# Programs as Singularities

Daniel Murfet, Will Troiani

June 20, 2025

## Abstract

We develop a correspondence between the structure of Turing machines and the structure of singularities of real analytic functions, based on connecting the Ehrhard-Regnier derivative from linear logic with the role of geometry in Watanabe's singular learning theory. The correspondence works by embedding ordinary (discrete) Turing machine codes into a family of "noisy" codes which form a smooth parameter space. On this parameter space we consider a potential function which has Turing machines as critical points. By relating the Taylor series expansion of this potential at such a critical point to combinatorics of error syndromes, we relate the local geometry to internal structure of the Turing machine.

The potential in question is the negative log-likelihood for a statistical model, so that the structure of the Turing machine and its associated singularity is further related to Bayesian inference. Two algorithms that produce the same predictive function can nonetheless correspond to singularities with different geometries, which implies that the Bayesian posterior can discriminate between distinct algorithmic implementations, contrary to a purely functional view of inference. In the context of singular learning theory our results point to a more nuanced understanding of Occam's razor and the meaning of simplicity in inductive inference.

> All of this will lead to theories which are much less rigidly of an all-or-none nature than past and present formal logic. They will be of a much less combinatorial, and much more analytical, character. In fact there are numerous indications to make us believe that this new system of formal logic will move closer to another discipline which has been little linked in the past with logic. This is thermodynamics, primarily in the form it was received from Boltzmann, and is that part of theoretical physics which comes nearest in some of its aspects to manipulating and measuring information.

J. von Neumann, Collected Works V, [43, p. 304]

# Contents

# 1 Introduction

We present a perspective on programs which relates them to the geometry of singularities in real algebraic geometry [2, 22, 21]. This continues a line of work on algebraic semantics of linear logic [30, 15, 13] and connections of those semantics to geometry [31, 38] and statistical learning theory [17, 45]. In this introduction we explain the key ideas of the present work without requiring any familiarity with this background material.

The concept of an *algorithm* or *program* had a long gestation. From a modern perspective we can see two primary threads: the refinement of logical debate and structured arguments starting with Aristotle and the refinement of the idea of an algebraic manipulation or calculation starting with Al-Khwārizmī. Over time these threads came together in the work of Leibniz and Boole [7] then later Gentzen [19], Turing [39], Brouwer-Heyting-Kolmogorov [37], Church [12], Curry and Howard [24] and others to give us the modern synthesis of logic and computation. However, with the rise of machine learning and artificial intelligence the conceptual landscape under this synthesis has shifted, as already anticipated by Turing [40] and von Neumann [42, 43]. Soon most sophisticated cognition on Earth, including the construction of programs, will occur in machines. Thus it may be appropriate to expand the proper domain of logic to include understanding the laws of thought that emerge in the minds of large learning machines *other* than ourselves.[1]

One way to take this seriously is to search for an embedding of the discrete world of proofs and programs into the continuous world of machine learning and statistical learning theory. We know one relevant example: our brains are continuous but can parametrise effectively discrete forms of reasoning and symbol manipulation [42]. Our main desiderata for this embedding should be that it be a *homomorphism*, that is, the internal structure of proofs and programs should be reflected in the geometric structure of the continuous space near the image of that program. When the geometry in question is that of the singularity of a potential function, there is a long tradition in mathematics and physics of relating the "internal structure" of points to the geometry of singularities [10, 9, 21, 1].

In this paper we study such a homomorphism, that is, a structure preserving map from programs to singularities. In [14, 17, 45] the map was constructed by considering *synthesis problems*, that is, the Bayesian inference of a (noisy) Turing machine code given input-output examples in the spirit of Solomonoff [35]. In this paper we develop a language of *error syndromes* in which to describe this map and exhibit the sense in which it is structure-preserving. Finally, we explain how this correspondence offers a new view on Bayesian statistics, which we term *structural Bayesianism*. This point of view emphasises that according to standard principles of model selection we should not only prefer models that predict well but which further have an internal structure which is optimal. This provides some theoretical grounding for approaches that attempt to perform structural inference on neural networks, another important class of singular models [23, 44].

---

[1] Interpretability is the new (old) logic.

## 1.1   Overview

The paper in overview:

- Section 2: we recall the general setup of synthesis problems and smooth relaxations of the execution of a UTM, and modulo the details of that smooth relaxation how we define a statistical model (Section 2.1, Section 2.2). We then recall the definition of learning coefficients (Section 2.3) and how this geometric invariant controls the Bayesian learning process according to singular learning theory (Section 2.4).

- Section 3: having associated to a Turing machine $M$ a singularity $([M], L)$ we proceed to introduce what it means to study the geometry of such an object. This leads to the definition of *influence functions* $g_\mathbf{i}(x)$.

- Section 4: we recall the basics of linear logic and its semantics (Section 4.1). Then we give a sketch of how in this semantics inputs are copied and distributed throughout the "programs" that are one interpretation of linear logic proofs, and how when the inputs are distributions this has the semantics of "resample whenever you use an input" which leads to the fundamental notion of an error syndrome (Section 4.2). This is further expanded in Section 4.3 which concludes with Theorem 4.8. This says that derivatives of denotations of plain proofs (an important case being the proof that encodes the execution of a UTM for $t$ timesteps) can be thought of in terms of counts of error syndromes.

- Section 5: we introduce a plain proof $\psi$ which encodes reading off the final state of a simulated machine after the execution of a particular UTM for $t$ steps (Section 5.2) and then apply Section 4 to that proof to define derivatives and express them in terms of error syndromes (Section 5.3).

- Section 6: we express the Taylor series expansion of the polynomial function $H(w)$ at a code $[M]$ in terms of counts of error syndromes (Theorem 6.3).

- Section 7: using Theorem 6.3 we explain how to make three separate connections between program structure and geometry (summarised in Section 1.1.3 below).

- Section 8: in the conclusion we draw some general lessons from this connection between programs and singularities for Bayesian inference and interpretability and introduce the philosophy of structural Bayesianism.
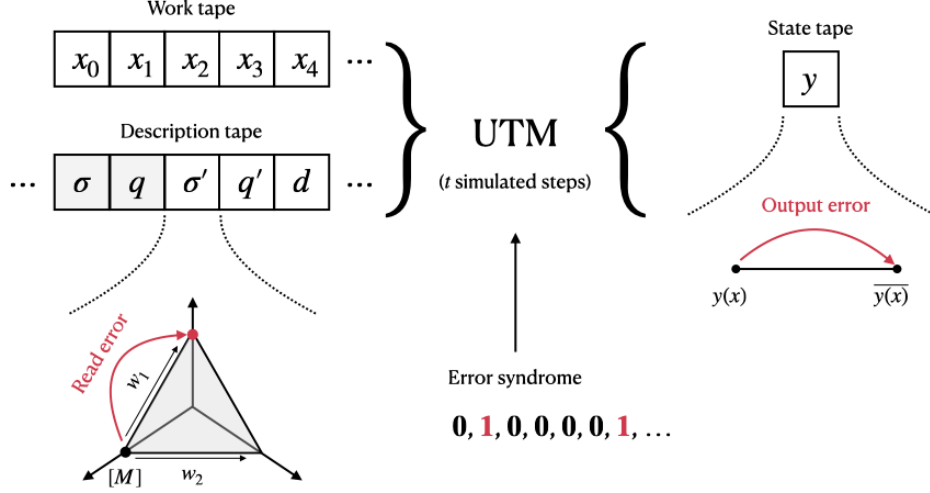
Figure 1: The parameter space $W$ with coordinates $w_1, \ldots, w_d$ parametrises "noisy codes" which specify for each entry on the description tape of a UTM a distribution over possible values. When executing such a noisy Turing machine code on an input $x$ for $t$ steps of the simulated machine the UTM can experience read errors when accessing the code on the description tape, and the pattern of such errors determines an error syndrome. Any particular error syndrome may flip the output of the simulated machine from the correct output $y = y(x)$ to the incorrect output $\overline{y(x)}$.

In the rest of this section we sketch the main results. First we recall the general situation of Bayesian inference of Turing machine codes considered in [14, 17, 45], which is depicted in Figure 1.

### 1.1.1 Inductive Inference of TMs

We suppose there is a computable function $y = y(x)$ with $y \in \{0, 1\}$ and $x \in \Sigma^*$ for some alphabet of symbols $\Sigma$ where in this paper $\Sigma^*$ denotes strings of positive length. That is, we are given a *language* $L = \{x \in \Sigma^* \mid y(x) = 1\}$. The learning problem, which we refer to as a *synthesis problem*, is to infer a Turing machine code that explains a given dataset $D_n = \{(x_i, y_i)\}_{i=1}^n$ of examples with $y_i = y(x_i)$ and $x_i \sim q(x)$ for some given distribution over inputs with finite support $I \subseteq \Sigma^*$. We refer to a Turing machine $M$ as a *classical solution* if $M(x) = y(x)$ for all $x \in I$.[2]

We can perform Bayesian inference to assign probabilities to various Turing machines $M$ represented by their codes $[M]$ for a particular universal Turing machine (UTM) which we denote $\mathcal{U}$. However, this approach does not "see" the internal structure of a Turing machine. On the theory that internal structure is revealed by the pattern of responses of a system to perturbations, we consider perturbations of the dynamical system $\mathcal{U}$ simu-

---

[2]We will be more precise about the meaning of $M(x)$ below, but note that it will involve a timeout.

lating $M$ of the following kind: when the UTM performs an operation, it *reads from the description tape with some probability of error*, obtaining enough samples to simulate the machine from the first step. That is, we view the connection of the description tape to the rest of the UTM as a noisy channel. The code $w$ for a *noisy Turing machine* specifies the precise errors that can occur and their probability. In this way we can include the ordinary Turing machine codes in a larger continuous space $W$. On this space we define the negative log-likelihood

$$L_n(w) = -\frac{1}{n} \sum_{i=1}^{n} \log p(y_i|x_i, w) \tag{1}$$

for any noisy Turing machine code $w \in W$, and its average $L(w) = \mathbb{E}_{D_n}[L_n(w)]$ over all possible datasets. This is an analytic function and the global minima of $L$ include Turing machines $M$ which are solutions in the above sense. The singularity assigned in [17, 45] to the pair consisting of $M$ and the synthesis problem is the germ $([M], L)$ (see Definition 2.4 below). Of course it remains to explain precisely how to assign probabilities to $y_i \in \{0, 1\}$ given an input $x_i$ and a noisy Turing machine code $w$ (we will get to this in Section 2.1).

### 1.1.2 Error Syndromes

With this preparation we can sketch the idea of error syndromes, influence functions and how they relate the structure of $M$ to the geometry of $([M], L)$. An *error syndrome* $\gamma$ records the pattern of errors encountered while $\mathcal{U}$ simulates $M$ for some fixed number $t$ of steps. Let $M$ be a classical solution and let $\mathcal{U}(x, \gamma)$ denote the "output" (the entry on the state tape of the UTM) obtained on input $x$ with error syndrome $\gamma$ when simulating $M$ for $t$ steps. If we only allow a single error on the $i$th entry of the description tape the *influence function of weight one* is (Section 6.3)

$$g_i(x) = \left| \left\{ \gamma \mid \mathcal{U}(x, \gamma) \neq y(x) \right\} \right|. \tag{2}$$

That is, $g_i \in \mathbb{N}^I$ assigns to each input $x$ the number of those error syndromes, with one error localised to the $i$th entry of the description tape, which cause an output error. Up to second-order the main structural information there is to know about the germ $([M], L)$ is whether the singularity is nondegenerate, that is, whether its Hessian is invertible. One can show that this geometry of $L$ is the same as the geometry of a polynomial function $H$ and our results are formulated about this polynomial (Section 3).

The Hessian of $H$ can be described in terms of influence functions (Corollary 6.4)

$$\frac{1}{2} \frac{\partial^2}{\partial w_i \partial w_j} H \Big|_{w=[M]} = \mathbb{E}_x \left[ g_i(x) g_j(x) \right] \tag{3}$$

where $\mathbb{E}_x[-]$ denotes expectation with respect to $q(x)$ and $w_1, \ldots, w_d$ are natural coordinates on the space of noisy TM codes $W$ at $[M]$. It follows that $([M], L)$ is nondegenerate

if and only if the set of influence functions is linearly independent, or what is the same, if *every bit of the code of M influences the computation in an independent way.* This provides the first hint of a nontrivial relation between the geometry of $([M], L)$ and the structure of the Turing machine (see Section 6.4).

More generally for a nonzero multi-index $\mathbf{k} = (k_1, \ldots, k_d)$ we have (Lemma 3.3)

$$\frac{\partial^{|\mathbf{k}|}}{\partial w_1^{k_1} \cdots \partial w_d^{k_d}} H \bigg|_{w=[M]} = \sum_{\substack{\mathbf{i},\mathbf{j} \neq 0 \\ \mathbf{i}+\mathbf{j}=\mathbf{k}}} C(\mathbf{i},\mathbf{j}) \, \mathbb{E}_x \Big[ g_{\mathbf{i}}(x) g_{\mathbf{j}}(x) \Big] \tag{4}$$

where the $C$ are some combinatorial factors and $g_{\mathbf{i}}(x)$ is a general influence function, which can be expressed in terms of counts of error syndromes subject to the condition that the number of errors in each entry of the description tape is at most the integer specified in $\mathbf{i}$ (Theorem 6.3). In this precise sense, the combinatorics of error syndromes controls the geometry of $H$ and thus of $L$.

### 1.1.3   Programs as Singularities

We claim that typical Turing machine codes are highly degenerate (see Section 7.4 for an example). That is, the effects of different entries on the description tape are highly correlated in terms of the effect that errors in these bits have on the output. We claim further that the pattern of these correlations reflects internal structure of Turing machines. Since there is no general formal definition of "internal structure" of programs to compare to, evidence for this claim takes the form of the development (in this paper and its sequels) of examples of program structure, together with a study of how this structure is reflected in correlations of influence functions (Figure 2).

In this paper we give three examples:

- If a Turing machine $M$ has run-time error correction then the Taylor series expansion of $H$ at $[M]$ has many vanishing lower order terms; this leads to upper bounds on the learning coefficient of $L$ (Section 7.1).

- It is apparent from (3) that Turing machines which process subsets of the input using independent control paths should have that structure reflected in their geometry (see Section 7.2).

- In Section 7.4 we study a Turing machine $\mathtt{detectA}^{(0)}$ which recognises the language consisting of all strings in $\{A, B\}$ which contain an $A$. We compute explicitly the polynomial $H$ to second order and diagonalise its Hessian in local coordinates to exhibit how the geometry relates to the structure of the machine.

8
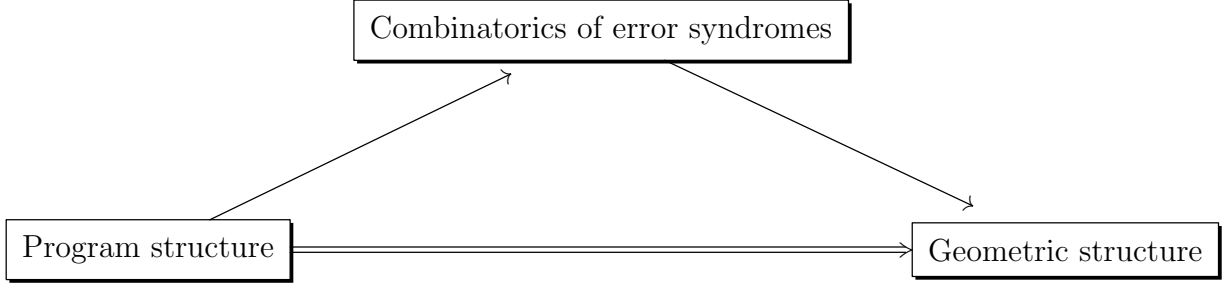
Figure 2: The relationship between program structure and geometry we establish works by relating both to the combinatorics of error syndromes, which are patterns of flips in bits on the description tape of a UTM which affect the output of the simulated machine.

| Symbol | Description | Reference |
|:---:|:---|:---:|
| $\gamma$ | Error syndrome | Definition 4.9 |
| $\mathcal{U}$ | Staged pseudo-universal Turing machine | Section 5.1 |
| $W$ | Parameter space of noisy Turing machines | Section 2 |
| $d$ | Dimension of $W$ | Equation (10) |
| $W^{\texttt{code}}$ | Set of (deterministic) Turing machine codes | Section 2 |
| $[M]$ | The code of a Turing machine $M$ | Definition 2.2 |
| $\Delta\text{step}^t$ | Execution of a noisy Turing machine for $t$ steps | Equation (11) |
| $L(w)$ | Average negative log-likelihood function | Section 2.2 |
| $K(w)$ | KL divergence | Section 2.2 |
| $H(w)$ | Polynomial function comparable to $K$ | Definition 2.8 |
| $g_{\mathbf{i}}(x)$ | Influence function measuring the effect of errors of pattern $\mathbf{i}$ on input $x$ | Definition 3.2 |
| $A^{\mathbf{s}}(x)$ | Count of weight $\mathbf{s}$ error syndromes that cause output errors on input $x$ | Definition 5.8 |
| $\mathbb{E}_x[-]$ | Expectation with respect to $q(x)$ | Section 3 |
| $\lambda([M], q)$ | Local learning coefficient of a Turing machine $M$ | Section 6 |
| $\psi$ | Plain proof in intuitionistic linear logic | Definition 4.2 |
| $\pi$ | Linear proof (usually the linear part of $\psi$) | Definition 4.2 |
| $f_\psi^\tau$ | Polynomial computing the denotation of $\psi$ | Definition 4.5 |
| $[\![\cdot]\!]$ | Denotational semantics operator | Section 4.1 |
| $\mathcal{U}(x, \gamma)$ | Evaluation of $\mathcal{U}$ input $x$ and error syndrome $\gamma$ | Definition 5.6 |

Table 1: Key symbols used throughout the paper and their descriptions.

# 2 Background

## 2.1 From Turing Machine to Singularity

We recall the treatment of inductive inference in [14, 17], assuming familiarity with Turing machines at the level of [3]. Our conventions for Turing machines are as in [14, §6.1]. We denote by $\Sigma$ the set of symbols and $Q$ the set of states. The set $\Sigma$ is assumed to contain some designated blank symbol $\square$ which is the only symbol that is allowed to occur infinitely often on the tape. Often one also designates a starting state.

The *configuration* of a Turing machine $M$ is an element $\big((\sigma_u)_{u\in\mathbb{Z}}, q\big)$ of $\Sigma^{\mathbb{Z}} \times Q$ where $q$ is the current state and the symbol in the square in position $u$ relative to the head is $\sigma_u$ (so $\sigma_0$ is the symbol currently under the tape head, $\sigma_{-1}$ is the symbol immediately to the left of the tape head). Observe that the configuration of the tape actually lies in the smaller set of functions which are finitely supported, in the following sense:

**Definition 2.1.** We write

$$\Sigma^{\mathbb{Z},\square} = \{f : \mathbb{Z} \longrightarrow \Sigma \,|\, f(u) = \square \text{ except for finitely many } u\}.$$

Given a finite alphabet $\Sigma$ and set of states $Q$ a Turing machine $M$ for this alphabet and set of states is identified with its *transition function*

$$\delta : \Sigma \times Q \longrightarrow \Sigma \times Q \times \{\mathrm{L}, \mathrm{S}, \mathrm{R}\}. \tag{5}$$

The interpretation is that when the machine reads the symbol $\sigma$ and is in state $q$ it writes $\sigma'$, transitions to state $q'$ and moves in direction $d$ (Left, Stay or Right) where $\delta(\sigma, q) = (\sigma', q', d)$. We sometimes write $\delta_0(\sigma, q), \delta_1(\sigma, q), \delta_2(\sigma, q)$ for $\sigma', q', d$ respectively.

The set of Turing machines with this alphabet and these states is

$$W^{\texttt{code}} = \prod_{\sigma,q} \Sigma \times Q \times \{\mathrm{L}, \mathrm{S}, \mathrm{R}\} \tag{6}$$

where unless specified otherwise, $\sigma$ always ranges over $\Sigma$ and $q$ over $Q$. We think of $W^{\texttt{code}}$ as a set of *codes* since the tuples $(\sigma, q, \sigma', q', d)$ are, when encoded on the description tape of a Universal Turing Machine (UTM) for all $\sigma, q$, a code for the Turing machine with the given transition function.

**Definition 2.2.** If $M$ is a Turing machine with alphabet $\Sigma$ and set of states $Q$, then the point in $W^{\texttt{code}}$ corresponding to $M$ is the *code of $M$* and is denoted $[M]$.

In this paper $\Sigma, Q$ are viewed as fixed and all Turing machines have the same set of symbols and states. Thus all Turing machines have codes in $W^{\texttt{code}}$. See Appendix A.7 for some comments on this restriction.

**Definition 2.3.** Given a set $Z$ we denote by $\Delta Z$ the *simplex of $Z$* which is

$$\Delta Z := \big\{f : Z \longrightarrow [0,1] \,|\, \sum_{z\in Z} f(z) = 1\big\}. \tag{7}$$

A *noisy Turing machine* is given by a function

$$\delta : \Sigma \times Q \longrightarrow \Delta\Sigma \times \Delta Q \times \Delta\{\mathrm{L}, \mathrm{S}, \mathrm{R}\} \tag{8}$$

so the set of all noisy Turing machines is

$$W := \prod_{\sigma, q} \Delta\Sigma \times \Delta Q \times \Delta\{\mathrm{L}, \mathrm{S}, \mathrm{R}\} \,. \tag{9}$$

This is a manifold with corners of dimension

$$d := \dim W = \sum_{\sigma, q} (|\Sigma| - 1 + |Q| - 1 + 2) = |\Sigma||Q|(|\Sigma| + |Q|) \,. \tag{10}$$

Again, we do not distinguish between a machine and its code, which are the points of $W$. The interpretation of a noisy Turing machine code is that when the UTM attempts to simulate the machine on an input $x$, it has some probability of *read errors* when it accesses the description tape. For instance if the second component of $\delta(\sigma, q)$ is $\mathbf{q}' \in \Delta Q$ then when the UTM attempts to determine what state to transition the simulated machine to when it reads $\sigma$ and is in state $q$, it receives from the description tape a sample from the distribution $\mathbf{q}'$. That is, communication between the description tape and other parts of the UTM are a *noisy channel* (see Appendix A.5 for some elaboration).[3]

Different ways of thinking about the operation of the UTM in the presence of this noise channel lead to different *smooth relaxations* [14], by which we mean in particular the specification of smooth functions $\Delta\mathrm{step}^t$ for all integers $t \geq 1$ which make the diagram

$$\begin{array}{ccc}
\Sigma^* \times W & \xrightarrow{\ \Delta\mathrm{step}^t\ } & \Delta Q \\
\uparrow & & \uparrow \\
\Sigma^* \times W^{\mathtt{code}} & \xrightarrow[\mathrm{step}^t]{} & Q
\end{array} \tag{11}$$

commute, where the bottom row is the usual execution of the UTM for $t$ steps of the simulated machine, and the vertical maps are the canonical inclusions. We read

$$\Delta\mathrm{step}^t(x, w) = \sum_y \Delta\mathrm{step}^t(x, w)_y \cdot y \in \Delta Q \tag{12}$$

as saying that when the UTM executes the noisy code $w$ on a (deterministic) input $x \in \Sigma^*$ for $t$ steps, the distribution over the state that results from propagating this uncertainty about the code to uncertainty about the final state of the simulated machine is

---

[3]There is an existing literature on noisy Turing machines, with some variations in the definition. Unlike [5] we do not consider random errors in every part of the configuration of the Turing machine. For example, we do not introduce random errors on the tape squares. Our noise is restricted to determining the tuples in the description to execute.

$\{\Delta \text{step}^t(x,w)_y\}_{y \in Q}$. If there is no uncertainty in the code, there is no uncertainty in the state. The precise smooth relaxation we have in mind is recalled in Section 5.2.

This becomes a problem of inductive inference [14, §7] if we observe the final state $y$ of the execution of some unknown machine when run on some inputs $x \in \Sigma^*$, and we wish to know *which Turing machine* explains these observations. If our model of the propagation of uncertainty is (12) then the conditional distribution of the state $y \in Q$ given $x, w$ is[4]

$$p(y|x,w) = \Delta \text{step}^t(x,w) \tag{13}$$

where we assume $t$ is fixed; for a more general class of synthesis problem see [45] and also Appendix A.3. According to Bayes' rule if we observe a set of inputs and final states $D_n = \{(x_i, y_i)\}_{i=1}^n$ the probability $p(w|D_n)$ of a given noisy code $w$ is

$$p(w|D_n) = \frac{p(D_n|w)\varphi(w)}{p(D_n)} = \frac{1}{Z_n}\varphi(w)\prod_{i=1}^n p(y_i|x_i, w) \tag{14}$$

where $\varphi(w)$ is our prior belief in codes, and $Z_n := p(D_n)$ is a normalising constant which is independent of $w$. Thus the degree of our belief in $w$ is controlled by the quantity

$$L_n(w) = -\sum_{i=1}^n \frac{1}{n}\log p(y_i|x_i, w) \tag{15}$$

which is called the *negative log-likelihood*, via the relationship

$$p(w|D_n) = \frac{1}{Z_n}\exp(-nL_n(w))\varphi(w), \tag{16}$$

$$Z_n = \int \exp(-nL_n(w))\varphi(w)dw. \tag{17}$$

Our philosophical position here is that ultimately the explanations we wish to consider are the (honest) Turing machine codes $w \in W^{\text{code}}$, but in executing these codes with the UTM we cannot avoid some small (perhaps extremely small) possibility of this execution being corrupted by read errors of the kind modelled by small perturbations within $W$.

Given Turing machines $M_1, M_2$ we compare codes $[M_1], [M_2] \in W^{\text{code}}$ by comparing the conditional probability that, given $D_n$, the explanation lies in a small neighbourhood $\mathcal{W}_i$ of $[M_i]$. These probabilities are

$$p(\mathcal{W}_i \mid D_n) = \frac{1}{Z_n}\int_{\mathcal{W}} \exp(-nL_n(w))\varphi(w)dw. \tag{18}$$

It turns out that, asymptotically as $n$ becomes large, this comparison is determined by the geometry at these points of the *average negative log-likelihood*

$$L = -\sum_{x \in I} q(x)\sum_{y \in Q} q(y|x)\log p(y|x,w).$$

---

[4]For expository purposes we give here a simplified form of the model, see Section 2.2 for details.

We explain how this comparison works in detail in Section 2.4. For now we take this as motivation to introduce the mathematical concept which captures the behaviour of a function at a point:

**Definition 2.4.** A *germ* of continuous (resp. analytic) real-valued functions on a space $X$ is an equivalence class of pairs $(U, f)$ where $U$ is an open neighbourhood of $x$, $f : U \longrightarrow \mathbb{R}$ is continuous (resp. analytic) and we say two such pairs are equivalent $(U, f) \sim (V, g)$ if there exists an open neighbourhood $Z$ of $x$ with $Z \subseteq U \cap V$ and $f(z) = g(z)$ for all $z \in Z$.

This provides the basic setting where we associate to a Turing machine $M$ the germ $([M], L)$ of an analytic function, and some property of this germ is relevant to the question of whether or not we should prefer $M$ as an explanation of the observed data, relative to some other candidate Turing machine.

We only consider *deterministic* synthesis problems for which there is a *true function* $y = y(x)$ from $I$ to $Q$ and the true conditional distribution $q(y|x) = \delta_{y=y(x)}$ puts all probability mass on this correct output. We suppose that two states $\{0, 1\} \subseteq Q$ have been identified (which we could read as for example reject, accept) and that $y(x) \in \{0, 1\}$ for all $x \in I$. For $z \in \{0, 1\}$ we sometimes write $\overline{z} = z + 1$ modulo 2.

**Definition 2.5.** Given a Turing machine $M$ we let

$$M(x) := \text{step}^t(x, [M]) \in Q$$

denote the state after executing $M$ for $t$ steps on input $x$.[5]

**Definition 2.6.** A Turing machine $M$ is a *classical solution* if $M(x) = y(x)$ for all $x \in I$.

**Remark 2.7.** By commutativity of (11) to be a classical solution is equivalent to

$$\Delta\text{step}^t\big(x, [M]\big)_{y(x)} = 1 \,, \qquad \forall x \in I \,.$$

**Definition 2.8.** We define

$$H(w) = \sum_x q(x) \, p\big(y \neq y(x) \mid x, w\big)^2 \tag{19}$$

where $y = y(x)$ is the true output.

To be explicit

$$p\big(y \neq y(x) \mid x, w\big) = \sum_{z \in Q, z \neq y(x)} \Delta\text{step}^t(x, w)_z \,. \tag{20}$$

For the smooth relaxation $\Delta\text{step}^t$ we use in this paper, the function $H$ is a polynomial function of $w$. By construction the zeros of $H$ among the classical codes $W^{\texttt{code}}$ are precisely

---

[5] We may also assume there is some common initialisation state $\underline{\text{init}} \in Q$.

the classical solutions. One can show [17, Lemma D.2] that the zeros of $H$ all lie on the boundary of the probability simplex, but we will not need that result here.

When things are set up correctly (see Section 2.2 below) the average log-likelihood $L(w)$ of the model is comparable to $H(w)$ up to an additive constant and so it suffices for many purposes to study the geometry of the polynomial function $H(w)$.

**Definition 2.9.** The *algebraic singularity* associated to a classical solution $M$ is the function germ $([M], H)$. This singularity depends on the UTM, the smooth relaxation (11) and the time cutoff $t$.

## 2.2  Details of the Model

For somewhat technical reasons the model we actually use is a slight variation on (13). The reader is advised to skip this section on a first reading. Before giving the complete definition of the model, here are the two issues with the naive definition in (13):

- **Correct vs Incorrect.** It is convenient to read the prediction from a single tape square of the UTM. Since we focus in this paper on the case of recognising languages, this should be a state of the simulated machine, as in (13). However, it is unnecessary to restrict the machine to two states *during operation*. So we simply measure the probability assigned by the code to the *correct* final state and everything else.

- **Moving off the Boundary.** It is convenient to analyse a polynomial $H(w)$ that is comparable to the (analytic, but not polynomial) KL divergence $K(w)$. This is possible (see Appendix B.2) but requires that we move the true distribution $q$ and the model $p$ "off the boundary" of the probability simplex $\Delta Q$. From a Bayesian point of view this can be thought of as insisting that zero probabilities are not realistic, so we introduce some (very small) chance of error.

Given the setting of the previous section, we define

$$p^{\mathbb{Z}/2}(z|x,w) = \begin{cases} p(z|x,w) & z = y(x) \\ \sum_{y \in Q, y \neq y(x)} p(y|x,w) & z \neq y(x) \end{cases}$$

where $p(y|x,w)$ is as in (13). That is, $p^{\mathbb{Z}/2}(z|x,w)$ is a distribution over $z \in \{0,1\}$ and

$$p^{\mathbb{Z}/2}(y(x)|x,w) = p(y(x)|x,w).$$

For $0 < \mu < 1$ let $\varepsilon_\mu : \Delta Q \longrightarrow \Delta Q$ move a probability distribution $\mathbf{x}$ a fraction $\mu$ towards the barycenter $\mathbf{b}$, i.e. the maximum entropy distribution (see Appendix B.3)

$$\varepsilon_\mu(\mathbf{x}) = (1 - \mu)\mathbf{x} + \mu\mathbf{b}.$$

We fix some small value of $\mu$, perhaps so small that with high probability the distinction from zero is never noticed in our experiments. The true distribution and statistical model we use in this paper are

$$q_\mu(y|x) = \varepsilon_\mu q(y|x)\,, \qquad p_\mu^{\mathbb{Z}/2}(y|x,w) = \varepsilon_\mu p^{\mathbb{Z}_2}(y|x,w)\,.$$

We let $L_\mu$ be the average log-likelihood defined with $q_\mu, p_\mu$.

**Definition 2.10.** We say that two analytic functions $f(w), g(w)$ on an open subset $U \subseteq \mathbb{R}^d$ are *comparable* (or *comparable up to constants*) if there exist $c, d > 0$ such that

$$c \cdot g(x) \le f(x) \le d \cdot g(x)\,, \qquad \forall x \in U\,.$$

It is easy to check that this is an equivalence relation on the set of analytic functions on $U$, and we write $f \sim g$ when $f, g$ are comparable.

Recall that we say the true distribution $q_\mu$ is *realisable* by the model $p_\mu$ if there exists $w \in W$ with $p_\mu(y|x,w) = q_\mu(y|x)$ for all $x \in I, y \in \{0,1\}$. But since $\varepsilon_\mu$ is injective that is just to say that $p^{\mathbb{Z}/2}(y|x,w) = q(y|x)$ for all $x, y$ which is to say that $p(y(x)|x,w) = 1$ for all $x \in I$. Since this is independent of $\mu$ we will simply say that the *true function is realisable*. In particular this is true if any classical solution exists.

**Lemma 2.11.** If the true function is realisable by $w^*$ then the KL divergence

$$K_\mu(w) = D_{\mathrm{KL}}(q_\mu \,\|\, p_\mu^{\mathbb{Z}/2}(w)) \tag{21}$$

is comparable to the polynomial $\frac{1}{2}H(w)$ in some open neighbourhood of $w^*$. In particular, the average log-likelihood $L_\mu(w)$ is comparable to $\frac{1}{2}H(w) + L_0^\mu$ where $L_0^\mu$ is a constant, the entropy of the true distribution $q_\mu$.

*Proof.* By Lemma B.6 applied to $q_\mu, p_\mu$ we find that $K_\mu(w)$ is comparable to $\frac{1}{2}H_\mu(w)$ in some neighbourhood of $w^*$, where $H_\mu(w)$ is defined by

$$H_\mu(w) = \sum_{x \in I} q(x) \sum_{z \in \mathbb{Z}/2} \left( p_\mu^{\mathbb{Z}/2}(z|x,w) - q_\mu(z|x) \right)^2\,.$$

By Appendix B.3 this is equal to

$$\tfrac{1}{2}(1-\mu)^2 \sum_{x \in I} q(x) \left[ \left( p^{\mathbb{Z}/2}(y(x)|x,w) - 1 \right)^2 + p^{\mathbb{Z}/2}\!\left(\overline{y(x)} \mid x, w\right)^2 \right]$$

$$= (1-\mu)^2 \sum_{x \in I} q(x)\, p^{\mathbb{Z}/2}\!\left(\overline{y(x)} \mid x, w\right)^2$$

which is $(1-\mu)^2 H(w)$, proving the claim. $\qquad \square$

Note that the zeros of $K_\mu(w)$ are by injectivity of $\varepsilon_\mu$ the same those of $D_{\mathrm{KL}}(q\|p(w))$. In the rest of this document we write $K, L$ for $K_\mu, L_\mu$.

**Definition 2.12.** The *analytic singularities* associated to a classical solution $M$ are the function germs $([M], K)$, $([M], L)$. These singularities depends on the UTM, the smooth relaxation (11), the time cutoff $t$ and $\mu > 0$.

Note that the germ is independent of $\mu$ up to the comparability relation. Since $K, L$ differ by a constant there is no essential difference and we use both in the sequel.

## 2.3 Learning Coefficients

Given a compact semi-analytic subset $\Omega \subseteq \mathbb{R}^d$, an analytic real-valued function $f$ on $\Omega$ and a smooth positive function $\varphi : \Omega \longrightarrow \mathbb{R}$ we consider the zeta function

$$\zeta(z) = \int_\Omega |f(w)|^z \varphi(w) dw \,. \tag{22}$$

This defines a holomorphic function on the half-plane $\text{Re}(z) > 0$ in $\mathbb{C}$ and provided $f$ vanishes somewhere on $\Omega$ the zeta function can be analytically continued to a meromorphic function on all of $\mathbb{C}$ with poles on the negative real line which are rational numbers and have a greatest element [47, §4.5], [26, Corollary 3.10]. If $-\lambda$ is the largest pole then $\lambda$ is called the *real log canonical threshold* of the pair $f, \varphi$ and is denoted $\text{RLCT}_W(f; \varphi)$.

In the setting of singular learning theory [46] we specify a triple $(p, q, \varphi)$ consisting of a model $p(y|x, w)$ a true distribution $q(y|x), q(x)$ and a prior distribution $\varphi(w)$ over some semi-analytic subset $W \subseteq \mathbb{R}^d$ which we refer to as parameter space. These have been defined in Section 2. Recall that $q(x)$ is a distribution supported on $I \subseteq \Sigma^*$ and we assume $q(y|x)$ is associated to what [17] calls a *deterministic synthesis problem*, that is, it is the distribution associated to a function $y = y(x)$. The prior $\varphi(w)$ does not play a significant role in this paper and may be chosen to be a (normalised) Lebesgue measure.

Let $K : W \longrightarrow \mathbb{R}$ denote the KL divergence associated to this choice of triple

$$K(w) = \int q(y|x) \log \frac{q(y|x)}{p(y|x, w)} q(x) dx \,. \tag{23}$$

Note that with

$$L(w) = - \int q(y|x) \log p(y|x, w) q(x) dx$$

we have $K(w) = -H(q) + L(w)$ where $H(q)$ is the entropy.

Taking $f = K$ in (22) we obtain the real log canonical threshold, which is also called the *learning coefficient* [47, §7] and denoted

$$\lambda = \text{RLCT}_W(K; \varphi) \,.$$

By [26, Proposition 3.9] this *global* learning coefficient is the minimum

$$\lambda = \min_{x \in \mathcal{V}} \text{RLCT}_{W_x}(K; \varphi)$$

over the set $\mathcal{V} = \{w \in W \mid K(w) = 0\}$. Here $W_x$ is a compact semi-analytic neighbourhood of $x \in \mathcal{V}$, which by the proof of [26, Proposition 3.9] may be chosen sufficiently small that the RLCT is independent of the choice of neighbourhood. When $x = [M]$ is the code of a Turing machine which is a classical solution we call this the *local learning coefficient* of $M$ associated to the synthesis problem encoded by $q$.

**Definition 2.13.** The *local learning coefficient* of $M$ for the synthesis problem $q$ is

$$\lambda([M], q) := \mathrm{RLCT}_{W_{[M]}}(K; \varphi) \tag{24}$$

where $W_{[M]}$ is a sufficiently small compact semi-analytic neighbourhood of $[M]$.

**Lemma 2.14.** Let $H$ be the polynomial function of Definition 2.8. Then

$$\mathrm{RLCT}_{W_{[M]}}(K; \varphi) = \mathrm{RLCT}_{W_{[M]}}(H; \varphi) \,.$$

*Proof.* Follows from Lemma 2.11 and [46, §7]. □

**Definition 2.15.** We define the *Newton polyhedron* of $M$ to be

$$\mathcal{P}(M) = \mathrm{conv}\left\{\mathbf{k} \in \mathbb{N}^d \,\Big|\, \frac{\partial^{|\mathbf{k}|}}{\partial w_1^{k_1} \cdots \partial w_d^{k_d}} H\Big|_{w=[M]} \neq 0\right\}. \tag{25}$$

The Newton polyhedron of $M$ is the convex hull of the set of monomials $w^{\mathbf{k}}$ represented as points $\mathbf{k}$ in $\mathbb{R}^d$ with nonzero coefficients in the Taylor series expansion of $H$ at $[M]$. The *distance* $l = l(M)$ of the Newton polyhedron is

$$l(M) = \inf\{s \geq 0 \mid (s, s, \ldots, s) \in \mathcal{P}(M)\} \,. \tag{26}$$

Note that by Theorem 6.3 the Newton polyhedron encodes some information about the error syndromes with weight $\leq \mathbf{k}$ which cause output errors in the execution of $M$.

**Proposition 2.16.** $\lambda([M], q) \leq 1/l(M)$.

*Proof.* Let $I = \langle H \rangle$ be the ideal generated by $H$ in the ring of germs of analytic functions at $[M]$ (let us change coordinates so this is at the origin). Then using the notation of [26, Proposition 4.3] it follows from [26, Proposition 4.8] that

$$\lambda([M], q) = \mathrm{RLCT}_0(I; 1) \leq \mathrm{RLCT}_0(\mathrm{mon}(I); 1) \,.$$

Now $\mathrm{mon}(I)$ is generated by $w^{\mathbf{k}}$ with $\frac{\partial^{|\mathbf{k}|}}{\partial w_1^{k_1} \ldots \partial w_d^{k_d}} H\big|_{w=[M]} \neq 0$ and hence the claim follows from [26, Theorem 4.18]. □

## 2.4 Free Energy and Comparison of Classical Solutions

In this paper we establish relations between internal structure of a Turing machine $M$ and the geometry of the associated singularities $([M], H), ([M], L), ([M], K)$. By further linking the geometry of these singularities, quantified by the local learning coefficient $\lambda([M], q)$, to the Bayesian learning process for the synthesis problem, we link the internal structure of $M$ with the behaviour of this process.

The central role played by the learning coefficient $\lambda$ in controlling the asymptotic form of the Bayesian posterior is captured by the *free energy*, which is the negative logarithm of the marginal likelihood [46]. For a dataset $D_n = \{(x_i, y_i)\}_{i=1}^n$ of size $n$ with $(x, y) \sim q(x, y)$.[6] Given a prior $\varphi(w)$ over codes $w \in W$, the *posterior* is defined by (16) with normalising constant $Z_n$. The free energy is defined to be

$$F_n := -\log Z_n = -\log \int_W \exp(-nL_n(w))\varphi(w)dw. \tag{27}$$

Under some technical conditions the asymptotic behaviour of $F_n$ as $n \to \infty$ is governed by Watanabe's free energy formula [47, §6.3], which asserts that

$$F_n = nL_n(w^*) + \lambda \log n + O_p(\log \log n), \tag{28}$$

where $w^* \in W$ minimises the average negative log-likelihood $L(w)$ and $\lambda$ is the learning coefficient. In this paper we assume the true distribution is realisable, so that the relevant conditions are satisfied.

This provides a natural asymptotic comparison between competing candidate explanations in the synthesis problem: given two classical solutions $M_1, M_2$ with respective codes $[M_1], [M_2] \in W^{\texttt{code}}$, the posterior probability mass contained in a small neighbourhood of each solution can be approximated in terms of the asymptotic behaviour of the free energies associated to those neighbourhoods. Our treatment follows [46, §7.6], [48], [11] and we refer to these references for full details. In the context of program synthesis these ideas were previously discussed in [17, 45].

Let $\mathcal{W}_1, \mathcal{W}_2 \subseteq W$ be sufficiently small compact semi-analytic neighbourhoods of $[M_1], [M_2]$ respectively, and let

$$F_n(\mathcal{W}_i) := -\log \int_{\mathcal{W}_i} \exp(-nL_n(w))\varphi(w)dw.$$

Then the posterior mass in $\mathcal{W}_i$ is approximately $\exp(-F_n(\mathcal{W}_i))$ and

$$F_n(\mathcal{W}_i) = nL_n([M_i]) + \lambda([M_i], q) \log n + O_p(\log \log n).$$

Since both $M_1, M_2$ are classical solutions $L([M_1]) = L([M_2])$. Thus in the large-sample limit the posterior probability assigned to each classical solution is controlled by the local

---

[6]In this section $n$ means the number of samples, elsewhere in the paper it is $n = |\Sigma|$.

learning coefficient, and the posterior will increasingly concentrate *around the solution with the smaller learning coefficient*. This explains why we are interested in understanding the kinds of internal structures of Turing machines $M$ which lead to lower learning coefficients $\lambda([M], q)$: the posterior will tend to prefer such machines among all other classical solutions for the same synthesis problem.

**Remark 2.17.** There can be nontrivial competition between the order $n$, $\log n$, $\log \log n$ and constant order terms in the asymptotic expansion of the free energy. Particularly for program synthesis we expect that some of these lower order terms may play a significant role, and the emphasis here on the learning coefficient should be viewed as an appropriate starting point for the theory but not the complete story.

**Remark 2.18.** While the discussion above compares classical solutions, the Bayesian posterior assigns nonzero mass to any neighbourhood in $W$ with positive prior measure. In particular, a machine $M'$ which is *not* a classical solution but which fits the data well on average and has a small learning coefficient $\lambda([M'], q)$ may dominate a classical solution $M$ in the posterior for small to moderate $n$, despite being wrong in the limit.

   This phenomenon is due to the competition between the loss term $nL_n([M'])$ and the complexity term $\lambda([M']) \log n$ in the free energy. Such competition can induce *Bayesian transitions* in the posterior, where the dominant contribution to the marginal likelihood shifts from one region of parameter space to another as $n$ increases [46, §7.6], [11].

# 3   Geometry

Given a synthesis problem we have associated in Section 2.1 to a classical solution $M$ the germ $([M], L)$ of an analytic function. The claim made in the introduction is that there is some relation between the "structure" of $M$ and the geometry of the germ. In this section we explain in a self-contained way what we mean by this geometry, and sketch the first layer of the relation to the structure of $M$.

   The germ $([M], L)$ is equivalent to the information of all the Taylor series coefficients

$$\frac{\partial^{|\alpha|} L}{\partial^{\alpha_1} w_1 \cdots \partial^{\alpha_d} w_d}\bigg|_{w=[M]} \tag{29}$$

where $w_1, \ldots, w_d$ are the coordinates of the parameter space $W$ and for $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}^d$ we write $|\alpha| = \sum_{i=1}^{d} \alpha_i$. Thus, roughly speaking, the geometric information comes in *layers* with the $r$th layer corresponding to the information in the Taylor series coefficients with $|\alpha| \leq r$. In this section we analyse the first two layers. For the higher layers we need more algebraic geometry. As explained in the previous section, we find it easier to analyse the geometric information in the Taylor series expansion of the polynomial function $H$, and deduce from this geometric information about $L, K$ (see Section 2.2).

**Remark 3.1.** In this section the choice of coordinates $w_1, \ldots, w_d$ is arbitrary. Note that the $w_i$ are *not* probabilities themselves, since these are not all independent. Later we will describe explicit coordinates in a neighbourhood of a Turing machine code.

We assume that there is a given true function $y = y(x)$ from $I$ to $\{0,1\} \subseteq Q$ and that $M$ is a classical solution. Thus $H([M]) = 0$ and $\nabla H([M]) = 0$. The second-order information in the germ is contained in the Hessian matrix

$$\mathrm{Hess}(H, [M]) = \left( \frac{\partial^2 H}{\partial w_i \partial w_j} \bigg|_{w=[M]} \right)_{1 \leq i,j \leq d}. \tag{30}$$

The geometry we are interested in does not depend on any choice of local coordinates. Since $\mathrm{Hess}(H, [M])$ is real symmetric and therefore diagonalisable, that means we are interested in the eigenvalues and eigenvectors of this matrix.

**Definition 3.2.** Given $\mathbf{i} \in \mathbb{N}^d$ the *influence function*

$$g_{\mathbf{i}} : \Sigma^* \longrightarrow \mathbb{R}$$

is defined by

$$g_{\mathbf{i}}(x) = \frac{\partial^{|\mathbf{i}|}}{\partial w_1^{i_1} \cdots \partial w_d^{i_d}} \, p\big(y \neq y(x) \mid x, w\big)\bigg|_{w=[M]}. \tag{31}$$

When $\mathbf{i} = e_i$ is the $i$th standard basis vector we write $g_i(x)$ for $g_{\mathbf{i}}(x)$.

In the coordinates that we will choose later at the point $[M]$ in $W$, each index $1 \leq i \leq d$ will correspond to a particular direction to vary a particular bit in the code $[M]$ away from its given value, and the derivative $g_i(x)$ measures the rate of change of the probability of the incorrect outputs with respect to this variation.

By the assumption that $M$ is a classical solution, we have $g_{\mathbf{0}}(x) = 0$ where $\mathbf{0} = (0, \ldots, 0)$ means no derivatives.

**Lemma 3.3.** Let $|\mathbf{k}| > 0$. Then

$$\frac{\partial^{|\mathbf{k}|}}{\partial w_1^{k_1} \cdots \partial w_d^{k_d}} H \bigg|_{w=[M]} = \sum_{\substack{\mathbf{i},\mathbf{j}\neq 0 \\ \mathbf{i}+\mathbf{j}=\mathbf{k}}} C(\mathbf{i}, \mathbf{j}) \, \mathbb{E}_x \Big[ g_{\mathbf{i}}(x) g_{\mathbf{j}}(x) \Big] \tag{32}$$

where $C(\mathbf{i}, \mathbf{j}) = \prod_{s=1}^d \binom{i_s + j_s}{i_s}$, $\mathbb{E}_x[-]$ denotes expectation with respect to $q(x)$ and $\mathbf{i}, \mathbf{j} \in \mathbb{N}^d$.

*Proof.* For the duration of this proof we write

$$g(x, w) = p\big(y \neq y(x) \mid x, w\big).$$

Recall that given functions $p(w), q(w)$ we have

$$\frac{\partial^{|\mathbf{k}|}}{\partial w_1^{k_1} \cdots \partial w_d^{k_d}} \Big[ p(w)q(w) \Big] = \sum_{0 \leq \mathbf{i} \leq \mathbf{k}} \Big\{ \prod_{s=1}^d \binom{k_s}{i_s} \Big\} \frac{\partial^{|\mathbf{i}|}}{\partial w_1^{i_1} \cdots \partial w_d^{i_d}} p(w) \frac{\partial^{|\mathbf{j}|}}{\partial w_1^{j_1} \cdots \partial w_d^{j_d}} q(w)$$

where $\mathbf{i} \leq \mathbf{k}$ means $i_s \leq k_s$ for $1 \leq s \leq d$ and $\mathbf{j} = \mathbf{k} - \mathbf{i}$. Recall that $\binom{0}{0} = 1$ by convention.

Substituting $p(w) = q(w) = g(x, w)$ we have

$$\frac{\partial^{|\mathbf{k}|}}{\partial w_1^{k_1} \cdots \partial w_d^{k_d}} H(w) = \sum_x q(x) \sum_{\substack{\mathbf{i},\mathbf{j} \geq 0 \\ \mathbf{i}+\mathbf{j}=\mathbf{k}}} C(\mathbf{i}, \mathbf{j}) \frac{\partial^{|\mathbf{i}|}}{\partial w_1^{i_1} \cdots \partial w_d^{i_d}} g(x, w) \frac{\partial^{|\mathbf{j}|}}{\partial w_1^{j_1} \cdots \partial w_d^{j_d}} g(x, w) \,.$$

Evaluating at $w = [M]$ we obtain

$$\frac{\partial^{|\mathbf{k}|}}{\partial w_1^{k_1} \cdots \partial w_d^{k_d}} H(w) \Big|_{w=[M]} = \sum_x q(x) \sum_{\substack{\mathbf{i},\mathbf{j} \geq 0 \\ \mathbf{i}+\mathbf{j}=\mathbf{k}}} C(\mathbf{i}, \mathbf{j}) g_{\mathbf{i}}(x) g_{\mathbf{j}}(x) \,.$$

Since $g_{\mathbf{0}}(x) = 0$ we obtain the result. $\qquad\square$

In particular for $1 \leq i \leq d$

$$\frac{\partial}{\partial w_i} H \Big|_{w=[M]} = 0 \,.$$

That is, $[M]$ is a critical point of $H$, which is why we call the germ $([M], H)$ a *singularity*. Another important special case is when $\mathbf{k} = e_i + e_j$ for $1 \leq i, j \leq d$. Then when $i \neq j$ we have two summands indexed by $(\mathbf{i}, \mathbf{j})$ equal to $(e_i, e_j), (e_j, e_i)$ and when $i = j$ just one summand with $\mathbf{i} = \mathbf{j} = e_i$, so in either case

$$\frac{\partial^2}{\partial w_i \partial w_j} H \Big|_{w=[M]} = 2 \, \mathbb{E}_x \Big[ g_i(x) g_j(x) \Big] \,. \tag{33}$$

Let $I \subseteq \Sigma^*$ be the set of allowed inputs. If we write $P$ for the $|I| \times d$ matrix

$$P = \big( g_i(x) \big)_{x,i}$$

then we have shown

$$\mathrm{Hess}(H, [M]) = 2 P^T Q P = 2 (Q^{1/2} P)^T (Q^{1/2} P) \tag{34}$$

where $Q$ is the diagonal matrix with entries $q(x)$. This means that the spectrum of the Hessian is the same as the singular values of $Q^{1/2} P$. In particular the Hessian and $Q^{1/2} P$ have the same rank, which is the rank of $P$.

**Lemma 3.4.** Let $V$ be the space of functions in $\mathbb{R}^I$ spanned by $g_1, \ldots, g_d$. Then the rank of $\mathrm{Hess}(H, [M])$ is equal to $\dim(V)$. In particular, the Hessian is nondegenerate if and only if the $g_i$ are linearly independent.

We have seen in Section 2 how inductive inference on Turing machines, viewed from the perspective of embedding ordinary codes into noisy codes, boils down to the comparison of integrals of the posterior over regions of parameter space (18) and how the asymptotics of those comparisons are sensitive to the geometry of the analytic function $L : W \longrightarrow \mathbb{R}$ around a code $[M]$. That is, to the geometry of the germ $([M], L)$. Studying the geometry

of this germ has now lead us to the following question: what does it mean *computationally* for there to be a linear dependence relation

$$\sum_{i=1}^{d} a_i g_i(x) = 0, \qquad \forall x \in I. \tag{35}$$

In the rest of the paper we explore this question, which we answer in Section 6.4.

# 4   Derivatives and Errors for Plain Proofs

## 4.1   Linear Logic

In this paper *linear logic* means *first-order intuitionistic linear logic* with connectives $\otimes, \&, \multimap, !$ along with the corresponding introduction rules and cut-elimination transformations from [28, 6]. Our use of linear logic follows [15, 16, 14]. For an introduction we recommend [20, 28, 29].

In this paper the role of linear logic is that by encoding the step function of a UTM into this logic, and then making use of the semantics of linear logic in vector spaces, we obtain a natural smooth relaxation of the execution of the UTM in the sense of (11). In particular, this provides a definition of the probability $p(y \neq y(x)|x, w)$ of an output error on some input $x$ when $w$ is a noisy Turing machine. The reader who is willing to take this probabilistic model at face value is encouraged to skip this section on a first reading.

Fix an algebraically closed field $\Bbbk$.[7] Given formulas (or synonymously, *types*) $A, B$ the definition of the *Sweedler semantics* [29, 30, 13] of linear logic in the category of vector spaces over $\Bbbk$, which we denote by $[\![-]\!]$, associates a vector space $[\![X]\!]$ to each atomic formula $X$, and to $A, B$, the vector spaces determined by the following rules:

$$[\![A \otimes B]\!] = [\![A]\!] \otimes [\![B]\!]$$
$$[\![A \& B]\!] = [\![A]\!] \oplus [\![B]\!]$$
$$[\![A \multimap B]\!] = \mathrm{Hom}_{\Bbbk}([\![A]\!], [\![B]\!])$$
$$[\![!A]\!] = ![\![A]\!]$$

where $!V$ denotes the universal cocomutative coassociative coalgebra of $V$ (referred to here as simply the *cofree coalgebra*). The universal morphism is usually denoted $d_V : !V \longrightarrow V$ or just $d$. An explicit description of the cofree coalgebra is [30]

$$!V = \bigoplus_{P \in V} \mathrm{Sym}_P(V) \tag{36}$$

where $\mathrm{Sym}_P(V) = \mathrm{Sym}(V)$ is the symmetric algebra with its canonical structure as a coalgebra. If $V$ is of dimension $n$ and $e_1, \ldots, e_n$ is a basis for $V$ then as a vector space

---

[7] We take $\Bbbk = \mathbb{C}$ but all coefficients are taken to be real.

there is an isomorphism $\mathrm{Sym}(V) \cong \Bbbk[e_1, \ldots, e_n]$. The denotations of proofs are defined such that if $\rho$ is a proof of $A$, written $\rho : A$, then $[\![\rho]\!] \in [\![A]\!]$. In particular $\varphi : A \vdash B$ then $[\![\varphi]\!] : [\![A]\!] \longrightarrow [\![B]\!]$ is a linear transformation. We sometimes write $\varphi(\rho) : B$ for the cut of $\varphi, \rho$ and think in this way of $\varphi$ as defining a function from cut-elimination equivalence classes of proofs of $A$ to such equivalence classes of proofs of $B$. If

$$\psi : !A_1, \ldots, !A_r \vdash B$$

is a proof, and $\rho_i : A_i$, then we also sometimes write

$$\psi(\rho_1, \ldots, \rho_r) : B \tag{37}$$

for the proof obtained by cutting $\psi$ against the promotions of the $\rho_i$ to proofs of $\vdash !A_i$.

These denotations of proofs are constructed using the structural maps of the functors $\otimes, \&, \multimap$ as well as $!$. We briefly recall the latter. Given $v_1, \ldots, v_s \in V$, the corresponding vector in $\mathrm{Sym}_P(V)$ is written using a ket

$$|v_1, \ldots, v_s\rangle_P := v_1 \otimes \ldots \otimes v_s \in \mathrm{Sym}_P(V). \tag{38}$$

The identity element $1 \in \mathrm{Sym}_P(V)$ is denoted by $|\emptyset\rangle_P$. For a subset $I = \{i_1, \ldots, i_p\}$ we denote by $v_I$ the sequence $v_{i_1}, \ldots, v_{i_p}$ and $I^c$ is the complement of $I$. With this notation, the universal map $d$ is defined by

$$d|\emptyset\rangle_P = P, \quad d|v\rangle_P = v, \quad d|v_1, \ldots, v_s\rangle_P = 0 \quad s > 1 \tag{39}$$

and the comultiplication on $!V$ is defined by

$$\Delta|v_1, \ldots, v_s\rangle_P = \sum_{I \subseteq \{1,\ldots,s\}} |v_I\rangle_P \otimes |v_{I^c}\rangle_P \tag{40}$$

where $I$ ranges over all subsets including the empty set. In particular

$$\Delta|\emptyset\rangle_P = |\emptyset\rangle_P \otimes |\emptyset\rangle_P. \tag{41}$$

We write $A^n = A \& \ldots \& A$ where there are $n$ copies of $A$.

### 4.1.1 Proofs

A *proof* in linear logic is a decorated tree that represents the application of deduction rules to instances of axioms; see [20, 28]. Following the Brouwer-Heyting-Kolmogorov interpretation [37] of intuionistic proof, or the Curry-Howard correspondence [24], we can think of proofs in intuitionistic linear logic as algorithms which take inputs in the form of proofs of their hypotheses and produce an output which is the proof of their conclusion.

In general a proof in linear logic, viewed as an algorithm, will use a different number of copies of its inputs depending on other inputs. For example the multiplication of Church numerals iterates one of its inputs a number of times specified by another of its inputs. Such a proof is *not* plain, in the sense of the following definitions. The defining characteristic of a plain proof is that the number of copies of each input required to compute the output is a *property of the proof itself* and does not depend on the inputs.

**Definition 4.1.** A *linear proof* will mean a proof in linear logic which only uses the connectives $\otimes, \&, \multimap$ with the corresponding introduction rules.

Throughout in the hypotheses $n\,A$ means $\overbrace{A, \ldots, A}^{n \text{ times}}$.

**Definition 4.2.** A proof $\psi$ of a sequent $!A_1, \ldots, !A_r \vdash B$ for $r \geq 0$ is *plain* if it is equivalent under cut-elimination to

$$
\begin{array}{c}
\pi \\
\vdots \\
\dfrac{n_1\,A_1, \ldots, n_r\,A_r \vdash B}{\dfrac{n_1\,!A_1, \ldots, n_r\,!A_r \vdash B}{!A_1, \ldots, !A_r \vdash B}\;\text{ctr/wk}}\;\text{der}
\end{array}
$$

for some linear proof $\pi$ and non-negative integers $n_1, \ldots, n_r$ where for $n_i > 1$ in the final step there is a corresponding contraction rule, and if $n_i = 0$ the final step involves a weakening rule. We refer to $n_i$ as the *degree* of $A_i$. We call $\pi$ the *linear part* of $\psi$.

**Remark 4.3.** In this paper all plain proofs have positive degrees $n_i > 0$ for $1 \leq i \leq r$.

A proof is *component-wise plain* if it is obtained by tensoring together a sequence of plain proofs; see [16, Definition 3.5].

**Example 4.4.** Many simple data types such as booleans, integers and binary integers have a natural encoding as types in linear logic. Some of these are reviewed in [13, §3], among which are the Church numerals $\underline{m}$, $m \geq 0$, (for some given formula $A$) of type

$$
\underline{m} : \mathbf{int}_A = !(A \multimap A) \multimap (A \multimap A)
$$

and the integers $\underline{i}$, $0 \leq i < n$ for fixed $n > 0$ of generalised Boolean type

$$
\underline{i} : {}_n\mathbf{bool}_A = A^n \multimap A.
$$

These proofs are all plain. Generally we write types with bold letters, and denote proofs with underlines. The proof $\underline{i}$ encodes a projection onto one of $n$ factors.

### 4.1.2   Denotations of Plain Proofs

The denotation of a plain proof $\psi$ is defined by a set of polynomials $f_\psi^\tau$ which will play an important role in the sequel and which we now define. Here $\psi : !A_1, \ldots, !A_r \vdash B$ denotes a plain proof with linear part $\pi : n_1 A_1, \ldots, n_r A_r \vdash B$ as in Definition 4.2. Suppose given finite sets of proofs $\mathcal{P}_i$ of $A_i$ and $\mathcal{Q}$ of $B$. Assume that $\{[\![\nu]\!]\}_{\nu \in \mathcal{Q}}$ is linearly independent in $[\![B]\!]$ and that

$$
\{\pi(X_1, \ldots, X_r) \mid X_i \in \mathcal{P}_i^{n_i}\} \subseteq \mathcal{Q}. \tag{42}
$$

Our polynomials have variables $\{x_j^i\}_{1 \leq i \leq r, 1 \leq j \leq n_i}$.

The way denotations work is that

$$[\![\psi]\!]\Big(|\emptyset\rangle_{[\![\rho_1]\!]} \otimes \cdots \otimes |\emptyset\rangle_{[\![\rho_r]\!]}\Big) = [\![\psi(\rho_1,\ldots,\rho_r)]\!]$$

$$= [\![\pi(\overbrace{\rho_1,\ldots,\rho_1}^{n_1 \text{ times}},\ldots,\overbrace{\rho_r,\ldots,\rho_r}^{n_r \text{ times}})]\!]$$

where $\psi(\rho_1,\ldots,\rho_r)$ recall is defined by cutting $\psi$ against the promotions of the $\rho_i : A_i$ as in (37). However there is more information in the denotation $[\![\psi]\!]$ than is in this formula, and it is this information that is surfaced when we take derivatives.

**Definition 4.5.** Given $\tau \in \mathcal{Q}$ we define

$$f_\psi^\tau = \sum_{\gamma_1,\ldots,\gamma_r} \delta_{\tau=\mathrm{eval}(\pi,\gamma)} \prod_{i=1}^r \prod_{j=1}^{n_i} x_{\gamma_i(j)}^i \tag{43}$$

where $\gamma_i$ ranges over all functions $\{1,\ldots,n_i\} \longrightarrow \mathcal{P}_i$, and $\mathrm{eval}(\pi,\gamma)$ means

$$\mathrm{eval}(\pi,\gamma) = \pi(\gamma_1(1),\ldots,\gamma_1(n_1),\ldots,\gamma_r(1),\ldots,\gamma_r(n_r)), \tag{44}$$

The next lemma says that $[\![\psi]\!]$ is computed by these polynomials $f_\psi^\tau$. Given a set $\mathcal{A}$ we denote by $\Bbbk\mathcal{A}$ the free vector space on the set, that is, the vector space with the elements of $\mathcal{A}$ as a basis.

**Lemma 4.6.** There exists a unique function $F_\psi$ making the diagram

$$
\begin{array}{ccc}
![\![A_1]\!] \otimes \ldots \otimes ![\![A_r]\!] & \xrightarrow{\;[\![\psi]\!]\;} & [\![B]\!] \\
\iota \uparrow & & \uparrow [\![-]\!] \\
\Bbbk\mathcal{P}_1 \times \ldots \times \Bbbk\mathcal{P}_r & \xrightarrow{\;F_\psi\;} & \Bbbk\mathcal{Q}
\end{array}
$$

commute, where $\iota$ maps an element

$$\Big(\sum_{\rho\in\mathcal{P}_1} \lambda_\rho^1 \rho, \ldots, \sum_{\rho\in\mathcal{P}_r} \lambda_\rho^r \rho\Big) \in \Bbbk\mathcal{P}_1 \times \ldots \times \Bbbk\mathcal{P}_r \tag{45}$$

to the tensor

$$|\emptyset\rangle_{\sum_{\rho\in\mathcal{P}_1} \lambda_\rho^1 [\![\rho]\!]} \otimes \ldots \otimes |\emptyset\rangle_{\sum_{\rho\in\mathcal{P}_r} \lambda_\rho^r [\![\rho]\!]}. \tag{46}$$

The function $F_\psi$ maps (45) to

$$\sum_{\tau\in\mathcal{Q}} f_\psi^\tau\Big|_{x_\rho^i = \lambda_\rho^i} \cdot \tau. \tag{47}$$

*Proof.* See [16, Proposition 3.8]. □

The papers [14], [15], [16] give an interpretation of the polynomials $f_\psi^\tau$ using a probabilistic semantics. Recall from Definition 2.3 that associated to every finite set $Z$ there is a simplex $\Delta Z$. The following is [14, Proposition 5.3].

25

**Proposition 4.7.** There is a unique function

$$\Delta\psi : \Delta\mathcal{P}_1 \times \ldots \times \Delta\mathcal{P}_r \longrightarrow \Delta\mathcal{Q} \tag{48}$$

which makes the diagram

$$\begin{array}{ccc}
![\![A_1]\!] \otimes \ldots \otimes ![\![A_r]\!] & \xrightarrow{[\![\psi]\!]} & [\![B]\!] \\
\text{inc} \big\uparrow & & \big\uparrow [\![-]\!] \\
\Delta\mathcal{P}_1 \times \ldots \times \Delta\mathcal{P}_r & \xrightarrow{\Delta\psi} & \Delta\mathcal{Q}
\end{array} \tag{49}$$

commute.

That is, probability distributions are sent to probability distributions by $[\![\psi]\!]$. By uniqueness of such a function $\Delta\psi$ has the same equation as $F_\psi$. Note that $\Delta\mathcal{P} \subseteq \Bbbk\mathcal{P}$ and so the left hand vertical arrow in (49) is a restriction of the $\iota$ in Lemma 4.6.

In the next section we explain how to think about this function $\Delta\psi$ and what it means to propagate uncertainty through $\psi$ (which we mean evaluate $\Delta\psi$ on input distributions, representing uncertainty about proofs in $\mathcal{P}_i$). See also [14, Section 5.1].

## 4.2 Evaluating Programs in Vector Spaces

From the point of view of linear logic the inputs to a Turing machine (encoded by its step function) are represented as proofs, and these inputs are "distributed" through the steps of the Turing machine by contraction rules, to eventually be "unpackaged" by dereliction rules and consumed by linear parts of the proof. Semantically, the contraction or copying step is represented by comultiplication $\Delta$ on the coalgebra $![\![A]\!]$ and the dereliction step by the counit $d : ![\![A]\!] \longrightarrow [\![A]\!]$. For background on the cofree coalgebra see [29, 30, 15].

For example if $\rho : A$ is a proof representing such an input (say some bit in the code of a Turing machine on the description tape of a UTM) then a pair of copies are made via

$$\Delta|\emptyset\rangle_{[\![\rho]\!]} = |\emptyset\rangle_{[\![\rho]\!]} \otimes |\emptyset\rangle_{[\![\rho]\!]} . \tag{50}$$

At the point where these copies are to be used, they are "unpackaged" by derelictions

$$(d \otimes d)\Delta|\emptyset\rangle_{[\![\rho]\!]} = [\![\rho]\!] \otimes [\![\rho]\!] . \tag{51}$$

Since the comultiplication is co-associative, we have $(1 \otimes \Delta)\Delta = (\Delta \otimes 1)\Delta$ and following standard practice we denote both maps by $\Delta^2$. Similarly, $\Delta^n$ denotes all possible ways of arranging $n$ comultiplications, one after another (which are the same linear map). Thus to make and consume $n$ copies we are computing

$$d^{\otimes n}\Delta^{n-1}|\emptyset\rangle_{[\![\rho]\!]} = d^{\otimes n}\left(|\emptyset\rangle_{[\![\rho]\!]} \otimes \cdots \otimes |\emptyset\rangle_{[\![\rho]\!]}\right) = [\![\rho]\!] \otimes \cdots \otimes [\![\rho]\!] . \tag{52}$$

A plain proof $\psi : \, ! \vdash B$ is, by definition, a linear part $\pi$ below which are derelictions and contractions (and possibly weakenings). Thus its denotation is, roughly speaking and accounting for only one input of type $A$, of the form $[\![\pi]\!]d^{\otimes n}\Delta^{n-1}$. Thus we may compute $[\![\psi]\!]|\emptyset\rangle_{[\![\rho]\!]}$ by applying the denotation $[\![\pi]\!]$ of the linear part to the above:

$$[\![\psi]\!]|\emptyset\rangle_{[\![\rho]\!]} = [\![\pi]\!]d^{\otimes n}\Delta^{n-1}|\emptyset\rangle_{[\![\rho]\!]} = [\![\pi(\rho,\ldots,\rho)]\!] \, . \tag{53}$$

To compute derivatives of denotations of proofs is to provide inputs other than vacuums $|\emptyset\rangle_{[\![\rho]\!]}$ [14, Corollary 4.5]. This has an easy to understand effect on the set of copies that are provided throughout the proof:

$$d^{\otimes n}\Delta^{n-1}|[\![\nu]\!]\rangle_{[\![\rho]\!]} = d^{\otimes n}\Big( \sum_{i=1}^{n}|\emptyset\rangle_{[\![\rho]\!]} \otimes \cdots \otimes \underbrace{|[\![\nu]\!]\rangle_{[\![\rho]\!]}}_{i\text{th}} \otimes \cdots \otimes |\emptyset\rangle_{[\![\rho]\!]}\Big)$$

$$= \sum_{i=1}^{n}[\![\rho]\!] \otimes \cdots \otimes \underbrace{[\![\nu]\!]}_{i\text{th}} \otimes \cdots \otimes [\![\rho]\!] \, . \tag{54}$$

That is, among all the $n$ copies of $\rho$ that are produced to be unpackaged and consumed in the proof, we substitute exactly one copy with $\nu$. This is the central idea in Ehrhard-Regnier's notion of derivative [18]. If we now apply a linear proof $\pi$ to this

$$[\![\pi]\!]d^{\otimes n}\Delta^{n-1}|[\![\nu]\!]\rangle_{[\![\rho]\!]} = \sum_{i=1}^{n}[\![\pi(\rho,\ldots,\nu,\ldots,\rho)]\!] \, . \tag{55}$$

Hence

$$[\![\pi]\!]d^{\otimes n}\Delta^{n-1}|[\![\nu]\!] - [\![\rho]\!]\rangle_{[\![\rho]\!]} = \sum_{i=1}^{n}\big\{ [\![\pi(\rho,\ldots,\nu,\ldots,\rho)]\!] - [\![\pi(\rho,\ldots,\rho,\ldots,\rho)]\!]\big\} \, . \tag{56}$$

Now suppose that we have a linear combination $\sum_{\rho} x_{\rho}\rho$. Then

$$d^{\otimes n}\Delta^{n-1}|\emptyset\rangle_{\sum_{\rho} x_{\rho}[\![\rho]\!]} = d^{\otimes n}\Big( |\emptyset\rangle_{\sum_{\rho} x_{\rho}[\![\rho]\!]} \otimes \cdots \otimes |\emptyset\rangle_{\sum_{\rho} x_{\rho}[\![\rho]\!]}\Big)$$

$$= \sum_{\rho_1,\ldots,\rho_n} x_{\rho_1} \cdots x_{\rho_n}[\![\rho_1]\!] \otimes \cdots \otimes [\![\rho_n]\!] \, .$$

If these copies are to be consumed in the linear part $\pi$ of the overall component-wise plain proof $\psi$ (which is roughly $\pi d^{\otimes n}\Delta^{n-1}$) then we end up with an output

$$[\![\psi]\!]|\emptyset\rangle_{\sum_{\rho} x_{\rho}[\![\rho]\!]} = [\![\pi]\!]d^{\otimes n}\Delta^{n-1}|\emptyset\rangle_{\sum_{\rho} x_{\rho}[\![\rho]\!]}$$

$$= \sum_{\rho_1,\ldots,\rho_n} x_{\rho_1} \cdots x_{\rho_n}[\![\pi(\rho_1,\ldots,\rho_n)]\!]$$

$$= \sum_{\tau}\Big\{ \sum_{\rho_1,\ldots,\rho_n} \delta_{\tau=\pi(\rho_1,\ldots,\rho_n)}x_{\rho_1} \cdots x_{\rho_n}\Big\}[\![\tau]\!]$$

$$= \sum_{\tau} f_{\psi}^{\tau}[\![\tau]\!] \tag{57}$$

where $f_\psi^\tau$ is the polynomial of Definition 4.5 in the special case where there is a single input and $\rho$ runs over the proofs chosen in $\mathcal{P}$. Note that if $\sum_\rho x_\rho \rho$ is a probability distribution, then we can define $p(\tau) := f_\psi^\tau$ in which case

$$p(\tau) = \sum_\gamma p(\tau|\gamma)p(\gamma) \tag{58}$$

where $p(\tau|\gamma) := \delta_{\tau=\pi(\rho_1,\ldots,\rho_n)}$ and $p(\gamma) := x_{\rho_1} \cdots x_{\rho_n}$. Here $\gamma$ ranges over sequences of proofs $\rho_1, \ldots, \rho_n$. Note that the linear part $\pi$ requires $n$ inputs, and when we evaluate $\psi$ on a linear combination (as opposed to the denotation of a proof) we can end up passing the linear part sequences other than $\rho, \ldots, \rho$ as in the normal evaluation (53).

Thus the polynomials $f_\psi^\tau$, which are the central algebraic objects, can be thought of as presenting the probabilities of output states $\tau$ by marginalising out some random variable $\gamma$ which represents a sequence of $n$ choices of proofs of type $A$. This random variable $\gamma$ is the central object of the present paper, and we explain in the next section how to think of it as an *error syndrome*.

## 4.3 Error Syndromes for Plain Proofs

### 4.3.1 Special Case

We begin with the case $\psi : !A \vdash B$ where $\psi$ has a single input with linear part $\pi : nA \vdash B$ and we restrict to the case where our set of allowed proofs is $\mathcal{P} = \{0, 1\}$. We consider a situation in which 0 is the "correct" input and 1 is an error. So the correct output is

$$\psi(0) = \pi(0, \ldots, 0)$$

and an error syndrome $\gamma$ is by definition an assignment to every computation path through the proof of either a 0 (meaning that this path experiences no error) or 1 (meaning that for this path there is an error). This explains the term *error syndrome* which is the standard terminology for such a pattern in the theory of error correcting codes.

**Theorem 4.8.** We have

$$[\![\psi]\!]|[\![1]\!] - [\![0]\!]\rangle_{[\![0]\!]} = \sum_{\tau \neq \psi(0)} A_\tau \left\{ [\![\tau]\!] - [\![\psi(0)]\!] \right\} \tag{59}$$

where $A_\tau$ is a non-negative integer defined for $\tau \neq \psi(\underline{0})$ by

$$A_\tau = \left| \left\{ 1 \leq i \leq n \mid \pi(0, \ldots, \overset{i}{1}, \ldots, 0) = \tau \right\} \right|$$

the count of the *number of weight one error syndromes that produce $\tau$*.

*Proof.* We consider a linear combination of these denotations in $[\![A]\!]$ with coefficients $x_0, x_1$ and then differentiate the output probabilities obtained using (57):

$$\left\{ \frac{\partial}{\partial x_1} - \frac{\partial}{\partial x_0} \right\} [\![\psi]\!] |\emptyset\rangle_{x_0 [\![0]\!] + x_1 [\![1]\!]} \bigg|_{x_0 = 1, x_1 = 0}$$

$$= \sum_\tau \sum_{\rho_1, \ldots, \rho_n} \delta_{\tau = \pi(\rho_1, \ldots, \rho_n)} \left\{ \frac{\partial}{\partial x_1} - \frac{\partial}{\partial x_0} \right\} \left( x_{\rho_1} \cdots x_{\rho_n} \right) \bigg|_{x_0 = 1, x_1 = 0} [\![\tau]\!]$$

$$= \sum_{i=1}^n \sum_\tau \sum_{\rho_1, \ldots, \rho_n} \delta_{\tau = \pi(\rho_1, \ldots, \rho_n)} \left\{ x_{\rho_1} \cdots \frac{\partial}{\partial x_1} (x_{\rho_i}) \cdots x_{\rho_n} - \right.$$

$$\left. x_{\rho_1} \cdots \frac{\partial}{\partial x_0} (x_{\rho_i}) \cdots x_{\rho_n} \right) \bigg|_{x_0 = 1, x_1 = 0} [\![\tau]\!]$$

$$= \sum_{i=1}^n \left\{ [\![\pi(0, \ldots, \overset{i}{1}, \ldots, 0)]\!] - [\![\psi(0)]\!] \right\}$$

$$= \sum_{\tau \neq \psi(0)} A_\tau \left\{ [\![\tau]\!] - [\![\psi(0)]\!] \right\} .$$

Note that under the prevailing hypotheses the set $\{[\![\tau]\!]\}_\tau$ is linearly independent, hence $\{[\![\tau]\!] - [\![\psi(0)]\!]\}_{\tau \neq \psi(0)}$ is linearly independent. The conclusion follows from [14, Corollary 4.5] which shows that the quantity calculated above is $[\![\psi]\!] |[\![1]\!] - [\![0]\!]\rangle_{[\![0]\!]}$. $\qquad\square$

In general we can express the derivatives of $f_\psi^\tau$ in terms of combinatorics of higher-weight error syndromes that produce $\tau$.

### 4.3.2 General Case

Let $\psi$ be a plain proof of a sequent $!A_1, \ldots, !A_r \vdash B$ as in Definition 4.2 with linear part $\pi : n_1 A_1, \ldots, n_r A_r \vdash B$ as in the following diagram:

$$\begin{array}{c} \pi \\ \vdots \\ \dfrac{n_1 A_1, \ldots, n_r A_r \vdash B}{\dfrac{n_1 !A_1, \ldots, n_r !A_r \vdash B}{!A_1, \ldots, !A_r \vdash B}} \, \text{der} \\ \text{ctr/wk} \end{array}$$

For $1 \leq i \leq r$ we assume given a finite set $\mathcal{P}_i$ of proofs of $A_i$ of size $p_i + 1 = |\mathcal{P}_i|$ and a set of proofs $\mathcal{Q}$ of $B$ satisfying the hypotheses of Lemma 4.6. We further assume given an enumeration of these proofs via bijections

$$\{0, 1, \ldots, p_i\} \overset{\cong}{\longrightarrow} \mathcal{P}_i \tag{60}$$

for each $1 \leq i \leq r$ and

$$\{0, 1, \ldots, |\mathcal{Q}| - 1\} \overset{\cong}{\longrightarrow} \mathcal{Q} . \tag{61}$$

Even if $A_i = A_j$ for $i \neq j$ and $\mathcal{P}_i = \mathcal{P}_j$ as sets, we still allow the enumerations of these sets to be different for $i$ and $j$. By abuse of notation we treat these enumerations as equalities, and refer to an element of $\mathcal{P}_i$ by its corresponding integer $0 \leq j \leq p_i$. We further assume these enumerations are chosen so that $\psi(0, \ldots, 0) = 0$. Following [14] we write

$$\Big( \sum_{j=0}^{p_i} x_j^i \cdot j \Big)_{i=1}^r \in \prod_{i=1}^r \Delta \mathcal{P}_i$$

for a generic input to the map $\Delta\psi$ of Proposition 4.7. Note that $\Delta\mathcal{P}_i \subseteq \mathbb{R}^{p_i}$ is a manifold with corners, where we give the ambient space the coordinates $\{x_j^i\}_{j=0}^{p_i}$. The distribution in $\Delta\mathcal{P}_i$ concentrated at 0 we simply denote by $\mathbf{0}$, that is,

$$\mathbf{0} = (1, 0, \ldots, 0) \in \Delta\mathcal{P}_i \,.$$

We write $\mathbf{0}$ for the sequence $(\mathbf{0})_{i=1}^r$ and $\mathbf{0} = (1, 0) \in \Delta\mathcal{Q}$. We are interested in

$$T_{\mathbf{0}}(\Delta\psi) : T_{\mathbf{0}} \Big( \prod_{i=1}^r \Delta\mathcal{P}_i \Big) \longrightarrow T_{\mathbf{0}}(\Delta\mathcal{Q})$$

and higher-order derivatives.

The linear part $\pi$ takes $n_i$ inputs of type $A_i$ for $1 \leq i \leq r$. These correspond to $n_i$ different *computation paths* in which different copies of the $i$th input are used differently in the algorithm represented by the proof.[8] On each of those paths we have $p_i$ distinct *types* of error (the elements of $\mathcal{P}_i$ other than 0). An error syndrome, as defined below, assigns to each computation path either a 0 (no error) or a particular type of error. In what follows $|S|$ denotes the cardinality of a finite set $S$.

**Definition 4.9.** An *error syndrome* $\gamma$ is a sequence of functions $\gamma_1, \ldots, \gamma_r$ where

$$\gamma_i : \{1, \ldots, n_i\} \longrightarrow \{0, \ldots, p_i\} \qquad 1 \leq i \leq r \,.$$

Hence $|\gamma_i^{-1}(j)|$ for $1 \leq i \leq r, 1 \leq j \leq p_i$ is the number of errors of type $j$ that occur in input $i$ in the syndrome. The total number of errors in the $i$th input is

$$|\gamma_i| = \Big| \big\{ 1 \leq c \leq n_i \mid \gamma_i(c) \neq 0 \big\} \Big| \,.$$

The *weight* of the error syndrome is

$$\mathrm{wt}(\gamma) = (\mathbf{s}^1, \ldots, \mathbf{s}^r) \in \prod_{i=1}^r \mathbb{N}^{p_i}$$

where

$$\mathbf{s}^i = (s_1^i, \ldots, s_{p_i}^i) \,, \qquad s_j^i = |\gamma_i^{-1}(j)| \,.$$

$$(x_0^1)^2(x_1^1)^3(x_2^1)^4(x_0^2)^2(x_1^2)^2(x_2^2)^5(x_0^r)^1(x_1^r)^6$$
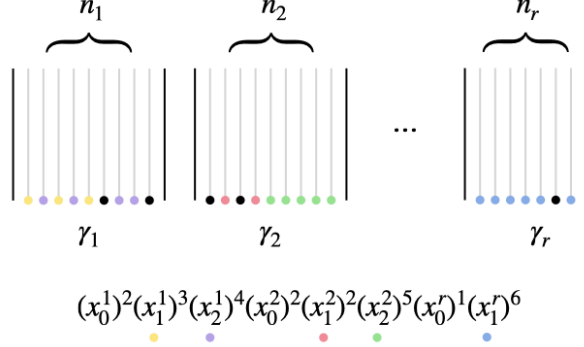
Figure 3: The plain proof $\psi$ has $r$ inputs, and $n_i$ computation paths leading from the $i$th input. In this depiction of an error syndrome $\gamma$, integers $1 \le j \le p_i$ are represented as colours with a different set of colours (proofs) for each $i$. Here for example $\gamma_1$ is the function corresponding to the integer sequence $1, 2, 1, 2, 1, 0, 2, 2, 0$. The corresponding factor in the monomial associated to $\gamma$ in $f_\psi^\tau$ is $(x_0^1)^2(x_1^1)^3(x_2^1)^4$ with $x_0^i$ factors being associated to "no error" (the integer 0). Note that different orderings of the integer sequence represent distinct error syndromes but contribute the same monomial; this is the origin of coefficients in $f_\psi^\tau$ other than 1.

**Definition 4.10.** Let $\gamma$ be an error syndrome. Then we define

$$\mathrm{eval}(\pi, \gamma) = \pi\Big(\gamma_1(1), \ldots, \gamma_1(n_1), \gamma_2(1), \ldots, \gamma_2(n_2), \ldots \gamma_r(1), \ldots, \gamma_r(n_r)\Big). \qquad (62)$$

**Theorem 4.11.** Let $\mathbf{k}^i \in \mathbb{N}^{p_i}$ for $1 \le i \le r$ and set $\mathbf{k} = (\mathbf{k}^1, \ldots, \mathbf{k}^r)$. Then

$$[\![\psi]\!]\Big(\bigotimes_{i=1}^{r} \Big| \big([\![1]\!] - [\![0]\!]\big)^{\otimes k_1^i}, \ldots, \big([\![p_i]\!] - [\![0]\!]\big)^{\otimes k_{p_i}^i}\Big\rangle_{[\![0]\!]}\Big) = \sum_{\tau \in \mathcal{Q}} \sum_{\mathbf{0} \le \mathbf{s} \le \mathbf{k}} S_\mathbf{s}^\mathbf{k} A_\tau^\mathbf{s} [\![\tau]\!] \qquad (63)$$

where $A_\tau^\mathbf{s}$ is a non-negative integer defined by

$$A_\tau^\mathbf{s} = \Big|\big\{\gamma \,\big|\, \mathrm{wt}(\gamma) = \mathbf{s} \text{ and } \mathrm{eval}(\pi, \gamma) = \tau\big\}\Big| \qquad (64)$$

the count of the *number of weight* $\mathbf{s}$ *error syndromes that produce* $\tau$. Here $\mathbf{s} \le \mathbf{k}$ means $s_j^i \le k_j^i$ for all $1 \le i \le r$, $1 \le j \le p_i$ and

$$S_\mathbf{s}^\mathbf{k} = \prod_{i=1}^{r} \Big\{\delta_{n_i \ge |\mathbf{k}^i|}(-1)^{|\mathbf{k}^i|-|\mathbf{s}^i|} \frac{(n_i - |\mathbf{s}^i|)!}{(n_i - |\mathbf{k}^i|)!} \prod_{j=1}^{p_i} \frac{k_j^i!}{(k_j^i - s_j^i)!}\Big\} \qquad (65)$$

where $|\mathbf{s}^i| = \sum_j s_j^i$, $|\mathbf{k}^i| = \sum_j k_j^i$.

---

[8]This will become more explicit in the special case of the encoding of the pseudo-UTM step function below, but the cut-elimination algorithm of of [16, Proposition 3.6] provides a general notion of computation path when $\psi$ is the cut of multiple component-wise plain proofs.

*Proof.* In a neighbourhood of $\mathbf{0} \in \Delta\mathcal{P}_i$ we can parametrise the probability simplex via a smooth immersion of manifolds with corners

$$\iota_i : [0, \varepsilon)^{p_i} \longrightarrow \Delta\mathcal{P}_i \tag{66}$$

$$(w_1^i, \ldots, w_{p_i}^i) \mapsto (1 - \sum_{j=1}^{p_i} w_j^i, w_1^i, \ldots, w_{p_i}^i) \tag{67}$$

with

$$T_{\mathbf{0}}(\iota)\Big(\frac{\partial}{\partial w_j^i}\Big) = \frac{\partial}{\partial x_j^i} - \frac{\partial}{\partial x_0^i} \in T_{\mathbf{0}}(\Delta\mathcal{P}_i)$$

for $1 \le j \le p_i$. The composite

$$\prod_{i=1}^{r}[0, \varepsilon)^{p_i} \xrightarrow{\ \prod_i \iota_i\ } \prod_{i=1}^{r} \Delta\mathcal{P}_i \xrightarrow{\ \Delta\psi\ } \Delta\mathcal{Q}$$

has derivative, writing $\iota = \prod_i \iota_i$ and letting $\mathbf{k}^i \in \mathbb{N}^{p_i}$

$$\prod_{i=1}^{r} \frac{\partial^{|\mathbf{k}^i|}}{\partial(w_1^i)^{k_1^i} \cdots \partial(w_{p_i}^i)^{k_{p_i}^i}} (\Delta\psi \circ \iota)\Big|_{w=\mathbf{0}} = \sum_{\tau \in \mathcal{Q}} \prod_{i=1}^{r} \frac{\partial^{|\mathbf{k}^i|}}{\partial(w_1^i)^{k_1^i} \cdots \partial(w_{p_i}^i)^{k_{p_i}^i}} f_\psi^\tau\Big|_{w=\mathbf{0}} \cdot \tau \tag{68}$$

Abbreviating $x_0^1 = 1, x_1^1 = 0, \ldots, x_0^2 = 1, \ldots$ to $\mathbf{x}_0 = 1, \mathbf{x}_{\ne 0} = 0$ we can write the coefficient of $\tau$ in this expression as

$$\prod_{i=1}^{r}\prod_{j=1}^{p_i} \Big[\frac{\partial}{\partial x_j^i} - \frac{\partial}{\partial x_0^i}\Big]^{k_j^i} f_\psi^\tau\Big|_{\mathbf{x}_0=1, \mathbf{x}_{\ne 0}=0} \tag{69}$$

$$= \prod_{i=1}^{r}\prod_{j=1}^{p_i}\sum_{s=0}^{k_j^i} \binom{k_j^i}{s}(-1)^{k_j^i - s}\Big[\frac{\partial}{\partial x_j^i}\Big]^s \Big[\frac{\partial}{\partial x_0^i}\Big]^{k_j^i - s} f_\psi^\tau\Big|_{\mathbf{x}_0=1, \mathbf{x}_{\ne 0}=0} \tag{70}$$

Hence by [14, Corollary 4.5], (68) is equal to the left hand side of (63), noting that the $|-\rangle$ symbols are multi-linear in their entries. On the other hand, we can by [14, Proposition 4.3] compute (68) by expanding $f_\psi^\tau$:

$$\sum_\gamma \delta_{\tau=\mathrm{eval}(\pi,\gamma)}\Big\{ \prod_{i=1}^{r}\prod_{j=1}^{p_i} \frac{\partial^{k_j^i}}{\partial(w_j^i)^{k_j^i}}\Big\} \Big(x_{\gamma_1(1)}^1 \cdots x_{\gamma_1(n_1)}^1 \cdots x_{\gamma_r(1)}^r \cdots x_{\gamma_r(n_r)}^r\Big)\Big|_{w=\mathbf{0}}$$

$$= \sum_\gamma \delta_{\tau=\mathrm{eval}(\pi,\gamma)}\Big\{ \prod_{i=1}^{r}\prod_{j=1}^{p_i} \frac{\partial^{k_j^i}}{\partial(w_j^i)^{k_j^i}}\Big\} \Big(\prod_{i=1}^{r}(1 - \sum_{j=1}^{p_i} w_j^i)^{n_i - |\gamma_i|}(w_1^i)^{|\gamma_i^{-1}(1)|} \cdots (w_{p_i}^i)^{|\gamma_i^{-1}(p_i)|}\Big)\Big|_{w=\mathbf{0}}$$

$$= \sum_\gamma \delta_{\tau=\mathrm{eval}(\pi,\gamma)} \prod_{i=1}^{r}\Big\{ \prod_{j=1}^{p_i} \frac{\partial^{k_j^i}}{\partial(w_j^i)^{k_j^i}} \Big((1 - \sum_{j=1}^{p_i} w_j^i)^{n_i - |\gamma_i|}(w_1^i)^{|\gamma_i^{-1}(1)|} \cdots (w_{p_i}^i)^{|\gamma_i^{-1}(p_i)|}\Big)\Big\}\Big|_{w=\mathbf{0}}.$$

$$\tag{71}$$

Next we apply Lemma 4.12 below to each $i$ separately. We find that (71) is equal to a sum over $\gamma$ of $\delta_{\tau=\mathrm{eval}(\pi,\gamma)}$ times the product from $i = 1$ to $r$ of the delta functions

$$\delta_{|\mathbf{k}^i|\leq n_i}\delta_{|\gamma_i^{-1}(1)|\leq k_1^i}\cdots\delta_{|\gamma_i^{-1}(p_i)|\leq k_{p_i}^i}$$

multiplied by the coefficient

$$(-1)^{|\mathbf{k}^i|-|\gamma_i|}\frac{(n_i-|\gamma_i|)!}{(n_i-|\mathbf{k}^i|)!}\prod_{j=1}^{p}\frac{k_j^i!}{(k_j^i-|\gamma_i^{-1}(j)|)!}\ .$$

Hence the coefficient of $\tau$ in (68) is

$$\sum_{\gamma}\delta_{\tau=\mathrm{eval}(\pi,\gamma)}\prod_{i=1}^{r}\left\{\delta_{n_i\geq|\mathbf{k}^i|}(-1)^{|\mathbf{k}^i|-|\gamma_i|}\frac{(n_i-|\gamma_i|)!}{(n_i-|\mathbf{k}^i|)!}\prod_{j=1}^{p_i}\frac{\delta_{k_j^i\geq|\gamma_i^{-1}(j)|}k_j^i!}{(k_j^i-|\gamma_i^{-1}(j)|)!}\right\}.$$

The combinatorial factor depends only on the weight of $\gamma$ so we can write (68) as

$$\prod_{i=1}^{r}\frac{\partial^{|\mathbf{k}^i|}}{\partial(w_1^i)^{k_1^i}\cdots\partial(w_{p_i}^i)^{k_{p_i}^i}}(\Delta\psi\circ\iota)\Big|_{w=0} = \sum_{\tau}\sum_{\gamma}\delta_{\tau=\mathrm{eval}(\pi,\gamma)}\delta_{\mathbf{k}\geq\mathrm{wt}(\gamma)}S_{\mathrm{wt}(\gamma)}^{\mathbf{k}}\tau$$

$$= \sum_{\tau}\sum_{\mathbf{s}\leq\mathbf{k}}S_{\mathbf{s}}^{\mathbf{k}}A_{\tau}^{\mathbf{s}}\tau$$

using the notation of (65), as claimed. □

**Lemma 4.12.** Given variables $y_1,\ldots,y_p$ and integers $b,c_1,\ldots,c_p\geq 0$ we have

$$\frac{\partial^{a_1}}{\partial y_1^{a_1}}\cdots\frac{\partial^{a_p}}{\partial y_p^{a_p}}\left\{\left(1-\sum_{j=1}^{p}y_j\right)^b y_1^{c_1}\cdots y_p^{c_p}\right\}\Big|_{\mathbf{y}=0}$$

$$= \delta_{|\mathbf{a}|-|\mathbf{c}|\leq b}\delta_{c_1\leq a_1}\cdots\delta_{c_p\leq a_p}(-1)^{|\mathbf{a}|-|\mathbf{c}|}\frac{b!}{(b-(|\mathbf{a}|-|\mathbf{c}|))!}\prod_{j=1}^{p}\frac{a_j!}{(a_j-c_j)!}$$

where $|\mathbf{a}|=\sum_{j=1}^{p}a_j$, $|\mathbf{c}|=\sum_{j=1}^{p}c_j$.

*Proof.* The left hand side is clearly zero unless $c_j\leq a_j$ for all $j$, in which case we get for each $1\leq j\leq p$ a factor of $\binom{a_j}{c_j}$ from all the ways of assigning $c_j$ copies of $\frac{\partial}{\partial y_j}$ to $y_j^{c_j}$ and then another factor of $c_j!$ from the $c_j$ derivatives of $y_j^{c_j}$ yielding the factor $\frac{a_j!}{(a_j-c_j)!}$. Then

$$\frac{\partial^{a_1-c_1}}{\partial y_1^{a_1-c_1}}\cdots\frac{\partial^{a_p-c_p}}{\partial y_p^{a_p-c_p}}\left(1-\sum_{j=1}^{p}y_j\right)^b\Big|_{w=0}$$

contributes the remaining factors. □

33

**Remark 4.13.** The sum in (63) includes $\mathbf{s} = \mathbf{0} = (0, \ldots, 0)$. However there is only one error syndrome $\gamma$ of weight $\mathbf{0}$, the constant function assigning 0 to every computation path. Moreover by hypothesis if $\gamma = \mathbf{0}$ denotes this syndrome then

$$\mathrm{eval}(\pi, \gamma) = \psi(\mathbf{0}) = 0$$

that is, $A_\tau^{\mathbf{0}} = \delta_{\tau = 0}$. We note that when $\mathbf{s} = 0$, the combinatorial factor simplifies

$$S_{\mathbf{0}}^{\mathbf{k}} = \prod_{i=1}^{r} \left\{ \delta_{n_i \geq |\mathbf{k}^i|} (-1)^{|\mathbf{k}^i|} \frac{n_i!}{(n_i - |\mathbf{k}^i|)!} \right\} .$$

**Remark 4.14.** Note that provided $|\mathbf{k}^i| \leq n_i$ for $1 \leq i \leq r$

$$S_{\mathbf{k}}^{\mathbf{k}} = \prod_{i=1}^{r} \prod_{j=1}^{p_i} k_j^i! .$$

# 5 Derivatives of a Universal Turing Machine

Recall that to define our probabilistic model $p(y|x, w)$ of final states $y$ of our UTM simulating a noisy Turing machine $w$ for $t$ steps on input $x$, we have in (11) of Section 2.1 assumed that we were given a smooth relaxation of the function

$$\Sigma^* \times W^{\mathtt{code}} \xrightarrow{\ \mathrm{step}^t\ } Q \ . \tag{72}$$

This is constructed as follows: first we encode $\Sigma^*$, $W^{\mathtt{code}}$, $Q$ as types and their elements as proofs, then we define a plain proof $\psi$ which computes $\mathrm{step}^t$ and then from this we obtain a denotation $\Delta\psi$ which by Proposition 4.7 makes the required diagram (11) commute. This $\Delta\psi$ is our smooth relaxation $\Delta\mathrm{step}^t$. Our model $p(y|x, w)$ has already been defined given this smooth relaxation (see (20) and Section 2.2) and the derivatives of $H$ are related to derivatives of $\Delta\psi$.

In this section we first define $\mathcal{U}$ (Section 5.1), then we define $\psi$ (Section 5.2) and then we specialise the discussion earlier of error syndromes for plain proofs to abstractly describe error syndromes for $\psi$ (Section 5.3).

## 5.1 Staged Pseudo-UTM

Recall that in a Turing machine with multiple tapes, at each timestep the heads for each tape read the current symbol and, as a function of that sequence of symbols and the current state, the machine transitions to a new state, writes to all of the tapes simultaneously and moves each head [3]. We assume our UTM has a description tape (which contains the specification of the TM to simulate), a staging tape (which is used to avoid overloading the states of the UTM with the description tape information), a state tape (which contains

the current state of the simulated machine) and a working tape (which is the state of the tape of the simulated machine).

An example of a pseudo-UTM $\mathcal{U}$ with these features is presented in this section. Note that this is not a universal machine because we assume the symbols and states of the simulated machine can be encoded on single tape squares. However, for the purposes of comparing any finite set of Turing machines this is sufficient (see Appendix A.7).

We now introduce the staged pseudo-UTM $\mathcal{U}$ following [17].[9] Simulating a Turing machine $M$ with tape alphabet $\Sigma_M$ and set of states $Q_M$ on a UTM generally requires the specification of an encoding of $\Sigma_M$ and $Q_M$ into strings in the tape alphabet $\Sigma_{\mathrm{UTM}}$ of the UTM. From the point of view of this paper it is not clear that this additional complexity is interesting, and it is a significant obstacle to doing calculations by hand. As such given $\Sigma, Q$ we consider a *staged pseudo-UTM* whose alphabet is

$$\Sigma_{\mathrm{UTM}} = \Sigma \cup Q \cup \{L, R, S\} \cup \{X, \square\}$$

where the union is disjoint and $\square$ is the blank symbol (which is distinct from the blank symbol of the simulated machines). We emphasise that there is no *theoretical* obstacle to considering a staged UTM of the same design but which makes use of encoded alphabets and states, or even to using a general UTM; we only restrict to the staged pseudo-UTM to make it tractable to reason about the resulting geometry.

Such a machine is capable of simulating any machine with tape alphabet $\Sigma$ and set of states $Q$ but cannot simulate arbitrary machines and is not a UTM in the standard sense. The adjective *staged* refers to the design of the UTM. The set of states is

$$\begin{aligned}
Q_{\mathrm{UTM}} = \{\, &\mathrm{compSymbol, compState, copySymbol, copyState, copyDir,} \\
&\neg\mathrm{compState}, \neg\mathrm{copySymbol}, \neg\mathrm{copyState}, \neg\mathrm{copyDir}, \\
&\mathrm{updateSymbol, updateState, updateDir, resetDescr}\,\}.
\end{aligned}$$

The UTM has four tapes numbered from 0 to 3, which we refer to as the *description tape*, the *staging tape*, the *state tape* and the *working tape* respectively. We write

$$n = |\Sigma|, \quad m = |Q|, \quad N = nm. \tag{73}$$

Then initially the description tape contains a string of the form

$$X\sigma_1 q_1 \sigma_1' q_1' d_1 \sigma_1 q_1 \sigma_1' q_1' d_1 \ldots \sigma_N q_N \sigma_N' q_N' d_N X,$$

corresponding to the tuples which define $M$, with the tape head initially on $\sigma_1$. The staging tape is initially a string $XXX$ with the tape head over the second $X$. The state

[9]We make the following amendment: we introduce a new symbol $e$, let the symbols $a, c, d, e$ stand for generic symbols which are not $X$, and let $b$ stand for a generic symbol (which may be $X$). In [17], there is only $a, b, c, d$ and all of these are required to not be $X$. Other than this, our presentation is identical to [17].

tape has a single square containing some distribution in $\Delta Q$, corresponding to the initial state of the simulated machine $M$, with the tape head over that square. Each square on the working tape is some distribution in $\Delta \Sigma$ with only finitely many distributions different from $\square$. The UTM is initialized in state compSymbol.

The specification of $\mathcal{U}$ is given in Figure 4. It consists of two phases; the *scan phase* (middle and right path), and the *update phase* (left path). During the scan phase, the description tape is scanned from left to right, and the first two squares of each tuple are compared to the contents of the working tape and state tape respectively. If both agree, then the last three symbols of the tuple are written to the staging tape (middle path), otherwise the tuple is ignored (right path). Once the $X$ at the end of the description tape is reached, the UTM begins the update phase, wherein the three symbols on the staging tape are then used to print the new symbol on the working tape, to update the simulated state on the state tape, and to move the working tape head in the appropriate direction. The tape head on the description tape is then reset to the initial $X$.

We define the *period* of the UTM to be the smallest nonzero time interval taken for the tape head on the description tape to return to the initial $X$, and the machine to reenter the state compSymbol. If the number of tuples on the description tape is $N$, then the period of the UTM is $P = 10N + 6$. Moreover, other than the working tape, the position of the tape heads are $P$-periodic.

## 5.2 Encoding of the Step Function

We encode the function (72) as a plain proof which can be described schematically as

$$\psi : \big\{\text{type encoding } \Sigma^*\big\}, \big\{\text{type encoding } W^{\texttt{code}}\big\} \vdash \big\{\text{type encoding } Q\big\}. \qquad (74)$$

This is an encoding in the sense that cutting $\psi$ against a proof encoding a string $x \in \Sigma^*$ on the work tape and a proof encoding $w \in W^{\texttt{code}}$ yields a proof equivalent under cut-elimination to $\text{step}^t(x, w)$, the state after $\mathcal{U}$ has simulated $w$ on input $x$ for $t$ steps.

In Appendix C we define a component-wise plain proof $_h\mathcal{U}\underline{\text{relstep}}^t$ which encodes the UTM $\mathcal{U}$ running for $t$ cycles, that is, simulating $t$ steps of a Turing machine (see Definition C.5). The component of this component-wise plain proof that produces the state of the simulated machine after $t$ steps has type

$$\mathbf{dscr} \otimes \mathbf{stg} \otimes !_m\mathbf{bool} \otimes !_n\mathbf{bool}^{\otimes 2h+1} \otimes !_{13}\mathbf{bool} \vdash {}_m\mathbf{bool}. \qquad (75)$$

The notation is defined in full in the appendix, but in summary:

- Proofs of type $\mathbf{dscr}$ encode configurations of the description tape, where

$$\mathbf{dscr} = \bigotimes_{i=1}^{N} \Big( \underbrace{!_n\mathbf{bool}}_{\sigma_i'} \otimes \underbrace{!_m\mathbf{bool}}_{q_i'} \otimes \underbrace{!_3\mathbf{bool}}_{d_i} \Big). \qquad (76)$$

- Proofs of $\mathbf{stg}$ encode configurations of the staging tape.
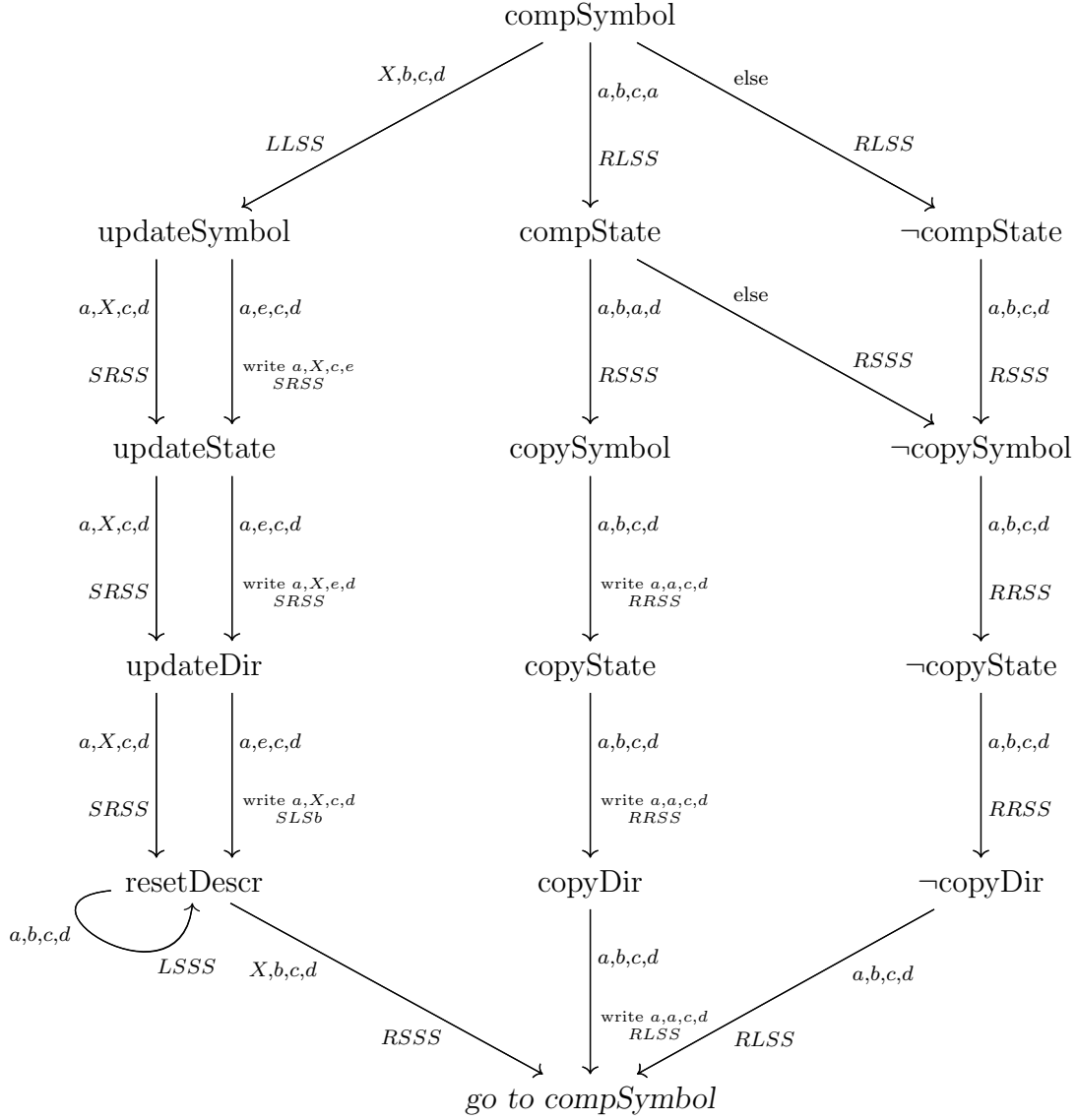
Figure 4: **The staged pseudo-UTM $\mathcal{U}$.** The nodes, except for *go to compSymbol*, are states of $\mathcal{U}$ and an arrow $q \to q'$ has the following interpretation: if the UTM is in state $q$ and sees the tape symbols (on the four tapes) as indicated by the source of the arrow, then the UTM transitions to state $q'$, writes the indicated symbols (or if there is no write instruction, simply rewrites the same symbols back onto the tapes), and performs the indicated movements of each of the tape heads with $R, S, L$ standing for Right, Stay, Left. The symbols $a, c, d, e$ stand for generic symbols which are not $X$, and $b$ stands for a generic symbol (which may be $X$).

- Proofs of $!_m\mathbf{bool}$ encode initial states of the simulated machine (i.e. elements of $Q$ which has size $m$).

- Proofs of $!_n\mathbf{bool}^{\otimes 2h+1}$ encode configurations of the work tape (that is, $2h+1$ elements of $\Sigma$ where we note that the work tape is encoded relative to the head, so this means there are $h$ entries to the right of the head and $h$ entries to the left).

- Proofs of $!_{13}\mathbf{bool}$ encode the state of the UTM itself (i.e. an element of $Q_{\mathrm{UTM}}$).

**Definition 5.1.** We define $\psi$ to be the plain proof obtained by cutting the component of $_h\mathcal{U}\mathrm{relstep}^t$ with type (75) against the following inputs

$$\underline{XXX} : \mathbf{stg}, \quad \underline{\mathrm{init}} : !_m\mathbf{bool}, \quad \underline{\mathrm{compSymbol}} : !_{13}\mathbf{bool}$$

which respectively encode putting $XXX$ on the staging tape, some initialisation state on the state tape and starting the UTM in the initial state compSymbol. Here the underlines refer to proofs encoding a given piece of data. This proof has type

$$\psi : \underbrace{\mathbf{dscr}}_{w} \otimes \underbrace{!_n\mathbf{bool}^{\otimes 2h+1}}_{x} \vdash {}_m\mathbf{bool}$$

realising the schematic in (74).[10]

We work with fixed sets of proofs (see [16, Appendix B] for definitions):

$$\mathcal{P}_i^{\mathrm{dscr}} = \begin{cases} \{[\![\underline{j}]\!]\}_{j=0,\dots,n-1} \subseteq [\![_n\mathbf{bool}]\!], & i = 1 \bmod 3, \\ \{[\![\underline{j}]\!]\}_{j=0,\dots,m-1} \subseteq [\![_m\mathbf{bool}]\!], & i = 2 \bmod 3, \\ \{[\![\underline{j}]\!]\}_{j=0,1,2} \subseteq [\![_3\mathbf{bool}]\!], & i = 0 \bmod 3. \end{cases} \tag{77}$$

**Remark 5.2.** These proofs are used to encode elements of $\Sigma, Q$ and directions to move. There is a subtlety: while all the squares of the sample "type" ($\sigma', q'$ or $d$) are encoded by the same formula and have the set of proofs, later it will be important that *different squares on the description tape use different enumerations* in the sense of (60). The enumerations are arbitrary, except that we always choose 0 to label the symbol, state or direction which appears in this position in the code $[M]$ of the chosen Turing machine $M$.

Since $\psi$ is plain [16, Lemma 3.4] we may pass to probability distributions (Proposition 4.7) to obtain the function

$$\Delta\psi : \Sigma^* \times \prod_{i=1}^{N} \left( \Delta\Sigma \times \Delta Q \times \Delta\{\mathrm{L}, \mathrm{S}, \mathrm{R}\} \right) \longrightarrow \Delta Q. \tag{78}$$

Since we do not allow uncertainty in the input $x$ we restrict the function to $\Sigma^* \subseteq \Delta\Sigma^*$.

**Definition 5.3.** The smooth relaxation $\Delta\mathrm{step}^t$ in (11) is $\Delta\psi$.

---

[10]On the left hand side of the turnstile tensors may be exchanged for commas, and freely permuted.

## 5.3  Error Syndromes for $\mathcal{U}$

Error syndromes for $\mathcal{U}$ are determined by the linear part $\pi$ of the proof $\psi$ encoding $\mathrm{step}^t$ defined in the previous section. Abstractly we know $\psi$ is a plain proof and thus there are uniquely determined integers $c, d$ (with various sub- and superscripts) which are the degrees of each input with

$$\pi : \bigotimes_{i=1}^{N} \left( c_i^{\mathrm{sym}} {}_n\mathbf{bool} \otimes c_i^{\mathrm{state}} {}_m\mathbf{bool} \otimes c_i^{\mathrm{dir}} {}_3\mathbf{bool} \right) \otimes \bigotimes_{i=-h}^{h} b_{i\ n}\mathbf{bool} \vdash {}_m\mathbf{bool} . \qquad (79)$$

However it is not necessarily trivial to compute these integers or enumerate the computation paths that they count. Later in Section 6.2 we do this explicitly when $t = 2$. In any case, we can now formally define *error syndromes* for $\mathcal{U}$ to be error syndromes in the sense of Definition 4.9 associated to the pair consisting of the plain proof $\psi$ of the previous section with linear part $\pi$.

**Remark 5.4.** Because of the way that the UTM is designed, the number of computation paths starting at a square on the description tape depends on the *type* of the square but not its position on the tape. That is, there exist integers $c^{\mathrm{sym}}, c^{\mathrm{state}}, c^{\mathrm{dir}}$ such that $c_i^{\mathrm{sym}} = c^{\mathrm{sym}}$, $c_i^{\mathrm{state}} = c^{\mathrm{state}}$ and $c_i^{\mathrm{dir}} = c^{\mathrm{dir}}$ for all $1 \leq i \leq N$.

It is important to keep in mind that since we study geometry *locally* at a code $[M]$ error syndromes are always considered relative to a particular Turing machine. To make this explicit, let us now:

**Fix a Turing machine** $M$ so that "errors" are variations away from $[M]$.

That is, error syndromes describe variations away from this code as a string on the description tape of the UTM (see Remark 5.2). We also do not consider derivatives with respect to the tape squares encoding the input $x$ and thus do not consider them in the definition of error syndromes. So to be explicit:

**Definition 5.5.** An *error syndrome* $\gamma$ for $\mathcal{U}$ is a sequence of functions $\gamma_1, \ldots, \gamma_{3N}$ where

$$\gamma_i : \{1, \ldots, n_i\} \longrightarrow \{0, \ldots, p_i\} \qquad 1 \leq i \leq 3N .$$

and

$$n_i = \begin{cases} c^{\mathrm{sym}} & i = 1 \bmod 3 \\ c^{\mathrm{state}} & i = 2 \bmod 3 \\ c^{\mathrm{dir}} & i = 3 \bmod 3 \end{cases} , \qquad p_i = \begin{cases} n - 1 & i = 1 \bmod 3 \\ m - 1 & i = 2 \bmod 3 \\ 2 & i = 3 \bmod 3 \end{cases}$$

where $\gamma_i(j) = 0$ is always interpreted as *no error* that is, whatever symbol, state or direction is in this position of the description of some fixed background machine $[M]$.

The tuples $\mathbf{s}, \mathbf{k}$ of Section 4.3.2 belong to

$$\prod_{i=1}^{r} \mathbb{N}^{p_i} = \prod_{a=1}^{N} \left[ \mathbb{N}^{|\Sigma|-1} \times \mathbb{N}^{|Q|-1} \times \mathbb{N}^2 \right]. \tag{80}$$

To make this more concrete, suppose $\mathbf{s} \in \prod_{i=1}^{r} \mathbb{N}^{p_i}$ is such a tuple and that $1 \le a \le N$ is the index of a tuple $\sigma q \sigma' q' d$ in $[M]$ and that

$$\mathbf{s} = \left( \cdots, \overbrace{\underbrace{\mathbf{u}}_{\mathbb{N}^{|\Sigma|-1}}, \underbrace{\mathbf{v}}_{\mathbb{N}^{|Q|-1}}, \underbrace{\mathbf{w}}_{\mathbb{N}^2}}^{a}, \cdots \right). \tag{81}$$

If $\mathrm{wt}(\gamma) = \mathbf{s}$ for some error syndrome $\gamma$, then out of all the computation paths leading from the description square containing $\sigma'$ in $[M]$ the count of those assigned errors of each type given by elements of $\Sigma \setminus \{\sigma'\}$ is given by $\mathbf{u}$, and similarly $\mathbf{v}, \mathbf{w}$ count assignments of errors of various types to computation paths out of $q', d$ respectively.

**Definition 5.6.** Given an input $x$ and error syndrome $\gamma$ we define

$$\mathcal{U}(x, \gamma) = \pi(\gamma, x) \tag{82}$$

where as in (62) the right hand side involves placing the outputs of all the $\gamma_i$ as inputs to $\pi$ in the correct positions. In particular if $\gamma = \mathbf{0}$ is the constant function then

$$\mathcal{U}(x, \mathbf{0}) = M(x).$$

**Remark 5.7.** Note that

$$\sum_{i=1}^{r} p_i = |Q||\Sigma|(|\Sigma| + |Q|) = d$$

the dimension of the parameter space $W$. Hence we may also write $\mathbf{s}, \mathbf{k} \in \mathbb{N}^d$.

With this preparation everything in Section 4.3 applies verbatim to the analysis of derivatives of the plain proof $\psi$. We denote by $A^{\mathbf{s}}(x)$ the error syndrome count from (64) with $\tau$ the alternative output to the correct output $M(x)$, that is:

**Definition 5.8.** Given $\mathbf{s} \in \prod_{i=1}^{r} \mathbb{N}^{p_i}$ and $x \in \Sigma^*$

$$A^{\mathbf{s}}(x) = \left| \left\{ \gamma \mid \mathrm{wt}(\gamma) = \mathbf{s} \text{ and } \mathcal{U}(x, \gamma) \ne M(x) \right\} \right| \tag{83}$$

denotes the count of weight $\mathbf{s}$ error syndromes $\gamma$ which, when we evaluate the UTM with $[M]$ on the description tape, deviations from the code given by $\gamma$, and $x$ on the work tape, produce the incorrect output.

**Remark 5.9.** Note that by definition, $A^{\mathbf{0}}(x) = 0$ for all $x$.

Since $\psi$ is defined by cutting $t$ component-wise plain proofs against one another, we can from the proof that the cut of component-wise plain proofs is component-wise plain [16, Proposition 3.6] derive the linear part $\pi$ of $\psi$ from the linear parts of each of these individual step functions. This gives us an explicit form of the computation paths through $\psi$ and thus the error syndromes. In the next section we present the computation paths in a special case.

# 6 Error and Geometry

In Section 2 we outlined a perspective on inductive inference over Turing machines, where we believe that a Turing machine is an explanation for our observations to the extent that the posterior concentrates around it in a continuous parameter space of "noisy" Turing machines. We sketched how our preferences for Turing machines are dominated, asymptotically in the number of samples $n$ from the generating process, by the geometry of the average log-likelihood $L$ at the code $[M]$ of our machine.

This geometry was related in Section 3 to the derivatives of the function that propagates uncertainty about Turing machine codes through our UTM to uncertainty about final states of those simulated machines. Section 4 showed that these derivatives can be understood in terms of the combinatorics of error syndromes (Theorem 4.8). If we put this all together, we can make some progress into understanding the underlying geometry of inductive inference.

As in Section 5.3, we now fix in addition to the staged pseudo-UTM $\mathcal{U}$ a particular classical solution $M$. That is, a Turing machine with $M(x) = y(x)$ for all $x \in I$.

## 6.1 Graphical Model

We consider the evolution of the random variables involved in the computation of $\mathcal{U}$ as encoded in linear logic in the previous section. This was previously discussed in [14, §6.2].

To each pair consisting of a tape square of $\mathcal{U}$ and a time step $\mu$ within the cycle there is a corresponding random variable. There is also a random variable for each time step $t$ ranging over all possible states of $\mathcal{U}$. The semantics of the encoding of the step function of $\mathcal{U}$ give the random variables at timestep $\mu + 1$ as functions of those at timestep $\mu$. This defines a directed graphical model (DGM) for one cycle of the UTM (so one simulated step) as shown in Figure 5. This has been simplified by the method in Section D.

We indicate in the left-most column the entry $[M]_i$ in the code of $M$ which is currently under the UTM head. One particular square, of $\sigma'$ type, is highlighted and denoted $\theta$. Paths in the DGM that start from $\theta$ (or other random variables on the description tape) and end at the state $q$ in some future cycle are the computation paths of $\mathcal{U}$ which are enumerated carefully in the next section. Further details are given in Appendix D.

## 6.2 Computation Paths for $\mathcal{U}$

In this section we enumerate the computation paths for the pseudo-UTM $\mathcal{U}$ from each of its $3N$ inputs (which recall represent squares on the description tape, which contains $N$ tuples) to the final state represented by the type $B$. Recall that we write description tape entries as $\sigma, q, \sigma', q', d$. For each index $1 \leq j \leq N$ we represent the corresponding tuple by $\sigma_j, q_j, \sigma'_j, q'_j, d_j$. Computation paths may be inferred from stacking two copies of Figure 5 to represent two cycles of the UTM.
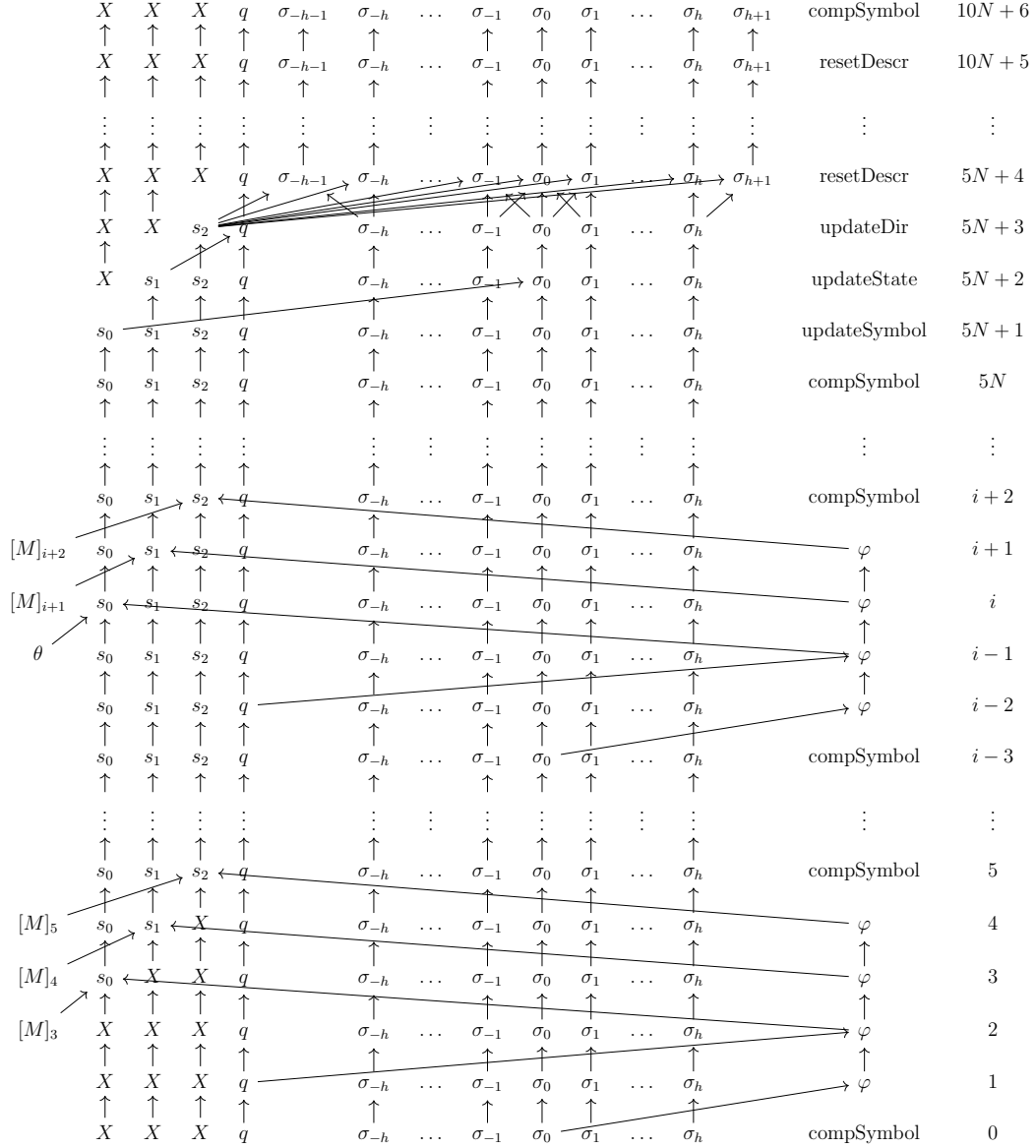
Figure 5: **Directed graphical model for a cycle of $\mathcal{U}$.** Shown is a complete cycle of the staged pseudo-UTM $\mathcal{U}$. Vertices represent random variables and arrows show the dependence relations. Columns are respectively random variables representing the squares on the description tape (column 1, counting from the left), staging tape (columns $2-4$), state tape (column 5), work tape squares, UTM state (second column from the right) and the timestep $\mu$ of the UTM within the cycle (rightmost column). Shown for reference is a random variable $\theta$ corresponding to a $\sigma'$ entry on the description tape.
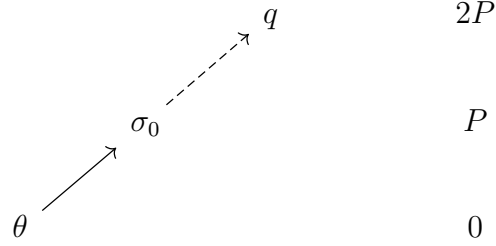
Figure 6: A collapsed version of two stacked diagrams as in Figure 5 (that is, two simulated timesteps of a TM, considering a possible error in either the symbol to write $\sigma'$ or the direction to move $d$) showing only paths that begin at $\theta$ and end at $q$. Dashed lines stand for $N$ computation paths (where $N$ is the number of tuples on the description tape) and $P$ for the number of UTM steps required to simulate a single step of $M$.
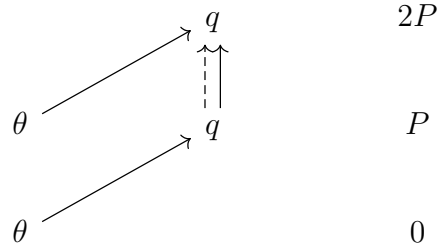


Figure 7: A collapsed version of two stacked diagrams as in Figure 5 (that is, two simulated timesteps of a TM, considering a possible error in the state to transition $q'$ to on the description tape) showing only paths that begin at $\theta$ and end at $q$.

In Figure 6 and Figure 7 we present a graphical model of the random variables involved in the forward propagation of uncertainty through the UTM from the description tape. The computation paths through the plain proof encoding $2P$ steps of the staged pseudo-UTM (so two steps of the simulated machine) which begin at a description tape entry $\theta$ and end at $q$ depend on whether $\theta$ is a $\sigma', q'$ or $d$-type square:

- $\sigma'$-**type**: there are computation paths from $\theta$ to $q$

$$\Gamma_j \qquad 1 \le j \le N \tag{84}$$

  as shown in Figure 6. This path goes through the $\sigma_0$ after one simulated timestep and then through the second entry on the staging tape to $q$. In the second cycle of the UTM, simulating the second step of the machine, the first two squares of each description tuple are read and compared to $\sigma_0, q$. One copy of $\sigma_0$ is used in each comparison, and the computation path $\Gamma_j$ represents the copy of $\sigma_0$ (which is produced using one copy of $\theta$) being used in the comparison with $\sigma_j$, the symbol on the $j$th tuple of the description tape. This comparison affects which state is written to the staging tape and eventually used to produce $q$.

- $q'$-**type**: there are computation paths from $\theta$ to $q$

$$\Lambda_j \qquad 1 \le j \le N \,, \qquad \Omega \,, \qquad \Xi \tag{85}$$

  as shown in Figure 7. The path $\Lambda_j$ goes through the $q$ after one simulated timestep and represents $\theta$ first being used to compute the state after one step, and then this output being used in the comparison with $q_j$ in order to update the staging tape. There is another computation path $\Omega$ where $\theta$ is used directly to produce the state in the second step. There is one additional computation path $\Xi$ where $\theta$ is used to compute the state after one simulated timestep, and then no match on the description tape is found, and so this state is carried through to the second simulated timestep.

- $d$-**type**: there are computation paths from $\theta$ to $q$

$$\Theta_j \qquad 1 \le j \le N \tag{86}$$

  as shown in Figure 6. As above this path goes through $\sigma_0$ and represents the copy of $\sigma_0$ (and thus $\theta$) used in the comparison with $\sigma_j$. The difference is that the original $\theta$ affects $\sigma_0$ in this case through the direction the tape head moves in the first step, rather than by writing a symbol.

## 6.3 Influence Functions

We return to the question of understanding the computational content of the influence functions (Definition 3.2) for $\mathbf{i} \in \mathbb{N}^d$ (see Remark 5.7)

$$g_{\mathbf{i}}(x) = \frac{\partial^{|\mathbf{i}|}}{\partial w_1^{i_1} \cdots \partial w_d^{i_d}} \, p\big(y \ne y(x) \mid x, w\big)\Big|_{w=[M]} \,. \tag{87}$$

44

However first we have to fix coordinates. In Section 3 the local coordinates $w_1, \ldots, w_d$ at $[M]$ in $W$ were arbitrary (Remark 3.1). However, now we have to fix the coordinates to be as specified in (67). That is, $w$ parametrises deviations in the probability simplex from $[M]$ (which is $w = \mathbf{0}$) with one coordinate for each position on the description tape and each symbol, or state or direction other than the one given in that position in $[M]$.

**Corollary 6.1.** For $\mathbf{0} \neq \mathbf{k} \in \mathbb{N}^d$

$$g_{\mathbf{k}}(x) = \sum_{\mathbf{0} \neq \mathbf{s} \leq \mathbf{k}} S_{\mathbf{s}}^{\mathbf{k}} A^{\mathbf{s}}(x) \tag{88}$$

where $S_{\mathbf{s}}^{\mathbf{k}}$ is as in (65) and $A^{\mathbf{s}}(x)$ is as in Definition 5.8.

*Proof.* Immediate from (68) in the proof of Theorem 4.11 since $\Delta \psi = \Delta \text{step}^t(x, w)$ when we take local coordinates around $[M]$. We can assume $\mathbf{0} \neq \mathbf{s}$ by Remark 5.9. $\square$

**Example 6.2.** Let $\mathbf{k} = e_k$ be the standard basis vector in $\mathbb{R}^d$ for some $1 \leq k \leq d$. Such an index corresponds to a tuple on the description tape, within that tuple one of the three squares $\sigma', q', d$ and within that square one of the possible values $z$ *other* than what is specified in $[M]$. Then $w_k = 0$ means the same value appearing in $[M]$ and $w_k = 1$ means that we replace that entry by some other symbol, state or direction.

Then we write $g_k(x)$ for $g_{\mathbf{k}}(x)$ and this function measures the infinitesimal variation in the probability of an output error caused by an infinitesimal variation in the code in this particular square, away from its value in $[M]$ towards $z$. We have (see Remark 4.14)

$$g_k(x) = S_{\mathbf{k}}^{\mathbf{k}} A^{\mathbf{k}}(x) = A^{\mathbf{k}}(x) = \left| \left\{ \gamma \mid \text{wt}(\gamma) = e_k \text{ and } \mathcal{U}(x, \gamma) \neq M(x) \right\} \right|.$$

We also denote this quantity simply by $A^k(x)$. To say that $\text{wt}(\gamma) = e_k$ is to say that $\gamma$ assigns no error to the computation paths from all other squares on the description tape, assigns to exactly one computation path from the square associated to $k$ an error, and that error is of type $z$. Note that since there are generally many computation paths, such an error syndrome is far from unique.

Combining this with Lemma 3.3 we obtain

**Theorem 6.3.** For $|\mathbf{k}| > 0$

$$\left. \frac{\partial^{|\mathbf{k}|}}{\partial w_1^{k_1} \cdots \partial w_d^{k_d}} H \right|_{w=[M]} = \sum_{\substack{\mathbf{i}, \mathbf{j} \neq 0 \\ \mathbf{i}+\mathbf{j}=\mathbf{k}}} \sum_{\substack{\mathbf{0} \neq \mathbf{s} \leq \mathbf{i} \\ \mathbf{0} \neq \mathbf{t} \leq \mathbf{j}}} C(\mathbf{i}, \mathbf{j}) S_{\mathbf{s}}^{\mathbf{i}} S_{\mathbf{t}}^{\mathbf{j}} \mathbb{E}_x \left[ A^{\mathbf{s}}(x) A^{\mathbf{t}}(x) \right] \tag{89}$$

where $C(\mathbf{i}, \mathbf{j}) = \prod_{s=1}^{d} \binom{i_s + j_s}{i_s}$, $\mathbb{E}_x[-]$ denotes expectation with respect to $q(x)$ and $\mathbf{i}, \mathbf{j} \in \mathbb{N}^d$.

*Proof.* We have by Lemma 3.3

$$\frac{\partial^{|\mathbf{k}|}}{\partial w_1^{k_1}\cdots \partial w_d^{k_d}} H\Big|_{w=[M]} = \sum_{\substack{\mathbf{i},\mathbf{j}\neq 0 \\ \mathbf{i}+\mathbf{j}=\mathbf{k}}} C(\mathbf{i},\mathbf{j})\, \mathbb{E}_x\Big[g_{\mathbf{i}}(x)g_{\mathbf{j}}(x)\Big]$$

$$= \sum_{\substack{\mathbf{i},\mathbf{j}\neq 0 \\ \mathbf{i}+\mathbf{j}=\mathbf{k}}} C(\mathbf{i},\mathbf{j})\, \mathbb{E}_x\Big[\Big(\sum_{0\neq \mathbf{s}\leq \mathbf{i}} S_{\mathbf{s}}^{\mathbf{i}} A^{\mathbf{s}}(x)\Big)\Big(\sum_{0\neq \mathbf{t}\leq \mathbf{j}} S_{\mathbf{t}}^{\mathbf{j}} A^{\mathbf{t}}(x)\Big)\Big]$$

$$= \sum_{\substack{\mathbf{i},\mathbf{j}\neq 0 \\ \mathbf{i}+\mathbf{j}=\mathbf{k}}} \sum_{\substack{0\neq \mathbf{s}\leq \mathbf{i} \\ 0\neq \mathbf{t}\leq \mathbf{j}}} C(\mathbf{i},\mathbf{j}) S_{\mathbf{s}}^{\mathbf{i}} S_{\mathbf{t}}^{\mathbf{j}}\, \mathbb{E}_x\Big[A^{\mathbf{s}}(x) A^{\mathbf{t}}(x)\Big] \qquad (90)$$

as claimed. $\qquad\qquad\square$

The only part of (89) that depends on the machine $M$ are the correlation functions $\mathbb{E}_x\big[A^{\mathbf{s}}(x)A^{\mathbf{t}}(x)\big]$. It is in this sense that the combinatorics and statistics of error syndromes determine the geometry of $H$ at $[M]$. An important special case is the Hessian of $H$:

**Corollary 6.4.** For $1 \leq i,j \leq d$ we have

$$\frac{\partial^2}{\partial w_i \partial w_j} H\Big|_{w=[M]} = 2\,\mathbb{E}_x\Big[A^i(x)A^j(x)\Big]. \qquad (91)$$

**Remark 6.5.** In the analysis of Boolean functions $f : \{0,1\}^n \longrightarrow \{0,1\}$ one often studies the *influence* of the $i$-th variable on $f$, denoted

$$\mathrm{Inf}_i(f) \;=\; \Pr_x\big[f(x) \neq f(x^{\oplus i})\big],$$

where $x^{\oplus i}$ is obtained from $x$ by flipping its $i$-th bit. That is, $\mathrm{Inf}_i(f)$ is the probability that flipping the $i$-th bit in the input flips the value of the function. A rich theory has developed around these notions, often called the *Fourier (or spectral) analysis of Boolean functions*, culminating in classical results such as the Poincaré inequality, KKL theorem, Friedgut's junta theorem, and so on [32].

The influence functions $g_i(x)$ of this paper are conceptually similar, but distinct, in that they measure the infinitesimal rate at which perturbing the $i$th bit in the code of the Turing machine $M$ changes the probability of an output error on $x$. By Example 6.2 this is a count of weight one error syndromes which induce an output error on $x$.

## 6.4 Meaning of Nondegeneracy

The germ $([M], H)$ is nondegenerate if and only if the functions $\{g_i\}_{i=1}^d$ are linearly independent (Lemma 3.4). Hence by Lemma 2.11 and Lemma B.1 the same is true of $([M], L)$. We can now understand the computational content of this condition: it says that the pattern of counts of error syndromes in the $i$th square of the description tape

which flip the output state cannot be computed from the pattern for the *other* squares of the description tape. In short, nondegeneracy means that *different parts of the code have different effects*.

More generally, the geometry of the singularity $([M], H)$ is determined by structure in the Taylor series expansion, which by Theorem 6.3 the same thing as structure in the combinatorics of error syndromes. That is, whenever we can show that the effect on the final state of errors in some bits *interact* with errors in other bits, we can expect this to show up in the combinatorics and thus potentially in the geometry of $([M], H)$. In this way, we predict a general correspondence between geometric structure of the singularity and the internal structure of the Turing machine. Several particular cases are presented in the next section.

# 7 Examples

## 7.1 Error-Correction

The concepts of fault-tolerance and error-correction have been present in theoretical computer science since its earliest foundational works [41]. Von Neumann demonstrated that a bounded level of noise or faults occurring at the level of individual elements can be systematically mitigated. His method involved structuring computations redundantly by executing each computational step multiple times independently and using majority voting. In this way he showed that local errors could be corrected with high probability.

Other frameworks for fault-tolerance have been developed, including the formal notion of fault-tolerant Turing machines [33, 36]. Such machines explicitly incorporate mechanisms for detecting and correcting faults that occur at the level of transitions, tape symbols, or internal states during computation. Under some conditions Çapuni and Gács [8, 49] have constructed general machines of this kind.[11]

**Definition 7.1.** For any integer $C \geq 0$ we say that a Turing machine $M$ can *correct errors of weight* $\leq C$ on $I \subseteq \Sigma^*$ if for any error syndrome $\gamma \in \mathbb{N}^d$ with $|\operatorname{wt}(\gamma)| \leq C$

$$\mathcal{U}(x, \gamma) = M(x) \qquad \forall x \in I. \tag{92}$$

Hidden in the notation is that $M(x)$ is the state after running the machine for a *fixed number $t$ of steps*. So informally this definition says: if we simulate $M$ on the UTM $\mathcal{U}$ for $t$ steps and, when this execution would make use of a sample from the distributions of symbols on the description tape, we make use of the outcomes provided by $\gamma$, the resulting state $\mathcal{U}(x, \gamma)$ is $M(x)$, that is, the outcome is the same as if there had been no errors (here *errors* are any time the sample we obtain is different from the entry in that

---

[11]We distinguish between noisy Turing machines and probabilistic Turing machines. While probabilistic Turing machines employ randomness deliberately as a computational resource, noisy or fault-tolerant Turing machines address a fundamentally different challenge: preserving computational correctness and universality despite the involuntary introduction of random faults.

position in $[M]$). We note again the unusual model of noisy execution of a UTM that we are employing (see Appendix A.5).

**Corollary 7.2.** If $M$ is able to correct errors of weight $\leq C$ then

$$\lambda([M], q) \leq \frac{d}{2(C+1)} \, . \tag{93}$$

*Proof.* If $\mathbf{i}, \mathbf{j} \neq 0$ and $\mathbf{s} \leq \mathbf{i}, \mathbf{t} \leq \mathbf{j}, \mathbf{i} + \mathbf{j} = \mathbf{k}$ then from $|\mathbf{s}| \geq C + 1, |\mathbf{t}| \geq C + 1$ we deduce

$$|\mathbf{k}| = |\mathbf{i}| + |\mathbf{j}| \geq |\mathbf{s}| + |\mathbf{t}| \geq 2C + 2 \, .$$

It follows from Theorem 6.3 that we have

$$\frac{\partial^{|\mathbf{k}|}}{\partial w_1^{k_1} \cdots \partial w_d^{k_d}} H \bigg|_{w=[M]} = 0$$

unless $|\mathbf{k}| \geq 2(C + 1)$ since by assumption $A^{\mathbf{s}}(x) = 0$ for all $|\mathbf{s}| \leq C$. Then the Newton polyhedron $\mathcal{P}(M)$ is the convex hull of points $\mathbf{k}$ with $|\mathbf{k}| \geq 2(C + 1)$. Since this set is itself convex, this means that

$$\mathcal{P}(M) \subseteq \{\mathbf{k} \, | \, |\mathbf{k}| \geq 2(C + 1)\}$$

and in particular $(s, s, \ldots, s) \in \mathcal{P}(M)$ implies $ds \geq 2(C + 1)$ and so $s > \frac{2(C+1)}{d}$. The distance $l(M)$ is the infimum of a subset of $[\frac{2(C+1)}{d}, \infty]$ and hence $l(M) \geq \frac{2(C+1)}{d}$. Thus $\frac{1}{l(M)} \leq \frac{d}{2(C+1)}$ and the claim now follows from Proposition 2.16. $\square$

Note that any Turing machine can correct $C = 0$ errors, in which case the upper bound of (93) is $\frac{d}{2}$, the universal bound on the learning coefficient [46, §7], [26, §4.2.3].

**Remark 7.3.** If $M$ does not make use of all the states in $Q$, it might be modified in such a way as to implement error-correction. This will on the one hand tend to *increase* the learning coefficient [14, §3] because the machine uses more tuples, but on the other hand by the Corollary the implementation of error-correction will tend to *decrease* the learning coefficient. It is an important open question whether error-correction can be made "net negative" in this sense for Turing machines.

It seems that simple majority voting schemes can be useful in this regard if we assume the error-correction itself is not subject to errors. However, as with [8] and the existing literature on similar questions, the real problem is to allow errors in the part of the description that implement error-correction. Answering this open question would have important implications for the character of programs that are given high probability by the posterior.

**Remark 7.4.** A relation between error sensitivity of neural networks in their forward pass and generalisation error has been noted by [4]. It has also been suggested that evolution in the presence of noise tends to produce error-correcting codes and modularity [27].

## 7.2 Control Flow

Suppose that $I$ can be written as a disjoint union $I = I_1 \sqcup I_2$ and the Turing machine $M$ on input $x$ first classifies whether $x$ belongs to $I_1$ or $I_2$ and then executes separate control paths for each subset. We divide the tuples (and thus the parameters $w_1, \ldots, w_d$) into three blocks, where the first and second blocks correspond to tuples in the first and second control path and the third block performs the classification of the input.

Then for $x \in I_1$ we have $g_j(x) = 0$ for any $j$ in the second block, and for $x \in I_2$ we have $g_i(x) = 0$ for any $i$ in the first block, since no error syndrome in one block can affect the output for inputs handled by the other block. Hence for $i$ in the first block and $j$ in the second we have by Corollary 6.4

$$\frac{1}{2} \frac{\partial^2}{\partial w_i \partial w_j} H \bigg|_{w=[M]} = \mathbb{E}_x \left[ g_i(x) g_j(x) \right] = 0 \,. \tag{94}$$

This means that there is a block structure in the Hessian of $H$

$$\mathrm{Hess}(H, [M]) = \begin{pmatrix} H_1 & 0 & A_{13} \\ 0 & H_2 & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \,. \tag{95}$$

This is the geometric signature of control flow in the Turing machine.

**Remark 7.5.** In general, increasing degeneracy of $H$ looks like vanishing of higher derivatives, or relations among these derivatives; in turn this by Theorem 6.3 takes the form of certain functions $f(x)$ with the property that $\mathbb{E}_x[f(x)] = 0$. That is, as *statistical patterns* on the sample space [34]. These combinatorial relations among error syndrome counts in return reflect the patterns by which parts of the Turing machine code coordinate to produce the output.

## 7.3 Nondegenerate Direction for `detectA`

We use the example of the synthesis problem `detectA` from [17] to illustrate the correspondence between geometry and structure. Recall that

$$\Sigma = \{0, 1, 2\} = \{\Box, A, B\},$$
$$Q = \{0, 1\} = \{\mathrm{reject}, \mathrm{accept}\} \,.$$

Consider the function $f : \{A, B\}^* \longrightarrow \{\mathrm{reject}, \mathrm{accept}\}$ that returns accept if the string contains at least one $A$, and reject otherwise. In Figure 8 we give two Turing machines that implement this function when the string is initially placed on the work tape with the head over the first symbol (arranged to the right on the tape). The machines are initialised in the reject state.

We consider errors which perturb the state $q'$ transitioned to by both machines, when $s = A$ and $q = \mathrm{reject}$. That is, the variation

$$(A, \mathrm{reject}, ?, \mathrm{accept} \to \mathrm{reject}, ?) \,. \tag{96}$$

| $\sigma$ | $q$ | $\sigma'$ | $q'$ | $d$ |
|---|---|---|---|---|
| □ | reject | □ | reject | Stay |
| □ | accept | □ | accept | Stay |
| $A$ | reject | $A$ | accept | Stay |
| $A$ | accept | $A$ | accept | Stay |
| $B$ | reject | $B$ | reject | Right |
| $B$ | accept | $B$ | accept | Stay |

(a) $\texttt{detectA}^{(0)}$

| $\sigma$ | $q$ | $\sigma'$ | $q'$ | $d$ |
|---|---|---|---|---|
| □ | reject | $B$ | reject | Right |
| □ | accept | □ | accept | Left |
| $A$ | reject | □ | accept | Left |
| $A$ | accept | □ | accept | Left |
| $B$ | reject | □ | reject | Right |
| $B$ | accept | □ | accept | Left |

(b) $\texttt{detectA}^{(1)}$

Figure 8: (a) A natural implementation of $\texttt{detectA}$. Each row is a tuple of the form $(\sigma, q, \sigma', q', d)$. (b) A second, distinct machine implementing the same function, implicitly given in [17, Example 3.5] and discovered there by MCMC-based program synthesis.

We let $\theta$ denote the random variable representing this $q'$-type square on the description tape (terminology of Section 6.2). Hence if we use the UTM to simulate the TMs for two steps, the computation paths from $\theta$ to the final state $q$ are

$$\Lambda_{\square,\text{reject}}, \ldots, \Lambda_{B,\text{accept}}, \Omega, \Xi$$

where we note $N = 6$ and we label computation paths by the associated $s, q$ for the tuple. Hence an error syndrome is a sequence $\gamma \in \{0, 1\}^8$ where we recall 0 is read as the symbol present in the code $[M]$ and 1 as the alternative presented in the variation (96). So for example $\gamma = (1, 0, 0, 0, 0, 0, 0, 0)$ (we write 10000000) means that we evaluate the UTM on a given input, using as our values of the entry $\theta$ the states

$$\Lambda_{\square,\text{reject}} = \text{reject}, \Lambda_{\square,\text{accept}} = \text{accept}, \ldots, \Lambda_{B,\text{accept}} = \text{accept}, \Omega = \text{accept}, \Xi = \text{accept}$$

where the fact that the first entry of $\gamma$ is 1 is what causes the first state here to be reject. What this means is that, while evaluating the UTM with $[\texttt{detectA}^{(0)}]$ on its description tape and $x$ on its work tape, in the second cycle of $P$ steps of the UTM when we go to compare $\square$, reject with $\sigma'_0, q'$ we use the state computed in the first step using the tuple

$$A, \text{reject}, A, \textcolor{red}{\text{reject}}, \text{Stay} \quad \text{rather than} \quad A, \text{reject}, A, \text{accept}, \text{Stay} .$$

In all other computation paths we use the tuple with accept, that is, without the error. For example on input $x = A$ with error syndrome 10000000, we have (writing $\sigma_0 = A, q = \text{reject}$ for the inital symbol and state, $\sigma'_0, q'$ for the symbol and state after one timestep of the simulated machine and $q''$ for the state after two timesteps)

$$q''(A) = q''\big(\sigma'_0(A, \text{reject}), q'(A, \text{reject}, \overset{\square,\text{reject}}{\theta}), \ldots, q'(A, \text{reject}, \overset{B,\text{accept}}{\theta}), \overset{7}{\theta}, \overset{8}{\theta}\big)$$
$$= q''\big(A, q'(A, \text{reject}, 1), q'(A, \text{reject}, 0), \ldots, q'(A, \text{reject}, 0), 0, 0\big)$$
$$= q''\big(A, \text{reject}, \text{accept}, \ldots, \text{accept}, 0, 0\big) = \text{accept} .$$

50

The syndrome $\gamma = 00100000$ is more interesting. This corresponds to an error occurring in the copy of $\theta$ used to compute $q'$ in the comparison of the current symbol and state with the first two squares $A$, reject of the third tuple of $\mathtt{detectA}^{(1)}$. Thus

$$q''(A) = q''\big(\sigma'_0(A, \text{reject}), \ldots, q'(A, \text{reject}, \overset{A,\text{reject}}{\theta}), q'(A, \text{reject}, \overset{A,\text{accept}}{\theta}), \ldots, \overset{7}{\theta}, \overset{8}{\theta}\big)$$
$$= q''\big(A, \ldots, q'(A, \text{reject}, 1), q'(A, \text{reject}, 0), \ldots, 0, 0\big)$$
$$= q''\big(A, \ldots, \text{reject}, \text{accept}, \ldots, 0, 0\big) = \text{accept} .$$

While scanning the description tape the UTM tries to compare $A$, reject with the current symbol and state, gets a match on the first and then (mistakenly) a match on the second (due to the error in computing $q'$, which should be accept after the first step, but due to the error in $\theta$ is instead reject). Hence the UTM writes accept to the staging tape. It then moves on to the comparison for the tuple beginning $A$, accept and *receives a second match*, writing accept to the staging tape again.

We see that none of the syndromes of weight one can cause an error (that is, flip the output from accept to reject) on the input $x = A$. However with syndrome $\gamma = 00000010$ and $x = BA$ there is an output error, as

$$q''(BA) = q''\big(\sigma'_0(B, \text{reject}), q'(B, \text{reject}, \overset{\square,\text{reject}}{\theta}), \ldots, q'_0(B, \text{reject}, \overset{B,\text{accept}}{\theta}), \overset{7}{\theta}, \overset{8}{\theta}\big)$$
$$= q''\big(A, \text{reject}, \ldots, \text{reject}, 1, 0\big) = \textcolor{red}{\text{reject}} .$$

Continuing in this way we fill out the tables in Figure 9.

As we have explained in Section 3, to second order the geometry of the singularity germ $([\mathtt{detectA}^{(j)}], L)$ for $j \in \{0, 1\}$ is closely related to the geometry of the configuration of the functions $\{g_i^{(j)}\}_{i=1}^d$ in the vector space of functions. From Figure 9 we compute, viewing the functions $g^{(j)}$ as vectors of coefficients in the basis of functions on $A, B, AB, BA, AA, BB$ (in that order)

$$g^{(0)} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}^T , \tag{97}$$
$$g^{(1)} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}^T \tag{98}$$

by counting the number of red entries for each $x$.

If we assume all the inputs have equal probability $\frac{1}{6}$, and computing the Hessian with respect to just the variable $w$ (which has value $w = 0$ when the state $q'$ in (96) is accept and the value $w = 1$ when it is reject) then by (34) we have for $j \in \{0, 1\}$

$$\text{Hess}(L, [\mathtt{detectA}^{(j)}]) = \frac{1}{3}(g^{(j)})^T g^{(j)} = \frac{1}{3} . \tag{99}$$

Thus in local coordinates around $[\mathtt{detectA}^{(j)}]$ we have $L = \frac{1}{3}w^2 + O(w^3)$. Even though these two Turing machines are distinct, their differences are irrelevant to errors in this particular tuple, so they have the same geometry in the $w$ direction.

| Syndrome | $x$ | |
|:---:|:---:|:---:|
| $*0$ | $A$ | accept |
| $\mathbf{0}1$ | $A$ | accept |
| $*0$ | $B$ | reject |
| $\mathbf{0}1$ | $B$ | reject |
| $*0$ | $AB$ | accept |
| $\mathbf{0}1$ | $AB$ | accept |
| $*0$ | $BA$ | accept |
| $\mathbf{0}1$ | $BA$ | reject |
| $*0$ | $AA$ | accept |
| $\mathbf{0}1$ | $AA$ | accept |
| $*0$ | $BB$ | reject |
| $\mathbf{0}1$ | $BB$ | reject |

(a) `detectA`$^{(0)}$

| Syndrome | $x$ | |
|:---:|:---:|:---:|
| $*0$ | $A$ | accept |
| $\mathbf{0}1$ | $A$ | accept |
| $*0$ | $B$ | reject |
| $\mathbf{0}1$ | $B$ | reject |
| $*0$ | $AB$ | accept |
| $\mathbf{0}1$ | $AB$ | accept |
| $*0$ | $BA$ | accept |
| $\mathbf{0}1$ | $BA$ | reject |
| $*0$ | $AA$ | accept |
| $\mathbf{0}1$ | $AA$ | accept |
| $*0$ | $BB$ | reject |
| $\mathbf{0}1$ | $BB$ | reject |

(b) `detectA`$^{(1)}$

Figure 9: Influence functions for weight one error syndromes for the two Turing machines implementing `detectA`. Here $*$ stands for any element of $\{0,1\}^6$ and $\mathbf{0} = 000000$.

## 7.4 Full geometry of `detectA`

In this section we complete the analysis by studying the geometry at `detectA`$^{(0)}$ to second-order in all coordinate directions. By Corollary 6.4 to compute the second-order derivatives of $H$ at $w = [M]$ we need to count error syndromes of weight one which change the output from $M(x) = y(x)$ (that is, cause an output error). As explained in Remark 6.2 we use $1 \le k \le d$ to index triples consisting of

- One among the $N = |\Sigma||Q|$ tuples on the description tape

- Within that tuple, one of the three positions $\sigma', q', d$

- For that position, a variation away from the entry of $[M]$ in that position.

For `detectA`$^{(0)}$ we have $d = 30$ and the enumeration is given in Figure 10. From now on indices $1 \le k \le d$ are associated with triples as indicated by this Figure.

For each $1 \le k \le d$ and input $x$, $A^k(x)$ is the number of computation paths leading from the description tape square associated to $k$ which, when the type of error indexed by $k$ is introduced, lead to an output error. We compute $A^k(x)$ for all $k$ and $x$ in

$$X = \{A, B, AB, BA, AA, BB\} \tag{100}$$

when we run the simulated machine for two steps $t = 2$. We use the notation for computation paths

$$\Gamma_j, \Lambda_j, \Omega, \Xi, \Theta_j \qquad 1 \le j \le N$$

introduced in Section 6.2.

**Lemma 7.6.** $A^k(x) = 0$ for all $k \notin \{3, 13, 18, 23, 24, 25\}$ and $x \in X$.

*Proof.* We explain a few cases in detail and leave the remaining cases to the reader. For $k = 1$ the relevant computation paths are of type $\Gamma_j$ (here the $j$ indexes tuples on the description tape and is unrelated to $k$). If in computing the symbol under the head $\sigma_0$ after one simulated timestep we use the variation $\square \to A$ for the first tuple in Figure 10, this does not affect $\sigma_0$ because the head never starts over a $\square$. Thus no output error can occur as a result of this variation, and so $A^k(x) = 0$ for $k \in \{1, 2, 6, 7\}$ and $x \in X$. The same logic applies to $k \in \{4, 5, 9, 10\}$ with $\Gamma_j$ replaced by $\Theta_j$.

Since the machine starts in state reject the variations $k \in \{16, 17, 19, 20, 26, 27, 29, 30\}$ similarly cannot affect $\sigma_0$ after one timestep along the computation paths $\Gamma_j, \Theta_j$ and so $A^k(x) = 0$ also in these cases for all $x \in X$.

When $k = 8$ we have to consider computation paths $\Lambda_j, \Omega, \Xi$. Along the former, again since the head never starts over $\square$, there can be no induced output error. The other computation path $\Omega$ is more subtle: note that $\mathtt{detectA}^{(0)}$ does not move the head once it transitions into state accept, so under normal operation it *cannot reach* the pair $(\square, \text{accept})$. Hence an error in $\Omega$ cannot cause an output error. For $\Xi$, we check each $x$ and observe that a match on the description tape is always found and so this computation path is never taken. Thus, $A^8(x) = 0$ for all $x$.[12]

For $k \in \{11, 12\}$ if $x$ begins with a $B$ this tuple is irrelevant. If it starts with $A$ then the machine transitions into state accept and it is irrelevant what is written to the tape as $\sigma_0$ (because the machine can never exit state accept). Hence errors of type $k$ on computation paths $\Gamma_j$ cannot cause an output error.[13] For similar reasons $A^k(x) = 0$ for all $k \in \{14, 15\}$ and $x \in X$.

For $k \in \{21, 22\}$ if $x$ begins with $A$ this tuple is irrelevant. If $x$ begins with $B$ then the machine stays in state reject and moves right, so $\sigma_0$ (the symbol under the head after one simulated timestep) is independent of what is written to the tape in the first step. Hence $A^k(x) = 0$ for all $x$.[14]

Finally when $k = 28$ we have to consider computation paths $\Lambda_j, \Omega, \Xi$. Since we start in state reject no error on path $\Lambda_j$ can cause an output error. For $\Omega$, there is no way for the machine under normal operation to reach state accept with the head over $B$, so again this cannot contribute an output error. Lastly, a match on the description tape is always found, so $\Xi$ cannot contribute an output error. Hence $A^k(x) = 0$ for all $x$ also in this case. $\square$

---

[12]Note how if we were to consider error syndromes of higher weight we would have to consider the possibility that the machine reaches $\square$, accept due to another error, thus making $\Omega$ for $k = 8$ relevant.

[13]Notice the possibility for an output error from higher weight syndromes implicit in this reasoning.

[14]Note that if we had an error of type 22 *and* an error of type 24 then the machine would write an $A$ and have the head stay over it, to be read in the second step. This suggests some third-order derivative may be nonzero, in a way that reflects the structure of the machine.

The reader is encouraged to work through the proof of Lemma 7.6 in detail, since this example illustrates the way in which *concretely* the behaviour of a Turing machine is reflected in combinatorics of error syndromes. It is quite intuitive why the variations that remain to be considered in Figure 10 (that is, variations other than the ones marked in blue which are ruled out by the Lemma) should be the most critical parts of the code.

| $\sigma$ | $q$ | $\sigma'$ | | $q'$ | $d$ | |
|---|---|---|---|---|---|---|
| $\square$ | reject | ${}^{1}\square \to A,$ | ${}^{2}\square \to B$ | ${}^{3}\text{reject} \to \text{accept}$ | ${}^{4}\text{Stay} \to \text{Left},$ | ${}^{5}\text{Stay} \to \text{Right}$ |
| $\square$ | accept | ${}^{6}\square \to A,$ | ${}^{7}\square \to B$ | ${}^{8}\text{accept} \to \text{reject}$ | ${}^{9}\text{Stay} \to \text{Left},$ | ${}^{10}\text{Stay} \to \text{Right}$ |
| $A$ | reject | ${}^{11}A \to \square,$ | ${}^{12}A \to B$ | ${}^{13}\text{accept} \to \text{reject}$ | ${}^{14}\text{Stay} \to \text{Left},$ | ${}^{15}\text{Stay} \to \text{Right}$ |
| $A$ | accept | ${}^{16}A \to \square,$ | ${}^{17}A \to B$ | ${}^{18}\text{accept} \to \text{reject}$ | ${}^{19}\text{Stay} \to \text{Left},$ | ${}^{20}\text{Stay} \to \text{Right}$ |
| $B$ | reject | ${}^{21}B \to \square,$ | ${}^{22}B \to A$ | ${}^{23}\text{reject} \to \text{accept}$ | ${}^{24}\text{Right} \to \text{Stay},$ | ${}^{25}\text{Right} \to \text{Left}$ |
| $B$ | accept | ${}^{26}B \to \square,$ | ${}^{27}B \to A$ | ${}^{28}\text{accept} \to \text{reject}$ | ${}^{29}\text{Stay} \to \text{Left},$ | ${}^{30}\text{Stay} \to \text{Right}$ |

Figure 10: Local coordinates at $[\texttt{detectA}^{(0)}]$ in the space of noisy Turing machines. The rows are indexed by $\sigma, q$ pairs (first two columns) and then for each of the three associated squares $\sigma', q, d$ on the description tape we enumerate the valid variations in the entry in that square away from its value in $\texttt{detectA}^{(0)}$. These variations are exactly the valid values taken by an error syndrome on that square. The variations also provide the local coordinates at $[\texttt{detectA}^{(0)}]$ in the space of noisy TM codes. Entries marked in blue do not contribute to error syndromes of weight one which can flip the output.

In what follows, we enumerate the tuples on the description tape following the rows of Figure 10. We sometimes view functions $X \to \mathbb{R}$ as vectors in the natural way, using the ordering in (100).

**Lemma 7.7.** For $k = 3$ the relevant computation paths are $\Lambda_j$ for $1 \le j \le N$ and $\Omega$ and so the error syndromes with weight one can be described by tuples

$$e_j = \left( \overset{\Lambda_1}{0}, \ldots, \overset{\Lambda_j}{1}, \ldots, \overset{\Lambda_N}{0}, \overset{\Omega}{0}, \overset{\Xi}{0} \right) \qquad 1 \le j \le N \tag{101}$$

$$e_{N+1} = \left( \overset{\Lambda_1}{0}, \ldots, \overset{\Lambda_j}{0}, \ldots, \overset{\Lambda_N}{0}, \overset{\Omega}{1}, \overset{\Xi}{0} \right) \tag{102}$$

$$e_{N+2} = \left( \overset{\Lambda_1}{0}, \ldots, \overset{\Lambda_j}{0}, \ldots, \overset{\Lambda_N}{0}, \overset{\Omega}{0}, \overset{\Xi}{1} \right) \tag{103}$$

The values of $\mathcal{U}(x, \gamma)$ are given by Table 2 and the function $A^3$ is

$$A^3 = \left( 0, 1, 0, 0, 0, 0 \right) \tag{104}$$

viewed as a vector in $\mathbb{R}^X$.

*Proof.* Recall $\mathcal{U}(x, \gamma)$ is the result of executing the UTM with error syndrome $\gamma$ on input $x$. When we write $\gamma = e_j$ in the context of the current lemma, we are implicitly considering the error syndrome that assigns 0 to all computation paths *other* than the ones starting at the position in the description tape indexed by $k = 3$, and just describing the values assigned by $\gamma$ to the computation paths starting at $k = 3$.

Note that since the head never starts over a blank symbol, $\gamma = e_j$ for $1 \leq j \leq N$ does not affect the output of the UTM. Hence all but the last column of Table 2 contain whatever output $M$ would give on that input. An error on the computation path $\Omega$ leads to error only when $x = B$ so that after the first step the machine has $\sigma_0 = \square, q = \text{reject}$ and then it erroneously transitions to accept. That is,

$$\mathcal{U}(x, e_{N+1}) = \begin{cases} M(x) & x \neq B \\ \text{accept} & x = B \end{cases} \tag{105}$$

The computation path $\Xi$ is never taken for the inputs considered here, so $\gamma = e_8$ does not affect the output of the UTM.

This completes the calculation of Table 2 and the values of $A^k(x)$ follow by summing the errors in each row. $\qquad\square$

| $x$ $\diagdown$ $\gamma$ | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |
|---|---|---|---|---|---|---|---|---|
| $A$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $B$ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | <span style="color:red">✓</span> | ✗ |
| $AB$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $BA$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $AA$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $BB$ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |

Table 2: Values of $\mathcal{U}(x, \gamma)$ for different $x$ and $\gamma$ and $k = 3$. To save space we use ✓ for accept, ✗ for reject, and we colour an output red if this is incorrect, i.e. an error.

**Lemma 7.8.** For $k = 13$ the relevant computation paths are $\Lambda_j$ for $1 \leq j \leq N, \Omega, \Xi$. With the notation of (101), (103) the function $A^{13}$ is

$$A^{13} = \big(0, 0, 0, 1, 0, 0\big) \tag{106}$$

viewed as a vector in $\mathbb{R}^X$.

*Proof.* For $x \in \{B, BB\}$ none of these computation paths are used. If $x = BA$ then errors in the $\Lambda_j$ computation paths cannot affect the output, but

$$\mathcal{U}(BA, e_7) = \text{reject}$$

is an output error induced by an error in $\Omega$. Otherwise the first symbol of $x$ is $A$, in which case errors on $\Omega$ cannot lead to output errors and we need only consider the computation paths $\Lambda_j$. The two relevant tuples on the description tape are $j \in \{3, 4\}$.

We have $\mathcal{U}(A*, e_3) = $ accept which is correct, even though an error occurred on the computation path $\Lambda_3$. In detail: if we use the modified tuple $(A, \text{reject}, A, \text{reject}, \text{Stay})$ in the first step we will have $\sigma_0 = A, q = $ reject after one timestep, which the UTM then compares with $(A, \text{reject})$ on the description tape during the scan phase and finds a match. It then writes accept to the $q'$ part of the staging tape (following the unmodified form of the third tuple). Note that the UTM finds a *second* match with $(A, \text{accept})$ and writes accept to the staging tape for a second time.

In contrast, we have $\mathcal{U}(A*, e_4) = $ accept. When checking $A, \text{accept}$ for a match on the description tape with $\sigma_0 = A, q = $ reject does not find a match. Hence the UTM does not match to *any* tuple and at the end of the scan phase $X$ remains on the staging tape, so it leaves the TM state unchanged with accept.

The computation path $\Xi$ is never taken for the inputs considered here, so $\gamma = e_8$ does not affect the output of the UTM. These calculations are summarised in Table 3. $\qquad\square$

| $\gamma$ \ $x$ | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |
|:---:|:---:|:---:|:---:|:---:|:---:|:---:|:---:|:---:|
| $A$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $B$ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| $AB$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $BA$ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ |
| $AA$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $BB$ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |

Table 3: Values of $\mathcal{U}(x, \gamma)$ for different $x$ and $\gamma$ and $k = 13$.

**Lemma 7.9.** For $k = 18$ the relevant computation paths are $\Lambda_j$ for $1 \leq j \leq N, \Omega$ and $\Xi$. With the notation of (101), (103) the function $A^{18}$ is

$$A^{18} = \begin{pmatrix} 1, 0, 1, 0, 1, 0 \end{pmatrix} \tag{107}$$

viewed as a vector in $\mathbb{R}^X$.

*Proof.* Again there can be no output errors when $x \in \{B, BB, BA\}$. However when $x$ begins with $A$ and we have the error of type $k = 18$ on computation path $\Omega$ we have $\mathcal{U}(A*, e_7) = $ reject which is an output error. Lastly, the computation path $\Xi$ is never taken. These results are in Table 4. $\qquad\square$

| $x$ \ $\gamma$ | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |
|---|---|---|---|---|---|---|---|---|
| $A$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ |
| $B$ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| $AB$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ |
| $BA$ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| $AA$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ |
| $BB$ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |

Table 4: Values of $\mathcal{U}(x,\gamma)$ for different $x$ and $\gamma$ and $k = 18$.

**Lemma 7.10.** For $k = 23$ the relevant computation paths are $\Lambda_j$ for $1 \le j \le N, \Omega$ and $\Xi$. With the notation of (101), (103) the function $A^{23}$ is

$$A^{23} = \big(0, 1, 0, 1, 0, 2\big) \tag{108}$$

viewed as a vector in $\mathbb{R}^X$.

*Proof.* Again there can be no output errors when $x \in \{A, AB, AA\}$. The behaviour of $\mathcal{U}$ on inputs beginning with $B$ is quite complex. Firstly if $x \in \{B, BA\}$ we have $\mathcal{U}(x, e_7) = M(x)$. But $\mathcal{U}(BB, e_7) = \text{accept}$. For the $\Lambda_j$ computation paths, we can ignore most columns depending on the second symbol in the input.

$$\mathcal{U}(B, e_1) = \text{reject}, \quad \mathcal{U}(B, e_2) = \text{accept}$$
$$\mathcal{U}(BA, e_3) = \text{reject}, \quad \mathcal{U}(BA, e_4) = \text{accept}$$
$$\mathcal{U}(BB, e_5) = \text{reject}, \quad \mathcal{U}(BB, e_6) = \text{accept} .$$

The computation path $\Xi$ is irrelevant. These calculations are summarised in Table 5. $\square$

| $x$ \ $\gamma$ | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ |
|---|---|---|---|---|---|---|---|---|
| $A$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $B$ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| $AB$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $BA$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $AA$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $BB$ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ |

Table 5: Values of $\mathcal{U}(x,\gamma)$ for different $x$ and $\gamma$ and $k = 23$.

**Lemma 7.11.** For $k = 24$ the relevant computation paths are $\Theta_j$ for $1 \leq j \leq N$ so the error syndromes with weight one can be described by tuples

$$e_j = \left( \overset{\Theta_1}{0}, \ldots, \overset{\Theta_j}{1}, \ldots, \overset{\Theta_N}{0} \right) \qquad 1 \leq j \leq N. \tag{109}$$

The values of $\mathcal{U}(x, \gamma)$ are given in Table 6

$$A^{24} = \left( 0, 0, 0, 2, 0, 0 \right) \tag{110}$$

viewed as a vector in $\mathbb{R}^X$.

*Proof.* For $x \notin \{B, BA, BB\}$ we have $\mathcal{U}(x, e_j) = M(x)$ for all $j$. For any $x$ that begins with $B$, the effect of introducing a $k = 24$ type error is to stay rather than move right in the first step, so that $\sigma_0 = B$ after one step. Thus $\mathcal{U}(BB, e_j) = M(BB) = $ reject for all $j$, as this is any case the symbol under the head under normal execution. We can ignore $j \in \{2, 4, 6\}$ since the machine is in state reject after one step. The remaining cases are

$$\mathcal{U}(B, e_1) = \text{reject}, \quad \mathcal{U}(B, e_3) = \text{reject}, \quad \mathcal{U}(B, e_5) = \text{reject}$$
$$\mathcal{U}(BA, e_1) = \text{accept}, \quad \mathcal{U}(BA, e_3) = \text{reject}, \quad \mathcal{U}(BA, e_5) = \text{reject}$$

These calculations are summarised in Table 6. $\qquad\square$

| $x$ \ $\gamma$ | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ |
|:---:|:---:|:---:|:---:|:---:|:---:|:---:|
| $A$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $B$ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| $AB$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $BA$ | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ |
| $AA$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $BB$ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |

Table 6: Values of $\mathcal{U}(x, \gamma)$ for different $x$ and $\gamma$ and $k = 24$.

**Lemma 7.12.** For $k = 25$ the relevant computation paths are $\Theta_j$ for $1 \leq j \leq N$ so the error syndromes with weight one can be described by tuples as in (109). Then

$$A^{25} = \left( 0, 0, 0, 1, 0, 0 \right) \tag{111}$$

viewed as a vector in $\mathbb{R}^X$.

*Proof.* For $x \notin \{B, BA, BB\}$ we have $\mathcal{U}(x, e_j) = M(x)$ for all $j$. For any $x$ beginning with $B$ introducing a $k = 24$ type error causes the head to move left rather than right in the first timestep, so $\mathcal{U}(B, e_j) = M(B) = $ reject for all $j$ since the symbol to the left

of the first symbol on the work tape is a blank. It is also easy to see that $\mathcal{U}(BB, e_j) = M(BB) = \text{reject}$ for all $j$. When $x = BA$ we have

$$\mathcal{U}(BA, e_2) = \mathcal{U}(BA, e_4) = \mathcal{U}(BA, e_6) = M(BA) = \text{accept}$$

since these tuples never match to the state reject after one timestep. Finally

$$\mathcal{U}(BA, e_1) = \text{accept}\,[15], \quad \mathcal{U}(BA, e_3) = \text{reject}, \quad \mathcal{U}(BA, e_5) = \text{accept}\,.$$

These calculations are summarised in Table 7. □

| $x$ \ $\gamma$ | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ |
|---|---|---|---|---|---|---|
| $A$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $B$ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| $AB$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $BA$ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ |
| $AA$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $B$ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |

Table 7: Values of $\mathcal{U}(x, \gamma)$ for different $x$ and $\gamma$ and $k = 25$.

Given the above we compute the functions $\{g_3, g_{13}, g_{18}, g_{23}, g_{24}, g_{25}\}$ and then the matrix $P$ of Section 3 is

$$
\begin{array}{c|cccccc|l}
 & 3 & 13 & 18 & 23 & 24 & 25 & \text{(24 zero columns)} \\
\hline
A & 0 & 0 & 1 & 0 & 0 & 0 & \\
B & 1 & 0 & 0 & 1 & 0 & 0 & \\
AB & 0 & 0 & 1 & 0 & 0 & 0 & \\
BA & 0 & 1 & 0 & 1 & 2 & 1 & \\
AA & 0 & 0 & 1 & 0 & 0 & 0 & \\
BB & 0 & 0 & 0 & 2 & 0 & 0 & \\
\end{array}
\tag{112}
$$

If we assume all the inputs have equal probability $\frac{1}{6}$ then by (34) we have

$$\text{Hess}(H, [\texttt{detectA}^{(0)}]) = \tfrac{1}{3} P^T P = \tfrac{1}{3} \begin{pmatrix} P' & 0 \\ 0 & 0 \end{pmatrix} \tag{113}$$

where (ordering the variables as in (112))

$$P' = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 2 & 1 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 1 & 1 & 0 & 6 & 2 & 1 \\ 0 & 2 & 0 & 2 & 4 & 2 \\ 0 & 1 & 0 & 1 & 2 & 1 \end{pmatrix}. \tag{114}$$

---

[15]Note that here the UTM executes two tuples.

From this we see that $e_{18}$ is an eigenvector of eigenvalue $\lambda_1 = 3$, and that

$$P'(e_{24} - 2e_{13}) = P'(e_{25} - e_{13}) = 0\,.$$

Hence zero is an eigenvalue of multiplicity two. Indeed the characteristic polynomial is

$$p(\lambda) = \lambda^2(\lambda - 3)(\lambda^3 - 13\lambda^2 + 41\lambda - 24)\,.$$

Let $\lambda_2, \lambda_3, \lambda_4$ denote the roots of the cubic factor, which are approximately $0.76, 3.73, 8.52$. Then in local coordinates $v_1, \ldots, v_{30}$ at $[\mathtt{detectA}^{(0)}]$ we have

$$H = v_1^2 + \sum_{i=2}^{4} \tfrac{1}{3}\lambda_i v_i^2 + O(v^3)\,. \tag{115}$$

Remarkably, there are *only four out of thirty* directions in the space of noisy TMs in which the function $H$ is nondegenerate. The majority of the possible weight one errors do not affect the performance of the machine (on inputs in $X$ when run for two steps).

Note that the behaviour of the machine when it is in state accept and reading an $A$ (which is the only way to be in state accept) is critical in our two step machine; introducing an error in the state part of this tuple is therefore one of the *most obvious ways to cause an output error* and we would therefore expect $H$ to be strongly curved in this direction. This is what the geometric analysis shows: the direction $v_1$ corresponding to $e_{18}$ is one of the natural coordinates associated to the eigenvectors of the Hessian of $H$, and it has a large positive eigenvalue. Thus we see a direct relationship between the local geometry of $H$ at $[\mathtt{detectA}^{(0)}]$ and the structure of the Turing machine.

**Remark 7.13.** Recall that it is the geometry of the average negative log-likelihood $L(w)$ that we are ultimately interested in, and the polynomial $H(w)$ is just a convenient way to access (some of) that geometry, since the two functions are comparable. Thus the above observations about $H$ in local coordinates should not be overinterpreted. However, it *is* true that (115) implies that the Hessian of $L$ also has rank 4 at $[\mathtt{detectA}^{(0)}]$ and moreover, we can relate their eigenvalues to some extent (see Appendix B).

**Remark 7.14.** It is clear by examining $P'$ that a basis for the kernel is $-2e_{13}+e_{24}, -e_{13}+e_{25}$. This gives two additional zero curvature directions in addition to the 24 identified in Lemma 7.6. By Lemma B.1 these are also directions in the kernel of the Hessian of $L$, that is, they are genuinely part of the geometry of the statistical learning problem.

In the notation of Figure 10 these directions are

$$-2\bigl(\text{accept} \overset{13}{\to} \text{reject}\bigr) + \text{Right} \overset{24}{\to} \text{Stay},$$

$$-1\bigl(\text{accept} \overset{13}{\to} \text{reject}\bigr) + \text{Right} \overset{25}{\to} \text{Left}\,.$$

Note that at the boundary point $[\mathtt{detectA}^{(0)}]$ of $W$ these tangent vectors point *outside* of the space of noisy TM codes. Nonetheless they indicate an interesting and non-obvious way to trade-off these two tuples, reflecting non-trivial structure of the machine.

# 8  Conclusion

One of the central lessons of logic and computer science for mathematics is to pay attention to the distinction between an *algorithm* and the *function* it computes. In statistical learning theory we often treat models as functions which assign probabilities to data, and ignore how those functions are implemented. In this section we explain how, within the framework of SLT, there is a natural sense in which the internal structure of *probabilistic models as algorithms* play an important role in Bayesian statistics.

To compare the evidence for two candidate models $p(y|x, w^1), p(y|x, w^2)$ of a conditional distribution $q(y|x)$ with respect to a dataset $D_n$, we integrate the posterior over neighbourhoods $\mathcal{W}_i$ of $w^i$. If for simplicity we assume that $w^1, w^2$ are both true parameters, then we know from SLT that asymptotically in $n$ this comparison is determined by the local geometry of the KL divergence $K(w)$ at each $w^i$ (see Section 2.4). This geometry is a reflection of *the pattern of variation of the probabilities* as $w$ is varied near $w^i$

$$p(y|x, w) \approx p(y|x, w^i) \text{ for } w \approx w^i. \tag{116}$$

In many cases, such as when the probabilistic model $p(y|x, w)$ is parametrised by a neural network or as here by a noisy TM, variations in the parameter $w$ are localised microscopic changes in the algorithm that computes these probabilities (e.g. weights associated to connections between specific neurons, or probabilities of specific errors in the UTM reading the Turing machine description). Thus the geometry of $K(w)$, and consequently the preferences of the posterior, are related to how such variations in the algorithm affect (or do *not* affect) the output probabilities. If the probabilities are computed by algorithms which have hidden variables, hierarchical structure or are composed of multiple information processing modules, then we expect these statistical models to be singular and that "the knowledge or grammar to be discovered corresponds to singularities" [46, §1.2].

To make this more precise: some changes in the algorithm underlying our model are either just irrelevant to the outputs, or can be compensated by coordinated changes in other parts, leading to changes in $w$ that do not change the output probabilities. These totally flat directions in the geometry of $K(w)$, or *degeneracies*, are signatures of the possible trade-offs among parts of the algorithm, and in turn reflect the internal structure. More refined information is obtained by considering coordinated changes in the algorithm that do not change the probabilities *up to some order* in a small perturbation parameter $\varepsilon$ measuring how far we deviate from the original algorithm.

In the framework developed in this paper we have studied concrete examples of both kinds of changes: in our study of the Turing machine detectA$^{(0)}$ we found that changes in 24 out of 30 directions at this code are irrelevant to outputs to second order, and in Remark 7.14 we exhibit two linearly independent coordinated changes in the code that together do not change the output probabilities up to second order.

## 8.1 Structural Bayesianism

All this leads to the hypothesis that the internal computational structure of Turing machines, and potentially other singular models such as neural networks, is encoded in the geometry of the KL divergence and may be discoverable from data [25, 23, 44].

We view this as a strong motivation for taking *interpretability* of singular models, including neural networks, seriously as a mathematical science. This philosophy, which we refer to as *structural Bayesianism*, says that the model selection principles at the heart of Bayesian statistics dictate in many cases that the posterior can "see" the internal structure of the algorithms computing probabilities in the model, and consequently the posterior can prefer some kinds of algorithmic structures over others. That is, the posterior concentrates probability mass on programs that not only predict *well* but predict *properly*.

# References

[1] Elionora Arnold, Sabir Medzhidovich Gusein-Zade, and Alexander N Varchenko. *Singularities of differentiable maps, volume 2: monodromy and asymptotics of integrals*. Springer Science & Business Media, 2012.

[2] Vladimir Igorevich Arnold. *Singularity theory*, volume 53. Cambridge University Press, 1981.

[3] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[4] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International conference on machine learning*, pages 254–263. PMLR, 2018.

[5] Eugene Asarin and Pieter Collins. Noisy turing machines. In *International Colloquium on Automata, Languages, and Programming*, pages 1031–1042. Springer, 2005.

[6] Nick Benton, Gavin Bierman, Valeria de Paiva, and Martin Hyland. A term calculus for intuitionistic linear logic. In Marc Bezem and Jan Friso Groote, editors, *Typed Lambda Calculi and Applications*, pages 75–90, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[7] George Boole. The laws of thought. *Collected Logical Works*, 2, 1854.

[8] Ilir Çapuni and Péter Gács. A turing machine resisting isolated bursts of faults. In *SOFSEM 2012: Theory and Practice of Computer Science: 38th Conference on Current Trends in Theory and Practice of Computer Science, Spindleruv Mlỳn, Czech Republic, January 21-27, 2012. Proceedings 38*, pages 165–176. Springer, 2012.

[9] Pierre Cartier. A mad day's work: from grothendieck to connes and kontsevich the evolution of concepts of space and symmetry. *Bulletin of the American Mathematical Society*, 38(4):389–408, 2001.

[10] Sergio Cecotti and Cumrun Vafa. On classification of n= 2 supersymmetric theories. *Communications in Mathematical Physics*, 158:569–644, 1993.

[11] Zhongtian Chen, Edmund Lau, Jake Mendel, Susan Wei, and Daniel Murfet. Dynamical versus Bayesian phase transitions in a toy model of superposition. *arXiv preprint arXiv:2310.06301*, 2023.

[12] Alonzo Church. A set of postulates for the foundation of logic. *Annals of mathematics*, 33(2):346–366, 1932.

[13] James Clift. Turing machines and differential linear logic. Master's thesis, MSc, University of Melbourne, 2017.

[14] James Clift and Daniel Murfet. Derivatives of Turing machines in linear logic. *arXiv preprint arXiv:1805.11813*, 2018.

[15] James Clift and Daniel Murfet. Cofree coalgebras and differential linear logic. *Mathematical Structures in Computer Science*, 30(4):416–457, 2020.

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

[17] James Clift, Daniel Murfet, and James Wallbridge. Geometry of program synthesis. *arXiv preprint arXiv:2103.16080*, 2021.

[18] Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1-3):1–41, 2003.

[19] Gerhard Gentzen. Untersuchungen über das logische schließen. i. *Mathematische zeitschrift*, 39(1):176–210, 1935.

[20] Jean-Yves Girard. Linear logic. *Theoretical computer science*, 50(1):1–101, 1987.

[21] Gert-Martin Greuel, Christoph Lossen, and Eugenii Shustin. Singularity theory. *Introduction to Singularities and Deformations*, pages 1–220, 2007.

[22] Gert-Martin Greuel, Christoph Lossen, and Eugenii I Shustin. *Introduction to singularities and deformations*. Springer Science & Business Media, 2007.

[23] Jesse Hoogland, George Wang, Matthew Farrugia-Roberts, Liam Carroll, Susan Wei, and Daniel Murfet. Loss landscape degeneracy drives stagewise development in transformers. *arXiv preprint arXiv:2402.02364*, 2025.

[24] William A Howard et al. The formulae-as-types notion of construction. *To HB Curry: essays on combinatory logic, lambda calculus and formalism*, 44:479–490, 1980.

[25] Edmund Lau, Zach Furman, George Wang, Daniel Murfet, and Susan Wei. The local learning coefficient: A singularity-aware complexity measure. *arXiv preprint arXiv:2308.12108*, 2024.

[26] Shaowei Lin. *Algebraic methods for evaluating integrals in Bayesian statistics*. University of California, Berkeley, 2011.

[27] Trevor McCourt, Ila R. Fiete, and Isaac L. Chuang. Noisy dynamical systems evolve error correcting codes and modularity, 2023.

[28] Paul-André Mellies. Categorical semantics of linear logic. *Panoramas et syntheses*, 27:15–215, 2009.

[29] Daniel Murfet. Logic and linear algebra: an introduction. *arXiv preprint arXiv:1407.2650*, 2014.

[30] Daniel Murfet. On sweedler's cofree cocommutative coalgebra. *Journal of Pure and Applied Algebra*, 219(12):5289–5304, 2015.

[31] Daniel Murfet and William Troiani. Elimination and cut-elimination in multiplicative linear logic. *arXiv preprint arXiv:2207.10871*, 2022.

[32] Ryan O'Donnell. *Analysis of Boolean functions*. Cambridge University Press, 2014.

[33] Nicholas Pippenger. On networks of noisy gates. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 30–38. IEEE, 1985.

[34] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[35] Ray J Solomonoff. A formal theory of inductive inference. part I. *Information and Control*, 7(1):1–22, 1964.

[36] Daniel A Spielman. Highly fault-tolerant parallel computation. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 154–163. IEEE, 1996.

[37] Anne Sjerp Troelstra et al. History of constructivism in the 20th century. *Set Theory, Arithmetic, and Foundations of Mathematics*, pages 150–179, 2011.

[38] William Troiani and Daniel Murfet. Linear logic and the hilbert scheme. *arXiv preprint arXiv:2502.08911*, 2025.

[39] Alan Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society Series 2*, 42:230–265, 1937.

[40] Alan Turing. Intelligent Machinery. 1948.

[41] John Von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata studies*, 34(34):43–98, 1956.

[42] John Von Neumann. *The Computer and the Brain.* 1958.

[43] John Von Neumann. *Collected Works*, volume 5. Pergamon Press, Oxford, 1963.

[44] George Wang, Jesse Hoogland, Stan van Wingerden, Zach Furman, and Daniel Murfet. Differentiation and specialization of attention heads via the refined local learning coefficient. *arXiv preprint arXiv:2410.02984*, 2024.

[45] Thomas Waring. *Geometric perspectives on program synthesis and semantics.* PhD thesis, MSc, The University of Melbourne, 2021.

[46] Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory.* Cambridge University Press, USA, 2009.

[47] Sumio Watanabe. *Mathematical Theory of Bayesian Statistics.* CRC Press, Taylor and Francis group, USA, 2018.

[48] Sumio Watanabe. Cross Validation, Information Criterion and Phase Transition. *Talk*, 2020.

[49] Ilir Çapuni and Peter Gács. A reliable turing machine, 2021.

# A    Frequently Asked Questions

So, you have questions.

## A.1    Does the geometry depend on the encoding?

We have described how to associate the germ of a singularity to a Turing machine $M$, or more precisely to the code of this machine $[M]$ for a particular pseudo-UTM. There are many choices involved in producing this geometric object, including: the choice of UTM, the choice of encoding of the step function of that UTM into linear logic, and the choice to use linear logic at all to construct the smooth extension. In this section we address some of the natural questions about these choices.

### A.1.1  Choice of UTM

There is no question of this geometry being *canonically* associated to $M$, and indeed it is our opinion that it is probably unreasonable to expect any such canonical association: at some conceptual level the geometry of $L$ (or $H$) at $[M]$ is determined by the particular way we choose to introduce errors into the execution, and this in turn is determined by *how* we think about executing that code. Different UTMs will lead to different geometries. In our understanding this is a reflection of the fact that the notion of "internal structure" in a Turing machine is not intrinsic to the tuples themselves, but to the dynamical system that represents their execution (that is, what a UTM does with that code for the TM).

The reader should recall that there are UTMs other than the "simulation type" UTM. For example, given any particular TM we can augment a simulation-type UTM $\mathcal{U}^{\mathrm{sim}}$ with special subroutines that recognise some part of the TM and consult a look-up table encoded in the UTM's specification in order to skip an arbitrary amount of computation and directly return the answer. Let us call this $\mathcal{U}^{\mathrm{cheat}}$. Any structure internal to the part of the TM $M$ that is skipped will not be visible in the geometry for $[M]$ on $\mathcal{U}^{\mathrm{cheat}}$.

Even if we stick to simulation-type UTMs, there are many variations. Some, like our staged pseudo-UTM, are convenient for doing calculations but may strike the reader as unnatural in various ways. We could hypothesise that some aspects of the geometry are independent of these variations to some degree, just like Kolmogorov complexity is independent of the chosen UTM to some degree, but there is no present theory which makes such claims and it is unclear what statement one could even hope for.

### A.1.2  Choice of Linear Logic

We choose to compute the propagation of uncertainty about Turing machine codes through the execution of a UTM by encoding the step function of the UTM into linear logic and using the Sweedler semantics [30, 16, 14]. This is somewhat roundabout and the reader is forgiven for wondering if "all this linear logic stuff is really necessary". The answer is No!

There is a (perhaps kinder) version of this paper which eschews linear logic completely, presents the computation paths of Appendix D as a *fait accompli* and presents a model of noisy execution of TM codes in those terms and derives geometry in that way. The reader is encouraged to get their hands dirty with the calculations in Section 7.4 to see how the associated mode of thought engages with computation paths and does not depend on linear logic at all. This is an entirely valid way of approaching the theory.

Nonetheless, our opinion is that while this presentation is easier to approach it is ultimately shallower, as it obscures the influence of Ehrhard-Regnier's idea of derivatives of proofs. Note that by Theorem 4.11 the geometry considered here is *an object of proof theory in its own right* (modulo the business with minus signs and the antipode). Moreover in that version of the paper, dear reader, you are asking where this notion of computation path comes from. There is some elegance (in the Kolmogorov complexity sense) of simply

saying "encode the UTM in linear logic" and then following that to its logical conclusion, but we accept that this elegance is only visible if you know linear logic (there's that UTM dependence of $K$). From a practical point of view one can think of linear logic as a useful term language to describe the algorithm for executing the UTM.

### A.1.3    Choice of Encoding

Once one has fixed a UTM and decided to use linear logic to construct a smooth relaxation of its step function, there are still multiple encodings into linear logic [16]. Variations are possible which would conceivably change the geometry of the final germ $([M], L)$. Once again, we think of these encodings as a compact *specification* of the computation paths which in turn determine how error can affect the simulation of the machine. We have provided one specification that we find natural, but we have no ground to assert that there are no other natural specifications. The claim of this paper is to construct *an example* of a structure preserving map from Turing machines to singularities. There is no claim, explicitly or implicitly, that this homomorphism is *unique*.

## A.2    Is everything in the geometry computationally meaningful?

Since every coefficient in the Taylor series expansion of $H$ has been given a computational interpretation in terms of correlations of error syndrome statistics in Theorem 6.3, it is possible to answer Yes in a formal sense provided we restrict this statement to the aspects of the geometry of $H$ that are shared with $L$ (Section B.1).

Our present intuition is that this answer seems overly strong, but since we don't have a clear definition of what "counts" as computational structure of a Turing machine it seems impossible to give a definitive answer. On the other hand, it seems plausible that geometry of the singularity $([M], L)$ that initially seems very remote from computer science may eventually be seen to be intimately connected to basic concepts in that field.

## A.3    Why is there a uniform number of time steps $t$

In our conceptual framing in terms of synthesis problems and statistical learning, we are interested in two classical solutions $M_1, M_2$ and how the Bayesian posterior distinguishes them. Let us suppose that for all $x \in I$ not only do $M_1(x) = y(x), M_2(x) = y(x)$ in the sense of the main text, but the machines both *halt* (i.e. enter some static state).

Then, given a dataset size $n$ the question is: does the posterior assign higher probability to a neighbourhood of $M_1$ or $M_2$? Since the dataset is finite and all inputs are finite strings, and *since $M_1, M_2$ are assumed to be classical solutions* and to halt, it may seem that there is some $t$ such that comparing them using this number of steps is sufficient. This is almost correct, but note that the local geometry is sensitive to execution traces of the machines that involve errors, and such traces *may fail to halt*. Thus the geometry can be sensitive to arbitrary numbers of timesteps even if the machines would usually halt on all allowed inputs within a given timeout.

More general synthesis problems are considered in [14, 45] but this introduces additional complexity and in this paper we follow [17] in making the simplification of fixing $t$. We admit this as a limitation of the current presentation. It would be interesting if there was an argument to the effect that non-stably halting machines are exponentially suppressed in the posterior, which would provide a more principled reason for fixing $t$.

## A.4  Why is the input distribution given finite support?

See the previous section: if we fix a finite number of timesteps then only a finite subset of the possible inputs can be computed. Note that finite numbers can be big: take $t \gg 0$ and $I \supseteq \Sigma^k$ for all $k \leq K$ for $K \gg 0$.

## A.5  How should I think about this model of noisy computation?

We refer the reader to Section 5.1 for a description of the staged pseudo-UTM, herein just "UTM". The remarks in this section expand on those in Section 1.1.1.

Recall that the squares of the description tape are of the form $\sigma, q, \sigma', q', d$. To compute the probabilities $p(y|x, w)$ as in Section 2.1 the string $x$ is placed on the work tape of the UTM, $w$ is arranged in the squares of type $\sigma', q', d$ on the description tape and then the (smooth relaxation of the) UTM is executed for $t$ steps before reading off the distribution over symbols $y$ on the state tape (meaning the tape which records the state of the simulated machine).

Note that we do not allow uncertainty in the inputs $x$. There is a distribution $q(x)$ over inputs, but this is irrelevant: the UTM only ever begins execution with a definite string $x$ on its work tape. We of course allow uncertainty on the squares of the description tape coming from $w$ (there is never uncertainty about the squares of type $\sigma, q$). While there is initially no uncertainty on the work, state or staging tapes, the uncertainty on the description tape propagates to these tapes as well. There may also be some uncertainty in the state of the UTM (e.g. between copySymbol and $\neg$ copySymbol) but the design of the staged pseudo-UTM mean that this uncertainty is "resolved" at the end of every cycle.

The upshot of using the encoding of the step function of the UTM into linear logic to propagate uncertainty is that, following the logic of Section 4.2, *whenever the UTM wants to read from a square on the description, work or staging tapes with uncertainty, it draws a sample.* But there is a catch: these samples must all be *independent*. What that means is that the UTM must recurse backwards to sample, in the previous step, all tape squares necessary to produce the tape square it originally wanted to read. These "requests for samples" propagate backwards through all previous steps until they cash out in (many) requests for samples from the original sources of uncertainty on the description tape.

These "chains of requests" from the final output state $y$ to the squares on the description tape are precisely the computation paths (Appendix D). Reading them forwards, these computation paths specify how one of a number of independent samples from some

error distribution on the description tape is to be routed through the execution of the UTM to be "consumed". This backwards chaining of requests is made precise by the cut-elimination rules for promotion and contraction, and this explains why linear logic is a particularly natural language to reason about this kind of noisy computation model.

## A.6  Why use TMs? Why not proofs or lambda terms?

The parameter space $W$ is part of the space of denotations of a linear logic formula, which has as proofs configurations of the description tape of a UTM. For a linear logician, it might seem natural to instead consider general proof synthesis for a general type $A$ and $W \subseteq \llbracket A \rrbracket$ relative to some proof $\psi : !A, !C \vdash B$ which plays the role of the (iterated) step function of the UTM in the present paper with some function $y : !C \vdash B$ as the analogue of the synthesis problem. And indeed, this geometrisation of proof synthesis is what [16, 14] were originally setting out to do!

There are however two problems: firstly, there is the technical problem of the propagation of uncertainty through non-plain proofs [16]. It is not currently known if this holds for proofs $\psi$ which are not plain. Putting that aside, the above theory could be constructed exactly as in this paper for any plain proof $\psi$, but then the question becomes: what are the natural examples we can actually compute with? Turing machine codes provide a natural "flat" class of examples that are short, can be constructed by hand, and which have a clear conceptual content.

## A.7  Why not a real UTM?

The staged pseudo-UTM $\mathcal{U}$ is not universal, because it assumes the alphabet symbols and states of the simulated machine can be encoded as single symbols and states of the UTM. That is, $\Sigma \cup Q \subseteq \Sigma_{\mathrm{UTM}}$. In a similar spirit to Appendix A.3, note that by enlarging $\Sigma, Q$ we can in this way accommodate any *finite* set of Turing machines as codes on our pseudo-UTM and this suffices for our purposes.

## A.8  Why not smooth relaxation $\mathcal{X}$?

There are many smooth relaxations of a UTM in the sense of Section 2.1. Maybe you have one in mind. Any such smooth relaxation gives rise to a KL divergence $K$ whose geometry is relevant to the learning process defined with respect to that smooth relaxation. For many such relaxations there will be a relation between the local learning coefficient and invariants like program length, as described in [17, 45] because un-used bits will often correspond to degeneracy.

However, generically a smooth relaxation $\mathcal{X}$ will have no deeper relation between the geometry of $K : W \longrightarrow \mathbb{R}$ and the internal structure of the machine being simulated $[M] \in W$, because there will be no *precise* and general relation between perturbations

away from $[M]$ within $W$ and computationally meaningful perturbations of the execution of $M$ as a dynamical system.

In our opinion, the nontrivial contribution being made in the series of papers [14, 17, 45] and continued here is to construct a smooth relaxation where such a precise relation *does* exist. One way of observing this is that the geometry of $K$ is here related to the derivatives of $H$ which are, by Theorem 4.11, denotations of *proofs in differential linear logic*. In some sense the geometry of the synthesis problem is an object of proof theory. Arguably the degeneracies observed in Section 7.4 arise from patterns that can also be expressed in the syntax. These are strong statements about the logical status of the geometry of $K$ which are not true of general smooth relaxations.

# B    Comparable Functions

We make some basic remarks about comparable functions (see Definition 2.10).

**Lemma B.1.** Let $f, g : \mathbb{R}^d \to \mathbb{R}$ be real analytic functions on an open neighbourhood $U$ of the origin, with local minima at the origin. Suppose there exist constants $c, d > 0$ such that

$$c \cdot f(x) \leq g(x) \leq d \cdot f(x) \tag{117}$$

for all $x \in U$. Let $Q_f$ and $Q_g$ be the Hessian matrices of $f$ and $g$ at the origin, respectively. Then:

(i)  $\ker(Q_f) = \ker(Q_g)$.

(ii)  For any eigenvector $v$ of $Q_f$ with eigenvalue $\lambda_f > 0$, we have $c \cdot \lambda_f \leq v^T Q_g v \leq d \cdot \lambda_f$.

*Proof.* Since $f$ and $g$ have local minima at 0, we have $f(0) = g(0) = 0$, $\nabla f(0) = \nabla g(0) = 0$, and both $Q_f$ and $Q_g$ are positive semidefinite. The Taylor expansions around the origin are:

$$f(x) = \frac{1}{2} x^T Q_f x + O(\|x\|^3) \tag{118}$$

$$g(x) = \frac{1}{2} x^T Q_g x + O(\|x\|^3) \tag{119}$$

For (i) it suffices to show $\ker(Q_f) \subseteq \ker(Q_g)$. Let $v \in \ker(Q_f)$, so $Q_f v = 0$ and $v^T Q_f v = 0$. For sufficiently small $t$, consider $x = tv \in U$:

$$f(tv) = \frac{1}{2} t^2 v^T Q_f v + O(t^3) = O(t^3) \tag{120}$$

By our growth rate assumption, we have:

$$g(tv) \geq c \cdot f(tv) = c \cdot O(t^3) = O(t^3) \tag{121}$$

$$g(tv) \leq d \cdot f(tv) = d \cdot O(t^3) = O(t^3) \tag{122}$$

However, we also know:

$$g(tv) = \frac{1}{2}t^2 v^T Q_g v + O(t^3) \tag{123}$$

For these equations to be consistent, we must have $v^T Q_g v = 0$. Since $Q_g$ is positive semidefinite, this implies $Q_g v = 0$, and therefore $v \in \ker(Q_g)$.

For (ii) let $v$ be an eigenvector of $Q_f$ with eigenvalue $\lambda_f > 0$, so $Q_f v = \lambda_f v$ and $v^T Q_f v = \lambda_f \|v\|^2$. Without loss of generality, we can assume $\|v\| = 1$.

For small $t$, consider $x = tv$:

$$f(tv) = \frac{1}{2}t^2 v^T Q_f v + O(t^3) = \frac{1}{2}t^2 \lambda_f + O(t^3) \tag{124}$$

$$g(tv) = \frac{1}{2}t^2 v^T Q_g v + O(t^3) \tag{125}$$

From our growth rate constraint, we have:

$$c \cdot f(tv) \leq g(tv) \leq d \cdot f(tv) \tag{126}$$

Substituting the expansions:

$$c \cdot \left(\frac{1}{2}t^2 \lambda_f + O(t^3)\right) \leq \frac{1}{2}t^2 v^T Q_g v + O(t^3) \leq d \cdot \left(\frac{1}{2}t^2 \lambda_f + O(t^3)\right) \tag{127}$$

Dividing by $\frac{1}{2}t^2$ and taking the limit as $t \to 0$, we obtain:

$$c \cdot \lambda_f \leq v^T Q_g v \leq d \cdot \lambda_f \tag{128}$$

This establishes the desired bounds on the quadratic form $v^T Q_g v$ for any eigenvector $v$ of $Q_f$ with positive eigenvalue $\lambda_f$. $\qquad\square$

**Remark B.2.** Lemma B.1 establishes that two real analytic functions with comparable growth rates near a common minimum point have Hessian matrices with identical kernels. Thus the ellipsoids defined by the quadratic forms have the same "flat directions" and further their curvatures in other directions maintain bounded ratios as

$$c \leq \frac{v^T Q_g v}{\lambda_f} \leq d.$$

Suppose that $v, v'$ are two eigenvectors of $Q_f$ with distinct eigenvalues $\lambda, \lambda'$. Define the curvatures of $Q_g$ in these directions as

$$\mu = v^T Q_g v, \qquad \mu' = (v')^T Q_g v'.$$

Then we have

$$\frac{\mu}{\mu'} = \frac{\mu}{\lambda} \cdot \frac{\lambda'}{\mu'} \cdot \frac{\lambda}{\lambda'} \leq \frac{d}{c} \cdot \frac{\lambda}{\lambda'}$$

71

and similarly with the lower bound, giving

$$\frac{c}{d} \cdot \frac{\lambda}{\lambda'} \le \frac{\mu}{\mu'} \le \frac{d}{c} \cdot \frac{\lambda}{\lambda'} \,. \tag{129}$$

This is applied in the following way: suppos the ratio of $\lambda$ to all other eigenvalues of $Q_f$ is large (i.e. $v$ is the most curved direction). Then the ratio of the curvatures $\frac{\mu}{\mu'}$ for $Q_g$ also can't be too small. In our applications, while $c$ can be chosen close to 1, $d$ may be large, so that (129) need not be very informative.

**Definition B.3.** Let $S : \mathbb{R}_{>0} \longrightarrow \mathbb{R}$ be defined by

$$S(u) = -\log u + u - 1 \,.$$

We will prove that $S$ is comparable to $\frac{1}{2}(u-1)^2$.

**Lemma B.4.** Given $0 < c < 1$ and $d > 1$ there exists $\delta > 0$ such that

$$c \cdot \tfrac{1}{2}(u-1)^2 \le S(u) \le d \cdot \tfrac{1}{2}(u-1)^2 \tag{130}$$

whenever $|1 - u| < \delta$.

*Proof.* We compute $S'(u) = -\frac{1}{u} + 1, S''(u) = \frac{1}{u^2}, S^{(3)}(u) = -\frac{2}{u^3}$ so $S(1) = 0, S'(1) = 0, S''(1) = 1$ and so for some $a$ between 1 and $u$

$$S(u) = \tfrac{1}{2}(u-1)^2 - \tfrac{1}{3a^3}(u-1)^3 \,.$$

For a constant $0 < c < 1$ we have $c \cdot \frac{1}{2}(u-1)^2 \le S(u)$ if and only if

$$\tfrac{1}{3a^3}(u-1)^3 \le \tfrac{1}{2}[1-c](u-1)^2$$

which is automatic if $u \le 1$ and for $u > 1$ holds if and only if

$$u \le 1 + \tfrac{3a^3}{2}\left[1 - c\right] \,. \tag{131}$$

Similarly if $1 < d$ we have $S(u) \le d \cdot \frac{1}{2}(u-1)^2$ if and only if

$$\tfrac{1}{2}\left[1 - d\right](u-1)^2 \le \tfrac{1}{3a^3}(u-1)^3$$

which is automatic if $u = 1$ and for $u \ne 1$ is equivalent to

$$1 + \tfrac{3a^3}{2}\left[1 - d\right] \le u \,. \tag{132}$$

It follows that given $c, d$ as described, (130) will hold provided

$$|1 - u| < \delta = \tfrac{3a^3}{2} \min\left\{|1 - c|, |1 - d|\right\} \tag{133}$$

as claimed. $\qquad\square$

Note that we can make the constants $c, d$ as close to 1 as we wish (respectively from below and above) by taking $\delta$ sufficiently small.

## B.1 Geometry of Comparable Functions

Let $U$ be an open neighbourhood of $0 \in \mathbb{R}^d$ and $f : U \longrightarrow \mathbb{R}$ a non-negative and non-constant analytic function with $f(0) = 0$. Let $g : U \longrightarrow \mathbb{R}$ be a comparable analytic function, say with $c, d > 0$ such that

$$cf(x) \leq g(x) \leq df(x), \qquad \forall x \in U. \tag{134}$$

By resolution of singularities (e.g. [46, Theorem 2.3]) there is a triple $(W, U, h)$ where $W$ is an open set in $\mathbb{R}^d$ which contains $0$, $U$ is a $d$-dimensional real analytic manifold, $h : U \longrightarrow W$ is a real analytic map, such that $h$ is proper and with

$$W_0 = \{x \in W \mid f(x) = 0\}, \qquad U_0 = \{u \in U \mid f(h(u)) = 0\}$$

$h$ induces an analytic isomorphism of $U \setminus U_0$ with $W \setminus W_0$. Further, for every point $P \in U_0$ there are local coordinates $u_1, \ldots, u_d$ of $U$ with origin $P$ such that

$$f(h(u)) = u_1^{2k_1} u_2^{2k_2} \cdots u_d^{2k_d}$$

where $k_1, \ldots, k_d$ are non-negative integers, and the Jacobian determinant of $x = h(u)$ is

$$g'(u) = b(u) u_1^{h_1} \cdots u_d^{h_d}$$

where $b(u)$ is nonzero in the local chart and $h_1, \ldots, h_d$ are non-negative integers.

By (134) we have in any such local chart that

$$c u_1^{2k_1} u_2^{2k_2} \cdots u_d^{2k_d} \leq g(h(u)) \leq d u_1^{2k_1} u_2^{2k_2} \cdots u_d^{2k_d}$$

and hence by [47, Theorem 2.6] there exists an open set within the coordinate chart around $P$ such that, on this open set, there is defined a real analytic function $t(u)$ with

$$g(h(u)) = t(u) f(h(u)).$$

That is to say, the ratio $t(u) = g(h(u))/f(h(u))$ is well-defined near $P$ on the resolution, *even though* it may be not be defined on the original space. Since $c \leq t(u) \leq d$ on this neighbourhood of $P$ we have that $t(u)$ is a unit in the ring of analytic functions. In paticular, in the local ring $\mathcal{O}_P^{\mathrm{an}}$ of germs of analytic functions on $U$ at $P$ we have that $f \circ h$ and $g \circ h$ generate the same ideal

$$\langle g(h(u)) \rangle = \langle f(h(u)) \rangle$$

and thus $\mathcal{O}_P^{\mathrm{an}} / \langle f \circ h \rangle \cong \mathcal{O}_P^{\mathrm{an}} / \langle g \circ h \rangle$. Thus any geometric property of $f$ that factors through the resolution is shared by $g$. This includes the real log canonical threshold (i.e. learning coefficient).

## B.2 Functions Comparable to the KL divergence

Let $q(x)$ be a probability distribution on a non-empty finite set $I$ with $q(x) > 0$ for all $x \in I$. Let $Q$ be a non-empty finite set. We suppose given a conditional distribution $q(y|x)$ over $y \in Q$ given $x \in I$ such that $q(y|x) > 0$ for all $y \in Q$. Let $p(y|x, w)$ be a model defined on some $W \subseteq \mathbb{R}^d$ with open interior. Define

$$H(w) = \sum_{x \in I} q(x)H(w, x), \qquad H(w, x) = \sum_{y \in Q} \Big( p(y|x, w) - q(y|x) \Big)^2,$$

$$L(w) = \sum_{x \in I} q(x)L(w, x), \qquad L(w, x) = -\sum_{y \in Q} q(y|x) \log p(y|x, w).$$

$$K(w) = \sum_{x \in I} q(x)K(w, x), \qquad K(w, x) = \sum_{y \in Q} q(y|x) \log \frac{q(y|x)}{p(y|x, w)}.$$

We assume $p(y|x, w)$ is continuous in $w \in W$ and that $p(y|x, w) > 0$ for all $x \in I, y \in Q$ and $w \in W$, so that the above expressions are well-defined.

**Remark B.5.** When the true distribution $q(x, y) = q(y|x)q(x)$ is realisable by a statistical model $q(y|x) = p(y|x, w^*)$ for some $w^*$ (which may be on the boundary of $W$) and we may replace $W$ by an infinitesimal neighbourhood of $w^*$. Since $p(y|x, w)$ is a continuous function of $w$ and there are only finitely many $x \in I$ and $y \in Q$, this means we may assume that given any $\varepsilon > 0$ we have simultaneously $|p(y|x, w) - q(y|x)| < \varepsilon$ for all $x, y$ and $w$. Since $q(y|x) > 0$ this means we may assume $p(y|x, w) > 0$ for all $w$, explaining the reason we adopt this hypothesis.

**Lemma B.6.** Given $0 < c < 1$ and $d > 1$ let $\delta > 0$ be as in (133). If

$$\left| \frac{p(y|x, w)}{q(y|x)} - 1 \right| < \delta \qquad \forall x \in I, y \in Q, w \in W \tag{135}$$

then $\frac{1}{2}H(w)$ is comparable to $K(w)$.

*Proof.* We can apply Lemma B.4 to $u = \frac{p(y|x,w)}{q(y|x)}$ to obtain

$$\frac{d}{2}\left( \frac{p(y|x, w)}{q(y|x)} - 1 \right)^2 \leq -\log \frac{p(y|x, w)}{q(y|x)} + \frac{p(y|x, w)}{q(y|x)} - 1 \leq \frac{d}{2}\left( \frac{p(y|x, w)}{q(y|x)} - 1 \right)^2$$

which we can multiply by $q(y|x)$ and sum over $y \in Q$ to obtain

$$\frac{c}{2}H'(w, x) \leq K(w, x) \leq \frac{d}{2}H'(w, x) \tag{136}$$

where

$$H'(w, x) = \sum_{y \in Q} q(y|x)\left( \frac{p(y|x, w)}{q(y|x)} - 1 \right)^2.$$

Thus $K(w, x)$ is comparable to $H'(w, x)$. Now

$$H'(w, x) = \sum_{y \in Q} q(y|x)^{-1} \Big( p(y|x, w) - q(y|x) \Big)^2$$

and if we let $C_x = \inf_{y \in Q} q(y|x)^{-1}$ and $D_x = \sup_{y \in Q} q(y|x)^{-1}$ then

$$C_x H(w, x) \leq H'(w, x) \leq D_x H(w, x)$$

so $H'(w, x)$ is comparable to $H(w, x)$. By transitivity $H(w, x)$ is comparable to $K(w, x)$, or more explicitly we have for all $x, w$

$$\frac{c\,C_x}{2} H(w, x) \leq K(w, x) \leq \frac{d\,D_x}{2} H(w, x)\,.$$

Multiplying through and taking the sum over $x \in I$

$$cC \cdot \tfrac{1}{2} H(w) \leq K(w) \leq dD \cdot \tfrac{1}{2} H(w)$$

where $C = \inf_{x \in I} C_x$, $D = \sup_{x \in I} D_x$ as desired. Of course if $K(w)$ is comparable to $\frac{1}{2} H(w)$ it is comparable to $H(w)$ but since we control $c, C, d, D$ this is more natural. $\square$

**Remark B.7.** In the proof note that $q(y|x) \leq 1$ so $q(y|x)^{-1} \geq 1$ and so $1 \leq C_x$ and thus $1 \leq C$. Hence, possibly by shrinking $\delta$ we can take the constant $cC$ in the lower bound as close as we like to 1. The upper bound is less trivial to deal with, because if there are $y$ for which $q(y|x)$ is very small, $D$ is large. For this reason the comparability of $K(w)$ to a sum of squared differences is not trivial when $q$ does not have full support. In Section B.3 we explain one way to deal with this.

**Remark B.8.** As in Remark B.5, we typically arrange for (135) to hold by shrinking $W$ to a small neighbourhood of a true parameter.

**Remark B.9.** Recall that $K(w) = L(w) - C$ where $C$ is the entropy of the true distribution (which does not depend on $w$). Thus $L(w)$ is comparable to $\frac{1}{2} H(w) + C$. If we assume we are working in a neighbourhood of a true parameter $w^*$ so that $K(w^*) = 0$ then $L(w^*) = C$ so that $L(w)$ is comparable to $\frac{1}{2} H(w) + L(w^*)$.

## B.3   Moving Away From the Boundary

We define for $\mu > 0$ the interpolation between a probability distribution and the barycenter $\mathbf{b} = \sum_{q \in Q} \frac{1}{|Q|} q$ of the probability simplex by

$$\varepsilon_\mu : \Delta Q \longrightarrow \Delta Q\,,$$
$$\varepsilon_\mu(\mathbf{x}) = (1 - \mu)\mathbf{x} + \mu \mathbf{b}$$

where we write probability distributions as vectors in $\Delta Q \subseteq \mathbb{R}^Q$ and denote them with bold letters. Note that

$$\|\varepsilon_\mu(\mathbf{x}) - \varepsilon_\mu(\mathbf{y})\|^2 = \|(1-\mu)(\mathbf{x}-\mathbf{y})\|^2 = (1-\mu)^2\|\mathbf{x}-\mathbf{y}\|^2 \,.$$

Let $I, Q$ and $q(x)$ be as in Section B.2. We assume given a conditional distribution $q(y|x)$ but we *do not* assume that $q(y|x) > 0$ for all $y \in Q$. Nonetheless the distribution

$$q_\mu(y|x) = \varepsilon_\mu q(y|x) \in \Delta Q$$

does have this property. Similarly, let $p(y|x, w)$ be a model parametrised by $w \in W$ which is continuous, but not necessarily positive on all $y$. We define

$$p_\mu(y|x, w) = \varepsilon_\mu p(y|x, w)$$

which once again, is positive on all $y$. Assuming the hypothesis of Lemma B.6 for $q_\mu, p_\mu$ we obtain that the KL divergence $K_\mu(w)$ between $q_\mu$ and $p_\mu$ is comparable to

$$\begin{aligned}
H_\mu(w) &= \sum_{x \in I} q(x)\|\varepsilon_\mu p(y|x, w) - \varepsilon_\mu q(y|x)\|^2 \\
&= \sum_{x \in I} q(x)(1-\mu)^2 \|p(y|x, w) - q(y|x)\|^2 \\
&= (1-\mu)^2 H(w) \,.
\end{aligned}$$

**Remark B.10.** The idea of moving distributions $p$ off the boundary in the context of program synthesis was introduced in [14], but the elegant trick of applying it to both $p, q$ is due to Waring [45].

## C  Details of the Step Function of $\mathcal{U}$

The proof $_h$relstep of Proposition [16, 5.5] encodes one step of a Turing machine in linear logic. In Section 5.1 we considered simulated Turing machines with tape alphabet given by some fixed set $\Sigma$, with set of states some fixed set $Q$. Here we fix these sets to be as follows, where $n, m \geq 1$:

$$\Sigma = \{0, \ldots, n-1\}, Q = \{0, \ldots, m-1\}, D = \{0, 1, 2\} = \{\texttt{Left}, \texttt{Stay}, \texttt{Right}\}. \quad (137)$$

and we consider a staged pseudo-UTM $\mathcal{U}$. There are $N = nm$ tuples associated to the transition function of $M$. The aim of this section is to adapt $_h$relstep, for $h > 0$, to a proof $_h\mathcal{U}$relstep sufficiently capturing and simulating the computation of $\mathcal{U}$ for a single simulated step of $M$ (which amounts to $10N + 6$ steps of $\mathcal{U}$).

One way to approach this would be to come up with a generic encoding of Turing machines with $k > 0$ tapes in linear logic and then specialise to the particular case of $\mathcal{U}$. This is possible, but we simplify things by making use of the following properties of $\mathcal{U}$:

(i) The description tape, staging tape, and state tape are of determined, finite length.

(ii) The head position of the description tape depends only on the time step $\mu$ of $\mathcal{U}$.

(iii) The head positions of the staging tape depends only on the state of $\mathcal{U}$.

(iv) The head position of the state tape never changes.

(v) The symbols and states of the simulated machine can be encoded in single tape squares of the $\mathcal{U}$.

Let $\mu, h \geq 0$ be an integers with $\mu \leq 10N + 6$. We define a type ${}_{h}^{\mu}\underline{\mathcal{U}\mathbf{tur}}$ along with a proof

$$
{}_{h}^{\mu}\underline{\mathcal{U}\text{relstep}} : {}_{h}^{\mu}\underline{\mathcal{U}\mathbf{tur}} \vdash {}_{h}^{\mu+1}\underline{\mathcal{U}\mathbf{tur}} \tag{138}
$$

encoding a single step of $\mathcal{U}$ at position $\mu$ within the overall cycle of the staged pseudo-UTM (see Section 5.1). By properties (i)-(v), we can reduce to a finite set of squares which are used in a single, fixed timestep of $\mathcal{U}$. For each of these squares we will construct a plain proof and another plain proof encoding the updates of the states of $\mathcal{U}$. Following [16], the proof ${}_{h}^{\mu}\underline{\mathcal{U}\text{relstep}}$ will then be given by promoting these proofs, taking a tensor product, and performing contractions. We construct the proof associated to these squares one tape at a time.

In the following the base type $A$ is chosen so that all required denotations are linearly independent; this is possible by [16, Appendix A]. Nothing depends on the choice of $A$ and we generally drop it from the notation and write **bool** for $\mathbf{bool}_A$.

We will make extensive use of the following two lemmas:

**Lemma C.1.** Given $k \geq 1$ and any function $f : \{0, ..., n-1\}^k \to \{0, ..., m-1\}$, there exists a proof $F$ of $k\,{}_n\mathbf{bool} \vdash {}_m\mathbf{bool}$ which encodes $f$.

*Proof.* The proof is by induction on $k$, with $k = 1$ being Lemma [16, 4.7]. Assuming the lemma holds for $k-1$ we for each $0 \leq z \leq n-1$ let $F_z$ be the proof of $(k-1)\,{}_n\mathbf{bool} \vdash {}_m\mathbf{bool}$ encoding the function

$$
f(-, z) : \{0, ..., n-1\}^{k-1} \to \{0, ..., m-1\} \,.
$$

Then the proof

$$
\cfrac{\cfrac{\cfrac{\begin{matrix} F_0 \\ \vdots \end{matrix}}{A^m, (k-1)\,{}_n\mathbf{bool} \vdash A} \quad ... \quad \cfrac{\begin{matrix} F_{n-1} \\ \vdots \end{matrix}}{A^m, (k-1)\,{}_n\mathbf{bool} \vdash A}}{A^m, (k-1)\,{}_n\mathbf{bool} \vdash A^n} {\scriptstyle \&R} \quad \cfrac{}{A \vdash A}}{\cfrac{A^m, (k-1)\,{}_n\mathbf{bool}, {}_n\mathbf{bool} \vdash A}{(k-1)\,{}_n\mathbf{bool}, {}_n\mathbf{bool} \vdash {}_m\mathbf{bool}} {\scriptstyle \multimap R}} {\scriptstyle \multimap L}
$$

77

encodes $f$. □

Similarly it follows that:

**Lemma C.2.** Let $n_1, n_2, m \geq 1$ and consider a function

$$f : \{0, \ldots, n_1 - 1\} \times \{0, \ldots, n_2 - 1\} \longrightarrow \{0, \ldots, m - 1\} \tag{139}$$

there exists a proof $F$ of ${}_{n_1}\mathbf{bool}, {}_{n_2}\mathbf{bool} \vdash {}_m\mathbf{bool}$ which encodes $f$.

## C.1   The description tape

The description tape of $\mathcal{U}$ is

$$X\sigma_1 q_1 \sigma'_1 q'_1 d_1 \ldots \sigma_N q_N \sigma'_N q'_N d_N X \tag{140}$$

The natural orders $0 < \ldots < n - 1, 0 < \ldots < m - 1$ on $\Sigma, Q$ respectively induce the lexicographical order on the set $\Sigma \times Q$ which means the information of the description tape can be compressed to tuples of size 3 instead of size 5 as follows:

$$\sigma'_1 q'_1 d_1 \ldots \sigma'_N q'_N d_N \tag{141}$$

where we have dropped the $X$ at the beginning and at the end of the tape as these are only used in the definition of $\mathcal{U}$ as a Turing machine and will not be used in the encoding. Here $\sigma'_i \in \Sigma$, $q'_i \in Q$, $d_i \in D$ for $1 \leq i \leq N$. We use the type **descr** and proofs $\mathcal{P}^i_{\mathrm{dscr}}$ introduced in Section 5.2.

For each $i = 1, \ldots, 3N$ we define a proof $S^{\mathrm{dscr}}_{\mu,i}$ of the following sequent:

$$S^{\mathrm{dscr}}_{\mu,i} : \begin{cases} {}_n\mathbf{bool} \vdash {}_n\mathbf{bool}, & i = 1 \bmod 3, \\ {}_m\mathbf{bool} \vdash {}_m\mathbf{bool}, & i = 2 \bmod 3, \\ {}_3\mathbf{bool} \vdash {}_3\mathbf{bool}, & i = 0 \bmod 3, \end{cases} \tag{142}$$

Each of these consist of an Axiom-rule encoding the identity function. For example, $S^{\mathrm{dscr}}_{\mu,1}$ is the following proof:

$$\frac{}{{}_n\mathbf{bool} \vdash {}_n\mathbf{bool}} \mathrm{Ax}$$

We define the proof associated to the description tape as the tensor of all of these:

$$S^{\mathrm{dscr}}_\mu = \bigotimes_{i=1}^{3N} S^{\mathrm{dscr}}_{\mu,i} : \mathbf{dscr} \vdash \mathbf{dscr}. \tag{143}$$

Note that this proof is independent of $\mu$.

## C.2 The staging tape

We label these squares $s_0, s_1, s_2$. If $\Sigma^i_{\text{stage}}$ denotes the set of symbols which may appear on square $s_i$ for $i = 0, 1, 2$, then:

$$\Sigma^i_{\text{stage}} = \begin{cases} \Sigma \cup \{X\}, & i = 0, \\ Q \cup \{X\}, & i = 1, \\ D \cup \{X\}, & i = 2. \end{cases} \tag{144}$$

We introduce the following type:

$$\mathbf{stg} = \underbrace{!_{n+1}\mathbf{bool}}_{\sigma'} \otimes \underbrace{!_{m+1}\mathbf{bool}}_{q'} \otimes \underbrace{!_4\mathbf{bool}}_{d} \tag{145}$$

where we have decorated each factor with the kind of information that is written to the staging tape during the execution of the UTM. We introduce sets of proofs:

$$\mathcal{P}^{\text{stg}}_i = \begin{cases} \{[\![\underline{j}]\!]\}_{j=0,\ldots,n} \subseteq [\![!_{n+1}\mathbf{bool}]\!], & i = 0, \\ \{[\![\underline{j}]\!]\}_{j=0,\ldots,m} \subseteq [\![!_{m+1}\mathbf{bool}]\!], & i = 1, \\ \{[\![\underline{j}]\!]\}_{0 \leq j \leq 3} \subseteq [\![!_4\mathbf{bool}]\!], & i = 2. \end{cases} \tag{146}$$

For each $i = 0, 1, 2$ we define a proof $S^{\text{stage}}_{\mu,i}$. By convention $X$ is encoded as $\underline{n}, \underline{m}, \underline{3}$.

We show the details for $i = 0$ and leave $i = 1, 2$ to the reader. Say $0 \leq \mu \leq 5N$. If $\mu \in \{0, 1, 3, 4\}$ mod 5 then $S^{\text{stage}}_{\mu,0}$ consists of a single Axiom-rule. Now say $\mu = 2$ mod 5. Recall that we have fixed a Turing machine $M$ with associated code $[M]$. Then $S^{\text{stage}}_{\mu,0}$ encodes the following function:

$$W^{\mathbf{code}} \times \Sigma \times Q_{\text{UTM}} \longrightarrow \Sigma$$

$$([M], \sigma, \varphi) \longmapsto \begin{cases} [M]_{\hat{\mu}}, & \varphi = \text{copySymbol}, \\ \sigma, & \varphi \neq \text{copySymbol} \end{cases}$$

where $\hat{\mu}$ is the entry of the description tape corresponding to the tuple in turn corresponding to $\mu$. This is a proof

$$S^{\text{stage}}_{\mu,0} : \mathbf{dscr} \otimes !_{n+1}\mathbf{bool} \otimes !_{13}\mathbf{bool} \vdash !_{n+1}\mathbf{bool}. \tag{147}$$

Existence follows from Lemmas C.1, C.2.

If $\mu = 5N + 1$ mod $P$ then $S^{\text{stage}}_{\mu,0}$ encodes the constant function mapping everything to $X$. For $5N + 2 \leq \mu \leq 10N + 6$ we define $S^{\text{stage}}_{\mu,0}$ to consist of an Axiom-rule. We define the proof associated to the staging tape as the tensor of all of these:

$$S^{\text{stg}}_\mu = S^{\text{stg}}_{\mu,0} \otimes S^{\text{stg}}_{\mu,1} \otimes S^{\text{stg}}_{\mu,2}. \tag{148}$$

This is a proof of the following sequents depending on $\mu$:

$$\begin{cases} \mathbf{stg} \vdash \mathbf{stg}, & \mu \leq 5N, \mu = 0, 1 \text{ mod } 5, \\ \mathbf{dscr} \otimes \mathbf{stg} \otimes !_{13}\mathbf{bool} \vdash \mathbf{stg}, & \mu \leq 5N, \mu = 2, 3, 4 \text{ mod } 5, \\ \mathbf{stg} \vdash \mathbf{stg}, & \mu \geq 5N \end{cases} \tag{149}$$

## C.3   The state tape

We consider
$$\mathcal{P}^{\text{state}} = \{[\![\underline{j}]\!]\}_{j=0,\dots,m-1} \subseteq [\![_m\mathbf{bool}]\!]. \tag{150}$$

We define a proof $S_\mu^{\text{state}}$. If $\mu \neq 5N + 2$ then $S_\mu^{\text{state}}$ consists of a single Axiom-rule. If $\mu = 5N + 2$ then we encode the function

$$(Q \cup \{X\}) \times Q \longrightarrow Q$$

$$(q, q') \longmapsto \begin{cases} q, & q \neq X, \\ q', & q = X. \end{cases}$$

This is a proof of the following sequents depending on $\mu$:

$$\begin{cases} !_m\mathbf{bool} \vdash !_m\mathbf{bool}, & \mu \neq 5N + 2, \\ !_{m+1}\mathbf{bool} \otimes !_m\mathbf{bool} \vdash !_m\mathbf{bool}, & \mu = 5N + 2. \end{cases} \tag{151}$$

## C.4   The working tape

Each square on the working tape is occupied by a symbol in $\Sigma$ and so for all $i = -h - 1, \dots, h + 1$ we take
$$\mathcal{P}^{\text{work}}_{h,i} = \{[\![\underline{j}]\!]\}_{j=0,\dots,n-1} \subseteq [\![_n\mathbf{bool}]\!]. \tag{152}$$

For $\mu \leq 5N + 2$ and $i = -h, \dots, h$ we define a proof $S^{\text{work}}_{\mu,i}$. For $\mu > 5N + 2$ and $i = -h - 1, \dots, h + 1$ we define a proof $S^{\text{work}}_{\mu,i}$. Say $0 \leq \mu \leq 5N + 1$ and $i$ is any value, then $S^{\text{work}}_{\mu,i}$ consists of just an Axiom-rule. Similarly for $5N + 1 < \mu < 10N + 6$ and $i$ any value. Similarly also if $\mu \in \{5N + 2, 5N + 3\}$ and $i \neq 0$. Now say $\mu = 5N + 2$ and $i = 0$. Then we encode the function

$$(\Sigma \cup \{X\}) \times \Sigma \longrightarrow \Sigma$$

$$(\sigma, \sigma') \longmapsto \begin{cases} \sigma, & \sigma \neq X, \\ \sigma' & \sigma = X. \end{cases}$$

Now say $\mu = 5N + 3$ and $i \in \{-h + 1, \dots, h - 1\}$. Then we need to encode the function

$$(D \cup \{X\}) \times \Sigma \times \Sigma \times \Sigma \longrightarrow \Sigma$$

$$(d, \sigma, \sigma', \sigma'') \longmapsto \begin{cases} \sigma, & d = \text{Left}, \\ \sigma', & d \in \{\text{Stay}, X\}, \\ \sigma'', & d = \text{Right}. \end{cases}$$

If $i = -h$ then we need to encode the function:

$$(D \cup \{X\}) \times \Sigma \times \Sigma \longrightarrow \Sigma$$

$$(d, \sigma, \sigma') \longmapsto \begin{cases} 0, & d = \text{Left}, \\ \sigma, & d \in \{\text{Stay}, X\}, \\ \sigma', & d = \text{Right}. \end{cases}$$

and similarly for $i = h$. Finally, if $i = -h - 1$ then we need to encode the function:

$$(D \cup \{X\}) \times \Sigma \longrightarrow \Sigma$$

$$(d, \sigma) \longmapsto \begin{cases} 0, & d = \text{Left}, \\ 0, & d \in \{\text{Stay}, X\}, \\ \sigma, & d = \text{Right}. \end{cases}$$

The proof associated to the working tape is the tensor of all these (with promotions and contractions)

$$\begin{cases} S_\mu^{\text{work}} = \bigotimes_{i=-h}^{h} S_{\mu,i}^{\text{work}}, & \mu \leq 5N + 2, \\ S_\mu^{\text{work}} = \bigotimes_{i=-h-1}^{h+1} S_{\mu,i}^{\text{work}}, & \mu = 5N + 3, \\ S_\mu^{\text{work}} = \bigotimes_{i=-h-1}^{h+1} S_{\mu,i}^{\text{work}}, & \mu \geq 5N + 4. \end{cases} \tag{153}$$

which is of the following type:

$$\begin{cases} !_n\mathbf{bool}^{\otimes 2h+1} \vdash !_n\mathbf{bool}^{\otimes 2h+1}, & \mu < 5N + 2, \\ !_{n+1}\mathbf{bool} \otimes !_n\mathbf{bool}^{\otimes 2h+1} \vdash {}_n\mathbf{bool}^{\otimes 2h+1}, & \mu = 5N + 2, \\ !_n\mathbf{bool}^{\otimes 2h+1} \vdash !_n\mathbf{bool}^{\otimes 2h+1}, & \mu = 5N + 3, \\ !_{n+1}\mathbf{bool} \otimes !_n\mathbf{bool}^{\otimes 2h+3} \vdash !_n\mathbf{bool}^{\otimes 2h+3}, & \mu \geq 5N + 4. \end{cases} \tag{154}$$

## C.5 The state of $\mathcal{U}$

There are 13 states of $\mathcal{U}$, so we consider

$$\mathcal{P}_i^{\text{state}} = \{[\![\underline{j}]\!]\}_{j=0,\dots,12} \subseteq [\![_{13}\mathbf{bool}]\!]. \tag{155}$$

We define a proof $S_\mu^{\mathcal{U}\text{state}}$.

Say $\mu \leq 5N$. Say $\mu = 1 \bmod 5$. Then $S_\mu^{\mathcal{U}\text{state}}$ encodes the function

$$\Sigma \longrightarrow Q_{\text{UTM}}$$

$$\sigma \longmapsto \begin{cases} \text{compState}, & \sigma = \sigma_{\hat{\mu}}, \\ \neg\text{compState}, & \sigma \neq \sigma_{\hat{\mu}}. \end{cases}$$

where $\hat{\mu}$ is the entry of $[M]$ associated to the timestep $\mu$.

Say $\mu = 2 \bmod 5$. Then $S_\mu^{\mathcal{U}\text{state}}$ encodes the function

$$Q \longrightarrow Q_{\text{UTM}}$$

$$q \longmapsto \begin{cases} \text{copySymbol}, & q = q_{\hat{\mu}}, \\ \neg\text{copySymbol}, & q \neq q_{\hat{\mu}}. \end{cases}$$

Now say $\mu > 5N$. Then each $S_\mu^{\mathcal{U}\text{state}}$ encodes a constant function as the state of $\mathcal{U}$ is determined completely by the timestep $\mu$. For example, if $\mu = 5N + 1$ then $\varphi = $ updateSymbol, etc. This is a proof

$$
\begin{cases}
!_n\mathbf{bool} \vdash !_{13}\mathbf{bool}, & \mu \leq 5N, \mu = 1 \bmod 5, \\
!_m\mathbf{bool} \vdash !_{13}\mathbf{bool}, & \mu \leq 5N, \mu = 2 \bmod 5, \\
!_{13}\mathbf{bool} \vdash !_{13}\mathbf{bool}, & \mu \leq 5N, \mu = 3, 4, 0 \bmod 5, \\
\vdash !_{13}\mathbf{bool}, & \mu > 5N.
\end{cases}
\tag{156}
$$

## C.6   The type ${}_h^\mu\underline{\mathcal{U}\mathbf{tur}}$

First we define the type such that proofs of that type include encodings of the configuration of the staged pseudo-UTM at various steps within its cycle.

**Definition C.3.** Define for $h \geq 0$ and $0 \leq \mu \leq 10N + 6$

$$
{}_h^\mu\underline{\mathcal{U}\mathbf{tur}} = \mathbf{dscr} \otimes \mathbf{stg} \otimes !_m\mathbf{bool} \otimes !_n\mathbf{bool}^{\otimes h'} \otimes !_{13}\mathbf{bool}
\tag{157}
$$

where

$$
h' = \begin{cases}
2h + 1, & \mu \neq 5N + 4, \\
2h + 3, & \mu = 5N + 4.
\end{cases}
\tag{158}
$$

Next we define the proof which encodes the mapping of configurations of the staged pseudo-UTM at position $\mu$ to $\mu + 1$ within the cycle.

**Definition C.4.** Define for $h \geq 0$ and $0 \leq \mu < 10N + 6$

$$
{}_h^\mu\underline{\mathcal{U}}\text{relstep} : {}_h^\mu\underline{\mathcal{U}\mathbf{tur}} \vdash {}_h^{\mu+1}\underline{\mathcal{U}\mathbf{tur}}.
\tag{159}
$$

as follows. We first take the promotion of each of the proofs associated to the tapes and also the proof associated to the state of $\mathcal{U}$:

$$
\mathrm{Prom}(S_\mu^{\text{dscr}}), \quad \mathrm{Prom}(S_\mu^{\text{stg}}), \quad \mathrm{Prom}(S_\mu^{\text{state}}), \quad \mathrm{Prom}(S_\mu^{\text{work}}), \quad \mathrm{Prom}(S_\mu^{\mathcal{U}\text{state}})
\tag{160}
$$

Then we tensor all of these together, and append appropriate Exchange- and Contraction-rules to arrive at

$$
\begin{aligned}
{}_h^\mu\underline{\mathcal{U}}\text{relstep} = \mathrm{Ctr} \Big( \mathrm{Ex} \Big( &\mathrm{Prom}(S_\mu^{\text{dscr}}) \otimes \mathrm{Prom}(S_\mu^{\text{stg}}) \\
&\otimes \mathrm{Prom}(S_\mu^{\text{state}}) \otimes \mathrm{Prom}(S_\mu^{\text{work}}) \otimes \mathrm{Prom}(S_\mu^{\mathcal{U}\text{state}}) \Big) \Big).
\end{aligned}
$$

Finally, we cut these proofs together for $\mu = 0, \mu = 1, \ldots, \mu = 10N + 5$ to get a proof computing a single cycle of the UTM.

**Definition C.5.** We define

$$_h\underline{\mathcal{U}\text{relstep}} = {}_h^{10N+5}\underline{\mathcal{U}\text{relstep}} \mid \ldots \mid {}_h^1\underline{\mathcal{U}\text{relstep}} \mid {}_h^0\underline{\mathcal{U}\text{relstep}} \tag{161}$$

and for any integer $t \geq 1$

$$_h\underline{\mathcal{U}\text{relstep}}^t = \underbrace{{}_{h+t}\underline{\mathcal{U}\text{relstep}} \mid \ldots \mid {}_{h+1}\underline{\mathcal{U}\text{relstep}} \mid {}_h\underline{\mathcal{U}\text{relstep}}}_{t \text{ proofs and } t-1 \text{ cuts}} \tag{162}$$

where $\alpha \mid \beta$ denotes the cut of two proofs $\alpha, \beta$.

**Remark C.6.** We recall that the cut of component-wise plain proofs is component-wise plain. Hence $_h\underline{\mathcal{U}\text{relstep}}^t$ is component-wise plain. If we perform cut-elimination on this proof we obtain a component-wise plain proof which is the tensor product of a family of proofs indexed by the active squares of $\mathcal{U}$. If we consider the plain proof associated to the only active square of the state tape, then we obtain a plain proof $\psi$ whose naive probabilistic extension is

$$\Delta\psi = \Delta\text{step}^t \tag{163}$$

of the body of the paper.

# D    Computation Paths for the UTM

We continue the treatment of computation paths for $\mathcal{U}$ from Section 6.1. The DGM for one simulated step of $T$ is shown in Figure 5. We perform the following simplification: in general, given a DGM, any vertex $B$ for which there exists a unique pair of edges $e_1, e_2$ such that $v$ is the target of $e_1$ and the source of $e_2$:

$$A \xrightarrow{e_1} B \xrightarrow{e_2} C \tag{164}$$

has conditional dependence relation given by:

$$\Pr(A, B, C) = \Pr(A|B)\Pr(B|C)\Pr(C). \tag{165}$$

If $\Pr(B|C) = \delta_{B=C}$ then (165) simplifies:

$$\begin{aligned}
\Pr(A|B)\Pr(B|C)\Pr(C) &= \Pr(A|B)\delta_{B=C}\Pr(C) \\
&= \Pr(A|C)\Pr(C).
\end{aligned}$$

In other words, in any DGM, a subgraph of the form (164) with $\Pr(B|C) = \delta_{B=C}$ can be replaced by the following:

$$A \longrightarrow C.$$

In this case we say that we have *collapsed* the DGM.

## D.1 Symbol write error

First we consider a symbol write error. Applying the above technique of collapsing the DGM we obtain the equivalent model shown in Figure 11, where in this and subsequent diagrams a dashed arrow means a collection of $N$ arrows and a squiggly arrow means a collection of $2N$ arrows.
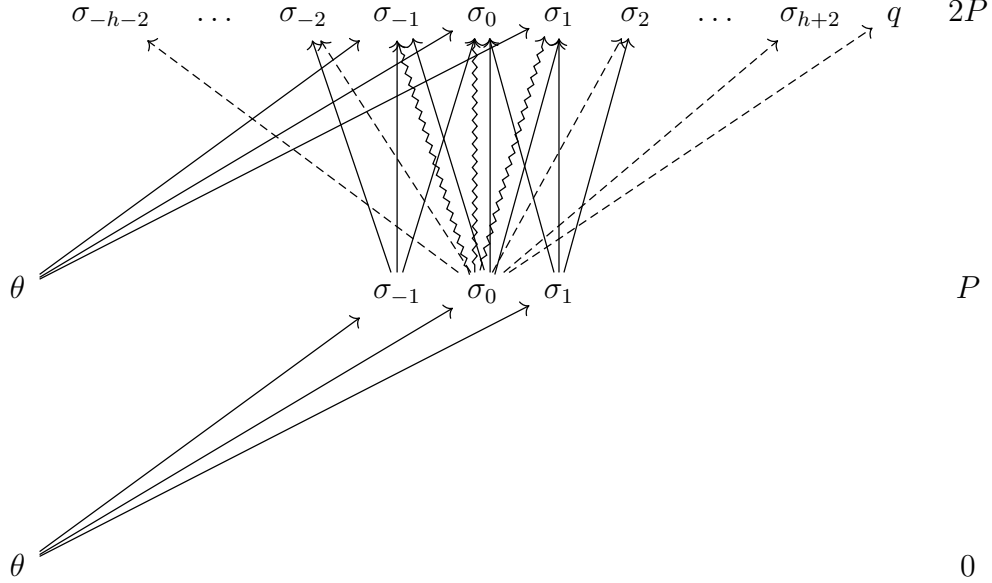


Figure 11: A collapsed version of two stacked diagrams as in Figure 5 (that is, two simulated timesteps of a TM, considering a possible error in the symbol to write square of the description tape) showing only paths that begin at $\theta$.

If we consider the simulated steps from $P = 1$ to $P = 3$ we obtain Figure 12.

The step from $P = 2$ to $P = 3$ gives the general pattern. The key point is that at step $P = 2$, there is now uncertainty in all of the squares on the working tape, and the square on the state tape. Except for the growth of the working tape, there is no further introduction of uncertainty, so we now have the general picture. If we only consider the paths which end at $q$ in the fourth simulated step, we obtain the equivalent model shown in Figure 13.

## D.2 State write error

Now we consider a state write error. This situation is similar to the above, but at time step $P = 1$ only the state square has uncertainty on it, not the $\sigma_0$ square on the working tape. However, at step $P = 2$, there is the same set of squares with uncertainty as in the symbol write error case. We obtain the DGM for $P = 1$ to $P = 3$ as shown in Figure 14.
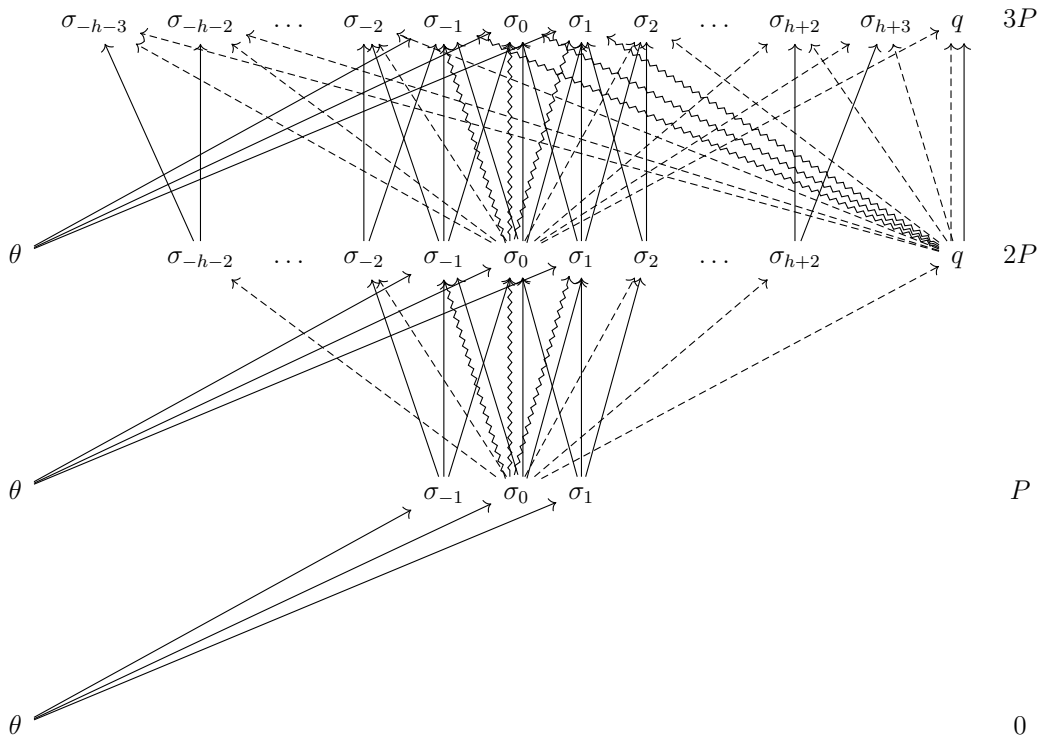
Figure 12: A collapsed version of three stacked diagrams as in Figure 5 (that is, three simulated timesteps of a TM, considering a possible error in the symbol to write square of the description tape) showing only paths that begin at $\theta$.
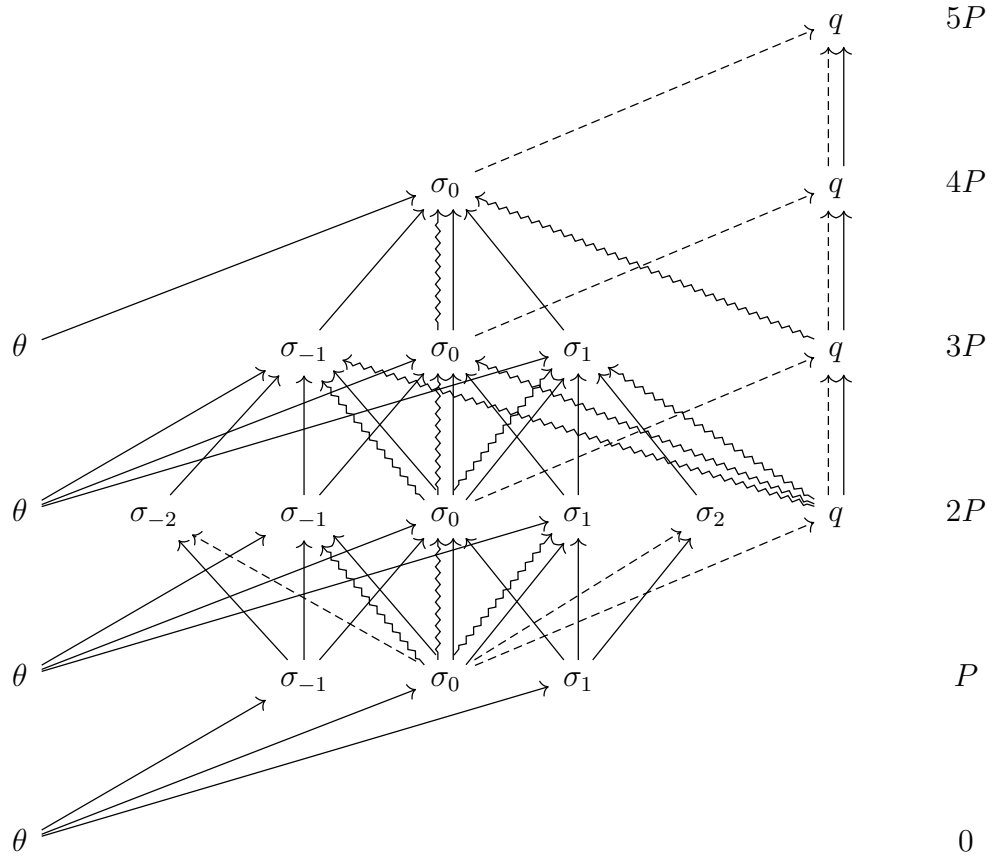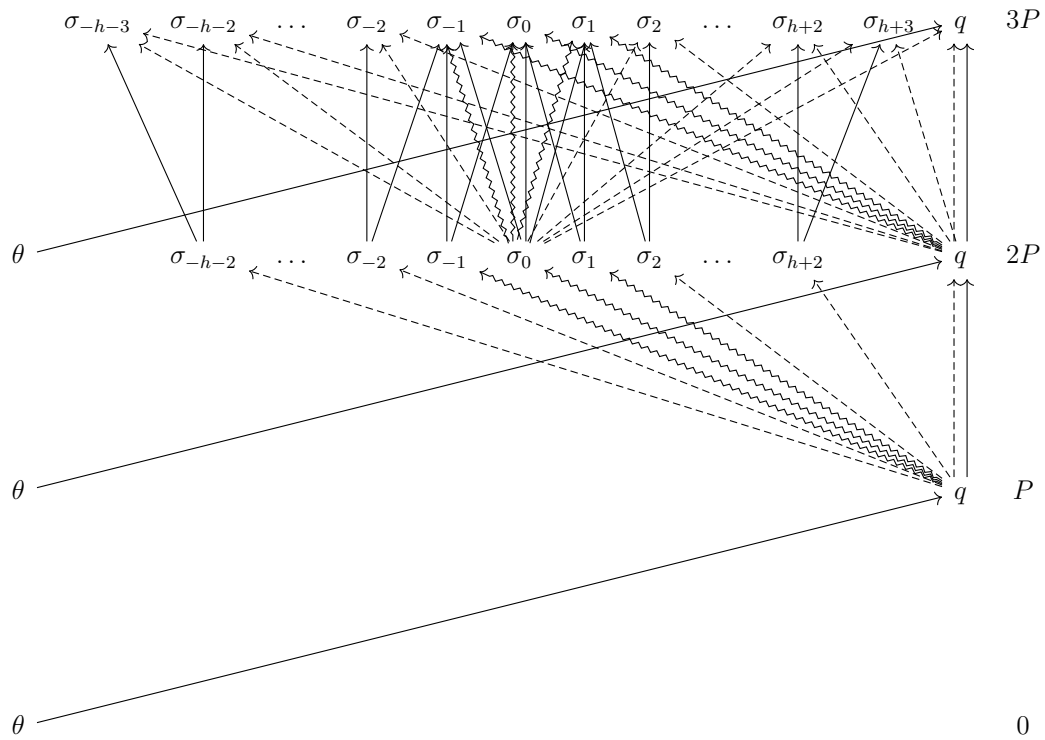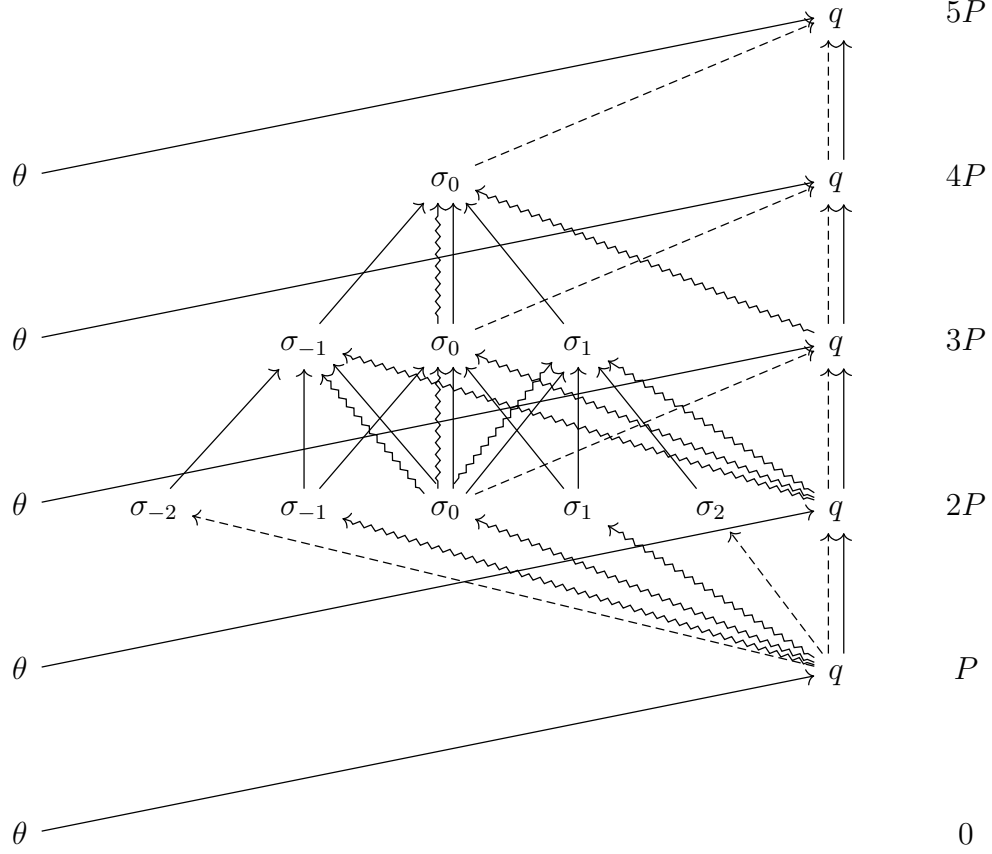
Figure 13: A collapsed version of five stacked diagrams as in Figure 5 (that is, five simulated timesteps of a TM, considering a possible error in the symbol to write square of the description tape) showing only paths that begin at $\theta$ and end at $q$.

Figure 14: A collapsed version of three stacked diagrams as in Figure 5 (that is, three simulated timesteps of a TM, considering a possible error in the state to transition to on the description tape) showing only paths that begin at $\theta$.

If we consider 5 simulated steps and only paths which end at $q$, we obtain Figure 15.



Figure 15: A collapsed version of five stacked diagrams as in Figure 5 (that is, five simulated timesteps of a TM, considering a possible error in the state to transition to on the description tape) showing only paths that begin at $\theta$ and end at $q$.

## D.3 Direction write error

Finally, we consider a direction write error. The simplified DGM for two simulated steps of $T$ is given in Figure 16.

If we consider 5 simulated steps and only paths which end at $q$, we obtain Figure 17.

**Remark D.1.** In all three cases, the trees from time step $2P$ and above are identical except for the arrows with source $\theta$.
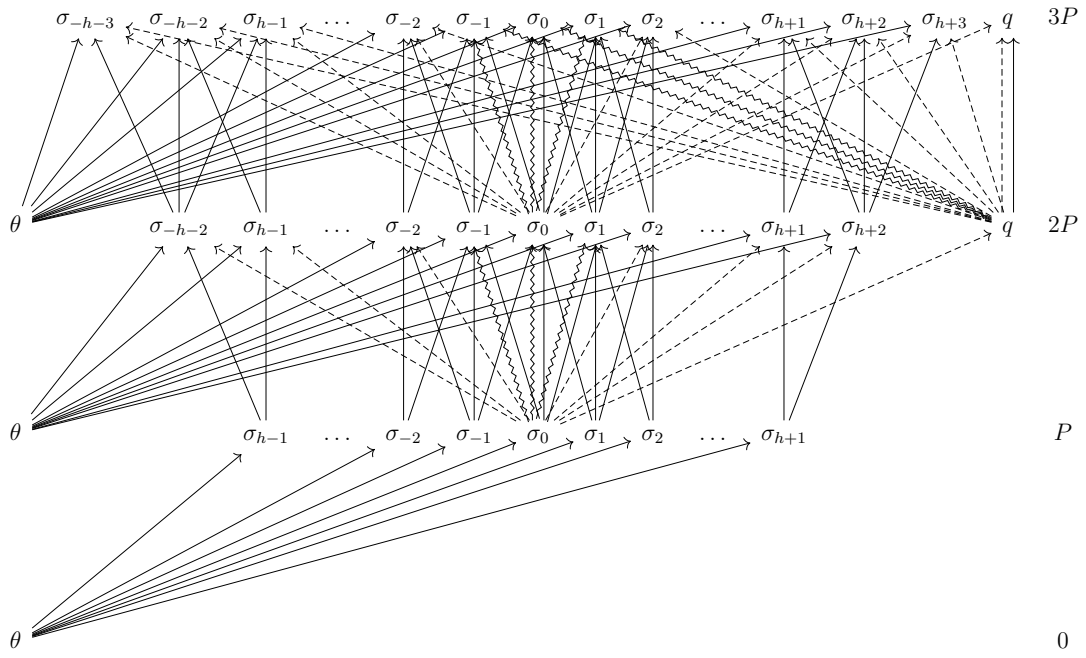
Figure 16: A collapsed version of three stacked diagrams as in Figure 5 (that is, three simulated timesteps of a TM, considering a possible error in the direction to move) showing only paths that begin at $\theta$.
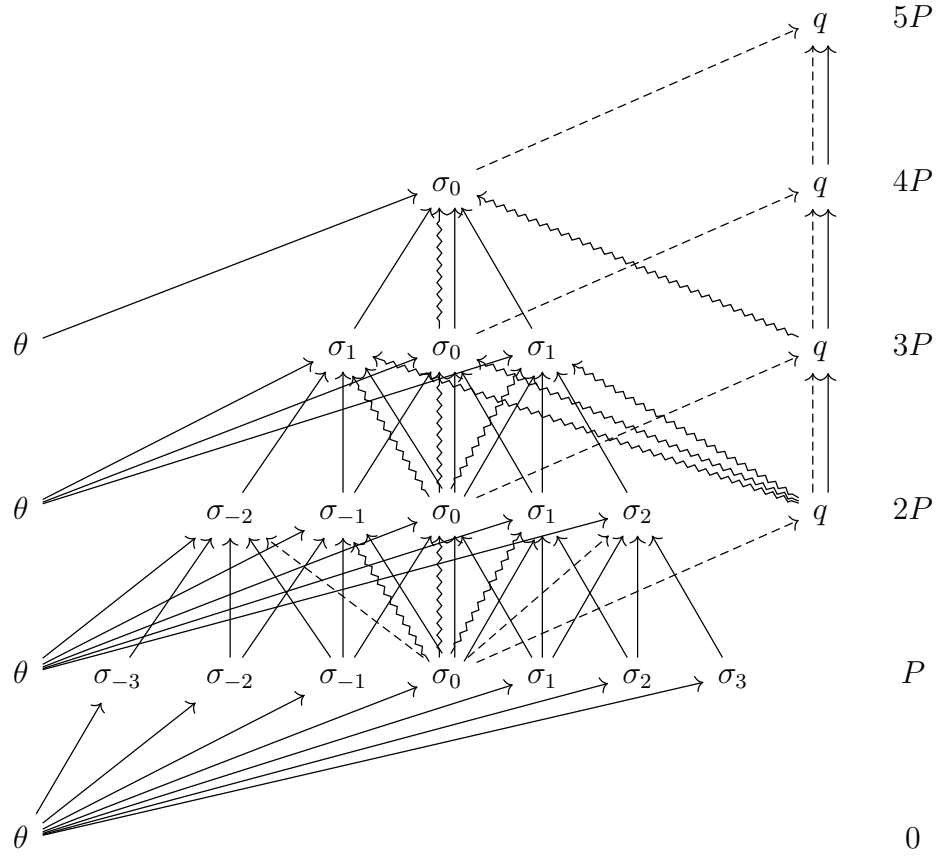
Figure 17: A collapsed version of five stacked diagrams as in Figure 5 (that is, five simulated timesteps of a TM, considering a possible error in the direction to move) showing only paths that begin at $\theta$ and end at $q$.