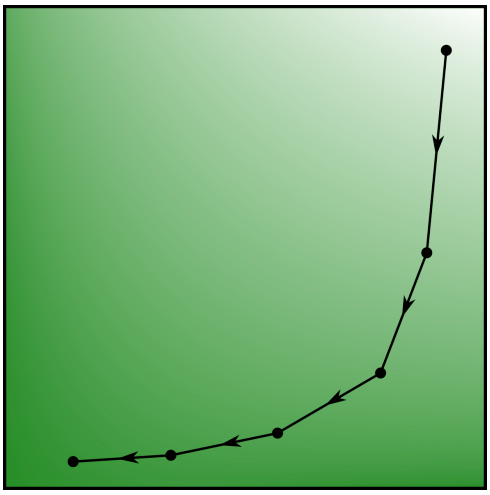

Figure 3: Schematic drawing of Hamiltonian Monte Carlo. The initial guess is in the top right, and the learning machine refines this towards the areas where the posterior is concentrated, represented by darker green.

The most important measure of difference between the true distribution q and an estimate p^* is the Kullback-Leibler (KL) function [Wat09, Section 1.1.2].

Definition 3.5. Given two distributions q and p, the *Kullback-Leibler divergence* is:

$$D_{\mathrm{KL}}(q \,||\, p) = \int dx\, q(x) \log \frac{q(x)}{p(x)}.$$

We note that $D_{\mathrm{KL}}(q \,||\, p) \geq 0$, with equality if and only if $q = p$ [Wat09, Theorem 1.1].

Starting with this definition, we define three measures of how well $p(x|w)$ approximates q. As usual we consider a collection of samples $D_n = \{X_1, \ldots, X_n\}$.

Definition 3.6. The log-loss function $L_n(w)$, KL divergence K and empirical KL divergence

K_n are respectively defined as:

$$L_n(w) = -\frac{1}{n}\sum_{i=1}^{n} \log p(X_i|w)$$
$$K(w) = \int dx\, q(x) \log \frac{q(x)}{p(x|w)}$$
$$K_n(w) = \frac{1}{n}\sum_{i=1}^{n} \log \frac{q(X_i)}{p(X_i|w)}$$

We note that $K_n(w) = -S_n + L_n(w)$, where the empirical entropy

$$S_n = -\frac{1}{n}\sum_{i=1}^{n} \log q(X_i)$$

is independent of w. Also, it is clear that $E_X[K_n(w)] = K(w)$, where the expectation is over our samples D_n.

With this in hand, we can restate our definition of the posterior in a way which draws out a productive analogy with statistical mechanics, where codes are equivalent to states of a physical system [Bal97]. Namely,

$$p^\beta(w|D_n) = \frac{1}{Z_n}\varphi(w)e^{-n\beta L_n(w)} = \frac{1}{Z_n^0}\varphi(w)e^{-n\beta K_n(w)}.$$

In the second equation we use the *reduced partition function* $Z_n^0 = e^{nS_n}Z_n$. We will discuss this in more detail in Appendix A, but for the moment it suffices to observe that this is the form taken by the Maxwell-Boltzmann distribution, with $nK_n(w)$ taking the place of the Hamiltonian (total energy) of a state. This motivates us to consider, as in [Wat20], the free energy as follows.

Definition 3.7. The free energy of a triple (q, p, φ) with samples D_n is:

$$F_n = -\log Z_n = -\log \int dw\ \varphi(w) \prod_{i=1}^{n} p(X_i|w)^\beta.$$

Likewise, the reduced free energy is:

$$F_n^0 = -\log Z_n^0 = -\log \int dw\ \varphi(w)e^{-n\beta K_n(w)}.$$

In [Wat09], Watanabe focusses in large part on the following four random variables, each of which measures the accuracy of either the Bayes or Gibbs estimation. We note that the Gibbs estimate involves a random choice of w^*, so we examine the expectation of the error for varying choices.

Definition 3.8. For a particular estimate p^* of q, the generalisation error is $D_{\mathrm{KL}}(q \mid\mid p^*)$. In this context we have two:

$$B_g = E_X\left[\log \frac{q(X)}{E_w^\beta[p(X_i|w)]}\right]$$
$$G_g = E_w^\beta[K(w)]$$

The training error relies on the particular collection of samples. Namely:

$$B_t = \frac{1}{n}\sum_{i=1}^{n} \log \frac{q(X_i)}{E_w^\beta\big[p(X_i|w)\big]}$$
$$G_t = E_w^\beta[K_n(w)]$$

We will not examine these errors in detail, but we observe the following fact as motivation for our interest in the free energy. In what follows, the unlabelled expectations are over the samples $D_n = \{X_1, \ldots, X_n\}$.

Theorem ([Wat09, Theorem 1.2])**.** For $n > 0$ and $\beta = 1$, we have:

$$B_g = E_{X_{n+1}}[F^0_{n+1}] - F^0_n,$$
$$E[B_g] = E[F^0_{n+1}] - E[F^0_n].$$

That is, we can study how F^0_n grows as a function of n as a proxy for how well our model $p(x|w)$ generalises.

In a more qualitative way, we can also consider a kind of local free energy. Again, in Appendix A, we will see that this mirrors the thermodynamic concept of *coarse graining*. For $w \in \mathcal{W}$, and $N_w \subset \mathcal{W}$ some neighbourhood, we can study integrals of the form:

$$F(N_w) = -\log \int_{N_w} dw\ \varphi(w) e^{-n\beta K_n(w)}.$$

If a certain value $w \in \mathcal{W}$ is assigned large probability (density) by the posterior, then $F(N_w)$ will be comparatively small. In all but the most trivial cases (see [Wat07]) the set of solutions $\mathcal{W}_0 \subset \mathcal{W}$ will contain more than one point, so it is natural to ask which of the solutions will be favoured for large n. As in thermodynamics, we expect solutions with smaller (local) free energy to be selected with higher probability.

As Watanabe remarks in §9.4 of [Wat18], to calculate asymptotic expansions, we may consider instead the *expected free energy* $\bar{F}$, which agrees with F^0_n up to terms order $O(1)$.

$$\bar{F}(n) := -\log \int dw\ \varphi(w) e^{-n\beta E[K_n(w)]} = -\log \int dw\ \varphi(w) e^{-n\beta K(w)}.$$

3.2 Asymptotic Methods

Motivated by the previous section, we now discuss asymptotic expansions as $n \to \infty$ for *Laplace integrals*, of the form:

$$Z(n) = \int_{\mathcal{W}} dw\ |\varphi(w)| e^{-n|f(w)|}.$$

Throughout, $\varphi(w)$ and $f(w)$ will be real analytic, and $\mathcal{W} \subset \mathbb{R}^d$ will be *semianalytic*, ie:

$$\mathcal{W} = \{w \in \mathbb{R}^d \mid g_1(w) \geq 0, \ldots, g_l(w) \geq 0\},$$

for some $g_1, \ldots, g_l$ real analytic. With K in place of $|f(w)|$, the logarithm of such expansions will give us information about the expected free energy discussed in the previous section.

It turns out there are constants C, λ and θ such that the leading term is:

$$Z(n) \approx Cn^{-\lambda}(\log n)^{\theta-1}, \quad n \to \infty.$$

Here, as in [Wat09, Theorem 4.7], the asymptotic equivalence is in the sense that, for some constants $a_1, a_2 > 0$ we have for any $n > 1$:

$$a_1 \frac{(\log n)^{\theta-1}}{n^\lambda} < Z(n) < a_1 \frac{(\log n)^{\theta-1}}{n^\lambda}.$$

After taking the logarithm, we write this as:

$$\log Z(n) = -\lambda \log(n) + (\theta - 1) \log\log(n) + O(1). \tag{8}$$

Our task, then, is to calculate the constants (λ, θ). By Mellin transform techniques, these constants are realised as the poles of a certain *zeta function*, which are then, theoretically at least, calculable by resolution of singularities. This is, basically, the content of [Wat09, Main Formula II].

Definition 3.9. Suppose given a compact semi-analytic set $\mathcal{W} \subset \mathbb{R}^d$, and functions f, φ real analytic on a neighbourhood of $\mathcal{W}$. If $f(w) = 0$ for some $w \in \mathcal{W}$, then we define the *real log canonical threshold* (RLCT) as $\mathrm{RLCT}_{\mathcal{W}}(f; \varphi) = (\lambda, \theta)$, where λ and θ are the coefficients in the asymptotic expansion of $Z(n)$, as above. If $|f(w)| > 0$ everywhere in $\mathcal{W}$, we set $\mathrm{RLCT}_{\mathcal{W}}(f; \varphi) = (\infty, -)$. We order pairs $(\lambda_1, \theta_1) < (\lambda_2, \theta_2)$ if $\lambda_1 < \lambda_2$, or $\lambda_1 = \lambda_2$ and $\theta_1 > \theta_2$. That is, such that for sufficiently large n:

$$\lambda_1 \log n - \theta_1 \log\log n < \lambda_2 \log n - \theta_2 \log\log n.$$

The RLCT, in [Lin11] and elsewhere, is defined in terms of the following function. To prove the statements in the lemma, one needs resolution of singularities (Theorem 3.12), but we state it now anyway in order to motivate the statements that follow.

Lemma 3.10. Given functions f, φ on $\mathcal{W}$ as above, then the following *zeta function* may be analytically continued to a meromorphic function of $z \in \mathbb{C}$:

$$\zeta(z) = \int_{\mathcal{W}} dw\, |f(w)|^{-z} |\varphi(w)|.$$

Its poles are isolated positive rational numbers, and $\mathrm{RLCT}_{\mathcal{W}}(f; \varphi) = (\lambda, \theta)$ is the smallest such pole, and its multiplicity.

Proof. The first statement is Corollary 3.10 in [Lin11] — and remains true if φ is the product of an analytic function and a smooth, positive function (in Lin's terms, φ is *nearly analytic*). The latter is the definition of the RLCT in [Lin11], and the equivalence of the two definitions follows from his Theorem 3.16. □

We first observe the following special case, where $\mathcal{W} = \mathbb{R}^d_{\geq 0}$ is the positive orthant, and f, φ are monomials. The process of resolution of singularities allows us to reduce the general case to this one. It should be remarked that this story, where resolution of singularities is a mere computational technique, misses the significance of the process a little. Watanabe's book centres

on extending statistical learning theory to the singular case, and the conceptual core of that process is the geometric process of resolution of singularities. In short, the RLCT is a measure of *how singular* the set $\mathcal{W}_0 \subset \mathcal{W}$ is (see [Mus11] for a discussion of the complex case). It is a profound observation that the geometry of the analytic function K controls the learning process in a fundamental way.

Proposition 3.11. Let $\mathbf{w} = (w_1, \ldots, w_d)$ be coordinates on $\mathbb{R}^d$ and let $\kappa = (\kappa_1, \ldots, \kappa_d)$ and $\tau = (\tau_1, \ldots, \tau_d)$ be vectors of non-negative integers. Set $\mathbf{w}^\kappa = w_1^{\kappa_1} \ldots w_d^{\kappa_d}$, $\mathbf{w}^\tau = w_1^{\tau_1} \ldots w_d^{\tau_d}$, and define:

$$\mathcal{W} = \mathbb{R}^d_{\geq 0} := \{\mathbf{w} \in \mathbb{R}^d \mid \forall i \; w_i \geq 0\}.$$

Then for a compactly supported smooth function $\phi(\mathbf{w})$, with $\phi(0) > 0$, we have

$$\mathrm{RLCT}_{\mathcal{W}}(\mathbf{w}^\kappa; \mathbf{w}^\tau \phi(\mathbf{w})) = (\lambda, \theta),$$

where we have set,

$$\lambda = \min_{1 \leq i \leq d} \left\{ \frac{\tau_i + 1}{\kappa_i} \right\},$$

and θ is the number of i for which this minimum is attained.

Proof. See [Lin11, Proposition 3.7], with more detail in [AVGZ85, Lemma 7.3]. For $\phi(\mathbf{w}) = 1$, we can integrate our zeta function explicitly (taking $\mathbf{w} \in [0, K]^d$ as ϕ is in fact compactly supported):

$$\begin{aligned}
\zeta(z) &= \int_{\mathcal{W}} d\mathbf{w} \, \mathbf{w}^{\tau - z\kappa} \\
&= \prod_{i=1}^{d} \int_0^K dw \, w^{\tau_i - z\kappa_i} \\
&= \prod_{i=1}^{d} \frac{K^{1+\tau_i - z\kappa_i}}{1 + \tau_i - z\kappa_i}.
\end{aligned}$$

In this situation we have poles for $1 + \tau_i - z\kappa_i = 0$, so the statement is clear. The general case follows by expanding ϕ into an N^{th} order Taylor series and remainder. The (non-zero) constant term contributes the smallest pole, and by increasing N, the term involving the remainder can be made analytic. □

With this result in mind, it is clear that the following theorem, originally due to Hironaka, is useful. Following Watanabe [Wat09, Theorem 2.3] and Lin [Lin11, Theorem 3.3], we state the version used by Atiyah in [Ati70]. For definitions of analytic spaces and manifolds, we refer to §0.1 of [Hir64], the original paper on this topic. The theorem is algorithmic (if labour-intensive), and proceeds by a series of transformations called blow-ups, which are defined and explained in §3.5 of [Wat09]. In §3.6 of the same, Watanabe provides some concrete examples.

Theorem 3.12 (Resolution of Singularities)**.** Let f be a non-constant real analytic function on a neighbourhood of the origin in $\mathbb{R}^d$, with $f(0) = 0$. Then there exists a triple (M, W, ρ) where:

- $W \subset \mathbb{R}^d$ is open, and contains 0,

- M is a d-dimensional real analytic manifold,
- $\rho : M \to W$ is a real analytic map.

The following also hold.

- ρ is proper, the inverse image of a compact set is compact.
- ρ is a real analytic isomorphism away from $\mathbb{V}_W(f)$. (That is, $M \backslash \mathbb{V}_M(f \circ \rho) \longrightarrow W \backslash \mathbb{V}_W(f)$.)
- Around any point $y \in \mathbb{V}_M(f \circ \rho)$, there are local coordinates $u = (u_1, \ldots, u_d)$ on some neighbourhood M_y, vectors κ and τ of non-negative integers, and strictly positive, real analytic functions a and h of u such that:

  $$f \circ \rho(u) = a(u)u^{\kappa},$$

  and the Jacobian determinant of ρ:

  $$|\rho'(u)| = h(u)u^{\tau}.$$

Corollary 3.13. Given non-constant analytic functions $f_1, \ldots, f_l$ in a neighbourhood of $0 \in \mathbb{R}^d$, all vanishing at the origin, there is a triple (M, W, ρ) as above which desingularises each f_i.

Proof. See [Wat09, Theorem 2.8]. Apply the original form of the Theorem to the product $f_1(w) \cdots f_l(w)$, then observe [Wat09, Theorem 2.7] that the resulting triple desingularises each f_i. □

Now we want to apply this theorem to calculate RLCTs: in short, it works as follows [Lin11, Lemma 3.8]. The statement is local, so we examine a particular point $w \in \mathbb{V}_{\mathcal{W}}(f)$, and desingularise f at w, using the theorem. We may also assume that the triple (M, W, ρ) desingularises φ and each of the analytic functions $g_1, \ldots, g_l$ (if they vanish at w) which define $\mathcal{W} \subset \mathbb{R}^d$. We can also show that the neighbourhood W can be shrunk to N_w such that $\rho^{-1}(N_w)$ is a union of coordinate neighbourhoods M_y as in the theorem. In each of these coordinates, the situation is as in Proposition 3.11: since the constraints are monomial, $\mathcal{M}_y = M_y \cap \rho^{-1}\mathcal{W}$ is a union of orthants, and the functions $f \circ \rho, \varphi \circ \rho$ are of the correct form. Using a partition of unity $\{\sigma_y\}$ subordinate to $\{\mathcal{M}_y\}$, we can write the zeta function as:

$$\zeta(z) = \sum_y \int_{\mathcal{M}_y} du\, |f \circ \rho(u)|^{-z} |\varphi \circ \rho(u)||\phi \circ \rho(u)|\sigma_y(u).$$

The RLCT (λ, θ) associated to ζ is simply the smallest such pair (using the ordering of Definition 3.9) associated to one of the integrals:

$$\zeta_y(z) = \int_{\mathcal{M}_y} du\, |f \circ \rho(u)|^{-z} |\varphi \circ \rho(u)||\phi \circ \rho(u)|\sigma_y(u),$$

which we may calculate as in the proposition. In particular, λ and θ are positive rational numbers, independent of the function ϕ. What we have shown is the following.

Lemma 3.14. Around any $w \in \mathcal{W}$ such that $f(w) = 0$, there is a neighbourhood $N_w \subset \mathcal{W}$ such that for all smooth functions ϕ with $\phi(w) > 0$ we have:

$$\mathrm{RLCT}_{N_w}(f; \varphi\phi) = \mathrm{RLCT}_{N_w}(f; \varphi).$$

These RLCTs are positive rational numbers.

This may seem to have achieved little, but the independence of ϕ allows us to express the "global" $\mathrm{RLCT}_{\mathcal{W}}(f; \varphi)$ in terms of the "local" $\mathrm{RLCT}_{N_w}(f; \varphi)$: this is [Lin11, Proposition 3.9]. We know that the large-n Laplace integral $Z(n)$ is controlled by $\mathcal{W}_0 = \mathbb{V}(f)$, as everywhere else the integrand is arbitrarily small. The intuitive content of this lemma is that 1) we can perform the integrals of interest in small neighbourhoods of solutions, and ignore a non-vanishing prior, and 2) $Z(n)$ is in fact controlled by the "deepest" [Lin11, §3.2.5] of these singularities, which minimise the local RLCT.

Lemma 3.15. For f, φ, ϕ functions on $\mathcal{W} \subset \mathbb{R}^d$, with the usual assumptions:

$$\mathrm{RLCT}_{\mathcal{W}}(f; \varphi\phi) = \min_{w \in \mathbb{V}_{\mathcal{W}}(f)} \mathrm{RLCT}_{N_w}(f; \varphi).$$

Proof. Noting that $\mathcal{W}$ is compact, pick a finite subcover $\{N_w \mid w \in S\}$ among the neighbourhoods N_w defined by Lemma 3.14. If $w \notin \mathbb{V}_{\mathcal{W}}(f)$, then choose N_w small enough that $f > 0$, so that $\mathrm{RLCT}_{N_w}(f; \varphi\phi) = (\infty, -)$. With a partition of unity $\{\sigma_w\}_{w \in S}$ subordinate to this cover, we can rewrite the zeta function in question as:

$$\zeta(z) = \sum_{w \in S} \int_{N_w} |f(v)|^{-z} |\varphi(v)\phi(v)| \sigma_w(v) dv.$$

Noting that we can choose $\sigma_w(w) > 0$, this implies:

$$\mathrm{RLCT}_{\mathcal{W}}(f; \varphi\phi) = \min_{w \in S} \mathrm{RLCT}_{N_w}(f; \varphi\phi\sigma_w) = \min_{w \in S} \mathrm{RLCT}_{N_w}(f; \varphi).$$

The minimum exists as S is finite. To finish, we observe that for any $w \in \mathcal{W}$, the zeta function defined by N_w cannot have poles that are not also present in ζ. As such,

$$\mathrm{RLCT}_{\mathcal{W}}(f; \varphi\phi) \leq \mathrm{RLCT}_{N_w}(f; \varphi\phi) = \mathrm{RLCT}_{N_w}(f; \varphi).$$

We take the minimum over $w \in \mathbb{V}_{\mathcal{W}}(f)$, as the other RLCTs are infinite. □

In summary, the large n asymptotics of Laplace-type integrals, of which our free energy is one, are controlled by the RLCT, which can be calculated as the poles of a certain zeta function. By resolution of singularities, these poles are at positive rational numbers, and may be calculated algorithmically.

3.3 Effective Parameters

Before we move on to the methods that we will use to calculate the RLCT, we develop in this section an intuition for its value as a measure of the *effective number of parameters* in a neighbourhood of a particular solution. The ideas here are contained in [Wat09, §7.1], and the discussion follows [Mur20b].

Let us consider a singular learning triple (q, p, φ), with KL divergence $K : \mathcal{W} \to \mathbb{R}$. Watanabe observes the following fact in [Wat09, Theorem 7.1 (4)]. Supposing that the set of true parameters is non-empty, and defining the following *volume function*:

$$V(t) = \int_{K(w)<t} dw\, \varphi(w),$$

we have that for arbitrary $a \in (0, 1)$:

$$\lambda = \lim_{t\to 0} \frac{\log[V(at)/V(t)]}{\log a}, \tag{9}$$

where $(\lambda, \theta) = \mathrm{RLCT}_{\mathcal{W}}(K; \varphi)$. This expresses λ as a measure of dimension. The function $V(t)$ measures the size of the set of "almost true parameters", and the limit of (9) measures a scaling exponent with decreasing "radius" t.

More precisely, suppose that $0 \in \mathcal{W}$ is a solution, and assume that for some coordinates $w_1, \ldots, w_d$ on a neighbourhood N of 0, we can write the KL divergence as:

$$K(w) = \sum_{i=1}^{d'} w_i^2, \tag{10}$$

for some $d' \leq d$. Then we can write (for the right choice of neighbourhood N):

$$\{w \in N \mid K(w) < t\} = B^{d'}_{t^{1/2}}(0) \times [-\epsilon, \epsilon]^{d-d'},$$

where $B^{d'}_{t^{1/2}}(0)$ is a d'-ball of radius $t^{1/2}$. For a uniform prior, we have:

$$V(t) \propto t^{d'/2},$$

so the local RLCT around 0 is:

$$\frac{\log[V(at)/V(t)]}{\log a} = \frac{\log a^{d'/2}}{\log a} = \frac{d'}{2}.$$

(This fact is also easily calculated using the arguments of Lemma 3.21, and the fact that $\mathrm{RLCT}_N(w_i^2, 1) = \frac{1}{2}$.) In this case,

$$\mathcal{W}_0 = \{w \in \mathcal{W} \mid K(w) = 0\} = \{0\} \times [-\epsilon, \epsilon]^{d-d'},$$

so d' is the *codimension* of $\mathcal{W}_0$. Therefore, around our solution there are 2λ coordinate directions which vary K: we say that our model has 2λ *effective parameters*.

Of course, we can't always write K in the form of (10), and as a result 2λ may not even be an integer. That being said, if $\mathcal{W}_0 = \{w \mid K(w) = 0\}$ is a $(d - d')$-dimensional submanifold of $\mathcal{W}$ around w_0, then by [Wat09, Theorem 7.3]:

$$\lambda \leq \frac{d'}{2}. \tag{11}$$

Observe that for any analytic K defined and positive in an open neighbourhood of 0, the Taylor series of K must only have quadratic terms and above. As such, (10) is the *sharpest* way

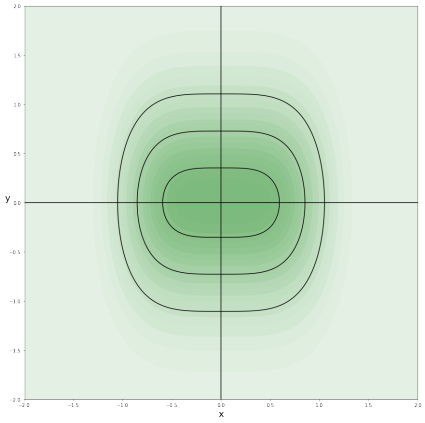

Figure 4: A plot of $e^{1-x^4-y^2}$. The model is more weakly coupled to the value of x, than to y.

K can vary from 0, in the sense that $x^n = o(x^2)$ as $x \to 0$ for $n > 2$. This explains the upper bound of (11). If K is comparable to a function of the form:

$$w_1^{m_1} + \cdots + w_{d'}^{m_{d'}},$$

for $m_i \geq 2$, then

$$\mathrm{RLCT}_N(K, 1) = \sum_{i=1}^{d'} \frac{1}{m_i}.$$

For this reason, we can interpret the quantity 2λ as a more refined measure than simply counting the normal directions to $\mathcal{W}_0$. If the parameter w_i appears with an index larger than 2, the model is more weakly coupled to its value, so it "effectively" accounts for less than one parameter (Figure 4).

As observed in [Wat13], this subsumes the *Bayesian Information Criterion*, which only applies to regular statistical models. In any case, if w_0 minimises the loss, the free energy is asymptotically

$$F_n = nL_n(w_0) + \lambda \log n + O(\log \log n).$$

(This extends (8) to the case where w_0 such that $K(w_0) = 0$ might not exist.) Then, to choose between statistical models p with varying minimal loss $L_n(w_0)$ and numbers of parameters, we choose the model which minimises this quantity (appealing to [Wat09, Theorem 1.2]). In the regular case, this amounts to minimising the value of:

$$\frac{k}{2} \log(n) + nL_n(w_0),$$

where k is the number of parameters. This is exactly the Bayesian Information Criterion, introduced by Gideon Schwartz in [Sch78].

3.4 Calculating RLCTs

It is often not feasible to calculate the resolution of singularities for a general function f. We are interested in the asymptotics of the expected free energy:

$$\bar{F}(n) := -\log \int dw\ \varphi(w) e^{-n\beta E[K_n(w)]} = -\log \int dw\ \varphi(w) e^{-n\beta K(w)}.$$

The function K (which will play the role of the phase f) is complicated, so we will introduce methods which reduce the complexity, following mostly §4 of [Lin11]. In particular, we will work on the tacit assumption that the model p is not overly complicated — in the case of program synthesis on a Turing Machine it will be a polynomial.

Lemma 3.16. Suppose given pairs (f_1, φ_1) and (f_2, φ_2) of functions on $\mathcal{W}$. If for every $w \in \mathcal{W}$ and some constants c, d:

$$|f_1(w)| \leq c|f_2(w)| \text{ and } |\varphi_1(w)| \geq d|\varphi_2(w)|,$$

then with $\mathrm{RLCT}_{\mathcal{W}}(f_i; \varphi_i) =: (\lambda_i, \theta_i)$,

$$(\lambda_1, \theta_1) \leq (\lambda_2, \theta_2),$$

using the usual ordering of Definition 3.9.

Proof. This is stated without proof in [Wat09, Remark 7.2]. If we define the Laplace integrals, for $i \in \{1, 2\}$:

$$Z_i(n) = \int_{\mathcal{W}} dw |\varphi_i(w)| e^{-n|f_i(w)|},$$

then as usual we have the asymptotic equivalences, as $n \to \infty$:

$$Z_i(n) \approx C_i n^{-\lambda_i} (\log n)^{\theta_i - 1}.$$

By monotonicity, we have:

$$Z_1(n) \geq \int_{\mathcal{W}} dw\, d|\varphi_2(w)| e^{-nc|f_2(w)|} = d \cdot Z_2(cn)$$

Taking logarithms, we have for sufficiently large n:

$$-\lambda_1 \log(n) + (\theta_1 - 1) \log\log(n) \geq -\lambda_2 \log(n) + (\theta_2 - 1) \log\log(n),$$

as the constants get absorbed in the $O(1)$ term. □

Corollary 3.17. If there are positive constants c_1 and c_2 such that $c_1|f_1| \leq |f_2| \leq c_2|f_1|$, then

$$\mathrm{RLCT}_{\mathcal{W}}(f_1; \varphi) = \mathrm{RLCT}_{\mathcal{W}}(f_2; \varphi).$$

We will refer to such functions [Lin11, §3.2.3] as *comparable*.

Using this, Lin demonstrates that the RLCT is, suitably interpreted, a property of the ideal generated by the phase function, inside the ring of real analytic functions on $\mathcal{W}$.

Definition 3.18. For a subset $\mathcal{W} \subset \mathbb{R}^d$, $\mathcal{A}_\mathcal{W}$ is the ring of real analytic functions on $\mathcal{W}$. For a (finitely-generated) ideal:

$$\langle f_1, \dots, f_r \rangle = I \subset \mathcal{A}_\mathcal{W},$$

we define $\mathrm{RLCT}_\mathcal{W}(I; \varphi)$, to be $\mathrm{RLCT}_\mathcal{W}(f; \varphi)$, where:

$$f(w) = \sum_{i=1}^{r} f_i(w)^2.$$

The next few results collect properties of this definition, as taken from [Lin11, §4.1]. **Caution**: this definition differs by a factor of two from the definition in [Lin11, Proposition 4.3]. As a result, for $f \in \mathcal{A}_\mathcal{W}$, and $\langle f \rangle$ the ideal it generates:

$$\begin{aligned} \mathrm{RLCT}_\mathcal{W}(\langle f \rangle; \varphi) &= (\lambda, \theta) \\ \mathrm{RLCT}_\mathcal{W}(f; \varphi) &= (2\lambda, \theta) \end{aligned} \tag{12}$$

as the first equation is associated to the zeta function:

$$\zeta(z) = \int_\mathcal{W} dw \, |f(w)^2|^{-z} |\varphi(w)| = \int_\mathcal{W} dw \, |f(w)|^{-2z} |\varphi(w)|.$$

We use this convention because, for our purposes, the RLCT of an ideal is interesting because of Lemma 3.22, which is easier to state in this way.

Lemma 3.19. The RLCT of a finitely generated ideal I is independent of the chosen set of generators.

Proof. This is [Lin11, Proposition 4.3]. In brief, if $\langle f_1, \dots, f_r \rangle = \langle g_1, \dots, g_s \rangle$, then for any j, there are functions $h_1, \dots, h_r \in \mathcal{A}_\mathcal{W}$ such that:

$$g_j^2 = (h_1 f_1 + \cdots + h_r f_r)^2 \leq (h_1^2 + \cdots + h_r^2)(f_1^2 + \cdots + f_r^2).$$

Using that $\mathcal{W}$ is compact, this implies that there is $c > 0$ such that:

$$\sum_{j=1}^{s} g_j^2 \leq c \sum_{i=1}^{r} f_i^2.$$

□

Corollary 3.20. If $I \subset J$ then $\mathrm{RLCT}_\mathcal{W}(I; \varphi) \leq \mathrm{RLCT}_\mathcal{W}(J; \varphi)$.

Proof. If $I = \langle f_1, \dots, f_n \rangle \subset J = \langle g_1, \dots, g_m \rangle$, then we might as well extend the generators of J to be $\langle f_1, \dots, f_n, g_1, \dots, g_m \rangle$. Then clearly,

$$\sum_{i=1}^{n} f_i^2 \leq \sum_{i=1}^{n} f_i^2 + \sum_{j=1}^{m} g_j^2,$$

so the result follows from Lemma 3.16. □

In many ways, the RLCT of an ideal is easier to apprehend. The following result allows us, in certain cases, to break calculations down into smaller units.

Lemma 3.21. Suppose we can write $\mathcal{W} = \mathcal{W}_1 \times \mathcal{W}_2$, for $\mathcal{W}_i$ compact and semianalytic. We can view $\mathcal{A}_{\mathcal{W}_i} \subset \mathcal{A}_{\mathcal{W}}$ by composing with projections. Then for $J_i \subset \mathcal{A}_{\mathcal{W}_i}$ ideals, and writing $(\lambda_i, \theta_i) = \mathrm{RLCT}_{\mathcal{W}_i}(J_i; \varphi)$, we have the formulae:

$$\mathrm{RLCT}_{\mathcal{W}}(J_1 + J_2; \varphi) = (\lambda_1 + \lambda_2, \theta_1 + \theta_2 - 1)$$

$$\mathrm{RLCT}_{\mathcal{W}}(J_1 J_2; \varphi) = \begin{cases} (\lambda_1, \theta_1) & \lambda_1 < \lambda_2 \\ (\lambda_2, \theta_2) & \lambda_1 > \lambda_2 \\ (\lambda_1, \theta_1 + \theta_2) & \lambda_1 = \lambda_2 \end{cases}$$

Proof. See [Lin11, Proposition 4.5]. The first follows by examining the asymptotic expansion of the associated Laplace integrals, and the second by calculating the poles of the zeta functions. □

The following result links the RLCT of a function ($g \circ f$ in this case) to the RLCT of a particular ideal — it will allow us to calculate $\mathrm{RLCT}_{\mathcal{W}}(K; \varphi)$ in many cases, without having to examine the logarithms in K. This technique applies to what Lin calls *regularly parametrised models*, as in [Lin11, Theorem 1.11].

Lemma 3.22. Suppose that $\mathcal{W} \subset \mathbb{R}^d$ and $\mathcal{W}' \subset \mathbb{R}^{d'}$ are compact and semi-analytic, and $f = (f_1, \ldots, f_{d'}) : \mathcal{W} \to \mathcal{W}'$ and $g : \mathcal{W}' \to \mathbb{R}$ are real analytic. Pick $\hat{w} \in \mathcal{W}$, set $\hat{f} = f(\hat{w})$, and let $N_{\hat{w}}$ be the neighbourhood defined by Lemma 3.14. Then if $g(\hat{f}) = 0$, $\nabla g(\hat{f}) = 0$ and the Hessian $\nabla^2 g(\hat{f})$ is positive definite, then:

$$\mathrm{RLCT}_{N_{\hat{w}}}(g \circ f; \varphi) = \mathrm{RLCT}_{N_{\hat{w}}}(\langle f_1 - \hat{f}_1, \ldots, f_{d'} - \hat{f}_{d'} \rangle; \varphi).$$

Proof. See [Lin11, Proposition 4.4]. The lemma follows from the fact that, in a small enough neighbourhood of $\hat{f}$, g is comparable (in the sense of Corollary 3.17) to a sum of squares:

$$(u_1 - \hat{f}_1)^2 + \cdots + (u_{d'} - \hat{f}_{d'})^2,$$

where $u_1, \ldots, u_{d'}$ are coordinates on $\mathcal{W}'$. The right-hand side is exactly the RLCT of f composed with such a sum of squares. □

To illustrate these techniques, we calculate an example RLCT, which we will use in Example 4.28.

Example 3.23. Let $\mathcal{W} = [0, 1]^2$, $f(x, y) = x^3 + y$ and set $\varphi = 1$. The Laplace integral in this case is

$$Z(n) = \int_{\mathcal{W}} dx dy \, e^{-n(x^3 + y)}.$$

By Lemma 3.15 the first order asymptotics of Z only depend on a neighbourhood of the origin, so we might as well extend the domain to $\mathbb{R}^2_{\geq 0}$, in which case we can evaluate explicitly:

$$\int_0^\infty \int_0^\infty dx dy \, e^{-n(x^3 + y)} = \int_0^\infty dx \, e^{-nx^3} \cdot \int_0^\infty dy \, e^{-ny} = n^{-4/3} \int_0^\infty du \, e^{-u^3}$$

As such, we can read off $\mathrm{RLCT}_{\mathcal{W}}(f;1) = (4/3, 1)$. On the other hand, we can examine the zeta function by resolution of singularities: we need to resolve the boundary conditions $x \geq 0, y \geq 0$ as well as f. It turns out (a slight extension of [Lin12, Example 9.1]) that the manifold $\mathcal{M}$ can be expressed as a union of four charts $U_1 \cup U_2 \cup U_3 \cup U_4$, each isomorphic to $\mathbb{R}^2$, with the map $\rho : \mathcal{M} \to \mathbb{R}^2$ given by (in coordinates (x_i, y_i) on U_i):

$$\begin{aligned}
\rho(x_1, y_1) &= (x_1 y_1, y_1) \\
\rho(x_2, y_2) &= (x_2 y_2, x_2^2 y_2^3) \\
\rho(x_3, y_3) &= (x_3, x_3^3 y_2^2) \\
\rho(x_4, y_4) &= (x_4 y_4, x_4 y_4^2)
\end{aligned}$$

These formulae are derived by blowing up the origin in $\mathbb{R}^2$ three times, first in the original domain then in any coordinate chart where $\mathbb{V}(f \circ \rho)$ remains singular. Without going into details, each blow-up produces two charts, which are related as follows:

$$\mathbb{R}^2 \longleftarrow \begin{cases} U_1 \\ V_1 \longleftarrow \begin{cases} V_2 \longleftarrow \begin{cases} U_2 \\ U_3 \end{cases} \\ U_4 \end{cases} \end{cases}$$

The integrand of the relevant zeta function is $(x^3 + y)^{-z} dx dy$, which in each of these coordinates is (picking out the relevant factor for our calculation):

$$\begin{aligned}
[y_1(x_1^3 y_1^2 + 1)]^{-z} y_1 dx_1 dy_1 &\sim y_1^{1-z} dx_1 dy_1 \\
[x_2^2 y_2^3 (x_2 + 1)]^{-z} x_2^2 y_2^3 dx_2 dy_2 &\sim x_2^{2-2z} y_2^{3-3z} dx_2 dy_2 \\
[x_3^3 (1 + y_3^2)]^{-z} 2 x_3^3 y_3 dx_3 dy_3 &\sim x_3^{3-3z} y_3^{1-z} dx_3 dy_3 \\
[x_4 y_4^2 (x_4^2 y_4 + 1)]^{-z} x_4 y_4^2 dx_4 dy_4 &\sim x_4^{1-z} y_4^{2-2z} dx_4 dy_4
\end{aligned}$$

In the first case, we get an RLCT of $(2, 1)$, in the second and third we get our expected minimum value $\mathrm{RLCT}_{\mathcal{W}}(f;1) = (4/3, 1)$, and in the final case we get $(3/2, 1)$.

Note that if 0 is not on the boundary, a simpler resolution of singularities works, viz:

$$\rho(u, v) = (v, u - v^3).$$

In this case, the zeta function becomes $|u|^{-z} du dv$ integrated in an open neighbourhood of the origin, so $\mathrm{RLCT}_{\mathbb{R}^2}(f, 1) = (1, 1)$. If we try this in the case above, then we get:

$$\zeta(z) = \int_0^1 \int_{v^3}^{v^3+1} du dv \, u^{-z} = \frac{1}{1-z} \int_0^1 dv \, (v^3 + 1)^{1-z} - v^{3-3z},$$

and the apparent singularity at $z = 1$ is removable. The two regions of integration are illustrated in Figure 5.

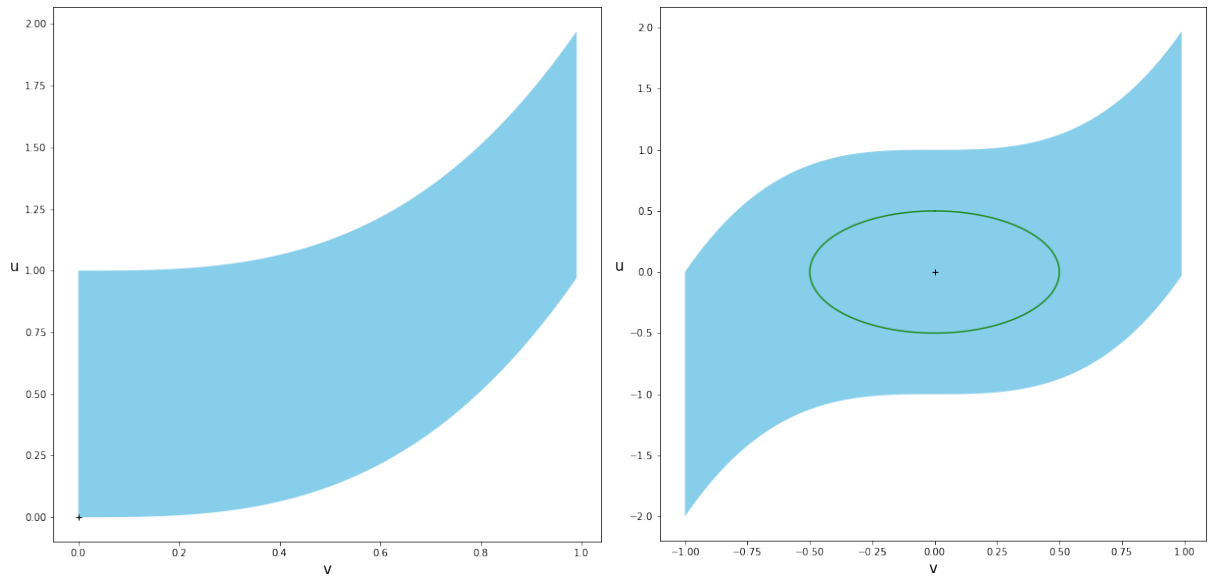

Figure 5: The integration regions in the $v - u$ plane, where the blue region is $\rho_i^{-1}(\mathcal{W})$. Case 1 is $\mathcal{W} = [0,1]^2$, and Case 2 $\mathcal{W} = [-1,1]^2$. In the latter case, we can shrink to the green region, on which Proposition 3.11 applies. Plots throughout use Matplotlib [Hun07].

4 Singular Learning on Turing Machines

We consider now a problem of general inductive inference on a Turing Machine, as in [CM19, Setup 7.1]. We start with an observer uncertain about the initial contents of (part of) the tape of a Turing Machine. The machine is run for some number of steps, and the contents of (some subset) of the tape are read off. From this, the observer tries to infer the initial tape sequence. This produces a singular learning triple in the sense of [Wat09] and Setup 3.1.

In the case that the machine is a UTM, our definition allows us to apply the results of Section 3 to a problem of program synthesis. In particular, if $\mathcal{W}$ parametrises possible programs, the process of *learning a program* is controlled by the geometry of the KL divergence, an analytic function

$$K : \mathcal{W} \longrightarrow \mathbb{R}.$$

To link this to algebraic geometry, we define in Section 4.4 an ideal of polynomials with the same RLCT as K. Our definitions provide the geometric perspective on programs that was discussed in the introduction.

4.1 Inference problems

We make small modifications to the setup of [CM19], in order to match with the formulation of SLT, as in [CMW21]. Before defining our inference problems in full generality, let us examine some special cases. Fix a Turing Machine M, and assume that a naive Bayesian observer is told what is on the tape after t time steps. Assume also that the observer is told the initial configuration, apart from the letter on square 0. Then, by varying $\mathbf{w} \in \Delta\Sigma$, the smooth relaxation defines a polynomial function:

$$\Delta\text{step}_M^t : \Delta\Sigma \longrightarrow (\Delta\Sigma)^{\mathbb{Z}}.$$

That is, given any belief $\mathbf{w} \in \Delta\Sigma$ about the contents of tape square 0, we have a corresponding belief for the contents of the tape after t steps. To invert this map is a problem of statistical inference: the value $\mathbf{w}$ parametrises a family of distributions, and we seek to find the value(s) $\mathbf{w}_0$ which correspond to the information in the true distribution (the tape configuration after some time).

The example to hold in mind (which we will return to in Section 4.3) is that of a pseudo-UTM $\mathcal{U}$, as in Definition 2.3. Suppose we want to find a TM that computes a function f, which we specify as a collection of input/output pairs $\{(x_i, y_i)\}$. Rather than searching for the "solution" TM itself, we search for a code, or *program*, $\mathbf{w} \in (\Sigma_{\mathcal{U}})^V$, which specifies this TM in the *programming language* of the UTM $\mathcal{U}$. The situation is similar to above: we have partial information about the tape configuration of $\mathcal{U}$, given by the pairs (x_i, y_i), and finding a TM to compute f is equivalent to finding a *program*, or code, $\mathbf{w} \in (\Sigma_{\mathcal{U}})^V$ which matches these observations.

This suggests some natural extensions. We might want to restrict the behaviour of the TM beyond requesting it compute a certain function. For example, we could place restrictions on the state (at varying times), or fix the tape configuration at intermediate times, rather than only fixing the halting distribution. Finally, we needn't consider only "deterministic" problems, where the final tape configuration is known for sure — rather, the "outputs" y_i could be distributions instead of sequences.

In what follows, the set $\mathcal{S} := [\mathbb{Z} \cup \{\text{state}\}] \times \mathbb{N} \times \Sigma^{U,\square}$ parametrises all of the possible (types of) constraints we could apply. The last two factors correspond to the time t and the input x, and the first to applying a constraint either to tape square $a \in \mathbb{Z}$, or to the state. That is, elements are $(a, t, x) \in \mathbb{Z} \times \mathbb{N} \times \Sigma^{U,\square}$, corresponding to fixing tape square a at time t with input x; or (state, t, x), which fixes the state at time t with input x. A weighting specifies which of these constraints will be applied, and their relative importance. For convenience, we split up the weighting — this will help us, say, if there is a single timeout for all inputs, or a fixed window in which we check the output.

Definition 4.1. By a *weighting*, we mean a set $U \subset \mathbb{Z}$, called the input window, and a distribution s over $\mathcal{S} = [\mathbb{Z} \cup \{\text{state}\}] \times \mathbb{N} \times \Sigma^{U,\square}$, which we present as follows.

- A distribution $s(x)$ over $\Sigma^{U,\square}$,
- A conditional distribution $s(t|x)$ over $\mathbb{N}$, and
- For each $(t, x) \in \mathbb{N} \times \Sigma^{U,\square}$, a pair $(p_{t,x}, \lambda_{t,x})$, where p is a distribution over $\mathbb{Z}$, and $\lambda \in [0, 1]$.

The value $\lambda_{t,x}$ controls the weighting on the state, where $\lambda_{t,x} = 0$ corresponds to $s(\text{state} \mid t, x) = 1$. For clarity, the weighting on an element $z \in \mathcal{S}$ is:

$$s(z) = \begin{cases} \lambda_{t,x} \cdot p_{t,x}(a) s(t|x) s(x), & z = (a, t, x) \in \mathbb{Z} \times \mathbb{N} \times \Sigma^{U,\square} \\ (1 - \lambda_{t,x}) \cdot s(t|x) s(x), & z = (\text{state}, t, x) \in \{\text{state}\} \times \mathbb{N} \times \Sigma^{U,\square} \end{cases}$$

The *support* of a weighting (U, s) is the subset

$$\mathcal{S}_+(s) = \{z \in \mathcal{S} \mid s(z) > 0\},$$

and the *target window* $Y_{t,x}$ for a pair (t, x) is

$$\text{supp}(p_{t,x}) = \{a \in \mathbb{Z} \mid p_{t,x}(a) \neq 0\}.$$

We will parametrise programs using the probability simplices $\Delta\Sigma$. However, we want to limit the letters which can be used, which we think of as defining a syntax for the programming language on the TM. The following definition is notation for a product of the simplices over this limited alphabet. In the setup of Section 3, this definition will play the role of $\mathcal{W}$.

Definition 4.2. Given a finite subset $V \subset \mathbb{Z}$ of tape squares, and a function $c : V \to \mathcal{P}(\Sigma)$ specifying the subset $c(i) \subset \Sigma$ of *allowed codes* on tape square i, we define the *code space*:

$$\mathscr{C}(V, c) = \prod_{i \in V} \Delta c(i).$$

Definition 4.3. Given disjoint subsets $U, V \subset \mathbb{Z}$, we define a sequence of distributions in $(\Delta\Sigma)^{\mathbb{Z},\square}$, for $x \in \Sigma^U$ and $\mathbf{w} \in (\Delta\Sigma)^V$:

$$(x + \mathbf{w})_i = \begin{cases} x_i & i \in U \\ \mathbf{w}_i & i \in W \\ \square & \text{otherwise} \end{cases}$$

Now we are ready for our definition of an inference problem. We have uncertainty on the tape squares V, and apply constraints in the situations (input, time, tape square or state) defined by the weighting (U, s). The state of belief $y_{t,x}$ defines the precise constraint we apply. For example, if $s(0, t, x) > 0$ and $\pi_0^{\text{tape}} y_{t,x} = 1 \cdot \sigma$, then the constraint is "with input x tape square 0 should hold the letter σ after t steps". A solution is $\mathbf{w} \in \mathscr{C}(V, c) \subset (\Delta\Sigma)^V$ which satisfies the constraints.

Definition 4.4. An *inference problem* P on a Turing Machine $M = (\Sigma, Q, \delta)$ consists of:

- A finite *code window* $V \subset \mathbb{Z}$,
- A weighting (U, s), with $U \cap V = \emptyset$.
- For every $t \in \mathbb{N}$ and $x \in \Sigma^{U,\square}$, a state of belief $y_{t,x} \in (\Delta\Sigma)^{\mathbb{Z},\square} \times \Delta Q$, and
- A specification of allowed codes $c : V \to \mathcal{P}(\Sigma)$, as in Definition 4.2.

An inference problem is *deterministic* if the constraints we apply have no uncertainty. That is, for every t, x positively weighted, we have:

$$\pi_a^{\text{tape}} y_{t,x} \in \Sigma \subset \Delta\Sigma, \quad \forall a \in Y_{t,x}$$
$$\pi^{\text{state}} y_{t,x} \in Q \subset \Delta Q$$

Definition 4.5. A *solution* to an inference problem is $\mathbf{w} \in \mathscr{C}(V, c)$ such that, for every positively weighted $(t, x) \in \mathbb{N} \times \Sigma^{U,\square}$, we have:

- if $\lambda < 1$ then $\pi^{\text{state}} \Delta\text{step}_M^t(x + \mathbf{w}, \underline{\text{init}}) = \pi^{\text{state}} y_{t,x}$
- if $\lambda > 0$ then for every $a \in Y_{t,x}$, $\pi_a^{\text{state}} \Delta\text{step}_M^t(x + \mathbf{w}, \underline{\text{init}}) = \pi_a^{\text{tape}} y_{t,x}$.

The set of solutions $\mathbf{w} \in \mathscr{C}(V, c)$ will be denoted $\mathcal{W}_0$.

Definition 4.6. The *support* $\mathcal{S}_+(P)$ of an inference problem P is the support of the weighting s:

$$\mathcal{S}_+(P) = \{z \in \mathcal{S} \mid s(z) > 0\}.$$

P is called *compact* if $\mathcal{S}_+(P)$ is a finite set.

Remark 4.7. Two inference problems found by varying some $y_{t,x}$ outside the target window $Y_{t,x}$ will be considered identical, as they have the same solutions, and (see below) the same KL divergence. We have assumed that:

$$\sum_{z \in \mathcal{S}} s(z) = 1,$$

but this is unimportant, so long as the sum is finite.

We make the general definition, rather than restricting ourselves to the specific case of program synthesis, as many of our results do not depend on that restriction. As such, it will make the high-level structure of the theory clearer. The definition of a UTM, and as such of program synthesis, depends on several choices of conventions [Hut04, Definition 2.6], so by defining a more general version of inductive inference we capture each of these possibilities. The simpler setting will also make it possible to calculate explicitly in small examples.

Example 4.8. Continuing from Example 2.11, we define a parametrised family P_α of inference problems, for $\alpha \in [0,1]$. As before, the observer to the Shift Machine is uncertain about whether the counter is 0 or 2, and whether the first tape square is A or B. Suppose now that the observer is told that the first tape square is an A after two time steps of the machine. The input on tape squares two and three is either a pair of A's, or an A and a B. That is, the two input scenarios are:

$$x^{(0)} = \ldots \square \underline{?}\, ?\, A\, A \square \ldots$$
$$x^{(1)} = \ldots \square \underline{?}\, ?\, A\, B \square \ldots$$

We "interpolate" between these two inputs with the constant $\alpha \in [0,1]$, so that:

$$s(x) = (1-\alpha) \cdot x^{(0)} + \alpha \cdot x^{(1)}.$$

Formally, we will apply two constraints. The input window is $U = \{2,3\}$, and with the two sequences $x^{(0)} = (A,A)$ and $x^{(1)} = (A,B) \in \Sigma^U$ and a timeout after two steps the weighting is:

$$s(a,t,x) = \begin{cases} (1-\alpha) & (a,t,x) = (1,2,x^{(0)}) \\ \alpha & (a,t,x) = (1,2,x^{(1)}) \\ 0 & \text{else} \end{cases}$$

The allowed codes are:

$$c(0) = \{0,2\} \text{ and } c(1) = \{A,B\}.$$

and the relevant part of the target distribution is:

$$\pi_1^{\text{tape}} y_{2,x^{(i)}} = 1 \cdot A.$$

We have that $\mathscr{C}(V,c) = \Delta\{0,2\} \times \Delta\{A,B\} \cong [0,1]^2$, where we use the coordinates h,k as in (7). For $i \in \{0,1\}$, define polynomials $S_A^{(i)}$ and $S_B^{(i)}$ by:

$$\pi_1^{\text{tape}} \Delta\text{step}_M^2(\mathbf{w} + x^{(i)}, \mathbf{q}) = S_A^{(i)} \cdot A + S_B^{(i)} \cdot B.$$

Using the calculation from that example, we have that:

$$S_A^{(0)} = 1 - (1-h)(1-h^2)(1-k) \qquad S_B^{(0)} = (1-h)(1-h^2)(1-k)$$
$$S_A^{(1)} = 1 - (1-h)(1-h^2)(1-k) - h^3 \qquad S_B^{(1)} = (1-h)(1-h^2)(1-k) + h^3$$

where $\mathbf{w} \in \mathscr{C}(V,c)$ corresponds to $(h,k) \in [0,1]^2$. With $\alpha > 0$, so that $x^{(1)}$ is positively weighted, the only solution is $(h,k) = (0,1)$. If $\alpha = 0$, we only need $(1-h)(1-h^2)(1-k) = 0$, so (h,k) is a solution if $h = 1$ or $k = 1$.

Definition 4.9. The *KL divergence* of an inference problem is:

$$K(\mathbf{w}) = \sum_{t,x} s(t,x)\lambda_{t,x} \sum_{a \in Y_{t,x}} p_{t,x}(a) D_{\text{KL}}\big(\epsilon_\mu \pi_a^{\text{tape}} y_{t,x} \;||\; \epsilon_\mu \pi_a^{\text{tape}} \Delta\text{step}_M^t(x + \mathbf{w}, \underline{\text{init}})\big)$$
$$+ \sum_{t,x} s(t,x)(1-\lambda_{t,x}) D_{\text{KL}}\big(\epsilon_\mu \pi^{\text{state}} y_{t,x} \;||\; \epsilon_\mu \pi^{\text{state}} \Delta\text{step}_M^t(x + \mathbf{w}, \underline{\text{init}})\big).$$

To avoid evaluating the KL divergence on the boundary, we use the injective map $\epsilon_\mu : \Delta Z \longrightarrow \Delta Z$ defined by:

$$\epsilon_\mu(\mathbf{x}) = (1-\mu)\mathbf{x} + \mu\mathbf{b},$$

where $\mathbf{b} = \sum_{z \in Z} \frac{1}{|Z|} \cdot z$ is the barycentre of the simplex, and $\mu \in (0,1)$. This follows [CM19], and is standard in machine learning: see for example the online documentation for Scikit-learn [PVG+11].

We define the RLCT of an inference problem to be $\mathrm{RLCT}_{\mathcal{W}}(K;\varphi)$, where φ is a some nonvanishing prior on the compact set $\mathcal{W} = \mathscr{C}(V,c)$[2].

Lemma 4.10. $K(\mathbf{w}) = 0$ if and only if $\mathbf{w} \in \mathscr{C}(V,c)$ is a solution.

Proof. Given two distributions $\mathbf{v}, \mathbf{w} \in \Delta Z$, we know that:

$$D_{\mathrm{KL}}(\mathbf{v} \,||\, \mathbf{w}) = 0 \iff \mathbf{v} = \mathbf{w}.$$

From this, and the fact that ϵ_μ is injective, the result follows. □

Example 4.11. The KL divergence of the inference problem defined in Example 4.8 is:

$$K^{(\alpha)}(h,k) = (1-\alpha)D_{\mathrm{KL}}(\epsilon_\mu A \,||\, \epsilon_\mu S^{(0)}) + \alpha D_{\mathrm{KL}}(\epsilon_\mu A \,||\, \epsilon_\mu S^{(1)}).$$

Calculating, for $i \in \{0,1\}$:

$$\begin{aligned}
D_{\mathrm{KL}}(\epsilon_\mu A \,||\, \epsilon_\mu S^{(i)}) &= -(1-\mu+\frac{\mu}{|\Sigma|})\log\left[\frac{(1-\mu)S_A^{(i)} + \frac{\mu}{|\Sigma|}}{1-\mu+\frac{\mu}{|\Sigma|}}\right] - \frac{\mu}{|\Sigma|}\log\left[\frac{(1-\mu)S_B^{(i)} + \frac{\mu}{|\Sigma|}}{\frac{\mu}{|\Sigma|}}\right] \\
&= -(1-\mu+\frac{\mu}{|\Sigma|})\log\left[1 - \frac{1-\mu}{1-\mu+\frac{\mu}{|\Sigma|}}(1-S_A^{(i)})\right] - \frac{\mu}{|\Sigma|}\log\left[1 + \frac{1-\mu}{\frac{\mu}{|\Sigma|}}S_B^{(i)}\right] \\
&= -(1-\mu+\frac{\mu}{|\Sigma|})\log\left[1 - \frac{1-\mu}{1-\mu+\frac{\mu}{|\Sigma|}}S_B^{(i)}\right] - \frac{\mu}{|\Sigma|}\log\left[1 + \frac{1-\mu}{\frac{\mu}{|\Sigma|}}S_B^{(i)}\right]
\end{aligned}$$

4.2 Links to statistical learning

In this subsection we will make explicit the way our formulation of an inference problem fits into the theory of statistical learning outlined in Section 3. Intuitively, we can understand samples from the true distribution q as observations of some TM. The state space, $\mathcal{X}$ in (13), contains every possible observation of this TM. Then, the singular learning triple we define compares samples from q to some actual TM, with $\mathbf{w} \in \mathcal{W} := \mathscr{C}(V,c)$ parametrising codes on its tape.

Definition 4.12. The *singular learning triple* (q,p,φ) associated to an inference problem P is defined as follows. Recall that the support of the weighting s is:

$$\mathcal{S}_+(P) = \{(a,t,x) \in \mathcal{S} \mid s(a,t,x) > 0\}.$$

[2]Uniform with respect to the Lesbesgue measure, say, or more generally a Dirictlet distribution [Mac19, §23.5].

The state space is discrete, corresponding to a disjoint union of copies of Q and Σ:

$$\mathcal{X} = \coprod_{z \in \mathcal{S}_+(P)} X_z, \quad \text{where } X_z = \begin{cases} \Sigma & z \in \mathbb{Z} \times \mathbb{N} \times \Sigma^{U,\square} \\ Q & z \in \{\text{state}\} \times \mathbb{N} \times \Sigma^{U,\square} \end{cases} \tag{13}$$

The true distribution, corresponding to the two cases $z = (a, t, x)$ or $z = (\text{state}, t, x)$, is:

$$\begin{aligned} q(\sigma, (a, t, x)) &= s(a, t, x) \cdot \epsilon_\mu \pi_a^{\text{tape}} y_{t,x}(\sigma) \\ q(m, (\text{state}, t, x)) &= s(\text{state}, t, x) \cdot \epsilon_\mu \pi^{\text{state}} y_{t,x}(m). \end{aligned} \tag{14}$$

Noting that we have assumed V is finite, set $N = \sum_{i \in V} |c(i)|$. Then, the model p will be parametrised by the code space $\mathcal{W} := \mathscr{C}(V, c)$, which is semianalytic and compact:

$$\mathcal{W} = \mathscr{C}(V, c) = \prod_{i \in V} \Delta c(i) \subset \mathbb{R}^N.$$

Corresponding to $\mathbf{w} \in \mathcal{W}$, we have the model:

$$\begin{aligned} p(\sigma, (a, t, x)|\mathbf{w}) &= s(a, t, x) \cdot \epsilon_\mu \pi_a^{\text{tape}} \Delta\text{step}_M^t(x + \mathbf{w}, \underline{\text{init}})(\sigma) \\ p(m, (\text{state}, t, x)|\mathbf{w}) &= s(\text{state}, t, x) \cdot \epsilon_\mu \pi^{\text{state}} \Delta\text{step}_M^t(x + \mathbf{w}, \underline{\text{init}})(m) \end{aligned} \tag{15}$$

For convenience, we define the target state $y(z) \in \Delta X_z$ for any $z \in \mathcal{S}_+(P)$, so that for $x \in X_z$:

$$q(x, z) = s(z) \cdot \epsilon_\mu y(z)(x).$$

These are the projections $\pi_a^{\text{tape}} y_{t,x}$ and $\pi^{\text{state}} y_{t,x}$ to which we compare our model $p(-, z|\mathbf{w})$. In a similar way, by $p(z, \mathbf{w})$ we mean the model of (15) without the weighting and perturbation ϵ_μ (this will only be used in the following lemma).

Lemma 4.13. The KL divergence associated to an inference problem P, in Definition 4.9, agrees with the KL divergence of its singular learning triple.

Proof. Using the notation of the previous definition, the KL divergence associated to P is:

$$K(\mathbf{w}) = \sum_{z \in \mathcal{S}_+} s(z) D_{\text{KL}}(\epsilon_\mu y(z) \,||\, \epsilon_\mu p(z, \mathbf{w})),$$

whereas the triple (q, p, φ) is $D_{\text{KL}}(q \,||\, p(\mathbf{w}))$. Expanding the latter expression:

$$\begin{aligned} D_{\text{KL}}(q \,||\, p(\mathbf{w})) &= \sum_{z \in \mathcal{S}_+} \sum_{x \in X_z} s(z) \cdot \epsilon_\mu y(z)(x) \log \left[\frac{s(z) \epsilon_\mu y(z)(x)}{s(z) \epsilon_\mu p(z, \mathbf{w})(x)} \right] \\ &= \sum_{z \in \mathcal{S}_+} s(z) \sum_{x \in X_z} \epsilon_\mu y(z)(x) \log \left[\frac{\epsilon_\mu y(z)(x)}{\epsilon_\mu p(z, \mathbf{w})(x)} \right] \\ &= \sum_{z \in \mathcal{S}_+} s(z) D_{\text{KL}}(\epsilon_\mu y(z) \,||\, \epsilon_\mu p(z, \mathbf{w})). \end{aligned}$$

□

With this in mind, it will be sensible to split up our old definition of the KL divergence, viz:

$$K(\mathbf{w}) = \sum_{z \in \mathcal{S}_+} s(z) D_{\mathrm{KL}}(\epsilon_\mu y(z) \,||\, \epsilon_\mu p(z, \mathbf{w})) =: \sum_{z \in \mathcal{S}_+} s(z) K_z(\mathbf{w}). \tag{16}$$

The following proposition is included for two reasons. Firstly, in the previous definition we assume that the distribution over constraints $s(z)$ is known, as it appears in both the true distribution and in the model. The following result demonstrates that, for a compact synthesis problem, only the support of s affects the RLCT. Secondly, later on we will see that, in the compact case, the RLCT can be calculated using a certain ideal, which also only depends on the support of s.

Proposition 4.14. Fix a compact inference problem P with weighting (U, s). If $s' : \mathcal{S} \to [0, \infty)$ has the same support as s, then the inference problem P' found by changing the weighting to (U, s') has the same RLCT as P.

Proof. Let K (resp. K') be the KL divergence of P (resp. P'). As the common support $\mathcal{S}_+$ of s and s' is finite, define positive constants c_1 and c_2 by

$$c_1 = \min\{s(z)/s'(z) \mid z \in \mathcal{S}_+\}$$
$$c_2 = \max\{s(z)/s'(z) \mid z \in \mathcal{S}_+\}.$$

Then, we see that K and K' are comparable, as,

$$c_1 K' \leq K \leq c_2 K'.$$

Therefore, by Corollary 3.17 $\mathrm{RLCT}_{\mathcal{W}}(K; \varphi) = \mathrm{RLCT}_{\mathcal{W}}(K'; \varphi)$, which proves the proposition. □

The link outlined in this section has practical implications. We will see in Section 4.5 that the formulation of synthesis problems in [CMW21] fits into our definition, which implies relevance of their practical experiments, where a learning machine attempts synthesise programs from a sequence of input/output pairs. In light of this, the results of Section 3 apply to a suitably designed learning machine examining one of our inference problems. This motivates the following definitions.

Definition 4.15. Given a compact inference problem P with associated triple (q, p, φ), the *expected free energy* $\bar{F}(n)$ and *RLCT* (λ, θ) of the inference problem are defined as for the associated statistical learning triple. That is, if $K : \mathcal{W} \to \mathbb{R}$ is the KL divergence of P, then:

$$\bar{F}(n) = -\log \int_{\mathcal{W}} dw\, e^{-n\beta K(w)} \varphi(w)$$

and

$$(\lambda, \theta) = \mathrm{RLCT}_{\mathcal{W}}(K, \varphi).$$

By the proposition, this agrees with the remark in Definition 4.9.

The expected free energy $\bar{F}$ is asymptotically equivalent to $\lambda \log(n) - (\theta - 1) \log\log n$, and predicts the generalisation error of a Bayesian learning machine, as in Section 3.1.

4.3 Program Synthesis on Turing Machines

The motivating example, which we consider in this subsection, is that of a synthesis problem on a pseudo-UTM. This exhibits *program synthesis* as a form of the statistical inference we have considered so far.

Definition 4.16. A *synthesis problem* on a pseudo-UTM $\mathcal{U}$ is an inference problem to which we apply the following constraints.

- The input and code windows are $U \cup U'$ and V respectively, as specified in the design of the UTM (Definition 2.3). If we index V in increasing order, $a_1, \dots, a_{3MN}$, the allowed codes are:
$$c(a_i) = \begin{cases} \Sigma & i \equiv 1 \mod 3 \\ Q & i \equiv 2 \mod 3 \\ \{-1, 0, 1\} & i \equiv 0 \mod 3 \end{cases}$$
- If $x \in (\Sigma_{\mathcal{U}})^{U \cup U'}$ is a positively weighted input, then it decomposes as $x' + z$: an input string $x' \in \Sigma^U$ over the simulation alphabet, and $z \in (\Sigma_{\mathcal{U}})^{U'}$ the background initialisation.
- We put no constraints on the state, that is, $\lambda_{t,x} = 1$ for every positively weighted pair (t, x).
- We examine the tape configuration inside the subset U: that is, the target window $Y_{t,x} = \mathrm{supp}(p_{t,x}) \subset U$.
- The tape part of the target state $y_{t,x}$ lies in the Σ-face of the simplices $\Delta\Sigma_{\mathcal{U}}$: that is, the final state should be a distribution over strings in the simulation alphabet.

A synthesis problem is called *deterministic* if the underlying inference problem is deterministic. That is, if the target state $\pi^{\mathrm{tape}} y_{t,x} \in \Sigma^{U,\square} \subset (\Delta\Sigma)^{U,\square}$.

Note that the code space $\mathcal{W} = \mathscr{C}(V, c) = (\Delta\Sigma \times \Delta Q \times \Delta\{-1, 0, 1\})^{MN}$, so the dimension is:

$$\dim \mathcal{W} = [(M-1) + (N-1) + (3-1)]MN = (M+N)MN.$$

Remark 4.17. We will often specify a deterministic synthesis problem with a subset $D \subset \Sigma^{U,\square}$ and functions $\tau : D \to \mathbb{N}$ and $f : D \to \Sigma^{U,\square}$. Then the positively weighted pairs are $(t, x) = (\tau(x'), x' + z)$ for $x' \in D$, and the target state of belief $y_{t,x}$ has $\pi^{\mathrm{tape}} y_{t,x} = f(x')$. This is the most intuitive form of program synthesis: it asks a learning machine to find a code $\mathbf{w} \in (\Delta\Sigma \times \Delta Q \times \{-1, 0, 1\})^{MN}$ which specifies a TM computing the function f within the timeout τ.

Here the timeout is applied on the UTM — the relationship between time steps on the simulated machine M, and those on $\mathcal{U}$ is a function of the specific UTM, but as $\mathcal{U}$ is part of the data, we will glide over this subtlety. In our general definition, we only demand that the pseudo-UTM mimic the halting behaviour of the simulated machine. As discussed in Remark 2.12, in particular examples we can often do better, in which case it makes sense to apply constraints at intermediate times as well. Similarly, certain pseudo-UTMs might store the state of the simulated machine on the tape. In this case, we might relax the last two requirements on the target window and state (see Example 4.29).

As a preliminary link between this formulation of program synthesis, and the classical, combinatorial version, we now examine the relationship between the RLCT and the Kolmogorov complexity of a given synthesis problem. In particular, this justifies our claim that the geometry of K around a given solution contains semantic information about that code.

Speaking loosely, the Kolmogorov complexity $\mathfrak{c}(\mathcal{D})$ of data $\mathcal{D}$ is the length of the smallest program computing $\mathcal{D}$. For example, let $\mathcal{D}_1 = (3, 1, 4, 1, 5, \dots)$ be the sequence of the first N digits in the decimal expansion of π, and suppose that $\mathcal{D}_2$ is a sequence of N genuinely random integers. Then $\mathfrak{c}(\mathcal{D}_1) < \mathfrak{c}(\mathcal{D}_2)$. In the first case, a short program might use an expansion like:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \dots,$$

whereas a program generating $\mathcal{D}_2$ would have to manually list all N digits. Of course, the value of $\mathfrak{c}(\mathcal{D})$ depends on the programming language used, but only up to an additive constant [Hut04, §2.2.3]. In our context of synthesis problems, we can make a simple definition. The formulation and results here follow [CMW21, §3].

Definition 4.18. Given a synthesis problem P, we suppose that all input strings x take values in the restricted alphabet $\Sigma_{\text{in}} \subset \Sigma$. Then, classical codes $w \in \mathcal{W}$ may specify Turing Machines $T = (\Sigma', Q', \delta)$ for any Σ', Q', where $\Sigma_{\text{in}} \subset \Sigma' \subset \Sigma$ and $Q' \subset Q$, by assigning any arbitrary value to the tuples corresponding to (σ, q) with $\sigma \notin \Sigma'$ or $q \notin Q'$. The Kolmogorov complexity $\mathfrak{c}(P)$ of P is the infimum of $|\Sigma'| \cdot |Q'|$ over Turing Machines T which are solutions to P.

Proposition 4.19. Consider a synthesis problem P on a pseudo-UTM $\mathcal{U}$, with KL divergence K and RLCT $(\lambda, \theta) = \text{RLCT}_{\mathcal{W}}(K; \varphi)$. Recalling the integers $N = |\Sigma|$ and $M = |Q|$ associated to $\mathcal{U}$, we have:

$$\lambda \leq \frac{1}{2}(M + N)\mathfrak{c}(P).$$

Proof. Let $T = (\Sigma', Q', \delta)$ be the TM attaining the infimum defining $\mathfrak{c}(P)$, and $w \in \mathcal{W}$ the associated code. Define $N' = |\Sigma'|$ and $M' = |Q'|$. Let $\tilde{T} = (\Sigma, Q, \tilde{\delta})$ be any TM using the full simulation alphabet and states, but with transition function $\tilde{\delta}$ agreeing with δ on the "fixed" tuples associated to

$$(\sigma, q) \in \Sigma' \times Q' \subset \Sigma \times Q. \tag{17}$$

Then the code $\tilde{w}$ associated to $\tilde{T}$ is also a solution to P — including if there is uncertainty on the tuples corresponding to $(\sigma, q) \in (\Sigma \times Q) \setminus (\Sigma' \times Q')$. This implies that there are $3(MN - M'N')$ squares on which the code may be freely assigned, defining a linear submanifold $\tilde{\mathcal{W}}$ with

$$\dim \tilde{\mathcal{W}} = (M-1)(MN - M'N') + (N-1)(MN - M'N') + (3-1)(MN - M'N').$$

Observing that $\dim \mathcal{W} = (M+N)MN$, this implies that:

$$\operatorname{codim} \tilde{\mathcal{W}} = (M+N)M'N' = (M+N)\mathfrak{c}(P).$$

Since $w \in \tilde{\mathcal{W}}$, [Wat09, Theorem 7.3] implies that the local RLCT $(\lambda', \theta') = \text{RLCT}_{N_w}(K, \varphi)$ satisfies $\lambda' \leq \frac{1}{2}\operatorname{codim} \tilde{\mathcal{W}}$ for any neighbourhood N_w of w. By Lemma 3.15 this value bounds the global RLCT, which implies the result. □

Remark 4.20. This matches the discussion of Section 3.3, as the Kolmogorov complexity is clearly a function of the number of effective parameters. If $T = (\Sigma', Q', \delta)$ has $M'N'$ minimal among solutions of P, then any variation in the parts of its code w corresponding to $(\sigma, q) \in \Sigma' \times Q'$ must change the model. Each such tuple is associated to $M + N$ parameters, for the values of $\delta(\sigma, q) \in \Delta\Sigma \times \Delta Q \times \Delta\{-1, 0, 1\}$, meaning 2λ is indeed related to the number of effective parameters.

We have defined the Kolmogorov complexity with reference to *classical codes*, with no uncertainty in the relevant tuples. *A priori*, this does not interact at all with non-classical codes. In the case of a deterministic synthesis problem, the situation is made clear by the following lemma: if we find a non-classical solution in the interior of some face, the classical codes at the vertices of that face are also solutions.

Lemma 4.21. Consider a solution $\mathbf{w}$ to a deterministic inference problem, and some $i \in V$, such that the projection $\mathbf{w}_i \in \Delta c(i)$ onto the i^{th} tape square lies in the interior of a face of that simplex. Then every $\mathbf{w}'$ found by varying $\mathbf{w}_i$ within that face is also a solution.

Proof. Fix a deterministic inference problem as above. $\mathbf{w}$ is a solution if and only if, for every positively weighted pair $(t, x) \in \mathbb{N} \times \Sigma^{U,\square}$, every $a \in Y_{t,x}$, and every $q' \neq q_{t,x}, \sigma' \neq f_{t,x}(a)$ we have:

$$\begin{aligned} \pi^{\text{state}} \Delta\text{step}^t_M(x + \mathbf{w}, \underline{\text{init}})(q') &= 0 \\ \pi^{\text{tape}}_a \Delta\text{step}^t_M(x + \mathbf{w}, \underline{\text{init}})(\sigma') &= 0. \end{aligned} \tag{18}$$

Each term in the polynomials (2) and (3) which define the smooth relaxation has positive coefficients. Therefore, since the variables $\{\mathbf{y}_i(\sigma) \mid i \in \mathbb{Z}, \sigma \in \Sigma\}$ and $\{\mathbf{q}(s) \mid s \in Q\}$ take non-negative values, its vanishing is controlled by the vanishing of some subset of the variables.

In the situation of this lemma, all variables are fixed, barring the set $\{\mathbf{w}_i(\sigma) \mid i \in W, \sigma \in c(i)\}$ (recall that $c(i) \subset \Sigma$ is the set of allowed codes on tape square i). As such, each of the equations in (18) is a polynomial in these variables, with non-negative coefficients. If $\mathbf{w}'$ is as in the statement, then:

$$\mathbf{w}_i(\sigma) = 0 \implies \mathbf{w}'_i(\sigma) = 0.$$

As such, if $\mathbf{w}$ is such that (18) is satisfied for some t, x, a, q' and σ', then $\mathbf{w}'$ satisfies the same condition. □

Corollary 4.22. If $\mathbf{w}$ is a solution to a non-trivial deterministic synthesis problem, then for at least one $i \in V$, $\mathbf{w}_i$ lies on the boundary of the simplex $\Delta c(i)$.

Proof. Suppose not. Then for every i, $\mathbf{w}_i$ lies in the interior of the $c(i)$ face of $\Delta c(i)$, and so, by the above, we may vary $\mathbf{w}_i$ to any point in $\Delta c(i)$. This implies that every $\mathbf{w} \in \mathscr{C}(V)$ is a solution, so the synthesis problem is trivial. □

4.4 Fibre ideal

The complicated function K is not necessary to specify the collection of solutions $\mathcal{W}_0 = \mathbb{V}_{\mathcal{W}}(K)$. In this section we study an ideal generated by polynomials which cuts out the same set. This gives the link between compact synthesis problems and algebraic geometry.

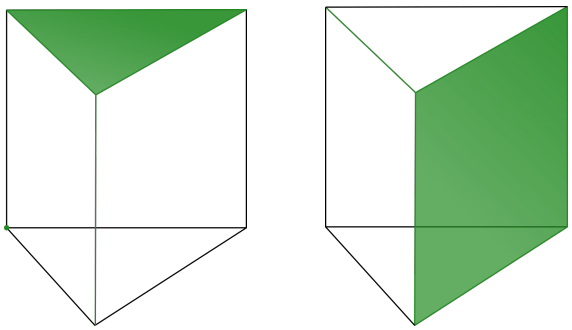

Figure 6: Possible solution sets $\mathcal{W}_0$ to a deterministic inference problem, where $\mathcal{W} \cong \Delta^2 \times \Delta^1$. According to Lemma 4.21, where points on the interior of a face are included, every point in the face is a solution.

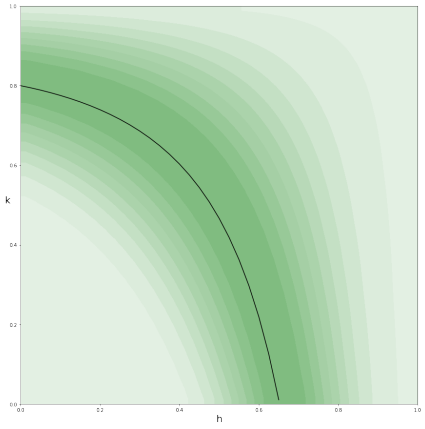

Figure 7: Plot of the posterior distribution corresponding to a non-deterministic inference problem P_0', obtained from P_0 (Example 4.8) by changing the target distribution to $4/5 \cdot A + 1/5 \cdot B$. $\mathcal{W}_0$ is marked by the black line.

Definition 4.23. The fibre ideal $I \subset \mathcal{A}_{\mathcal{W}}$ of an inference problem is generated by the following polynomials in the coordinates of $\mathbf{w}$:

$$\begin{aligned} &\pi_a^{\text{tape}} \Delta\text{step}_M^t(x + \mathbf{w}, \underline{\text{init}})(\sigma) - \pi_a^{\text{tape}} y_{t,x}(\sigma) && (a, t, x) \in \mathcal{S}_+, \sigma \in \Sigma \\ &\pi^{\text{state}} \Delta\text{step}_M^t(x + \mathbf{w}, \underline{\text{init}})(m) - \pi^{\text{state}} y_{t,x}(m) && (\text{state}, t, x) \in \mathcal{S}_+, m \in Q \end{aligned} \tag{19}$$

I defines the same vanishing locus $\mathbb{V}_{\mathcal{W}}(I) = \mathcal{W}_0$ as the KL divergence K. Since V is finite, I is generated by polynomials in a finite number of variables, so it is finitely generated by the Hilbert Basis Theorem [Hil90] — even if $\mathcal{S}_+$ is infinite.

Theorem 4.24. Fix a compact synthesis problem P with fibre ideal $I_P \subset \mathcal{A}_{\mathcal{W}}$, and KL divergence $K : \mathcal{W} \to \mathbb{R}$. Then, in a (small enough) neighbourhood N_w of any solution $w \in \mathcal{W}_0$:

$$\text{RLCT}_{N_w}(K, \varphi) = \text{RLCT}_{N_w}(I, \varphi).$$

Proof. In the notation of Definition 4.12, we can factor K as:

$$\mathcal{W} \xrightarrow{\prod p(z,w)} \prod_{z \in \mathcal{S}_+} \Delta X_z \xrightarrow{\prod D_{\text{KL}}(y(z) \,||\, -)} \prod_{z \in \mathcal{S}_+} \mathbb{R} \xrightarrow{\sum s(z) \cdot -} \mathbb{R}$$

We lose nothing by projecting off a coordinate from each simplex ΔX_z, so set $\Delta_0 X_z \subset \mathbb{R}^{|X_z|-1}$ to be the resulting sets, which have non-empty interior. In order to apply Lemma 3.22, with $\mathcal{W}' = \prod_{z \in \mathcal{S}} \Delta_0 X_z$, we will show that the composite of the last two maps has positive definite Hessian, when evaluated at the true distribution q. As each $s(z) > 0$, it suffices to prove this fact for each map (as in (16)):

$$K_z : \Delta_0 X_z \longrightarrow \mathbb{R}.$$

If we write $q(z) = (q_1, \ldots, q_n) \in \Delta X_z$, the Hessian in question is:

$$\nabla^2 K_z = \text{diag}\left(\frac{1}{q_1}, \ldots, \frac{1}{q_{n-1}}\right) + \frac{1}{q_n} \begin{pmatrix} 1 & \ldots & 1 \\ \vdots & & \vdots \\ 1 & \ldots & 1 \end{pmatrix}.$$

This matrix is positive definite, as, with $\begin{pmatrix} a_1 & \ldots & a_{n-1} \end{pmatrix} \neq 0$:

$$\begin{pmatrix} a_1 & \ldots & a_{n-1} \end{pmatrix} \nabla^2 K_z \begin{pmatrix} a_1 \\ \ldots \\ a_{n-1} \end{pmatrix} = \sum_{i=1}^{n-1} \frac{1}{q_i} a_i^2 + \frac{1}{q_n} \left(\sum_{i=1}^{n-1} a_i\right)^2 > 0.$$

If I is the ideal generated by the polynomials (in $\mathbf{w}$):

$$\begin{aligned} &\epsilon_\mu \pi_a^{\text{tape}} \Delta\text{step}_M^t(x + \mathbf{w}, \underline{\text{init}})(\sigma) - \epsilon_\mu \pi_a^{\text{tape}} y_{t,x}(\sigma) && (a, t, x) \in \mathcal{S}_+, \sigma \in \Sigma \\ &\epsilon_\mu \pi^{\text{state}} \Delta\text{step}_M^t(x + \mathbf{w}, \underline{\text{init}})(m) - \epsilon_\mu \pi^{\text{state}} y_{t,x}(m) && (\text{state}, t, x) \in \mathcal{S}_+, m \in Q, \end{aligned}$$

Then the above analysis implies that $\mathrm{RLCT}_{N_w}(K,\varphi) = \mathrm{RLCT}_{N_w}(I,\varphi)$. The function ϵ_μ only contributes a non-zero constant. For example, in the former case, the polynomial is:

$$\begin{aligned}\epsilon_\mu \pi_a^{\mathrm{tape}} \Delta\mathrm{step}_M^t(x+\mathbf{w}, \underline{\mathrm{init}})(\sigma) - \epsilon_\mu \pi_a^{\mathrm{tape}} y_{t,x}(\sigma) \\ = (1-\mu)\big[\pi_a^{\mathrm{tape}} \Delta\mathrm{step}_M^t(x+\mathbf{w}, \underline{\mathrm{init}})(\sigma) - \pi_a^{\mathrm{tape}} y_{t,x}(\sigma)\big].\end{aligned}$$

As such, the ideal I is exactly the fibre ideal I_P. □

Remark 4.25. In the terminology of [Lin11, §1.5], the proof of the theorem demonstrates that the model p is regularly parametrised, which follows from his Example 1.9. The Hessian of the KL divergence is called the *Fisher Information Metric*, and the condition that it be positive definite is one of the conditions for a model to be "regular". In our case, $\nabla^2 K$ is not (necessarily) positive-definite, so the model is singular.

Remark 4.26. From the definition of $\mathrm{RLCT}_{\mathcal{W}}(I;\varphi)$, the ideal I is interchangeable with the polynomial given by the sum of the squares of its generators. However, using the ideal gives us freedom to pick a convenient set of generators, by [Lin11, Proposition 4.3]. This will simplify calculations.

Corollary 4.27. For any inference problem, the RLCT of its fibre ideal I, on a neighbourhood N_w any solution w, is a lower bound for the RLCT of K on N_w.

Proof. Associate to a subset S' of $\mathcal{S}_+$ the ideal $I_{S'}$ generated by the polynomials in (19) with $z \in S'$. For any S', $I_{S'} \subset I$. Using the Hilbert basis theorem, there is a finite subset $F \subset \mathcal{S}_+$ such that $I_F = I$, and as such,

$$\mathrm{RLCT}_{N_w}(I_F;\varphi) = \mathrm{RLCT}_{N_w}(I;\varphi).$$

As in (16), we can rewrite the KL divergence as:

$$K(\mathbf{w}) = \sum_{z\in\mathcal{S}} s(z) K_z(\mathbf{w}),$$

where $K_z : \Delta X_z \to \mathbb{R}$ are non-negative. By Theorem 4.24, if we set

$$K_F(\mathbf{w}) = \sum_{z\in F} s(z) K_z(\mathbf{w}),$$

then $\mathrm{RLCT}_{N_w}(K_F;\varphi) = \mathrm{RLCT}_{N_w}(I_F;\varphi)$. ($K_F$ is not strictly the KL divergence of an inference problem, as s is not a probability distribution. As observed in Remark 4.7 this is not important, and the proof of the Theorem goes through regardless.) Observing that $K_F \leq K$ and applying Lemma 3.16, this implies

$$\mathrm{RLCT}_{N_w}(I;\varphi) = \mathrm{RLCT}_{N_w}(K_F;\varphi) \leq \mathrm{RLCT}_{N_w}(K;\varphi).$$

□

4.5 Examples

Example 4.28. We wrap up the discussion of the Shift Machine, using the techniques developed in the previous section. First let us recall the definitions and results of Examples 2.11 and 4.8. The machine is initialised to one of the configurations we represented as:

$$x^{(0)} = \ldots \square \, \nu_{\text{counter}} \, \nu_{string} \, A \, A \square \ldots$$

$$x^{(1)} = \ldots \square \, \nu_{\text{counter}} \, \nu_{string} \, A \, B \square \ldots$$

where the unknown distributions are parametrised by $(h, k) \in [0, 1]^2$, according to:

$$\begin{aligned} \nu_{\text{counter}} &= (1-h) \cdot 0 + h \cdot 2 \\ \nu_{\text{string}} &= (1-k) \cdot B + k \cdot A. \end{aligned}$$

The machine is run for two steps, and the distribution is read off one square to the right of the head. In the two input cases $x^{(i)}$ the distribution we read off is $S^{(i)}(A) \cdot A + S^{(i)}(B) \cdot B$, where:

$$\begin{aligned} S^{(0)}(B) &= (1-h)(1-h^2)(1-k) \\ S^{(1)}(B) &= (1-h)(1-h^2)(1-k) + h^3 \end{aligned}$$

We set the true distribution to be $1 \cdot A$ in both cases.

We now calculate the fibre ideal and RLCT of the inference problems P_α in Example 4.8. We have two elements $(1, \tau, x^{(i)}) \in \mathcal{S}_+$, so the non-zero generating polynomials associated to $(1, \tau, x^{(i)})$ are:

$$\begin{aligned} S^{(i)}(A) - 1 \quad &\text{if } z = (1, \tau, x^{(i)}), \sigma = A \\ S^{(i)}(B) \quad &\text{if } z = (1, \tau, x^{(i)}), \sigma = B \end{aligned}$$

As $S^{(i)}(B) = 1 - S^{(i)}(A)$, we have that the fibre ideal I_α for varying $\alpha \in [0, 1]$ is:

$$I_\alpha = \begin{cases} \langle (1-h)(1-h^2)(1-k) \rangle & \alpha = 0 \\ \langle (1-h)(1-h^2)(1-k), h^3 \rangle & \alpha \in (0, 1) \\ \langle (1-h)(1-h^2)(1-k) + h^3 \rangle & \alpha = 1 \end{cases}$$

When $\alpha = 0$ the global RLCT is of the form (up to a linear coordinate change) of Proposition 3.11, namely the vanishing of a monomial in the positive orthant. That is:

$$\text{RLCT}_{\mathcal{W}}(\langle (1+h)(1-h)^3(1-k) \rangle; 1) = \text{RLCT}_{\mathbb{R}^2_{\geq 0}}(x^6 y^2; 1) = (1/6, 1).$$

(Note that we consider the square of our generator, and that the factor $(1+h)$ is positive on $\mathcal{W}$.) In the local case, around some solution w_0, we have one of $w_0 = (1, k)$, $w_0 = (h, 1)$ or $w_0 = (1, 1)$. In the first and last cases, the local RLCT achieves the global minimum, but in the second we have (for $h \neq 1$):

$$\text{RLCT}_{N_{(h,1)}}((1-h)^6(1-k)^2; 1) = (1/2, 1),$$

as the highest order vanishing monomial is $(1-k)^2$. This indicates that the posterior will favour solutions with $h = 1$ (as shown in Figure 8).

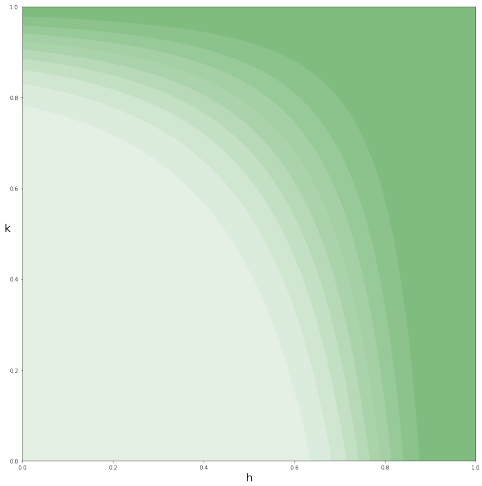

Figure 8: A graph of $e^{-nK^{(0)}(h,k)}$, to which the posterior at is proportional at $\alpha = 0$. Observe that it is larger in a neighbourhood of $h = 1$ than the same neighbourhood around $k = 1$.

For $\alpha \in (0, 1)$, we have a unique solution $w_0 = (0, 1)$. In a small neighbourhood N of $(0, 1)$, $1 - h$ is a unit, so the ideal in question is $\langle 1 - k, h^3 \rangle = \langle 1 - k \rangle + \langle h^3 \rangle \subset \mathcal{A}_N$. Using the sum rule of Lemma 3.21 we have $\mathrm{RLCT}_N(I_\alpha; 1) = (2/3, 1)$.

For $\alpha = 1$, again $\mathcal{W}_0 = \{(0, 1)\}$. Observe that, on N, our generator $(1 - h)^2(1 - k) + h^3$ is comparable to $1 - k + h^3$. As such, and using a linear coordinate change that takes $w_0 \mapsto (0, 0) \in N'$:

$$\mathrm{RLCT}_N(I_\alpha; 1) = \mathrm{RLCT}_{N'}(\langle x^3 + y \rangle; 1) = (2/3, 1).$$

Here we use the calculation of Example 3.23, and the relationship observed in (12).

Example 4.29 (`parityCheck` and `detectA`)**.** In [CMW21], two synthesis problems are considered from a practical point of view, which we can now fit into our definitions. Two slight complications should be noted. Firstly, the *staged pseudo-UTM* defined in Appendix E of their paper is multi-tape, but this can be reconciled with our single-tape presentation using the techniques of [Xu21]. Secondly, the synthesis problems place constraints on the final state of the simulated machine, but this is minor as the pseudo-UTM they consider has a tape location holding this state.

First, consider `parityCheck`. For this example, the pseudo-UTM has simulation alphabet and states:

$$\Sigma = \{\square, A, B, X\}$$
$$Q = \{\mathrm{reject}, \mathrm{accept}, \mathrm{getNextAB}, \mathrm{getNextA}, \mathrm{getNextB}, \mathrm{gotoStart}\},$$

with the lexicographic ordering on $\Sigma \times Q$ induced from the above. The code and input windows are on different tapes, and the background initialisation is described in [CMW21, Appendix E]. Here $N = |\Sigma| = 4$ and $M = |Q| = 6$.

Let $a_q \in \mathbb{Z}$ be the index of the tape square holding the state of the simulated machine, and let $T \in \mathbb{N}$ be the timeout on the UTM corresponding to 42 steps of the simulated machine [CMW21, Table 3]. The input strings x will be sampled from $\{A, B\}^l$, where $1 \leq l \leq b$, where

for the three experiments $b \in \{5, 6, 7\}$. More specifically, for some fixed b, the weighting $s(x)$ over inputs is:

$$s = \frac{1}{b} \sum_{l=1}^{b} \sum_{x \in \{A,B\}^l} 2^{-l} \cdot x.$$

For any x with $s(x) > 0$, we have a single timeout $s(t = T|x) = 1$, and $p_{T,x}(a_q) = 1$ is the only tape square constrained. The target $\pi^{\text{tape}}_{a_q} y_{t,x}$ is accept if x contains an equal number of A and B, and reject otherwise (recall that the simulation states Q are contained in the alphabet $\Sigma_{\mathcal{U}}$ of the UTM).

This synthesis problem asks a learning machine to find a code

$$\mathbf{w} \in (\Delta\Sigma \times \Delta Q \times \Delta\{-1, 0, 1\})^{MN},$$

which checks the parity of an input string. The total number of parameters is $(M + N)MN = 240$, but the statistical estimates of the RLCT for $b = 5, 6, 7$ are $\lambda = 4.4, 4.0$ and 3.9 respectively.

The situation for `detectA` is similar. Now we consider a pseudo-UTM with simulation alphabet and states:

$$\Sigma = \{\square, A, B\}$$
$$Q = \{\text{reject}, \text{accept}\},$$

With the code and input windows as above. Similarly, we apply a constraint to the state of the simulated machine, held on tape square a_q, now at time T corresponding to 10 steps of the simulated machine. The input strings x are sampled from $\{A, B\}^*$ in the same way, with $1 \leq l \leq b < 10$. Having constructed the weighting in the same way, the target distribution is accept if x contains an A, and reject otherwise.

In this simple example, we can illustrate explicitly the discussion about effective parameters in Section 3.3, extending the treatment of Kolmogorov complexity. Suppose that the (simulated) machine is initialised to the state "reject", and with the first character of x under the head. Then, there is an obvious solution which scans rightwards through the string until it finds an A. We can represent such a transition function δ in a table.

	$\square$	A	B
reject	$(\square, \text{reject}, 0)$	$(A, \text{accept}, 0)$	$(B, \text{reject}, 1)$
accept	$(\square, \text{accept}, 0)$	$(A, \text{accept}, 0)$	$(B, \text{accept}, 0)$

Some of these parameters don't matter, however. For any values in the spaces left blank the following is also a solution:

	$\square$	A	B
reject	$(\square, \text{reject}, -)$	$(-, \text{accept}, -)$	$(B, \text{reject}, 1)$
accept	$(-, \text{accept}, -)$	$(-, \text{accept}, -)$	$(-, \text{accept}, -)$

This defines an 18-dimensional submanifold of $\mathcal{W}$, which has dimension 30. By the results of Section 3.3, this implies that the RLCT of `detectA` is bounded above by $\frac{1}{2}(30 - 18) = 6$. Even

with relatively short Markov chains, estimates of the RLCT consistent with this prediction are found in [CMW21, Table 1], which demonstrates that the semantic information we have extracted by hand is present in the dynamics of the learning process.

5 Lattice of Inference Problems

In this section, we explore the structure of the collection of inference problems on a fixed Turing Machine. The results of the previous section allow us to extract geometric information about programs, by seeing them as singularities of some fibre ideal. The question that remains is "which fibre ideal?", or what is the same, "which inference problem?" The definition of the fibre ideal suggests a natural lattice structure on inference problems, which we claim illuminates the semantic content of this choice.

The *specialisation preorder* underlying our lattice is developed in Section 5.1, and in Section 5.2 we define a function from codes into inference problems. Lattices of this form arise in program semantics and in logic (see Remark 5.12), which suggests the structure we develop in this section is natural. On the other hand, the process of learning (as in [Tur04]) is reflected by limits in the lattice. To back up our semantic interpretation, we provide some example situations where the geometric information that inference problems provide helps to get at the meaning behind the irrelevancies of syntax.

Appendix A will expand on the relationship encapsulated by specialisation, in terms of thermodynamics. In particular, the process of learning is represented by second-order phase transitions.

In this section, fix a TM M, code window V and specification of allowed codes c.

5.1 Specialisation Preorder

We recall some notations for an inference problem P. The support $\mathcal{S}_+(P)$ is the support of the weighting s, as in Definition 4.6. For $a \in \mathbb{Z} \cup \{\text{state}\}$, and $(t, x) \in \mathbb{N} \times \Sigma^{U,\square}$ the *true distribution* $y(a, t, x)$, defined in Section 4.2, is the relevant part of the target distribution $y_{t,x}$. Specifically:

$$y(a, t, x) = \pi_a^{\text{tape}} y_{t,x}, \text{ for } a \in \mathbb{Z}$$
$$y(\text{state}, t, x) = \pi^{\text{state}} y_{t,x}$$

Define formal inference problems $\bot, \top$, called overspecified and underspecified respectively. With a choice of code space $\mathcal{W}$, define solution sets $\mathcal{W}_0(\bot) = \emptyset$ and $\mathcal{W}_0(\top) = \mathcal{W}$, and fibre ideals $I_\bot = \mathcal{A}_\mathcal{W}$, $I_\top = (0)$. Formally define $\text{RLCT}_\mathcal{W}(\bot; \varphi) = (\infty, -)$ and $\text{RLCT}_\mathcal{W}(\top; \varphi) = (0, -)$.

Definition 5.1. $\mathcal{I}^{\text{pre}}(M, V, c)$ is the set of inference problems on the machine M, with the fixed code window V and allowed codes c. We include the two formal problems $\bot, \top$.

Definition 5.2 (Specialisation Preorder)**.** Given ordinary $P, P' \in \mathcal{I}^{\text{pre}}(M, V, c)$, P' is said to be a *specialisation* of P, denoted $P \sqsubseteq P'$, if:

1. The support $\mathcal{S}_+(P')$ of P' contains that of P: $\mathcal{S}_+(P) \subset \mathcal{S}_+(P')$, and,
2. For each $z \in \mathcal{S}_+(P)$, the target states $y(z)$ and $y'(z)$ of the two synthesis problems agree.

In short, $P \sqsubseteq P'$ if P' applies all of the constraints that P does. If $P \sqsubseteq P'$, we will occasionally call P a *generalisation* of P'. For every $P \in \mathcal{I}^{\text{pre}}(M, V, c)$ define $P \sqsubseteq \bot$ and $\top \sqsubseteq P$.

In algebraic geometry, given a containment of ideals $I \subset J$, we have a reverse containment of their vanishing loci $\mathbb{V}(I) \supset \mathbb{V}(J)$. Having defined the fibre ideal $I_P \subset \mathcal{A}_\mathcal{W}$ associated to an

inference problem P, it is natural to consider the relationship between problems P and P' for which $I_P \subset I_{P'}$. Since $\mathbb{V}(I_P) = \mathbb{V}(K_P) = \mathcal{W}_0(P)$, this will imply that $\mathcal{W}_0(P) \supset \mathcal{W}_0(P')$. The preorder $\sqsubseteq$ we have defined fulfils this, as we see in Proposition 5.5. The symbol $\sqsubseteq$ should be read as "specialises to".

Our definition arises naturally out of the geometric interpretation of program synthesis that we have given. Alternatively, viewing inference problems as propositions, we will see in Remark 5.12 that specialisation $P \sqsubseteq P'$ is dual to the entailment "P' implies P."

Remark 5.3. The term specialisation is borrowed from topology, where the equivalent ordering is denoted $\rightsquigarrow$. For points x, x' in a topological space X, $x \rightsquigarrow x'$ if $x' \in \overline{\{x\}}$; ie if x' belongs to every closed set that contains x [Sta18, Tag 0060]. If $\mathfrak{p}, \mathfrak{q}$ are points (prime ideals) of the spectrum of a ring, then:

$$\begin{aligned} \mathfrak{p} \rightsquigarrow \mathfrak{q} &\iff \mathfrak{q} \in \overline{\{\mathfrak{p}\}} \\ &\iff \mathfrak{q} \in \mathbb{V}(\mathfrak{p}) \\ &\iff \mathfrak{p} \subset \mathfrak{q}. \end{aligned}$$

Example 5.4. For any inference problem P, there is an obvious generalisation Q corresponding to any subset $\mathcal{S}_+(Q) \subset \mathcal{S}_+(P)$. Namely, we set:

$$s_Q(z) = \begin{cases} \frac{1}{C} \cdot s_P(z), & z \in \mathcal{S}_+(Q) \\ 0 & \text{else} \end{cases}$$

with $C = \sum_{z \in \mathcal{S}_+(Q)} s_P(z)$, and keep the target distribution the same. This corresponds to simply omitting the constraints indexed by $z \in \mathcal{S}_+(P) \setminus \mathcal{S}_+(Q)$.

Proposition 5.5. The ordering $\sqsubseteq$ is a preorder on the set $\mathcal{I}^{\text{pre}}(M, V, c)$. For any inference problems $P, P' \in \mathcal{I}^{\text{pre}}(M, V, c)$, if $P \sqsubseteq P'$ then $I_P \subset I_{P'}$ and $\mathcal{W}_0(P) \supset \mathcal{W}_0(P')$. If P is compact, then specialisation $P \sqsubseteq P'$ does not decrease the RLCT.

Proof. The statements are clear in the case of the formal problems.

That $(\mathcal{I}^{\text{pre}}(M, V, c), \sqsubseteq)$ is a preorder follows from the facts that (a) $\subset$ is a partial order, and (b) equality is transitive. It fails to be a partial order as it only depends on the support of the weighting.

The second statement is immediate from the definitions. The generators of I_P are a subset of the generators of $I_{P'}$, proving the first assertion, and the second follows from the fact that any solution of P' must also satisfy the constraints applied by P.

The final statement follows directly from the previous, using Theorem 4.24 for the equals sign, and Corollaries 3.20 and 4.27 for the inequalities:

$$\text{RLCT}_{\mathcal{W}}(P; \varphi) = \text{RLCT}_{\mathcal{W}}(I_P; \varphi) \leq \text{RLCT}_{\mathcal{W}}(I_{P'}; \varphi) \leq \text{RLCT}_{\mathcal{W}}(P'; \varphi).$$

□

It is sometimes simpler to think of a synthesis problem $P \in \mathcal{I}^{\text{pre}}(M, V, c)$ as consisting of its support $\mathcal{S}_+$, and the family $y(z) \in \Delta X_z$ of target states: that is, to ignore specifics of the weighting $s(z)$. As we have shown, for compact P the RLCT is indeed independent of these details, and so, at least up to leading order in n, the expected free energy $\bar{F}(n)$ is independent also. This motivates the following definition, which is easily seen to be equivalent. The ordering $\sqsubseteq$ descends to the quotient, and defines a partial order on $\mathcal{I}(M, V, c)$ [Vic89, Proposition 3.2.4].

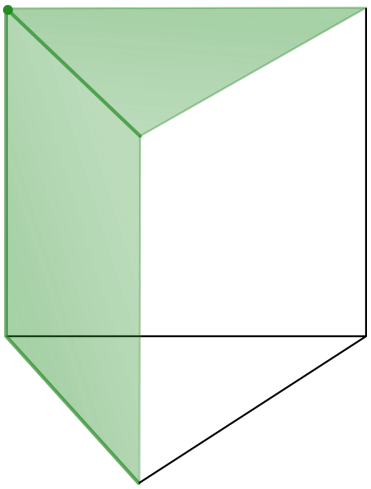

Figure 9: Solution sets $\mathcal{W}_0 \subset \mathcal{W} \cong \Delta^2 \times \Delta^1$ to inference problems $P \sqsubseteq Q \sqsubseteq R$.

Definition 5.6. The set $\mathcal{I}(M, V, c)$ is the quotient of $\mathcal{I}^{\text{pre}}(M, V, c)$ by the equivalence relation $\sim$, defined by:

$$P \sim P' \iff P \sqsubseteq P' \text{ and } P' \sqsubseteq P. \tag{20}$$

We identify problems $P \in \mathcal{I}^{\text{pre}}(M, V, c)$ with their equivalence class in $\mathcal{I}(M, V, c)$.

Beyond the order, the partially-ordered set of ideals in the ring $\mathcal{A}_{\mathcal{W}}$ the structure of a lattice. For ideals I and J, the *meet*, or greatest lower bound, is the intersection $I \cap J$, and the *join*, or least upper bound, is the sum $I + J$. We can extend this structure to the collection of synthesis problems.

Definition 5.7 (Sum)**.** Two ordinary inference problems $P, Q \in \mathcal{I}^{\text{pre}}(M, V, c)$ are said to be *consistent* if on the intersection of their supports, $z \in \mathcal{S}_+(P) \cap \mathcal{S}_+(Q)$, their target states agree: $y_P(z) = y_Q(z)$. For consistent P, Q, we define their *sum* as the inference problem applying all the constraints of P and Q. More precisely,

- The support of $P+Q$ is the union of the supports of P and Q: $\mathcal{S}_+(P+Q) = \mathcal{S}_+(P) \cup \mathcal{S}_+(Q)$,
- For $z \in \mathcal{S}_+(P+Q)$, the weighting is $s_{P+Q}(z) = \frac{1}{2}(s_P(z) + s_Q(z))$,
- The target state $y_{P+Q}(z)$ is whichever of $y_P(z)$ or $y_Q(z)$ is defined.

If P and Q are inconsistent, define $P + Q = \bot$, and let $P + \bot = \bot$, $P + \top = P$. Observe that $P, Q \sqsubseteq P + Q$. Indeed, if $T \in \mathcal{I}^{\text{pre}}(M, V, c)$ is any inference problem such that $P \sqsubseteq T$ and $Q \sqsubseteq T$, then $P + Q \sqsubseteq T$.

Definition 5.8 (Intersection)**.** For ordinary inference problems $P, Q \in \mathcal{I}^{\text{pre}}(M, V, c)$, define:

$$\mathcal{S}_+(P \cap Q) = \{z \in \mathcal{S}_+(P) \cap \mathcal{S}_+(Q) \mid y_P(z) = y_Q(z)\}.$$

If this set is empty, set $P \cap Q = \top$. For z in this set, define:

$$s_{P \cap Q}(z) = \frac{1}{C}(s_P(z) + s_Q(z)) \quad \text{where } C = \sum_{z' \in \mathcal{S}_+(P \cap Q)} s_P(z') + s_Q(z').$$

In the formal case, make the expected definitions $P \cap \top = \top$ and $P \cap \bot = P$. Again, $P \cap Q \sqsubseteq P, Q$, and is universal with this property.

Example 5.9. Recall the family P_α of inference problems on the Shift Machine, defined in Example 4.8. For $\alpha \in (0,1)$, we have a pair of specialisations $P_0 \sqsubseteq P_\alpha$ and $P_1 \sqsubseteq P_\alpha$, which come from adding a constraint on one of the inputs $x^{(i)}$. $P_{\frac{1}{2}}$ is precisely the sum $P_0 + P_1$.

Proposition 5.10. For synthesis problems $P, Q \in \mathcal{I}^{\text{pre}}(M, V, c)$, $I_{P+Q} = I_P + I_Q$ and $I_{P \cap Q} \subset I_P \cap I_Q$.

Proof. First examine the sum. If P and Q are consistent, the statement is clear, as the set of generators of I_{P+Q} is the union of the generating sets of I_P and I_Q. Similarly, it is clear in the formal case.

If P and Q are inconsistent, let $z \in \mathcal{S}_+(P) \cap \mathcal{S}_+(Q)$, with $y_P(z) \neq y_Q(z)$. If $y_P(z)$ and $y_Q(z)$ differ on the coordinate $x \in X_z$, and $p(x|\mathbf{w})$ is the model associated to the triple (M, V, c), we have:

$$p(x, z|\mathbf{w}) - y_P(x, z) \in I_P$$
$$p(x, z|\mathbf{w}) - y_Q(x, z) \in I_Q$$

As such, the sum $I_P + I_Q$ contains the constant $y_P(x, z) - y_Q(x, z) \neq 0$. Since this constant is a unit, $I_P + I_Q = \mathcal{A}_\mathcal{W} = I_\bot$, so we are done.

For the intersection and in the ordinary case, observe that the generators of $I_{P \cap Q}$ are a subset of each of the generating sets for I_P and I_Q. The claim is clear in the formal case. □

Proposition 5.11. With the induced ordering $\sqsubseteq$, $\mathcal{I}(M, V, c)$ is a complete lattice. For $P, Q \in \mathcal{I}^{\text{pre}}(M, V, c)$, the equivalence classes of the supremum and infimum of $\{P, Q\} \subset \mathcal{I}(M, V, c)$ are represented by $P + Q$ and $P \cap Q$ respectively.

Proof. The second statement is immediate from the universality claims in Definitions 5.7 and 5.8. It suffices, then, to prove the existence of arbitrary least upper bounds [DP02, Theorem 2.31].

Take some set $\{P_i\}_{i \in I} \subset \mathcal{I}^{\text{pre}}(M, V, c)$ of inference problems. if any pair P_i, P_j are inconsistent, the supremum is $\bot$. Otherwise, define $P \in \mathcal{I}^{\text{pre}}(M, V, c)$ by:

$$\mathcal{S}_+(P) = \bigcup_{i \in I} \mathcal{S}_+(P_i),$$

and with $y(z)$ agreeing with $y_{P_i}(z)$ for every i. Choosing any weighting s_P, P defines an equivalence class in $\mathcal{I}(M, V, c)$ with $P_i \sqsubseteq P$ for every i. As before, P is universal with this property, so:

$$P = \sup_{i \in I} \{P_i\}.$$

□

Motivated by the second part of this statement, we will denote arbitrary least upper bounds by a sum.

Remark 5.12. As we have alluded to, we can make sense of the lattice structure on $\mathcal{I}(M, V, c)$ by seeing inference problems as propositions. Here we have two distinct perspectives on program synthesis, one from geometry and one from logic, which give rise to the same structure. This suggests that the formulation of inductive inference that leads to this lattice structure is canonical in some way — or at least makes clear some very natural concepts.

In logic, especially in quantum physics, propositions (or "yes-no experiments") about a physical system are given a lattice structure, with $p \leq q$ whenever a proof of p suffices to demonstrate that q is true [OP93, pp.1-2]. More precisely, suppose that the state of a quantum system is given by a ray in the Hilbert space $\mathfrak{S}$. Then, an argument of von Neumann [vN55, §III.5] says that a proposition about this state is an observable (roughly a self-adjoint operator [vN55, §IV.2]), with eigenvalues 0 and/or 1. This is precisely a projection p onto a closed linear subspace $\mathfrak{M}_p$. Then, the partial order $p \leq q$ comes from $pq = p$ [OP93], which implies $\mathfrak{M}_p \subset \mathfrak{M}_q$.

In program synthesis, a state is a code $\mathbf{w} \in \mathcal{W}$, and an equivalence class $P \in \mathcal{I}(M, V, c)$ is a proposition about this state. The closed subspaces $\mathfrak{M}_p$ correspond to solution sets $\mathcal{W}_0(P)$, and as such our relation $P \sqsubseteq P'$ might be interpreted as $P \geq P'$, as in this case $\mathcal{W}_0(P) \supset \mathcal{W}_0(P')$ by Proposition 5.5.

Using the finer relation on $\mathcal{I}^{\text{pre}}(M, V, c)$, and the results of SLT, we can do better than a projection onto $\mathcal{W}_0$. Namely, gradient descent against K gives a way of *flowing* all of $\mathcal{W}$ onto the solution set. Similarly, integration against the posterior distribution refines restriction to the solution set (and indeed formally reduces to it as $\beta \to \infty$ [Wat09, Remark 1.13]). By rediscovering a natural logical structure geometrically, this discussion vindicates our method for studying program synthesis.

5.2 Mathematical semantics of programs

As alluded to in the introduction, Scott finds in [Sco77] a mathematical interpretation of *data types* as complete lattices. To apply his ideas, we interpret our lattice $\mathcal{I}(M, V, c)$ in these terms, and this understanding extends to codes $\mathbf{w}$ via a map $[\![-]\!] : \mathscr{C}(V, c) \to \mathcal{I}(M, V, c)$. In the case where M is a UTM, this justifies the claim that synthesis problems provide a semantics of programs. The geometry we have developed enriches this semantics with information about the construction of a program by *learning*.

Before we define in detail the Scott-style semantics we obtain from our preorder, we sketch an example of how the geometry of learning can give us semantic information about programs. The role of semantics, here and elsewhere, is somewhat analogous to that of topological invariants: in theory, two spaces can be differentiated by examining their open sets directly, but in many cases it is easier to differentiate them based on their, say, homology.

Example 5.13. Fix a pseudo-UTM $\mathcal{U}$ and a synthesis problem P, and let $\mathbf{w} \in \mathcal{W}$ be some solution. If we let the simulation states of $\mathcal{U}$ be:

$$Q = \{\text{init}, \text{halt}, q_1, \ldots, q_n\},$$

then any permutation $f \in S_n$ of $\{1, \ldots, n\}$ acts naturally on Q, and as such on $\mathcal{W}$. Specifically, if $\delta_{\mathbf{w}}$ is the transition function coded by $\mathbf{w}$, then set:

$$\delta_{f(\mathbf{w})}(\sigma, q_{f(i)}) = (\sigma', q_{f(i')}, d), \quad \text{where } \delta_{\mathbf{w}}(\sigma, q_i) = (\sigma', q_{i'}, d).$$

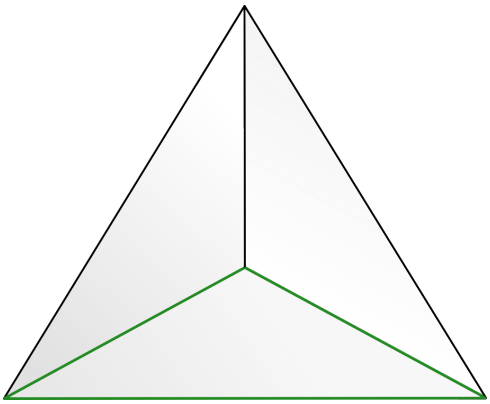

Figure 10: A symmetrical arrangement of solutions $\mathcal{W}_0$ inside $\mathcal{W} \cong \Delta^3$. If the germs $(\mathbf{w}_i, K)$ in the bottom corners are isomorphic, via a rotation of $\mathcal{W}$, then we expect that the solutions they code for are not semantically different.

Assuming that only the states init and halt appear in P, the other "working" states $q_1, \ldots, q_n$ are entirely interchangeable, and so the maps f we define make no difference to the tape configuration of the coded machine. Therefore, the syntactically different solutions $\mathbf{w}$ and $f(\mathbf{w})$ have no semantic difference.

This identity between solutions is detected by the geometry. Let K be the KL divergence of P. Then, as permuting the states does not affect the tape configuration, we have $K \circ f(\mathbf{w}) = K(\mathbf{w})$. Therefore, the local geometry around $\mathbf{w}$ and $f(\mathbf{w})$ is identical. (Precisely, the complex space germs $(\mathbf{w}, K)$ and $(f(\mathbf{w}), K)$ are isomorphic [GLS07].)

This is not a deep example — the equivalent in `Python` might be changing the names of some variables — but it does illustrate how examining the geometry of K can allow us to "forget" the irrelevancies of syntax.

In this simple case, the isomorphism of complex space germs has clear semantic content. More generally, structuring the geometry we have defined will help to provide these interpretations, and the Scott-style semantics that follows from Section 5.1 will provide this structure. Before making this link more formal, we sketch Scott's definitions.

Setup 5.14. In [Sco77] Dana Scott develops an early mathematical semantics of computation[3]. In particular, a *data type* D is a complete lattice, where for two objects $x, y \in D$, $x \sqsubseteq y$ means that the information in y is *consistent* with that of x, and possibly more detailed. The idea is that we specify objects $y \in D$ by a sequence of approximations:

$$\cdots \sqsubseteq x_i \sqsubseteq x_{i+1} \sqsubseteq \ldots$$

so that y is the limit — or the least upper bound — of the directed set $\{x_i\}$. This is clearer in an example. Scott defines $\mathcal{R}$, the type of real numbers, to consist of closed intervals $[x, x'] \subset \mathbb{R}$, with ordering:

$$[x, x'] \sqsubseteq [y, y'] \text{ iff } x \leq y \leq y' \leq x'.$$

[3]Interestingly, it was a version of Scott's idea that led Jean-Yves Girard to define Linear Logic, which ultimately underlies the smooth relaxation we have defined [Gir11, §8].

The usual real numbers embed as the one-point intervals $x = [x, x]$, which we specify as the limit (intersection, in this case) of a sequence of intervals with rational endpoints:

$$[q_1, q_1'] \sqsubseteq [q_2, q_2'] \sqsubseteq \cdots \sqsubseteq [x, x].$$

The crux of Scott's idea is that the lattice has an *effectively given basis*[4] (Axiom 5 of *loc. cit.*), from which we specify general elements as a directed limit. In the example, the basis is the collection of intervals $[q, q']$ with rational endpoints, which is countable and recursively enumerable. A *computable* element of the data type is one specified as the limit of a sequence of elements from the basis.

On a surface level, there is a clear analogy with the lattice we have defined. Inference problems P consist of data, and if $P \sqsubseteq P'$ then P' adds to the data present in P. The following result demonstrates that any inference problem P may be approximated by compact inference problems. This is our suggestion for a counterpart to Scott's basis in program synthesis.

Proposition 5.15 (Compact approximation)**.** Each inference problem $P \in \mathcal{I}(M, V, c)$ is the directed limit of compact inference problems.

Proof. Let some $P \in \mathcal{I}(M, V, c)$ be given. For each $z \in \mathcal{S}_+(P)$, let $B_{z,P}$ be the *basic* inference problem with support $\mathcal{S}_+(B_{z,P}) = \{z\}$, and target state $y_{B_{z,P}}(z) = y_P(z)$. By definition,

$$B_{z,P} \sqsubseteq P.$$

Let $\mathcal{J}_P \subset \mathcal{I}(M, V, c)$ be the set of inference problems containing every $B_{z,P}$, and all of their finite sums. To give a definite representative, if $Q = \sum_{i=1}^n B_{z_i,P}$, we can take the weighting from P:

$$s_Q(z) = \frac{1}{C} \begin{cases} s_P(z) & z \in \{z_1, \ldots, z_n\} \\ 0 & \text{else} \end{cases} \qquad \text{where } C = \sum_{i=1}^n s_P(z_i).$$

Since $\mathcal{S}_+(Q) = \{z_1, \ldots, z_n\}$, every such finite join is compact. Also, $Q \sqsubseteq P$.

Suppose that $T \in \mathcal{I}(M, V, c)$ is such that $Q \sqsubseteq T$ for every $Q \in \mathcal{J}_P$. This implies that:

$$\mathcal{S}_+(P) \subset \mathcal{S}_+(T),$$

as the latter set contains $\mathcal{S}_+(B_{z,P}) = \{z\}$ for every $z \in \mathcal{S}_+(P)$. Also, the two problems must be compatible, as T is assumed to be compatible with every $\mathcal{S}_+(B_{z,P})$. Therefore, $P \sqsubseteq T$, which implies that P is a limit of the directed set $\mathcal{J}_P$. □

We now want to use the lattice $\mathcal{I}(M, V, c)$ to understand *programs*, which applies in the case that M is a UTM. In aid of this, the following definition embeds codes $\mathbf{w}$ into the lattice.

Definition 5.16. For any $\mathbf{w} \in \mathscr{C}(V, c)$, define $[\![\mathbf{w}]\!] \in \mathcal{I}(M, V, c)$ to be supported on all of $\mathcal{S}$, and have the target state of belief $y_{t,x} = \Delta\text{step}_M^t(x + \mathbf{w})$. That is, for the two cases of $z \in \mathcal{S}$ define the target distribution to be exactly what makes $\mathbf{w}$ a solution:

$$\begin{aligned} y(a, t, x) &= \pi_a^{\text{tape}} \Delta\text{step}_M^t(x + \mathbf{w}), \quad a \in \mathbb{Z} \\ y(\text{state}, t, x) &= \pi^{\text{state}} \Delta\text{step}_M^t(x + \mathbf{w}). \end{aligned} \tag{21}$$

As defined, $[\![\mathbf{w}]\!]$ is an equivalence class. Picking a representative amounts to picking a distribution s which is supported on the whole of $\mathcal{S}$.

[4] The meaning of "effectively given" is left deliberately vague.

Having embedded codes into the lattice $\mathcal{I}(M, V, c)$, the following proposition expresses a kind of duality between codes $\mathbf{w} \in \mathscr{C}(V, c)$ and inference problems.

Proposition 5.17. The function $[\![-]\!]$ defines a Galois connection between programs and inference problems, in the sense that:

$$\mathbf{w} \in \mathcal{W}_0(P) \iff P \sqsubseteq [\![\mathbf{w}]\!].$$

Proof. Suppose that $\mathbf{w} \in \mathcal{W}_0(P)$. Automatically, $\mathcal{S}_+(P) \subset \mathcal{S}_+([\![\mathbf{w}]\!]) = \mathcal{S}$, so we need only verify that they are compatible. This is the requirement that the target state $y_P(z) = y_{[\![\mathbf{w}]\!]}(z)$ for each $z \in \mathcal{S}_+(P)$, but by the definitions of (21) this is precisely the requirement that $\mathbf{w}$ is a solution to P.

For the reverse implication, we use Proposition 5.5, which implies that:

$$\mathcal{W}_0(P) \supset \mathcal{W}_0([\![\mathbf{w}]\!]).$$

Again, by definition $\mathbf{w} \in \mathcal{W}_0([\![\mathbf{w}]\!])$, so we win. □

Remark 5.18. This proposition fits with our inspiration from algebraic geometry: to fully understand the structure of the set of codes on a given machine, we have to consider "non-closed points" — and we add in the non-closed points by considering equations that cut out subsets of the ambient space (affine n-space, or the code space $\mathcal{W}$). Indeed, in the topological version of specialisation outlined in Remark 5.3 (which doesn't include the upper bound $\perp$), closed points are maximal elements under $\sqsubseteq$. The elements $[\![\mathbf{w}]\!]$ are also maximal, in the sense that, for $P \in \mathcal{I}(M, V, c)$:

$$[\![\mathbf{w}]\!] \sqsubseteq P \iff P = \perp \text{ or } P = [\![\mathbf{w}]\!]. \tag{22}$$

Therefore, the map $[\![-]\!]$ corresponds closely to the inclusion

$$\operatorname{MSpec} A \hookrightarrow \operatorname{Spec} A,$$

of the closed points into the full spectrum. Compact inference problems correspond to finitely generated ideals. In our logical interpretation, (22) shows that $[\![\mathbf{w}]\!]$ are the *atoms* in the lattice of propositions [DP02, §5.2].

Having interpreted synthesis problems on a UTM as a semantics of programs (in the sense of Scott), the compact problems provide a stock whose geometry we readily understand — via Theorem 4.24. The process of approximation in Proposition 5.15 is analogous, in the discrete version of this theory, to expressing the graph $\{(x, Tx) \mid x \in \Sigma^*\}$ of a program T as a union of its finite subsets:

$$\{(x, Tx)\} = \{x_1, Tx_1\} \cup \{x_2, Tx_2\} \cup \dots$$

Using our results on Singular Learning, we can do better. The data of an inference problem contain more than a specification of its solutions, as they package information about the learning process. To understand the semantic content of this approximation, we borrow from Turing. In this remark, we will stick to the case of program synthesis on a pseudo-UTM, but the ideas are not specific to this setting.

Remark 5.19 (Compact approximation is education)**.** The document [Tur04] is Turing's investigation of the "question as to whether it is possible for machinery to show intelligent behaviour". He offers the following:

> If we are trying to produce an intelligent machine, and are following the human model as closely as we can, we should begin with a machine with very little capacity to carry out elaborate operations or to react in a disciplined manner to orders (taking the form of interference). Then by applying appropriate interference, mimicking education, we should hope to modify the machine until it could be relied on to produce deliberate reactions to certain commands.

This is particularly interesting to us as by "interference", Turing means "provision of information", which he calls *paper interference*. He proposes that we start with a random, but sufficiently large, architecture[5], and modify it by paper interference to mimic the required behaviour [Tur04, §8].

This model of education provides an interpretation of compact approximation. We start with a random code $\mathbf{w}_0 \in \mathscr{C}(V, c) = \mathcal{W}$, which we see as a solution to the underspecified inference problem $\top$ (Definition 5.1). Then we consider a chain of specialisations, where each one adds a single constraint.

$$\top = P_0 \sqsubseteq P_1 \sqsubseteq P_2 \sqsubseteq \dots \tag{23}$$

Each constraint corresponds to a tighter specification of the function which we demand the solution compute. Therefore, at each stage the solution set (possibly) shrinks:

$$\mathcal{W} = \mathcal{W}_0(\top) \supset \mathcal{W}_0(P_1) \supset \mathcal{W}_0(P_2) \supset \dots$$

and the Bayesian posterior concentrates around the smaller set. As a proxy for this kind of concentration, we have used the expected free energy, which measures the (minus log) average of the posterior around a point — this is discussed in more detail in Appendix A. For large sample sizes, the free energy is controlled by the RLCT, and solutions with a smaller local RLCT will be favoured by the posterior.

In general, by Proposition 5.5 we expect that at certain stages the specialisation $P_n \sqsubseteq P_{n+1}$ will increase the RLCT:

$$\mathrm{RLCT}_{\mathcal{W}}(P_n; \varphi) < \mathrm{RLCT}_{\mathcal{W}}(P_{n+1}; \varphi). \tag{24}$$

Let $\mathbf{w}_n$ and $\mathbf{w}_{n+1}$ be the solutions which achieve the minimum of Lemma 3.15, so that

$$\mathrm{RLCT}_{\mathcal{W}}(P_n; \varphi) = \mathrm{RLCT}_{N_{\mathbf{w}_n}}(P_n; \varphi),$$

for a small neighbourhood $N_{\mathbf{w}_n}$ (likewise for $n+1$). The inequality (24) implies that $\mathbf{w}_n$ will be the favoured solution to P_n, but *not* to P_{n+1}. By Section 4.3 and Proposition 4.19, the change to a solution with larger RLCT can be interpreted as an increase in the complexity of the solution.

Supposing that our learning machine obtains a sequence of solutions $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2, \dots$, the geometric information in the fibre ideals I_{P_i} encodes the process of refinement in the sequence of solutions — or, in Turing's language, the increasing organisation of the machine. By the analysis of Appendix A the dynamical content of this process is a sequence of second-order phase transitions.

[5] Turing's formulation of "B-type machines" is remarkably similar to a neural network.

This is compact approximation. Compact, because after n steps in (23), meaning n paper modifications, we refine our machine to the solution of a compact inference problem P_n, which applies n constraints:

$$|\mathcal{S}_+(P_n)| = n.$$

If the desired end result of the education process is a program $\mathbf{w}$, then what we have described is a directed limit:

$$\sum_{i=0}^{\infty} P_i = [\![\mathbf{w}]\!].$$

To conclude, we sketch another example, where our proto-semantics could be used to *distinguish*, rather than identify, two algorithms.

Example 5.20. Fix a pseudo-UTM $\mathcal{U}$, which we assume to be sufficiently powerful to encode both the Karatsuba and ordinary multiplication algorithms [Ber01, KO62], and let $\mathbf{w}_1$ and $\mathbf{w}_2$ be their respective codes. Certainly these codes are syntactically different, but this doesn't necessarily illuminate the qualitative difference in their *ideas* (semantics). For example, it is not immediately clear that the difference isn't of the same kind as Example 5.13.

However, given that the two algorithms are indeed essentially different, if P is an inference problem to which $\mathbf{w}_1$ and $\mathbf{w}_2$ are solutions, the local geometry of K_P (or I_P) around $\mathbf{w}_1$ and $\mathbf{w}_2$ will differ. If we view programs as the limit of a learning process, as in the previous remark, then this difference encodes a qualitative difference between the algorithms. The programs will be learned differently, so they *are* different.

Karatsuba multiplication is faster, but only for large numbers [KO62]. As such, by varying the inputs and timeouts for the synthesis problem P, one or other of the codes $\mathbf{w}_1, \mathbf{w}_2$ may be the only solution. For example, on small numbers there will be a timeout for which $\mathbf{w}_1$ is not a solution, and vice-versa for large numbers. Understanding the way in which varying the inference problem varies the solutions is the purpose of the lattice $\mathcal{I}(\mathcal{U}, V, c)$. For example, the intersection $[\![\mathbf{w}_1]\!] \cap [\![\mathbf{w}_2]\!]$ is an inference problem lax enough so that both codes are solutions, and the same is true for any P for which $P \sqsubseteq [\![\mathbf{w}_1]\!] \cap [\![\mathbf{w}_2]\!]$.

We claimed, in the introduction, that by viewing programs as the limit of a learning process, we can use SLT to understand semantic differences between programs. This section approaches that goal, starting from Scott's axiom that the objects of our theory should be complete lattices. In this case, the elements of $\mathcal{I}(M, V, c)$ are viewed as approximations to codes $\mathbf{w} \in \mathscr{C}(V, c)$. With this in mind, program synthesis is equivalent to computation of a program (in the language of [Sco77, pp.173-174]).

We have enriched Scott's lattice by linking it to algebraic geometry, and Watanabe's insights [Wat09] allow us to extract information about how the learning process will play out in practice. This can help us to understand qualitative differences between programs, as in Example 5.20.

On the other hand, it is natural to understand elements of a set as potential singularities of some function [KS08, p.10]. The geometric perspective we provide on program synthesis realises this, where a code (program) $\mathbf{w}$ is a singularity of the KL divergence associated to any inference problem where $P \sqsubseteq [\![\mathbf{w}]\!]$. Apart from being useful in a practical sense, in order to synthesise programs, we have argued that this perspective can provide semantic information about codes. To realise this, we need to pick an inference problem, in order to produce the ideal I_P. This

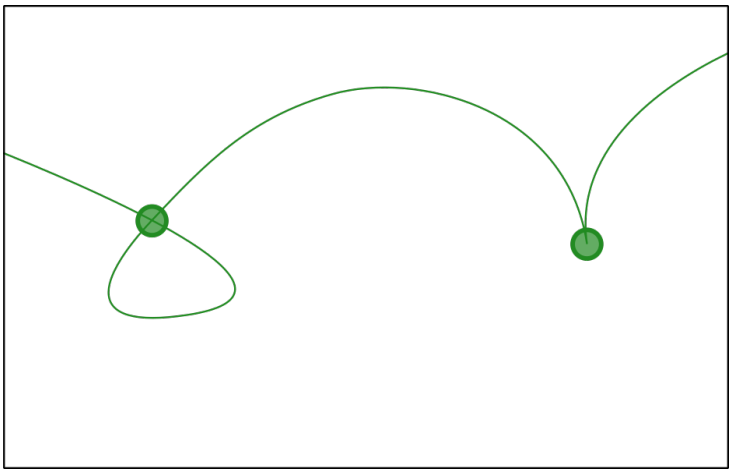

Figure 11: Two non-isomorphic germs of some analytic function K, with vanishing set marked in green.

is the purpose of the lattice, which makes clear the relationships between different choices of inference problem P.

Of course, the fact that we can define the lattice does not mean that it has any meaning. However, given that its structure matches that of lattices in program semantics [Sco77], logic (Remark 5.12) and geometry, it appears that this structure is somehow canonical. Beyond this, the interpretation of Remark 5.19 fits into our claim that we should understand programs as the limit of a learning process.

The discussion of this section offers directions for future work. Most importantly, we are yet to consider morphisms of inference problems. Continuous functions on lattices are central to Scott's work, and morphisms between geometric objects are central in algebraic geometry. Applying techniques from either of these areas to program synthesis could be very interesting.

A Thermodynamics and Phase Transitions

As we briefly alluded to in Section 3.1, there is a strong analogy between statistical learning and thermodynamics. In this section, we will make this analogy more precise, in order to discuss *phase transitions*. Speaking loosely, the idea is to understand the way that the learning machine chooses between solutions that are different in *kind*, as hyperparameters of the problem or estimator are changed. This matches our intention from the introduction. We want to distinguish between semantically different solutions to some inference problem, and thermodynamics allows us to interpret these solutions as phases of a physical system. Given the discussion of the previous section, we will focus on a hyperparameter which switches constraints on or off, like α in Example 4.8.

Before we can think about learning and inference in this context, we will do significant violence to the deep theory of thermodynamics and statistical mechanics, in order to give a brief overview of the terms of reference. A systematic reference is [Cal85], and [Pen05, Chapter 27] provides an excellent intuition for the idea of *coarse graining*, which we will use here.

The theory of thermodynamics deals with the *macroscopic properties* of a system, measurements of which are "extremely slow on the atomic scale of time, and ... extremely coarse on the atomic scale of distance." [Cal85, p.5] Imagine, for example, the temperature of a volume of gas, as opposed to the momentum of a specific particle. The *microstate* of such a system, which might have on the order of 10^{24} coordinates, is rapidly changing, but the macroscopic coordinates — energy, temperature, volume — are chosen to remain relatively stable. We can interpret this, following [Pen05, §27.3], as dividing the phase space of the system into boxes of macroscopically indistinguishable states. As such, everything else being equal, the system will be found in the box with the largest volume. We quantify this using the *entropy*: for a state $\mathcal{R}$, we define $S_{\mathcal{R}} = -k_B \log V_{\mathcal{R}}$, where $V_{\mathcal{R}}$ is the phase-space volume of all the microstates with macrostate $\mathcal{R}$, and k_B is a constant. With this definition in hand, the vagaries of our discussion become less important. As Penrose argues in §27.4, the logarithm in the formula means that the entropy is not sensitive to the specific coarse graining we choose.

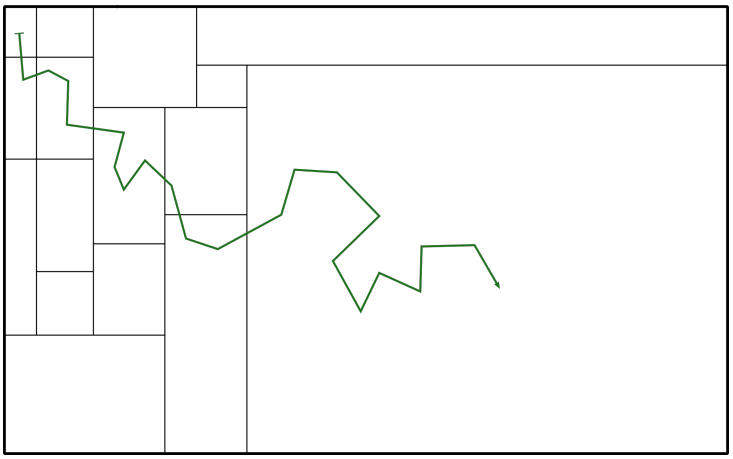

Figure 12: A coarse graining of some phase space, and some possible microscopic state transitions of the associated system. With high probability the system will end up in the box with the largest phase space volume.

Thermodynamics	Statistical Learning
Phase space	Code space $\mathcal{W}$
Energy	$H(w) := nK_n(w) - \frac{1}{\beta}\log\varphi(w)$
Boltzmann distribution	Bayesian posterior
Macrostate $\mathcal{R}$	Neighbourhoods N_w of solutions $w \in \mathcal{W}_0$
Microscopic state transitions	Hamiltonian Monte-Carlo

Table 1: A thermodynamic dictionary for Singular Learning Theory.

We can match this formulation with our theory of statistical learning, using a *canonical ensemble*. The only difference is that now the energy of microstates is allowed to vary, so we use a probability measure on the phase space, where a microstate x with energy E_x is assigned probability proportional to $e^{-E_x/(k_BT)}$. This is the Boltzmann distribution [Cal85, Chapter 16]. Then, the relevant "potential" associated to a macrostate $\mathcal{R}$ is (writing $\beta = (k_BT)^{-1}$):

$$\mathcal{F}_{\mathcal{R}}(T) := -k_B \log \int_{x\in\mathcal{R}} dx\, e^{-\beta E_x}.$$

This is termed the Helmholtz free energy[6], which we recognise as the same sort of free energy integral we have already considered. To match it up more precisely, Table 1 provides a *thermodynamic dictionary* for SLT, following [Mur20a]. In particular, the family of microstates that the thermodynamic system transitions through could correspond to the family of samples generated by a Hamiltonian Monte Carlo process (Remark 3.4).

With this analogy in mind, we can reason about different kinds of solutions, rather than sticking to specific points in the code space $\mathcal{W}$. As for the canonical ensemble, the learning machine will pick the solution with the smallest local free energy: but this choice may vary as hyperparameters of the learning problem are changed. The right formulation of this is in the language of phase transitions.

Example 4.8 provides a good intuition for this concept. At $\alpha = 0$, define two solutions $\hat{w}_0 = (1,1)$ and $\hat{w}_1 = (0,1)$ of the inference problem P_0. Recall that these values of $(h,k) \in [0,1]^2$ correspond to believing, respectively, that the initial tape configuration is (where the underline indicates the head position):

$$\ldots \square \underline{2}\, A\, A\, A \square \ldots$$

$$\ldots \square \underline{0}\, A\, A\, A \square \ldots$$

Then, after two time steps the configurations will be:

$$\ldots \square\, 1\, A\, \underline{A}\, A \square \ldots$$

$$\ldots \square \underline{0}\, A\, A\, A \square \ldots$$

[6]Here the only *coordinate* that $\mathcal{F}$ depends on is T. We can introduce others, say a volume V, which provide their own version of coarse graining by restricting the integral to $V(x) = V$.

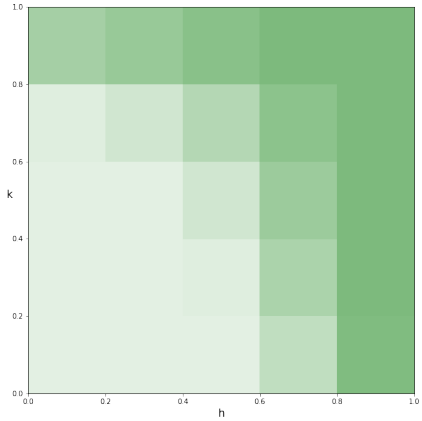

Figure 13: A plot of the free energy integrated over the boxes of a coarse graining on $\mathcal{W}$. Numerical integration uses SciPy [VGO+20].

The constraint applied by the problem P_0 is that the tape square to the right of the head is an A, so both $\hat{w}_0$ and $\hat{w}_1$ are solutions, but in different ways. This is what we mean by solutions different in kind: we have a strong intuition that there may be qualitatively different ways of computing the same function, ways that differ *semantically*. In the smooth case, we postulate that kinds of solutions correspond to local minima of the KL divergence — this is vindicated by the shift machine example. This is, granted, a simpler example than fully fledged program synthesis, but we expect the shape of the general theory to be similar. We will refer to different kinds of solutions as *phases*. This refers both to the solution $\hat{w}$, and to the associated macroscopic state, identified with a neighbourhood $N_{\hat{w}}$.

As we have discussed, the posterior favours $\hat{w}_0$, which is detected by its smaller local RLCT. By this we mean that, for large enough n ,the free energy around $\hat{w}_0$ will be significantly lower than around $\hat{w}_1$, so a random sample from the posterior is overwhelmingly more likely to be near $\hat{w}_0$ (Figure 13). With respect to some coarse graining of the code space, we expect that system will be found in the state, or phase, which contains $\hat{w}_0$. Lemma 3.15 vindicates our assumption that the specifics of this coarse graining are not especially important.

Now we imagine continuously changing the parameter α, which determines the weighting over the inputs on squares 2 and 3:

$$s(x) = (1-\alpha)\cdot(A,A) + \alpha\cdot(A,B).$$

Evaluated on the second input (which we called $x^{(1)}$ in Example 4.8), the respective tape configurations after two time steps are:

$$\ldots \square\, 1\, A\, \underline{A}\, B\, \square \ldots$$

$$\ldots \square\, \underline{0}\, A\, A\, A\, \square \ldots$$

Therefore, for any $\alpha > 0$, the "preferred" phase $\hat{w}_0$ is no longer a solution, so as we continuously change α, there is some value at which the macroscopic phase of the system changes to $\hat{w}_0$. This is a phase transition. The reason that this process is interesting is that, at least for small enough

α, the "identity" of the phase $\hat{w}_0$ is preserved, as a local minimum of F. We can demonstrate this as follows. First approximate the local free energy $\bar{F}_i$ around $\hat{w}_i$ as [Wat13]:

$$\bar{F}_i^\alpha(n) \approx n\beta K^\alpha(w_i) + \lambda_i \log(n) + (\theta_i - 1)\log\log(n).$$

(Roughly, this comes from applying the usual asymptotic methods to $K^\alpha(w) - K^\alpha(w_i)$.) Here the local RLCT (λ_i, θ_i) is of the level set:

$$\{w \mid K^\alpha(w) = K^\alpha(w_i)\},$$

and as such might vary with α. However, since λ is rational and F varies continuously with α, λ_i is locally constant (likewise θ_i is an integer). Since w_i is a local minimum of K^α, it is a local minimum of $\bar{F}$ as claimed. Then, as we raise α, the local (and global) minimum of F at $\hat{w}_0$ bifurcates into two, around the two phases. This identifies the phase transition as *second-order*.

The crux of this example was that P_α adds constraints not present in P_0, so we have a specialisation $P_0 \sqsubseteq P_\alpha$. More generally, suppose that $Q_0 \sqsubseteq Q_1$ is some specialisation, and that Q_0 admits simpler solutions than Q_1, in the sense that:

$$\mathrm{RLCT}_{\mathcal{W}}(Q_0; \varphi) < \mathrm{RLCT}_{\mathcal{W}}(Q_1; \varphi).$$

In full generality the RLCTs may be equal, but using our analysis of the Kolmogorov complexity, it is reasonable to consider a situation where the more specific problem requires a more complex (longer) program. Let $\hat{w}_i \in \mathcal{W}$ be the solution of Q_i which realises the minimum in the statement of Lemma 3.15. That is, for a small neighbourhood N_i of $\hat{w}_i$:

$$\mathrm{RLCT}_{\mathcal{W}}(Q_i; \varphi) = \mathrm{RLCT}_{N_i}(Q_i; \varphi).$$

Now we define an inference problem Q_α, for $\alpha \in (0, 1)$, which has the same support and target state as Q_1, but with weighting:

$$s_\alpha(z) = (1 - \alpha)s_0(z) + \alpha s_1(z).$$

Then we have precisely the same situation as our shift machine example. At $\alpha = 0$ both $\hat{w}_0$ and $\hat{w}_1$ are solutions, as $P_0 \sqsubseteq P_1$. Since the local RLCT of $\hat{w}_0$ is smaller than that of $\hat{w}_1$, the former is favoured by the posterior, and we have a unique local minimum of the free energy. Increasing α slightly, the minimum bifurcates as $\hat{w}_0$ is no longer a solution, but maintains its identity as a phase.

Example A.1. In a synthesis problem, specialisation corresponds a tighter specification of the synthesised program. We can return to the problem `detectA` of Example 4.29 to illustrate this. Recall that, in the general case we bounded the RLCT by observing that a solution $\mathbf{w}$ can vary within a codimension-12 submanifold of $\mathcal{W}$, which we schematically represented by the following table:

	$\square$	A	B
reject	$(\square, \text{reject}, -)$	$(-, \text{accept}, -)$	$(B, \text{reject}, 1)$
accept	$(-, \text{accept}, -)$	$(-, \text{accept}, -)$	$(-, \text{accept}, -)$

Now suppose that $\texttt{detectA}'$ is the inference problem defined in the same way, but limiting the input strings to length one. This relaxes the constraints on our solution, as the simulated machine $M_{\mathbf{w}}$ no longer needs to scan the whole string. Before, we required that $\delta_3(B, \text{reject}) = 1$, so that $M_{\mathbf{w}}$ moves rightwards through the string. If this is no longer necessary, a new table could be:

	$\square$	A	B
reject	$(\square, \text{reject}, -)$	$(-, \text{accept}, -)$	$(B, \text{reject}, -)$
accept	$(-, \text{accept}, -)$	$(-, \text{accept}, -)$	$(-, \text{accept}, -)$

(25)

This decreases the codimension of the submanifold in question, so that the new bound is:

$$\text{RLCT}_{\mathcal{W}}(\texttt{detectA}'; \varphi) \leq 5.$$

This is the situation we have discussed. Increasing the bound on the lengths defines a specialisation $\texttt{detectA}' \sqsubseteq \texttt{detectA}$, which we expect will increase the RLCT. We can impose these restraints continuously, by defining a family of weightings $s_\alpha(x)$ over the inputs $x \in \{A, B\}^*$:

$$s_\alpha = \frac{1-\alpha}{2} \cdot [A + B] \; + \; \frac{\alpha}{b} \sum_{l=1}^{b} \sum_{x \in \{A,B\}^l} 2^{-l} \cdot x.$$

Let $\hat{w}_0$ be the solution of $\texttt{detectA}'$ described by (25) with $\delta_3(B, \text{reject}) = 0$. Let $\hat{w}_1$ be identical, apart from setting $\delta_3(B, \text{reject}) = 1$. For α small and positive, $\hat{w}_1$ is not a solution, but fails only for a small number of samples from the true distribution q — as the weighting on sequences longer than 1 letter is small. As such, at $\alpha = 0$ we have a local minimum of the free energy around $\hat{w}_0$, which undergoes a second-order phase transition for $\alpha > 0$.

Semantically different solutions $\mathbf{w}_i \in \mathcal{W}_0$ have different local geometry, as detected by the germs $(\mathbf{w}_i, K)$. This implies that they are learned differently, as discussed in the earlier sections. As we add constraints to a synthesis problem, the algorithm required to solve it may change: if the timeouts get shorter, then we might require a more efficient algorithm. The argument of this section demonstrates that the passage from a simpler to a more complex solution is a second-order phase transition.

References

[Ati70] M. F. Atiyah. Resolution of singularities and division of distributions. *Communications on Pure and Applied Mathematics*, 23(2):145–150, 1970.

[AVGZ85] V. Arnold, A. Varchenko, and S. Gusein-Zade. *Singularities of differentiable maps.* Birkhauser, 1985.

[Bal97] V. Balasubramanian. Statistical inference, Occam's razor, and statistical mechanics on the space of probability distributions. *Neural Comput.*, 9(2):349–368, February 1997.

[Ber01] D. J. Bernstein. Multidigit multiplication for mathematicians. Available at `http://cr.yp.to/papers/m3.pdf`, 2001.

[BGJM11] S. Brooks, A. Gelman, G. Jones, and X. Meng. *Handbook of Markov Chain Monte Carlo.* CRC Press, 2011.

[Cal85] H. B. Callen. *Thermodynamics and an introduction to thermostatistics; 2nd ed.* Wiley, 1985.

[CM19] J. Clift and D. Murfet. Derivatives of Turing machines in Linear Logic. *arXiv preprint arXiv:1805.11813*, 2019.

[CM20] J. Clift and D. Murfet. Encodings of Turing machines in Linear Logic. *Mathematical Structures in Computer Science*, 30(4):379–415, April 2020.

[CMW21] J. Clift, D. Murfet, and J. Wallbridge. Geometry of program synthesis. *arXiv preprint arXiv:2103.16080*, 2021.

[DP02] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order, Second Edition.* Cambridge University Press, 2002.

[EG18] R. Evans and E. Grefenstette. Learning explanatory rules from noisy data. *arXiv preprint arXiv:1711.04574*, 2018.

[Ehr16] T. Ehrhard. An introduction to Differential Linear Logic: proof-nets, models and antiderivatives. *CoRR*, abs/1606.01642, 2016.

[GBS+16] A. L. Gaunt, M. Brockschmidt, R. Singh, N. Kushman, P. Kohli, J. Taylor, and D. Tarlow. Terpret: A probabilistic programming language for program induction. *arXiv preprint arXiv:1608.04428*, 2016.

[GHS12] S. Gulwani, W. R. Harris, and R. Singh. Spreadsheet data manipulation using examples. *Communications of the ACM*, 55(8):97–105, August 2012.

[Gir11] J.-Y. Girard. *The Blind Spot: Lectures on Logic.* European Mathematical Society Publishing House, 2011.

[GLS07] G. Greuel, C. Lossen, and E. Shustin. *Introduction to Singularities and Deformations.* Springer Monographs in Mathematics. Springer Berlin Heidelberg, 2007.

[GLT93] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and types*. Cambridge University Press, 1993.

[GM14] S. Gulwani and M. Marron. Nlyze: interactive programming by natural language for spreadsheet data analysis and manipulation. *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 2014.

[GSLT+18] J. Gottschlich, A. Solar-Lezama, N. Tatbul, M. Carbin, M. Rinard, R. Barzilay, S. Amarasinghe, J. B. Tenenbaum, and T. Mattson. The three pillars of machine programming. In *Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2018, pages 69–80. Association for Computing Machinery, 2018.

[Hil90] D. Hilbert. Über die theorie der algebraischen formen. *Math. Annalen*, 36:473–534, 1890.

[Hir64] H. Hironaka. Resolution of singularities of an algebraic variety over a field of characteristic zero: I. *Annals of Mathematics*, 79(1):109–203, 1964.

[HMU01] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[Hun07] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[Hut04] M. Hutter. *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Springer Science & Business Media, 2004.

[KAS16] K. Kurach, M. Andrychowicz, and I. Sutskever. Neural random-access machines. *arXiv preprint arXiv:1511.06392*, 2016.

[KO62] A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595–596, December 1962.

[KS08] M. Kontsevich and Y. Soibelman. Stability structures, motivic Donaldson-Thomas invariants and cluster transformations. *arXiv preprint arXiv:0811.2435*, 2008.

[Lib01] H. Liberman, editor. *Your Wish is My Command: Programming by Example*. Morgan Kaufmann Publishers Inc., 2001.

[Lin11] S. Lin. *Algebraic Methods for Evaluating Integrals in Bayesian Statistics*. PhD thesis, University of California, Berkeley, 2011.

[Lin12] S. Lin. Useful facts about RLCT, 2012. Available at https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.303.6529&rep=rep1&type=pdf.

[Mac19] D. J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2019.

[MGA13] M. Manshadi, D. Gildea, and J. Allen. Integrating programming by example and natural language programming. *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013*, pages 661–667, January 2013.

[Mur20a] D. Murfet. Deep learning theory lecture 2: Thermodynamics of singular learning theory, 2020. Private communication.

[Mur20b] D. Murfet. Singular learning theory 4: local RLCT, 2020. Private communication.

[Mus11] M. Mustata. IMPANGA lecture notes on log canonical thresholds. *arXiv preprint arXiv:1107.2676*, 2011.

[OP93] M. Ohya and D. Petz. *Quantum Entropy and Its Use*. Lecture Notes in Computer Science. Springer-Verlag, 1993.

[Pen05] R. Penrose. *The road to reality: a complete guide to the laws of the universe*. Vintage, 2005.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Ros39] B. Rosser. An informal exposition of proofs of Gödel's theorems and Church's theorem. *The Journal of Symbolic Logic*, 4(2):53–60, 1939.

[Sch78] G. Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461–464, 1978.

[Sco77] D. Scott. Outline of a mathematical theory of computation. *Kiberneticheskij Sbornik. Novaya Seriya*, 14:169–176, January 1977.

[Sta18] Stacks Project Authors. *Stacks Project*. https://stacks.math.columbia.edu, 2018.

[Tur39] A. Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, s2-45(1):161–228, 1939.

[Tur04] A. Turing. Intelligent machinery. 1948. In B. J. Copeland, editor, *The Essential Turing*, pages 395–423. Oxford University Press, 2004.

[VGO+20] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[Vic89] S. Vickers. *Topology via Logic*. Cambridge University Press, 1989.

[vN55] J. von Neumann. *Mathematical Foundations of Quantum Mechanics*. Goldstine Printed Materials. Princeton University Press, 1955. trans. R.T. Beyer.

[Wat07] S. Watanabe. Almost all learning machines are singular. In *2007 IEEE Symposium on Foundations of Computational Intelligence*, pages 383–388. IEEE, 2007.

[Wat09] S. Watanabe. *Algebraic geometry and statistical learning theory*. Cambridge University Press, 2009.

[Wat13] S. Watanabe. A widely applicable Bayesian information criterion. *Journal of Machine Learning Research*, 14:867–897, 2013.

[Wat18] S. Watanabe. *Mathematical Theory of Bayesian Statistics*. CRC Press, 2018.

[Wat20] S. Watanabe. Statistical learning theory 13 : Phase transition and prior effect, 2020.

[Xu21] A. K. Xu. Smooth relaxation preserving Turing machines, 2021. Private communication.