TOWARDS COMMON-SENSE REASONING VIA CONDITIONAL SIMULATION: LEGACIES OF TURING IN ARTIFICIAL INTELLIGENCE

CAMERON E. FREER, DANIEL M. ROY, AND JOSHUA B. TENENBAUM

Abstract. The problem of replicating the flexibility of human common-sense reasoning has captured the imagination of computer scientists since the early days of Alan Turing's foundational work on computation and the philosophy of artificial intelligence. In the intervening years, the idea of cognition as computation has emerged as a fundamental tenet of Artificial Intelligence (AI) and cognitive science. But what kind of computation is cognition?

We describe a computational formalism centered around a probabilistic Turing machine called QUERY, which captures the operation of probabilistic conditioning via *conditional simulation*. Through several examples and analyses, we demonstrate how the QUERY abstraction can be used to cast common-sense reasoning as probabilistic inference in a statistical model of our observations and the uncertain structure of the world that generated that experience. This formulation is a recent synthesis of several research programs in AI and cognitive science, but it also represents a surprising convergence of several of Turing's pioneering insights in AI, the foundations of computation, and statistics.

1. Introduction

In his landmark paper *Computing Machinery and Intelligence* [Tur50], Alan Turing predicted that by the end of the twentieth century, "general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted." Even if Turing has not yet been proven right, the idea of *cognition as computation* has emerged as a fundamental tenet of Artificial Intelligence (AI) and cognitive science. But what kind of computation—what kind of computer program—is cognition?

AI researchers have made impressive progress since the birth of the field over 60 years ago. Yet despite this progress, no existing AI system can reproduce any nontrivial fraction of the inferences made regularly by children. Turing himself appreciated that matching the capability of children, e.g., in language, presented a key challenge for AI:

> We hope that machines will eventually compete with men in all purely intellectual fields. But which are the best ones to start with? Even this is a difficult decision. Many people think that a very abstract activity, like the playing of chess, would be best. It can also be maintained that it is best to provide the machine with the best sense organs money can buy, and then teach it to understand and speak English. This process could follow the normal teaching of a child. Things would be pointed out and named, etc. Again I do not know what the right answer is, but I think both approaches should be tried. [Tur50, p. 460]

Indeed, many of the problems once considered to be grand AI challenges have fallen prey to essentially brute-force algorithms backed by enormous amounts of computation, often robbing us of the insight we hoped to gain by studying these challenges in the first place. Turing's presentation of his "imitation game" (what we now call "the Turing test"), and the problem of common-sense reasoning implicit in it, demonstrates that he understood the difficulty inherent in the open-ended, if commonplace, tasks involved in conversation. Over a half century later, the Turing test remains resistant to attack.

The analogy between minds and computers has spurred incredible scientific progress in both directions, but there are still fundamental disagreements about the nature of the computation performed by our minds, and how best to narrow the divide between the capability and flexibility of human and artificial intelligence. The goal of this article is to describe a computational formalism that has proved useful for building simplified models of common-sense reasoning. The centerpiece of the formalism is a universal probabilistic Turing machine called QUERY that performs *conditional simulation*, and thereby captures the operation of conditioning probability distributions that are themselves represented by probabilistic Turing machines. We will use QUERY to model the inductive leaps that typify common-sense reasoning. The distributions on which QUERY will operate are models of latent unobserved processes in the world and the sensory experience and observations they generate. Through a running example of medical diagnosis, we aim to illustrate the flexibility and potential of this approach.

The QUERY abstraction is a component of several research programs in AI and cognitive science developed jointly with a number of collaborators. This chapter represents our own view on a subset of these threads and their relationship with Turing's legacy. Our presentation here draws heavily on both the work of Vikash Mansinghka on "natively probabilistic computing" [Man09, MJT08, Man11, MR] and the "probabilistic language of thought" hypothesis proposed and developed by Noah Goodman [KGT08, GTFG08, GG12, GT12]. Their ideas form core aspects of the picture we present. The *Church* probabilistic programming language (introduced in [GMR+08] by Goodman, Mansinghka, Roy, Bonawitz, and Tenenbaum)

and various Church-based cognitive science tutorials (in particular, [GTO11], developed by Goodman, Tenenbaum, and O'Donnell) have also had a strong influence on the presentation.

This approach also draws from work in cognitive science on "theory-based Bayesian models" of inductive learning and reasoning [TGK06] due to Tenenbaum and various collaborators [GKT08, KT08, TKGG11]. Finally, some of the theoretical aspects that we present are based on results in computable probability theory by Ackerman, Freer, and Roy [Roy11, AFR11].

While the particular formulation of these ideas is recent, they have antecedents in much earlier work on the foundations of computation and computable analysis, common-sense reasoning in AI, and Bayesian modeling and statistics. In all of these areas, Turing had pioneering insights.

1.1. **A convergence of Turing's ideas.** In addition to Turing's well-known contributions to the philosophy of AI, many other aspects of his work—across statistics, the foundations of computation, and even morphogenesis—have converged in the modern study of AI. In this section, we highlight a few key ideas that will frequently surface during our account of common-sense reasoning via conditional simulation.

An obvious starting point is Turing's own proposal for a research program to pass his eponymous test. From a modern perspective, Turing's focus on learning (and in particular, induction) was especially prescient. For Turing, the idea of programming an intelligent machine entirely by hand was clearly infeasible, and so he reasoned that it would be necessary to construct a machine with the ability to adapt its own behavior in light of experience—i.e., with the ability to learn:

> Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one that simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain. [Tur50, p. 456]

Turing's notion of learning was inductive as well as deductive, in contrast to much of the work that followed in the first decade of AI. In particular, he was quick to explain that such a machine would have its flaws (in reasoning, quite apart from calculational errors):

> [A machine] might have some method for drawing conclusions by scientific induction. We must expect such a method to lead occasionally to erroneous results. [Tur50, p. 449]

Turing also appreciated that a machine would not only have to learn facts, but would also need to learn *how to learn*:

> Important amongst such imperatives will be ones which regulate the order in which the rules of the logical system concerned are to be applied. For at each stage when one is using a logical system, there is a very large number of alternative steps, any of which one is permitted to apply [...] These choices make the difference between a brilliant and a footling reasoner, not the difference between a sound and a fallacious one. [...] [Some such imperatives] may be 'given by authority', but others may be produced by the machine itself, *e.g.* by scientific induction. [Tur50, p. 458]

In addition to making these more abstract points, Turing presented a number of concrete proposals for how a machine might be programmed to learn. His ideas

capture the essence of supervised, unsupervised, and reinforcement learning, each major areas in modern AI.[1] In Sections 5 and 7 we will return to Turing's writings on these matters.

One major area of Turing's contributions, while often overlooked, is statistics. In fact, Turing, along with I. J. Good, made key advances in statistics in the course of breaking the Enigma during World War II. Turing and Good developed new techniques for incorporating evidence and new approximations for estimating parameters in hierarchical models [Goo79, Goo00] (see also [Zab95, §5] and [Zab12]), which were among the most important applications of Bayesian statistics at the time [Zab12, §3.2]. Given Turing's interest in learning machines and his deep understanding of statistical methods, it would have been intriguing to see a proposal to combine the two areas. Yet if he did consider these connections, it seems he never published such work. On the other hand, much of modern AI rests upon a statistical foundation, including Bayesian methods. This perspective permeates the approach we will describe, wherein learning is achieved via Bayesian inference, and in Sections 5 and 6 we will re-examine some of Turing's wartime statistical work in the context of hierarchical models.

A core latent hypothesis underlying Turing's diverse body of work was that processes in nature, including our minds, could be understood through mechanical—in fact, *computational*—descriptions. One of Turing's crowning achievements was his introduction of the a-machine, which we now call the Turing machine. The Turing machine characterized the limitations and possibilities of computation by providing a mechanical description of a human computer. Turing's work on morphogenesis [Tur52] and AI each sought mechanical explanations in still further domains. Indeed, in all of these areas, Turing was acting as a natural scientist [Hod97], building models of natural phenomena using the language of computational processes.

In our account of common-sense reasoning as conditional simulation, we will use probabilistic Turing machines to represent mechanical descriptions of the world, much like those Turing sought. In each case, the stochastic machine represents one's uncertainty about the generative process giving rise to some pattern in the natural world. This description then enables probabilistic inference (via QUERY) about these patterns, allowing us to make decisions and manage our uncertainty in light of new evidence. Over the course of the article we will see a number of stochastic generative processes of increasing sophistication, culminating in models of decision making that rely crucially on recursion. Through its emphasis on inductive learning, Bayesian statistical techniques, universal computers, and mechanical models of nature, this approach to common-sense reasoning represents a convergence of many of Turing's ideas.

1.2. **Common-sense reasoning via QUERY.** For the remainder of the paper, our focal point will be the probabilistic Turing machine QUERY, which implements a generic form of probabilistic conditioning. QUERY allows one to make predictions using complex probabilistic models that are themselves specified using probabilistic Turing machines. By using QUERY appropriately, one can describe various forms

[1]Turing also developed some of the early ideas regarding *neural networks*; see the discussions in [Tur48] about "unorganized machines" and their education and organization. This work, too, has grown into a large field of modern research, though we will not explore neural nets in the present article. For more details, and in particular the connection to work of McCulloch and Pitts [MP43], see Copeland and Proudfoot [CP96] and Teuscher [Teu02].

of learning, inference, and decision-making. These arise via Bayesian inference, and common-sense behavior can be seen to follow *implicitly* from past experience and models of causal structure and goals, rather than explicitly via rules or purely deductive reasoning. Using the extended example of medical diagnosis, we aim to demonstrate that QUERY is a surprisingly powerful abstraction for expressing common-sense reasoning tasks that have, until recently, largely defied formalization.

As with Turing's investigations in AI, the approach we describe has been motivated by reflections on the details of human cognition, as well as on the nature of computation. In particular, much of the AI framework that we describe has been inspired by research in cognitive science attempting to model human inference and learning. Indeed, hypotheses expressed in this framework have been compared with the judgements and behaviors of human children and adults in many psychology experiments. Bayesian generative models, of the sort we describe here, have been shown to predict human behavior on a wide range of cognitive tasks, often with high quantitative accuracy. For examples of such models and the corresponding experiments, see the review article [TKGG11]. We will return to some of these more complex models in Section 8. We now proceed to define QUERY and illustrate its use via increasingly complex problems and the questions these raise.

2. Probabilistic reasoning and QUERY

The specification of a probabilistic model can implicitly define a space of complex and useful behavior. In this section we informally describe the universal probabilistic Turing machine QUERY, and then use QUERY to explore a medical diagnosis example that captures many aspects of common-sense reasoning, but in a simple domain. Using QUERY, we highlight the role of conditional independence and conditional probability in building compact yet highly flexible systems.

2.1. An informal introduction to QUERY.

The QUERY formalism was originally developed in the context of the Church probabilistic programming language [GMR+08], and has been further explored by Mansinghka [Man11] and Mansinghka and Roy [MR].

At the heart of the QUERY abstraction is a probabilistic variation of Turing's own mechanization [Tur36] of the capabilities of human "computers", the Turing machine. A Turing machine is a finite automaton with read, write, and seek access to a finite collection of infinite binary tapes, which it may use throughout the course of its execution. Its input is loaded onto one or more of its tapes prior to execution, and the output is the content of (one or more of) its tapes after the machine enters its halting state. A *probabilistic* Turing machine (PTM) is simply a Turing machine with an additional read-only tape comprised of a sequence of independent random bits, which the finite automaton may read and use as a source of randomness.

Turing machines (and their probabilistic generalizations) capture a robust notion of deterministic (and probabilistic) computation: Our use of the Turing machine abstraction relies on the remarkable existence of *universal* Turing machines, which can simulate all other Turing machines. More precisely, there is a PTM UNIVERSAL and an encoding $\{e_s : s \in \{0,1\}^*\}$ of all PTMs, where $\{0,1\}^*$ denotes the set of finite binary strings, such that, on inputs s and x, UNIVERSAL halts and outputs the string t if and only if (the Turing machine encoded by) e_s halts and outputs t on input x. Informally, the input s to UNIVERSAL is analogous to a program

written in a programming language, and so we will speak of (encodings of) Turing machines and programs interchangeably.

QUERY is a PTM that takes two inputs, called the *prior program* P and *conditioning predicate* C, both of which are themselves (encodings of) PTMs that take no input (besides the random bit tape), with the further restriction that the predicate C return only a 1 or 0 as output. The semantics of QUERY are straightforward: first generate a sample from P; if C is satisfied, then output the sample; otherwise, try again. More precisely:

(1) Simulate the predicate C on a random bit tape R (i.e., using the existence of a universal Turing machine, determine the output of the PTM C, if R were its random bit tape);
(2) If (the simulation of) C produces 1 (i.e., if C *accepts*), then simulate and return the output produced by P, using the *same* random bit tape R;
(3) Otherwise (if C *rejects* R, returning 0), return to step 1, using an independent sequence R' of random bits.

It is important to stress that P and C share a random bit tape on each iteration, and so the predicate C may, in effect, act as though it has access to any intermediate value computed by the prior program P when deciding whether to accept or reject a random bit tape. More generally, any value computed by P can be recomputed by C and vice versa. We will use this fact to simplify the description of predicates, informally referring to values computed by P in the course of defining a predicate C.

As a first step towards understanding QUERY, note that if $\top$ is a PTM that always accepts (i.e., always outputs 1), then $\mathsf{QUERY}(\mathsf{P}, \top)$ produces the same distribution on outputs as executing P itself, as the semantics imply that QUERY would halt on the first iteration.

Predicates that are not identically 1 lead to more interesting behavior. Consider the following simple example based on a remark by Turing [Tur50, p. 459]: Let N_{180} be a PTM that returns (a binary encoding of) an integer N drawn uniformly at random in the range 1 to 180, and let $\mathsf{DIV}_{2,3,5}$ be a PTM that accepts (outputs 1) if N is divisible by 2, 3, and 5; and rejects (outputs 0) otherwise. Consider a typical output of

$$\mathsf{QUERY}(\mathsf{N}_{180}, \mathsf{DIV}_{2,3,5}).$$

Given the semantics of QUERY, we know that the output will fall in the set

$$\{30, 60, 90, 120, 150, 180\} \tag{1}$$

and moreover, because each of these possible values of N was *a priori* equally likely to arise from executing N_{180} alone, this remains true *a posteriori*. You may recognize this as the conditional distribution of a uniform distribution conditioned to lie in the set (1). Indeed, QUERY performs the operation of conditioning a distribution.

The behavior of QUERY can be described more formally with notions from probability theory. In particular, from this point on, we will think of the output of a PTM (say, P) as a random variable (denoted by φ_{P}) defined on an underlying probability space with probability measure $\mathbb{P}$. (We will define this probability space formally in Section 3.1, but informally it represents the random bit tape.) When it is clear from context, we will also regard any named intermediate value (like N) as a random variable on this same space. Although Turing machines manipulate binary representations, we will often gloss over the details of how elements of other

countable sets (like the integers, naturals, rationals, etc.) are represented in binary, but only when there is no risk of serious misunderstanding.

In the context of $\mathsf{QUERY}(\mathsf{P}, \mathsf{C})$, the output distribution of P, which can be written $\mathbb{P}(\varphi_{\mathsf{P}} \in \cdot\,)$, is called the *prior* distribution. Recall that, for all measurable sets (or simply *events*) A and C ,

$$\mathbb{P}(A \mid C) := \frac{\mathbb{P}(A \cap C)}{\mathbb{P}(C)}, \tag{2}$$

is the conditional probability of the event A given the event C, provided that $\mathbb{P}(C) > 0$. Then the distribution of the output of $\mathsf{QUERY}(\mathsf{P}, \mathsf{C})$, called the *posterior* distribution of φ_{P}, is the conditional distribution of φ_{P} given the event $\varphi_{\mathsf{C}} = 1$, written

$$\mathbb{P}(\varphi_{\mathsf{P}} \in \cdot \mid \varphi_{\mathsf{C}} = 1).$$

Then returning to our example above, the prior distribution, $\mathbb{P}(N \in \cdot\,)$, is the uniform distribution on the set $\{1, \ldots, 180\}$, and the posterior distribution,

$$\mathbb{P}(N \in \cdot \mid N \text{ divisible by 2, 3, and 5}),$$

is the uniform distribution on the set given in (1), as can be verified via equation (2).

Those familiar with statistical algorithms will recognize the mechanism of QUERY to be exactly that of a so-called "rejection sampler". Although the definition of QUERY describes an explicit algorithm, we do not actually intend QUERY to be executed in practice, but rather intend for it to define and represent complex distributions. (Indeed, the description can be used by algorithms that work by very different methods than rejection sampling, and can aid in the communication of ideas between researchers.)

The actual implementation of QUERY in more efficient ways than via a rejection sampler is an active area of research, especially via techniques involving Markov chain Monte Carlo (MCMC); see, e.g., [GMR+08, WSG11, WGSS11, SG12]. Turing himself recognized the potential usefulness of randomness in computation, suggesting:

> It is probably wise to include a random element in a learning machine. A random element is rather useful when we are searching for a solution of some problem. [Tur50, p. 458]

Indeed, some aspects of these algorithms are strikingly reminiscent of Turing's description of a random system of rewards and punishments in guiding the organization of a machine:

> The character may be subject to some random variation. Pleasure interference has a tendency to fix the character i.e. towards preventing it changing, whereas pain stimuli tend to disrupt the character, causing features which had become fixed to change, or to become again subject to random variation. [Tur48, §10]

However, in this paper, we will not go into further details of implementation, nor the host of interesting computational questions this endeavor raises.

Given the subtleties of conditional probability, it will often be helpful to keep in mind the behavior of a rejection-sampler when considering examples of QUERY. (See [SG92] for more examples of this approach.) Note that, in our example,

(a) Disease marginals

n	Disease	p_n
1	Arthritis	0.06
2	Asthma	0.04
3	Diabetes	0.11
4	Epilepsy	0.002
5	Giardiasis	0.006
6	Influenza	0.08
7	Measles	0.001
8	Meningitis	0.003
9	MRSA	0.001
10	Salmonella	0.002
11	Tuberculosis	0.003

(b) Unexplained symptoms

m	Symptom	ℓ_m
1	Fever	0.06
2	Cough	0.04
3	Hard breathing	0.001
4	Insulin resistant	0.15
5	Seizures	0.002
6	Aches	0.2
7	Sore neck	0.006

(c) Disease-symptom rates

$c_{n,m}$	1	2	3	4	5	6	7
1	.1	.2	.1	.2	.2	.5	.5
2	.1	.4	.8	.3	.1	.0	.1
3	.1	.2	.1	.9	.2	.3	.5
4	.4	.1	.0	.2	.9	.0	.0
5	.6	.3	.2	.1	.2	.8	.5
6	.4	.2	.0	.2	.0	.7	.4
7	.5	.2	.1	.2	.1	.6	.5
8	.8	.3	.0	.3	.1	.8	.9
9	.3	.2	.1	.2	.0	.3	.5
10	.4	.1	.0	.2	.1	.1	.2
11	.3	.2	.1	.2	.2	.3	.5

TABLE 1. Medical diagnosis parameters. (These values are fabricated.) (a) p_n is the marginal probability that a patient has a disease n. (b) ℓ_m is the probability that a patient presents symptom m, assuming they have no disease. (c) $c_{n,m}$ is the probability that disease n causes symptom m to present, assuming the patient has disease n.

every simulation of N_{180} generates a number "accepted by" $\mathsf{DIV}_{2,3,5}$ with probability $\frac{1}{30}$, and so, on average, we would expect the loop within QUERY to repeat approximately 30 times before halting. However, there is no finite bound on how long the computation could run. On the other hand, one can show that $\mathsf{QUERY}(\mathsf{N}_{180}, \mathsf{DIV}_{2,3,5})$ eventually halts with probability one (equivalently, it halts *almost surely*, sometimes abbreviated "a.s.").

Despite the apparent simplicity of the QUERY construct, we will see that it captures the essential structure of a range of common-sense inferences. We now demonstrate the power of the QUERY formalism by exploring its behavior in a medical diagnosis example.

2.2. **Diseases and their symptoms.** Consider the following prior program, DS, which represents a simplified model of the pattern of *Diseases and Symptoms* we might find in a typical patient chosen at random from the population. At a high level, the model posits that the patient may be suffering from some, possibly empty, set of diseases, and that these diseases can cause symptoms. The prior program DS proceeds as follows: For each disease n listed in Table 1a, sample an independent binary random variable D_n with mean p_n, which we will interpret as indicating whether or not a patient has disease n depending on whether $D_n = 1$ or $D_n = 0$, respectively. For each symptom m listed in Table 1b, sample an independent binary random variable L_m with mean ℓ_m and for each pair (n, m) of a disease and symptom, sample an independent binary random variable $C_{n,m}$ with mean $c_{n,m}$, as listed in Table 1c. (Note that the numbers in all three tables have been fabricated.) Then, for each symptom m, define

$$S_m = \max\{L_m, D_1 \cdot C_{1,m}, \ldots, D_{11} \cdot C_{11,m}\},$$

so that $S_m \in \{0, 1\}$. We will interpret S_m as indicating that a patient has symptom m; the definition of S_m implies that this holds when any of the variables on the right hand side take the value 1. (In other words, the max operator is playing the role of a logical OR operation.) Every term of the form $D_n \cdot C_{n,m}$ is interpreted as indicating whether (or not) the patient has disease n *and* disease n has caused symptom m. The term L_m captures the possibility that the symptom may present

itself despite the patient having none of the listed diseases. Finally, define the output of DS to be the vector $(D_1, \ldots, D_{11}, S_1, \ldots, S_7)$.

If we execute DS, or equivalently QUERY(DS, $\top$), then we might see outputs like those in the following array:

	Diseases											Symptoms						
	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
4	0	0	1	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

We will interpret the rows as representing eight patients chosen independently at random, the first two free from disease and not presenting any symptoms; the third suffering from diabetes and presenting insulin resistance; the fourth suffering from diabetes and influenza, and presenting a fever and insulin resistance; the fifth suffering from unexplained aches; the sixth free from disease and symptoms; the seventh suffering from diabetes, and presenting insulin resistance and aches; and the eighth also disease and symptom free.

This model is a toy version of the real diagnostic model QMR-DT [SMH+91]. QMR-DT is probabilistic model with essentially the structure of DS, built from data in the Quick Medical Reference (QMR) knowledge base of hundreds of diseases and thousands of findings (such as symptoms or test results). A key aspect of this model is the disjunctive relationship between the diseases and the symptoms, known as a "noisy-OR", which remains a popular modeling idiom. In fact, the structure of this model, and in particular the idea of layers of disjunctive causes, goes back even further to the "causal calculus" developed by Good [Goo61], which was based in part on his wartime work with Turing on the weight of evidence, as discussed by Pearl [Pea04, §70.2].

Of course, as a model of the latent processes explaining natural patterns of diseases and symptoms in a random patient, DS still leaves much to be desired. For example, the model assumes that the presence or absence of any two diseases is independent, although, as we will see later on in our analysis, diseases are (as expected) typically not independent conditioned on symptoms. On the other hand, an actual disease might cause another disease, or might cause a symptom that itself causes another disease, possibilities that this model does not capture. Like QMR-DT, the model DS avoids simplifications made by many earlier expert systems and probabilistic models to not allow for the simultaneous occurrence of multiple diseases [SMH+91]. These caveats notwithstanding, a close inspection of this simplified model will demonstrate a surprising range of common-sense reasoning phenomena.

Consider a predicate OS, for *Observed Symptoms*, that accepts if and only if $S_1 = 1$ and $S_7 = 1$, i.e., if and only if the patient presents the symptoms of a fever and a sore neck. What outputs should we expect from QUERY(DS, OS)? Informally, if we let μ denote the distribution over the combined outputs of DS and OS on a shared random bit tape, and let $A = \{(x, c) : c = 1\}$ denote the set of those pairs that OS accepts, then QUERY(DS, OS) generates samples from the conditioned distribution $\mu(\cdot \mid A)$. Therefore, to see what the condition $S_1 = S_7 = 1$ implies about the

plausible execution of DS, we must consider the conditional distributions of the diseases given the symptoms. The following conditional probability calculations may be very familiar to some readers, but will be less so to others, and so we present them here to give a more complete picture of the behavior of QUERY.

2.2.1. *Conditional execution.* Consider a $\{0,1\}$-assignment d_n for each disease n, and write $D = d$ to denote the event that $D_n = d_n$ for every such n. Assume for the moment that $D = d$. Then what is the probability that OS accepts? The probability we are seeking is the conditional probability

$$\mathbb{P}(S_1 = S_7 = 1 \mid D = d) \tag{3}$$
$$= \mathbb{P}(S_1 = 1 \mid D = d) \cdot \mathbb{P}(S_7 = 1 \mid D = d), \tag{4}$$

where the equality follows from the observation that once the D_n variables are fixed, the variables S_1 and S_7 are independent. Note that $S_m = 1$ if and only if $L_m = 1$ or $C_{n,m} = 1$ for some n such that $d_n = 1$. (Equivalently, $S_m = 0$ if and only if $L_m = 0$ and $C_{n,m} = 0$ for all n such that $d_n = 1$.) By the independence of each of these variables, it follows that

$$\mathbb{P}(S_m = 1 | D = d) = 1 - (1 - \ell_m) \prod_{n \,:\, d_n = 1} (1 - c_{n,m}). \tag{5}$$

Let d' be an alternative $\{0,1\}$-assignment. We can now characterize the *a posteriori* odds

$$\frac{\mathbb{P}(D = d \mid S_1 = S_7 = 1)}{\mathbb{P}(D = d' \mid S_1 = S_7 = 1)}$$

of the assignment d versus the assignment d'. By Bayes' rule, this can be rewritten as

$$\frac{\mathbb{P}(S_1 = S_7 = 1 \mid D = d) \cdot \mathbb{P}(D = d)}{\mathbb{P}(S_1 = S_7 = 1 \mid D = d') \cdot \mathbb{P}(D = d')}, \tag{6}$$

where $\mathbb{P}(D = d) = \prod_{n=1}^{11} \mathbb{P}(D_n = d_n)$ by independence. Using (4), (5) and (6), one may calculate that

$$\frac{\mathbb{P}(\text{Patient only has influenza} \mid S_1 = S_7 = 1)}{\mathbb{P}(\text{Patient has no listed disease} \mid S_1 = S_7 = 1)} \approx 42,$$

i.e., it is forty-two times more likely that an execution of DS satisfies the predicate OS via an execution that posits the patient only has the flu than an execution which posits that the patient has no disease at all. On the other hand,

$$\frac{\mathbb{P}(\text{Patient only has meningitis} \mid S_1 = S_7 = 1)}{\mathbb{P}(\text{Patient has no listed disease} \mid S_1 = S_7 = 1)} \approx 6,$$

and so

$$\frac{\mathbb{P}(\text{Patient only has influenza} \mid S_1 = S_7 = 1)}{\mathbb{P}(\text{Patient only has meningitis} \mid S_1 = S_7 = 1)} \approx 7,$$

and hence we would expect, over many executions of QUERY(DS, OS), to see roughly seven times as many explanations positing only influenza than positing only meningitis.

Further investigation reveals some subtle aspects of the model. For example, consider the fact that

$$\frac{\mathbb{P}(\text{Patient only has meningitis and influenza} \mid S_1 = S_7 = 1)}{\mathbb{P}(\text{Patient has meningitis, maybe influenza, but nothing else} \mid S_1 = S_7 = 1)} = 0.09 \approx \mathbb{P}(\text{Patient has influenza}), \tag{7}$$

which demonstrates that, once we have observed some symptoms, diseases are no longer independent. Moreover, this shows that once the symptoms have been "explained" by meningitis, there is little pressure to posit further causes, and so the posterior probability of influenza is nearly the prior probability of influenza. This phenomenon is well-known and is called *explaining away*; it is also known to be linked to the computational hardness of computing probabilities (and generating samples as QUERY does) in models of this variety. For more details, see [Pea88, §2.2.4].

2.2.2. *Predicates give rise to diagnostic rules.* These various conditional probability calculations, and their ensuing explanations, all follow from an analysis of the DS model conditioned on one particular (and rather simple) predicate OS. Already, this gives rise to a picture of how QUERY(DS, OS) implicitly captures an elaborate system of rules for what to believe following the observation of a fever and sore neck in a patient, assuming the background knowledge captured in the DS program and its parameters. In a similar way, every diagnostic scenario (encodable as a predicate) gives rise to its own complex set of inferences, each expressible using QUERY and the model DS.

As another example, if we look (or test) for the remaining symptoms and find them to all be absent, our new beliefs are captured by QUERY(DS, OS*) where the predicate OS* accepts if and only if $(S_1 = S_7 = 1) \wedge (S_2 = \cdots = S_6 = 0)$.

We need not limit ourselves to reasoning about diseases given symptoms. Imagine that we perform a diagnostic test that rules out meningitis. We could represent our new knowledge using a predicate capturing the condition

$$(D_8 = 0) \wedge (S_1 = S_7 = 1) \wedge (S_2 = \cdots = S_6 = 0).$$

Of course this approach would not take into consideration our uncertainty regarding the accuracy or mechanism of the diagnostic test itself, and so, ideally, we might expand the DS model to account for how the outcomes of diagnostic tests are affected by the presence of other diseases or symptoms. In Section 6, we will discuss how such an extended model might be learned from data, rather than constructed by hand.

We can also reason in the other direction, about symptoms given diseases. For example, public health officials might wish to know about how frequently those with influenza present no symptoms. This is captured by the conditional probability

$$\mathbb{P}(S_1 = \cdots = S_7 = 0 \mid D_6 = 1),$$

and, via QUERY, by the predicate for the condition $D_6 = 1$. Unlike the earlier examples where we reasoned backwards from effects (symptoms) to their likely causes (diseases), here we are reasoning in the same forward direction as the model DS is expressed.

The possibilities are effectively inexhaustible, including more complex states of knowledge such as, *there are at least two symptoms present*, or *the patient does not have both salmonella and tuberculosis*. In Section 4 we will consider the vast

number of predicates and the resulting inferences supported by QUERY and DS, and contrast this with the compact size of DS and the table of parameters.

In this section, we have illustrated the basic behavior of QUERY, and have begun to explore how it can be used to decide what to believe in a given scenario. These examples also demonstrate that rules governing behavior need not be explicitly described as rules, but can arise implicitly via other mechanisms, like QUERY, paired with an appropriate prior and predicate. In this example, the diagnostic rules were determined by the definition of DS and the table of its parameters. In Section 5, we will examine how such a table of probabilities itself might be learned. In fact, even if the parameters are learned from data, the structure of DS itself still posits a strong structural relationship among the diseases and symptoms. In Section 6 we will explore how this structure could be learned. Finally, many common-sense reasoning tasks involve making a *decision*, and not just determining what to believe. In Section 7, we will describe how to use QUERY to make decisions under uncertainty.

Before turning our attention to these more complex uses of QUERY, we pause to consider a number of interesting theoretical questions: What kind of probability distributions can be represented by PTMs that generate samples? What kind of conditional distributions can be represented by QUERY? Or represented by PTMs in general? In the next section we will see how Turing's work formed the foundation of the study of these questions many decades later.

3. Computable probability theory

We now examine the QUERY formalism in more detail, by introducing aspects of the framework of computable probability theory, which provides rigorous notions of computability for probability distributions, as well as the tools necessary to identify probabilistic operations that can and cannot be performed by algorithms. After giving a formal description of probabilistic Turing machines and QUERY, we relate them to the concept of a computable measure on a countable space. We then explore the representation of points (and random points) in uncountable spaces, and examine how to use QUERY to define models over uncountable spaces like the reals. Such models are commonplace in statistical practice, and thus might be expected to be useful for building a statistical mind. In fact, no generic and computable QUERY formalism exists for conditioning on observations taking values in uncountable spaces, but there are certain circumstances in which we can perform probabilistic inference in uncountable spaces.

Note that although the approach we describe uses a universal Turing machine (QUERY), which can take an arbitrary pair of programs as its prior and predicate, we do not make use of a so-called *universal* prior program (itself necessarily noncomputable). For a survey of approaches to inductive reasoning involving a universal prior, such as Solomonoff induction [Sol64], and computable approximations thereof, see Rathmanner and Hutter [RH11].

Before we discuss the capabilities and limitations of QUERY, we give a formal definition of QUERY in terms of probabilistic Turing machines and conditional distributions.

3.1. A formal definition of QUERY.

Randomness has long been used in mathematical algorithms, and its formal role in computations dates to shortly after the introduction of Turing machines. In his paper [Tur50] introducing the Turing test,

Turing informally discussed introducing a "random element", and in a radio discussion c. 1951 (later published as [Tur96]), he considered placing a random string of 0's and 1's on an additional input bit tape of a Turing machine. In 1956, de Leeuw, Moore, Shannon, and Shapiro [dMSS56] proposed probabilistic Turing machines (PTMs) more formally, making use of Turing's formalism [Tur39] for oracle Turing machines: a PTM is an oracle Turing machine whose oracle tape comprises independent random bits. From this perspective, the output of a PTM is itself a random variable and so we may speak of *the distribution of (the output of) a PTM*. For the PTM QUERY, which simulates other PTMs passed as inputs, we can express its distribution in terms of the distributions of PTM inputs. In the remainder of this section, we describe this formal framework and then use it to explore the class of distributions that may be represented by PTMs.

Fix a canonical enumeration of (oracle) Turing machines and the corresponding partial computable (oracle) functions $\{\varphi_e\}_{e\in\mathbb{N}}$, each considered as a partial function

$$\{0,1\}^\infty \times \{0,1\}^* \to \{0,1\}^*,$$

where $\{0,1\}^\infty$ denotes the set of countably infinite binary strings and, as before, $\{0,1\}^*$ denotes the set of finite binary strings. One may think of each such partial function as a mapping from an oracle tape and input tape to an output tape. We will write $\varphi_e(x,s)\downarrow$ when φ_e is defined on oracle tape x and input string s, and $\varphi_e(x,s)\uparrow$ otherwise. We will write $\varphi_e(x)$ when the input string is empty or when there is no input tape. As a model for the random bit tape, we define an independent and identically distributed (i.i.d.) sequence $R = (R_i : i \in \mathbb{N})$ of binary random variables, each taking the value 0 and 1 with equal probability, i.e, each R_i is an independent Bernoulli(1/2) random variable. We will write $\mathbb{P}$ to denote the distribution of the random bit tape R. More formally, R will be considered to be the identity function on the Borel probability space $(\{0,1\}^\infty, \mathbb{P})$, where $\mathbb{P}$ is the countable product of Bernoulli(1/2) measures.

Let s be a finite string, let $e \in \mathbb{N}$, and suppose that

$$\mathbb{P}\{\, r \in \{0,1\}^\infty \,:\, \varphi_e(r,s)\downarrow \,\} = 1.$$

Informally, we will say that the probabilistic Turing machine (indexed by) e halts almost surely on input s. In this case, we define the *output distribution of the eth (oracle) Turing machine on input string s* to be the distribution of the random variable

$$\varphi_e(R,s);$$

we may directly express this distribution as

$$\mathbb{P} \circ \varphi_e(\,\cdot\,, s)^{-1}.$$

Using these ideas we can now formalize QUERY. In this formalization, both the prior and predicate programs P and C passed as input to QUERY are finite binary strings interpreted as indices for a probabilistic Turing machine with no input tape. Suppose that P and C halt almost surely. In this case, the output distribution of QUERY(P, C) can be characterized as follows: Let $R = (R_i \,:\, i \in \mathbb{N})$ denote the random bit tape, let $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a standard pairing function (i.e., a computable bijection), and, for each $n, i \in \mathbb{N}$, let $R_i^{(n)} := R_{\pi(n,i)}$ so that $\{R^{(n)} \,:\, n \in \mathbb{N}\}$ are independent random bit tapes, each with distribution $\mathbb{P}$. Define

the nth *sample from the prior* to be the random variable

$$X_n := \varphi_{\mathsf{P}}(R^{(n)}),$$

and let

$$N := \inf \{\, n \in \mathbb{N} \,:\, \varphi_{\mathsf{C}}(R^{(n)}) = 1 \,\}$$

be the first iteration n such that the predicate C evaluates to 1 (i.e., accepts). The *output distribution of* $\mathsf{QUERY}(\mathsf{P}, \mathsf{C})$ is then the distribution of the random variable

$$X_N,$$

whenever $N < \infty$ holds with probability one, and is undefined otherwise. Note that $N < \infty$ a.s. if and only if C accepts with non-zero probability. As above, we can give a more direct characterization: Let

$$\mathcal{A} := \{\, R \in \{0,1\}^\infty \,:\, \varphi_{\mathsf{C}}(R) = 1 \,\}$$

be the set of random bit tapes R such that the predicate C accepts by outputting 1. The condition "$N < \infty$ with probability one" is then equivalent to the statement that $\mathbb{P}(\mathcal{A}) > 0$. In that case, we may express the output distribution of $\mathsf{QUERY}(\mathsf{P}, \mathsf{C})$ as

$$\mathbb{P}_{\mathcal{A}} \circ \varphi_{\mathsf{P}}^{-1}$$

where $\mathbb{P}_{\mathcal{A}}(\cdot) := \mathbb{P}(\,\cdot \mid \mathcal{A})$ is the distribution of the random bit tape conditioned on C accepting (i.e., conditioned on the event $\mathcal{A}$).

3.2. **Computable measures and probability theory.** Which probability distributions are the output distributions of *some* PTM? In order to investigate this question, consider what we might learn from simulating a given PTM P (on a particular input) that halts almost surely. More precisely, for a finite bit string $r \in \{0,1\}^*$ with length $|r|$, consider simulating P, replacing its random bit tape with the finite string r: If, in the course of the simulation, the program attempts to read beyond the end of the finite string r, we terminate the simulation prematurely. On the other hand, if the program halts and outputs a string t then we may conclude that all simulations of P will return the same value when the random bit tape begins with r. As the set of random bit tapes beginning with r has $\mathbb{P}$-probability $2^{-|r|}$, we may conclude that the distribution of P assigns at least this much probability to the string t.

It should be clear that, using the above idea, we may enumerate the (prefix-free) set of strings $\{r_n\}$, and matching outputs $\{t_n\}$, such that P outputs t_n when its random bit tape begins with r_n. It follows that, for all strings t and $m \in \mathbb{N}$,

$$\sum_{\{n \le m \,:\, t_n = t\}} 2^{-|r_n|}$$

is a lower bound on the probability that the distribution of P assigns to t, and

$$1 - \sum_{\{n \le m \,:\, t_n \neq t\}} 2^{-|r_n|}$$

is an upper bound. Moreover, it is straightforward to show that as $m \to \infty$, these converge monotonically from above and below to the probability that P assigns to the string t.

This sort of effective information about a real number precisely characterizes the *computable real numbers*, first described by Turing in his paper [Tur36] introducing Turing machines. For more details, see the survey by Avigad and Brattka connecting computable analysis to work of Turing, elsewhere in this volume [AB].

Definition 3.1 (computable real number)**.** A real number $r \in \mathbb{R}$ is said to be *computable* when its left and right cuts of rationals $\{q \in \mathbb{Q} : q < r\}, \{q \in \mathbb{Q} : r < q\}$ are computable (under the canonical computable encoding of rationals). Equivalently, a real is computable when there is a computable sequence of rationals $\{q_n\}_{n\in\mathbb{N}}$ that *rapidly converges to* r, in the sense that $|q_n - r| < 2^{-n}$ for each n.

We now know that the probability of each output string t from a PTM is a computable real (in fact, *uniformly in* t, i.e., this probability can be computed for each t by a single program that accepts t as input.). Conversely, for every computable real $\alpha \in [0, 1]$ and string t, there is a PTM that outputs t with probability α. In particular, let $R = (R_1, R_2, \dots)$ be our random bit tape, let $\alpha_1, \alpha_2, \dots$ be a uniformly computable sequence of rationals that rapidly converges to α, and consider the following simple program: On step n, compute the rational $A_n := \sum_{i=1}^{n} R_i \cdot 2^{-i}$. If $A_n < \alpha_n - 2^{-n}$, then halt and output t; If $A_n > \alpha_n + 2^{-n}$, then halt and output $t0$. Otherwise, proceed to step $n+1$. Note that $A_\infty := \lim A_n$ is uniformly distributed in the unit interval, and so $A_\infty < \alpha$ with probability α. Because $\lim \alpha_n \to \alpha$, the program eventually halts for all but one (or two, in the case that α is a dyadic rational) random bit tapes. In particular, if the random bit tape is the binary expansion of α, or equivalently, if $A_\infty = \alpha$, then the program does not halt, but this is a $\mathbb{P}$-measure zero event.

Recall that we assumed, in defining the output distribution of a PTM, that the program halted almost surely. The above construction illustrates why the stricter requirement that PTMs halt always (and not just almost surely) could be very limiting. In fact, one can show that there is no PTM that halts always and whose output distribution assigns, e.g., probability 1/3 to 1 and 2/3 to 0. Indeed, the same is true for all non-dyadic probability values (for details see [AFR11, Prop. 9]).

We can use this construction to sample from any distribution ν on $\{0, 1\}^*$ for which we can compute the probability of a string t in a uniform way. In particular, fix an enumeration of all strings $\{t_n\}$ and, for each $n \in \mathbb{N}$, define the distribution ν_n on $\{t_n, t_{n+1}, \dots\}$ by $\nu_n = \nu/(1 - \nu\{t_1, \dots, t_{n-1}\})$. If ν is computable in the sense that for any t, we may compute real $\nu\{t\}$ uniformly in t, then ν_n is clearly computable in the same sense, uniformly in n. We may then proceed in order, deciding whether to output t_n (with probability $\nu_n\{t_n\}$) or to recurse and consider t_{n+1}. It is straightforward to verify that the above procedure outputs a string t with probability $\nu\{t\}$, as desired.

These observations motivate the following definition of a computable probability measure, which is a special case of notions from computable analysis developed later; for details of the history see [Wei99, §1].

Definition 3.2 (computable probability measure)**.** A probability measure on $\{0, 1\}^*$ is said to be *computable* when the measure of each string is a computable real, uniformly in the string.

The above argument demonstrates that the samplable probability measures — those distributions on $\{0, 1\}^*$ that arise from sampling procedures performed by

probabilistic Turing machines that halt a.s. — coincide with computable probability measures.

While in this paper we will not consider the efficiency of these procedures, it is worth noting that while the class of distributions that can be sampled by Turing machines coincides with the class of computable probability measures on $\{0,1\}^*$, the analogous statements for polynomial-time Turing machines fail. In particular, there are distributions from which one can efficiently sample, but for which output probabilities are not efficiently computable (unless $\mathbf{P} = \mathbf{PP}$), for suitable formalizations of these concepts [Yam99].

3.3. **Computable probability measures on uncountable spaces.** So far we have considered distributions on the space of finite binary strings. Under a suitable encoding, PTMs can be seen to represent distributions on general countable spaces. On the other hand, many phenomena are naturally modeled in terms of continuous quantities like real numbers. In this section we will look at the problem of representing distributions on uncountable spaces, and then consider the problem of extending QUERY in a similar direction.

To begin, we will describe distributions on the space of *infinite* binary strings, $\{0,1\}^\infty$. Perhaps the most natural proposal for representing such distributions is to again consider PTMs whose output can be interpreted as representing a random point in $\{0,1\}^\infty$. As we will see, such distributions will have an equivalent characterization in terms of uniform computability of the measure of a certain class of sets.

Fix a computable bijection between $\mathbb{N}$ and finite binary strings, and for $n \in \mathbb{N}$, write $\bar{n}$ for the image of n under this map. Let e be the index of some PTM, and suppose that $\varphi_e(R, \bar{n}) \in \{0,1\}^n$ and $\varphi_e(R, \bar{n}) \sqsubseteq \varphi_e(R, \overline{n+1})$ almost surely for all $n \in \mathbb{N}$, where $r \sqsubseteq s$ for two binary strings r and s when r is a prefix of s. Then the *random point in* $\{0,1\}^\infty$ *given by* e is defined to be

$$\lim_{n\to\infty} (\varphi_e(R, \bar{n}), 0, 0, \dots). \tag{8}$$

Intuitively, we have represented the (random) infinite object by a program (relative to a fixed random bit tape) that can provide a convergent sequence of finite approximations.

It is obvious that the distribution of $\varphi_e(R, \bar{n})$ is computable, uniformly in n. As a consequence, for every basic clopen set $A = \{s : r \sqsubseteq s\}$, we may compute the probability that the limiting object defined by (8) falls into A, and thus we may compute arbitrarily good lower bounds for the measure of unions of computable sequences of basic clopen sets, i.e., c.e. open sets.

This notion of computability of a measure is precisely that developed in computable analysis, and in particular, via the Type-Two Theory of Effectivity (TTE); for details see Edalat [Eda96], Weihrauch [Wei99], Schröder [Sch07], and Gács [Gác05]. This formalism rests on Turing's oracle machines [Tur39]; for more details, again see the survey by Avigad and Brattka elsewhere in this volume [AB]. The representation of a measure by the values assigned to basic clopen sets can be interpreted in several ways, each of which allows us to place measures on spaces other than just the set of infinite strings. From a topological point of view, the above representation involves the choice of a particular basis for the topology, with an appropriate enumeration, making $\{0,1\}^\infty$ into a *computable topological space*; for details, see [Wei00, Def. 3.2.1] and [GSW07, Def. 3.1].

Another approach is to place a metric on $\{0,1\}^\infty$ that induces the same topology, and that is computable on a dense set of points, making it into a *computable metric space*; see [Hem02] and [Wei93] on approaches in TTE, [Bla97] and [EH98] in effective domain theory, and [Wei00, Ch. 8.1] and [Gác05, §B.3] for more details. For example, one could have defined the distance between two strings in $\{0,1\}^\infty$ to be 2^{-n}, where n is the location of the first bit on which they differ; instead choosing $1/n$ would have given a different metric space but would induce the same topology, and hence the same notion of computable measure. Here we use the following definition of a computable metric space, taken from [GHR10, Def. 2.3.1].

Definition 3.3 (computable metric space)**.** A *computable metric space* is a triple $(S, \delta, \mathcal{D})$ for which δ is a metric on the set S satisfying

(1) (S, δ) is a complete separable metric space;
(2) $\mathcal{D} = \{s(1), s(2), \dots\}$ is an enumeration of a dense subset of S; and,
(3) the real numbers $\delta(s(i), s(j))$ are computable, uniformly in i and j.

We say that an S-valued random variable X (defined on the same space as R) is an *(almost-everywhere) computable S-valued random variable* or *random point in S* when there is a PTM e such that $\delta(X_n, X) < 2^{-n}$ almost surely for all $n \in \mathbb{N}$, where $X_n := s(\varphi_e(R, \bar{n}))$. We can think of the random sequence $\{X_n\}$ as a *representation* of the random point X. A *computable probability measure* on S is precisely the distribution of such a random variable.

For example, the real numbers form a computable metric space $(\mathbb{R}, d, \mathbb{Q})$, where d is the Euclidean metric, and $\mathbb{Q}$ has the standard enumeration. One can show that computable probability measures on $\mathbb{R}$ are then those for which the measure of an arbitrary finite union of rational open intervals admits arbitrarily good lower bounds, uniformly in (an encoding of) the sequence of intervals. Alternatively, one can show that the space of probability measures on $\mathbb{R}$ is a computable metric space under the Prokhorov metric, with respect to (a standard enumeration of) a dense set of atomic measures with finite support in the rationals. The notions of computability one gets in these settings align with classical notions. For example, the set of naturals and the set of finite binary strings are indeed both computable metric spaces, and the computable measures in this perspective are precisely as described above.

Similarly to the countable case, we can use QUERY to sample points in *uncountable spaces* conditioned on a predicate. Namely, suppose the prior program P represents a random point in an uncountable space with distribution ν. For any string s, write $\mathsf{P}(s)$ for P with the input fixed to s, and let C be a predicate that accepts with non-zero probability. Then the PTM that, on input $\bar{n}$, outputs the result of simulating $\mathsf{QUERY}(\mathsf{P}(\bar{n}), \mathsf{C})$ is a representation of ν conditioned on the predicate accepting. When convenient and clear from context, we will denote this derived PTM by simply writing $\mathsf{QUERY}(\mathsf{P}, \mathsf{C})$.

3.4. **Conditioning on the value of continuous random variables.** The above use of QUERY allows us to condition a model of a computable real-valued random variable X on a predicate C. However, the restriction on predicates (to accept with non-zero probability) and the definition of QUERY itself do not, in general, allow us to condition on X itself taking a specific value. Unfortunately, the problem is not superficial, as we will now relate.

Assume, for simplicity, that X is also continuous (i.e., $\mathbb{P}\{X = x\} = 0$ for all reals x). Let x be a computable real, and for every computable real $\varepsilon > 0$, consider the (partial computable) predicate C_ε that accepts when $|X - x| < \varepsilon$, rejects when $|X - x| > \varepsilon$, and is undefined otherwise. (We say that such a predicate *almost* decides the event $\{|X - x| < \varepsilon\}$ as it decides the set outside a measure zero set.) We can think of $\mathsf{QUERY}(\mathsf{P}, \mathsf{C}_\varepsilon)$ as a "positive-measure approximation" to conditioning on $X = x$. Indeed, if P is a prior program that samples a computable random variable Y and $B_{x,\varepsilon}$ denotes the closed ε-ball around x, then this QUERY corresponds to the conditioned distribution $\mathbb{P}(Y \mid X \in B_{x,\varepsilon})$, and so provided $\mathbb{P}\{X \in B_{x,\varepsilon}\} > 0$, this is well-defined and evidently computable. But what is its relationship to the original problem?

While one might be inclined to think that $\mathsf{QUERY}(\mathsf{P}, \mathsf{C}_{\varepsilon=0})$ represents our original goal of conditioning on $X = x$, the continuity of the random variable X implies that $\mathbb{P}\{X \in B_{x,0}\} = \mathbb{P}\{X = x\} = 0$ and so C_0 rejects with probability one. It follows that $\mathsf{QUERY}(\mathsf{P}, \mathsf{C}_{\varepsilon=0})$ does not halt on any input, and thus does not represent a distribution.

The underlying problem is that, in general, conditioning on a null set is mathematically undefined. The standard measure-theoretic solution is to consider the so-called "regular conditional distribution" given by conditioning on the σ-algebra generated by X—but even this approach would in general fail to solve our problem because the resulting disintegration is only defined up to a null set, and so is undefined at points (including x). (For more details, see [AFR11, §III] and [Tju80, Ch. 9].)

There have been various attempts at more constructive approaches, e.g., Tjur [Tju74, Tju75, Tju80], Pfanzagl [Pfa79], and Rao [Rao88, Rao05]. One approach worth highlighting is due to Tjur [Tju75]. There he considers additional hypotheses that are equivalent to the existence of a continuous *disintegration*, which must then be unique at all points. (We will implicitly use this notion henceforth.) Given the connection between computability and continuity, a natural question to ask is whether we might be able to extend QUERY along the lines.

Despite various constructive efforts, no general method had been found for computing conditional distributions. In fact, conditional distributions are not in general computable, as shown by Ackerman, Freer, and Roy [AFR11, Thm. 29], and it is for this reason we have defined QUERY in terms of conditioning on the event $\mathsf{C} = 1$, which, provided that C accepts with non-zero probability as we have required, is a positive-measure event. The proof of the noncomputability of conditional probability [AFR11, §VI] involves an encoding of the halting problem into a pair (X, Y) of computable (even, absolutely continuous) random variables in $[0, 1]$ such that no "version" of the conditional distribution $\mathbb{P}(Y \mid X = x)$ is a computable function of x.

What, then, is the relationship between conditioning on $X = x$ and the approximations C_ε defined above? In sufficiently nice settings, the distribution represented by $\mathsf{QUERY}(\mathsf{P}, \mathsf{C}_\varepsilon)$ converges to the desired distribution as $\varepsilon \to 0$. But as a corollary of the aforementioned noncomputability result, one sees that it is noncomputable in general to determine a value of ε from a desired level of accuracy to the desired distribution, for if there were such a general and computable relationship, one could use it to compute conditional distributions, a contradiction. Hence although such

a sequence of approximations might converge in the limit, one cannot in general compute how close it is to convergence.

On the other hand, the presence of noise in measurements can lead to computability. As an example, consider the problem of representing the distribution of Y conditioned on $X+\xi = x$, where Y, X, and x are as above, and ξ is independent of X and Y and uniformly distributed on the interval $[-\varepsilon, \varepsilon]$. While conditioning on continuous random variables is not computable in general, here it is possible. In particular, note that $\mathbb{P}(Y \mid X + \xi = x) = \mathbb{P}(Y \mid X \in B_{x,\varepsilon})$ and so $\mathsf{QUERY}(\mathsf{P}, \mathsf{C}_\varepsilon)$ represents the desired distribution.

This example can be generalized considerably beyond uniform noise (see [AFR11, Cor. 36]). Many models considered in practice posit the existence of independent noise in the quantities being measured, and so the QUERY formalism can be used to capture probabilistic reasoning in these settings as well. However, in general we should not expect to be able to reliably approximate noiseless measurements with noisy measurements, lest we contradict the noncomputability of conditioning. Finally, it is important to note that the computability that arises in the case of certain types of independent noise is a special case of the computability that arises from the existence and computability of certain conditional probability densities [AFR11, §VII]. This final case covers most models that arise in statistical practice, especially those that are finite-dimensional.

In conclusion, while we cannot hope to condition on arbitrary computable random variables, QUERY covers nearly all of the situations that arise in practice, and suffices for our purposes. Having laid the theoretical foundation for QUERY and described its connection with conditioning, we now return to the medical diagnosis example and more elaborate uses of QUERY, with a goal of understanding additional features of the formalism.

4. Conditional independence and compact representations

In this section, we return to the medical diagnosis example, and explain the way in which conditional independence leads to compact representations, and conversely, the fact that efficient probabilistic programs, like DS, exhibit many conditional independencies. We will do so through connections with the Bayesian network formalism, whose introduction by Pearl [Pea88] was a major advancement in AI.

4.1. The combinatorics of QUERY.

Humans engaging in common-sense reasoning often seem to possess an unbounded range of responses or behaviors; this is perhaps unsurprising given the enormous variety of possible situations that can arise, even in simple domains.

Indeed, the small handful of potential diseases and symptoms that our medical diagnosis model posits already gives rise to a combinatorial explosion of potential scenarios with which a doctor could be faced: among 11 potential diseases and 7 potential symptoms there are

$$3^{11} \cdot 3^7 = 387\,420\,489$$

partial assignments to a subset of variables.

Building a table (i.e., function) associating every possible diagnostic scenario with a response would be an extremely difficult task, and probably nearly impossible if one did not take advantage of some structure in the domain to devise a more compact representation of the table than a structureless, huge list. In fact, much of

AI can be interpreted as proposals for specific structural assumptions that lead to more compact representations, and the QUERY framework can be viewed from this perspective as well: the prior program DS implicitly defines a full table of responses, and the predicate can be understood as a way to index into this vast table.

This leads us to three questions: Is the table of diagnostic responses induced by DS any good? How is it possible that so many responses can be encoded so compactly? And what properties of a model follow from the existence of an efficient prior program, as in the case of our medical diagnosis example and the prior program DS? In the remainder of the section we will address the latter two questions, returning to the former in Section 5 and Section 6.

4.2. **Conditional independence.** Like DS, every probability model of 18 binary variables implicitly defines a gargantuan set of conditional probabilities. However, unlike DS, most such models have no compact representation. To see this, note that a probability distribution over k outcomes is, in general, specified by $k-1$ probabilities, and so in principle, in order to specify a distribution on $\{0,1\}^{18}$, one must specify

$$2^{18} - 1 = 262\,143$$

probabilities. Even if we discretize the probabilities to some fixed accuracy, a simple counting argument shows that most such distributions have no short description.

In contrast, Table 1 contains only

$$11 + 7 + 11 \cdot 7 = 95$$

probabilities, which, via the small collection of probabilistic computations performed by DS and described informally in the text, parameterize a distribution over 2^{18} possible outcomes. What properties of a model can lead to a compact representation?

The answer to this question is *conditional independence*. Recall that a collection of random variables $\{X_i : i \in I\}$ is *independent* when, for all finite subsets $J \subseteq I$ and measurable sets A_i where $i \in J$, we have

$$\mathbb{P}\Big(\bigwedge_{i \in J} X_i \in A_i\Big) = \prod_{i \in J} \mathbb{P}(X_i \in A_i). \tag{9}$$

If X and Y were binary random variables, then specifying their distribution would require 3 probabilities in general, but only 2 if they were independent. While those savings are small, consider instead n binary random variables X_j, $j = 1, \ldots, n$, and note that, while a generic distribution over these random variables would require the specification of $2^n - 1$ probabilities, only n probabilities are needed in the case of full independence.

Most interesting probabilistic models with compact representations will not exhibit enough independence between their constituent random variables to explain their own compactness in terms of the factorization in (9). Instead, the slightly weaker (but arguably more fundamental) notion of conditional independence is needed. Rather than present the definition of conditional independence in its full generality, we will consider a special case, restricting our attention to conditional independence with respect to a discrete random variable N taking values in some countable or finite set $\mathcal{N}$. (For the general case, see Kallenberg [Kal02, Ch. 6].) We say that a collection of random variables $\{X_i : i \in I\}$ is *conditionally independent*

given N when, for all $n \in \mathcal{N}$, finite subsets $J \subseteq I$ and measurable sets A_i, for $i \in J$, we have

$$\mathbb{P}(\bigwedge_{i \in J} X_i \in A_i \mid N = n) = \prod_{i \in J} \mathbb{P}(X_i \in A_i \mid N = n).$$

To illustrate the potential savings that can arise from conditional independence, consider n binary random variables that are conditionally independent given a discrete random variable taking k values. In general, the joint distribution over these $n + 1$ variables is specified by $k \cdot 2^n - 1$ probabilities, but, in light of the conditional independence, we need specify only $k(n + 1) - 1$ probabilities.

4.3. **Conditional independencies in DS.** In Section 4.2, we saw that conditional independence gives rise to compact representations. As we will see, the variables in DS exhibit many conditional independencies.

To begin to understand the compactness of DS, note that the 95 variables

$$\{D_1, \ldots, D_{11};\ L_1, \ldots, L_7;\ C_{1,1}, C_{1,2}, C_{2,1}, C_{2,2}, \ldots, C_{11,7}\}$$

are independent, and thus their joint distribution is determined by specifying only 95 probabilities (in particular, those in Table 1). Each symptom S_m is then derived as a deterministic function of a 23-variable subset

$$\{D_1, \ldots, D_{11};\ L_m;\ C_{1,m}, \ldots, C_{11,m}\},$$

which implies that the symptoms are conditionally independent given the diseases. However, these facts alone do not fully explain the compactness of DS. In particular, there are

$$2^{2^{23}} > 10^{10^6}$$

binary functions of 23 binary inputs, and so by a counting argument, most have no short description. On the other hand, the max operation that defines S_m does have a compact *and efficient* implementation. In Section 4.5 we will see that this implies that we can introduce additional random variables representing intermediate quantities produced in the process of computing each symptom S_m from its corresponding collection of 23-variable "parent" variables, and that these random variables exhibit many more conditional independencies than exist between S_m and its parents. From this perspective, the compactness of DS is tantamount to there being only a small number of such variables that need to be introduced. In order to simplify our explanation of this connection, we pause to introduce the idea of representing conditional independencies using graphs.

4.4. **Representations of conditional independence.** A useful way to represent conditional independence among a collection of random variables is in terms of a directed acyclic graph, where the vertices stand for random variables, and the collection of edges indicates the presence of certain conditional independencies. An example of such a graph, known as a directed graphical model or Bayesian network, is given in Figure 1. (For more details on Bayesian networks, see the survey by Pearl [Pea04]. It is interesting to note that Pearl cites Good's "causal calculus" [Goo61]—which we have already encountered in connection with our medical diagnosis example, and which was based in part on Good's wartime work with Turing on the weight of evidence—as a historical antecedent to Bayesian networks [Pea04, §70.2].)

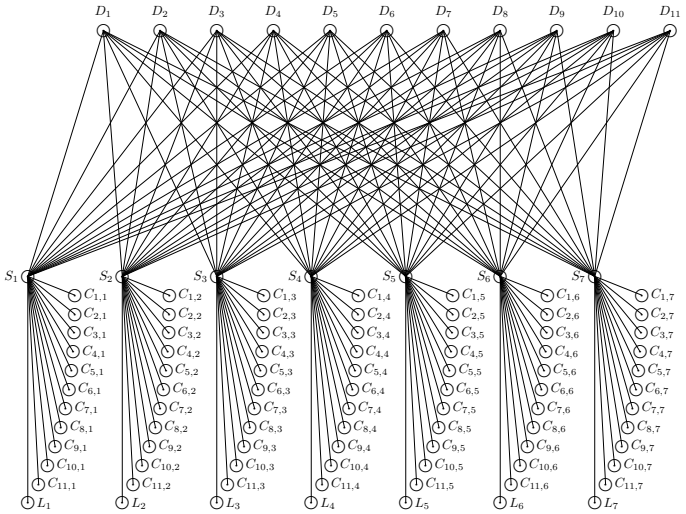

FIGURE 1. Directed graphical model representations of the conditional independence underlying the medical diagnosis example. (Note that the directionality of the arrows has not been rendered as they all simply point towards the symptoms S_m.)

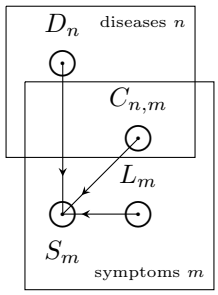

FIGURE 2. The repetitive structure Figure 1 can be partially captured by so-called "plate notation", which can be interpreted as a primitive `for`-loop construct. Practitioners have adopted a number of strategies like plate notation for capturing complicated structures.

Directed graphical models often capture the "generative" structure of a collection of random variables: informally, by the direction of arrows, the diagram captures, for each random variable, which other random variables were directly implicated in the computation that led to it being sampled. In order to understand exactly which conditional independencies are formally encoded in such a graph, we must introduce the notion of d-separation.

We determine whether a pair (x, y) of vertices are d-separated by a subset of vertices $\mathcal{E}$ as follows: First, mark each vertex in $\mathcal{E}$ with a $\times$, which we will indicate by the symbol $\bigotimes$. If a vertex with (any type of) mark has an unmarked parent,

mark the parent with a $+$, which we will indicate by the symbol $\bigoplus$. Repeat until a fixed point is reached. Let $\bigodot$ indicate unmarked vertices. Then x and y are d-separated if, for all (undirected) paths from x to y through the graph, one of the following patterns appears:

$$\bigodot \to \bigotimes \to \bigodot$$
$$\bigodot \leftarrow \bigotimes \leftarrow \bigodot$$
$$\bigodot \leftarrow \bigotimes \to \bigodot$$
$$\bigodot \to \bigodot \leftarrow \bigodot$$

More generally, if $\mathcal{X}$ and $\mathcal{E}$ are disjoint sets of vertices, then the graph encodes the conditional independence of the vertices $\mathcal{X}$ given $\mathcal{E}$ if every pair of vertices in $\mathcal{X}$ is d-separated given $\mathcal{E}$. If we fix a collection V of random variables, then we say that a directed acyclic graph G over V is a *Bayesian network* (equivalently, a directed graphical model) when the random variables in V indeed posses all of the conditional independencies implied by the graph by d-separation. Note that a directed graph G says nothing about which conditional independencies do *not* exist among its vertex set.

Using the notion of d-separation, we can determine from the Bayesian network in Figure 1 that the diseases $\{D_1, \ldots, D_{11}\}$ are independent (i.e., conditionally independent given $\mathcal{E} = \emptyset$). We may also conclude that the symptoms $\{S_1, \ldots, S_7\}$ are conditionally independent given the diseases $\{D_1, \ldots, D_{11}\}$.

In addition to encoding a set of conditional independence statements that hold among its vertex set, directed graphical models demonstrate that the joint distribution over its vertex set admits a concise factorization: For a collection of binary random variables $X_1, \ldots, X_k$, write $p(X_1, \ldots, X_k) : \{0,1\}^k \to [0,1]$ for the probability mass function (p.m.f.) taking an assignment $x_1, \ldots, x_k$ to its probability $\mathbb{P}(X_1 = x_1, \ldots, X_k = x_k)$, and write

$$p(X_1, \ldots, X_k \mid Y_1, \ldots, Y_m) : \{0,1\}^{k+m} \to [0,1]$$

for the *conditional* p.m.f. corresponding to the conditional distribution

$$\mathbb{P}(X_1, \ldots, X_k \mid Y_1, \ldots, Y_m).$$

It is a basic fact from probability that

$$\begin{aligned} p(X_1, \ldots, X_k) &= p(X_1) \cdot p(X_2 \mid X_1) \cdots p(X_k \mid X_1, \ldots, X_{k-1}) \\ &= \prod_{i=1}^{k} p(X_i \mid X_j,\ j < i). \end{aligned} \tag{10}$$

Such a factorization provides no advantage when seeking a compact representation, as a conditional p.m.f. of the form $p(X_1, \ldots, X_k \mid Y_1, \ldots, X_m)$ is determined by $2^m \cdot (2^k - 1)$ probabilities. On the other hand, if we have a directed graphical model over the same variables, then we may have a much more concise factorization. In particular, let G be a directed graphical model over $\{X_1, \ldots, X_k\}$, and write $\mathrm{Pa}(X_j)$ for the set of vertices X_i such that $(X_i, X_j) \in G$, i.e., $\mathrm{Pa}(X_j)$ are the parent vertices of X_j. Then the joint p.m.f. may be expressed as

$$p(X_1, \ldots, X_k) = \prod_{i=1}^{k} p(X_i \mid \mathrm{Pa}(X_i)). \tag{11}$$

Whereas the factorization given by (10) requires the full set of $\sum_{i=1}^{k} 2^{i-1} = 2^k - 1$ probabilities to determine, this factorization requires $\sum_{i=1}^{k} 2^{|\mathrm{Pa}(X_i)|}$ probabilities, which in general can be exponentially smaller in k.

4.5. **Efficient representations and conditional independence.** As we saw at the beginning of this section, models with only a moderate number of variables can have enormous descriptions. Having introduced the directed graphical model formalism, we can use DS as an example to explain why, roughly speaking, the output distributions of efficient probabilistic programs exhibit many conditional independencies.

What does the efficiency of DS imply about the structure of its output distribution? We may represent DS as a small boolean circuit whose inputs are random bits and whose 18 output lines represent the diseases and symptom indicators. Specifically, assuming the parameters in Table 1 were dyadics, there would exist a circuit composed of constant-fan-in elements implementing DS whose size grows linearly in the number of diseases and in the number of symptoms.

If we view the input lines as random variables, then the output lines of the logic gates are also random variables, and so we may ask: what conditional independencies hold among the circuit elements? It is straightforward to show that the circuit diagram, viewed as a directed acyclic graph, is a directed graphical model capturing conditional independencies among the inputs, outputs, and internal gates of the circuit implementing DS. For every gate, the conditional probability mass function is characterized by the (constant-size) truth table of the logical gate.

Therefore, if an efficient prior program samples from some distribution over a collection of binary random variables, then those random variables exhibit many conditional independencies, in the sense that we can introduce a polynomial number of additional boolean random variables (representing intermediate computations) such that there exists a constant-fan-in directed graphical model over all the variables with constant-size conditional probability mass functions.

In Section 5 we return to the question of whether DS is a good model. Here we conclude with a brief discussion of the history of graphical models in AI.

4.6. **Graphical models and AI.** Graphical models, and, in particular, directed graphical models or Bayesian networks, played a critical role in popularizing probabilistic techniques within AI in the late 1980s and early 1990s. Two developments were central to this shift: First, researchers introduced compact, computer-readable representations of distributions on large (but still finite) collections of random variables, and did so by explicitly representing a graph capturing conditional independencies and exploiting the factorization (11). Second, researchers introduced efficient graph-based algorithms that operated on these representations, exploiting the factorization to compute conditional probabilities. For the first time, a large class of distributions were given a formal representation that enabled the design of general purpose algorithms to compute useful quantities. As a result, the graphical model formalism became a lingua franca between practitioners designing large probabilistic systems, and figures depicting graphical models were commonly used to quickly communicate the essential structure of complex, but structured, distributions.

While there are sophisticated uses of Bayesian networks in cognitive science (see, e.g., [GKT08, §3]), many models are not usefully represented by a Bayesian network.

In practice, this often happens when the number of variables or edges is extremely large (or infinite), but there still exists special structure that an algorithm can exploit to perform probabilistic inference efficiently. In the next three sections, we will see examples of models that are not usefully represented by Bayesian networks, but which have concise descriptions as prior programs.

5. Hierarchical models and learning probabilities from data

The DS program makes a number of implicit assumptions that would deserve scrutiny in a real medical diagnosis setting. For example, DS models the diseases as *a priori* independent, but of course, diseases often arise in clusters, e.g., as the result of an auto-immune condition. In fact, because of the independence and the small marginal probability of each disease, there is an *a priori* bias towards mutually exclusive diagnoses as we saw in the "explaining away" effect in (7). The conditional independence of symptoms given diseases reflects an underlying casual interpretation of DS in terms of diseases *causing* symptoms. In many cases, e.g., a fever or a sore neck, this may be reasonable, while in others, e.g., insulin resistance, it may not.

Real systems that support medical diagnosis must relax the strong assumptions we have made in the simple DS model, while at the same time maintaining enough structure to admit a concise representation. In this and the next section, we show how both the structure and parameters in prior programs like DS *can be learned from data*, providing a clue as to how a mechanical mind could build predictive models of the world simply by experiencing and reacting to it.

5.1. **Learning as probabilistic inference.**

The 95 probabilities in Table 1 eventually parameterize a distribution over 262 144 outcomes. But whence come these 95 numbers? As one might expect by studying the table of numbers, they were designed by hand to elucidate some phenomena and be vaguely plausible. In practice, these parameters would themselves be subject to a great deal of uncertainty, and one might hope to use data from actual diagnostic situations to learn appropriate values.

There are many schools of thought on how to tackle this problem, but a hierarchical Bayesian approach provides a particularly elegant solution that fits entirely within the QUERY framework. The solution is to generalize the DS program in two ways. First, rather than generating one individual's diseases and symptoms, the program will generate data for $n + 1$ individuals. Second, rather than using the fixed table of probability values, the program will start by randomly generating a table of probability values, each independent and distributed uniformly at random in the unit interval, and then proceed along the same lines as DS. Let DS$'$ stand for this generalized program.

The second generalization may sound quite surprising, and unlikely to work very well. The key is to consider the combination of the two generalizations. To complete the picture, consider a past record of n individuals and their diagnosis, represented as a (potentially partial) setting of the 18 variables $\{D_1, \ldots, D_{11}; S_1, \ldots, S_7\}$. We define a new predicate OS$'$ that accepts the $n + 1$ diagnoses generated by the generalized prior program DS$'$ if and only if the first n agree with the historical records, and the symptoms associated with the $n + 1$'st agree with the current patient's symptoms.

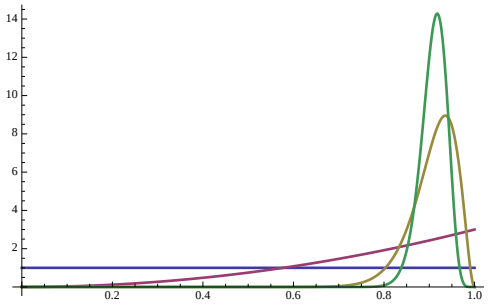

FIGURE 3. Plots of the probability density of Beta(a_1, a_0) distributions with density $f(x; \alpha_1, \alpha_0) = \frac{\Gamma(\alpha_1+\alpha_0)}{\Gamma(\alpha_1)\Gamma(\alpha_0)} x^{\alpha_1-1}(1-x)^{\alpha_0-1}$ for parameters $(1,1)$, $(3,1)$, $(30,3)$, and $(90,9)$ (respectively, in height). For parameters $\alpha_1, \alpha_0 > 1$, the distribution is unimodal with mean $\alpha_1/(\alpha_1+\alpha_0)$.

What are typical outputs from $\mathsf{QUERY}(\mathsf{DS}', \mathsf{OS}')$? For very small values of n, we would not expect particularly sensible predictions, as there are many tables of probabilities that could conceivably lead to acceptance by OS'. However, as n grows, some tables are much more likely to lead to acceptance. In particular, for large n, we would expect the hypothesized marginal probability of a disease to be relatively close to the observed frequency of the disease, for otherwise, the probability of acceptance would drop. This effect grows exponentially in n, and so we would expect that the typical accepted sample would quickly correspond with a latent table of probabilities that match the historical record.

We can, in fact, work out the conditional distributions of entries in the table in light of the n historical records. First consider a disease j whose marginal probability, p_j, is modeled as a random variable sampled uniformly at random from the unit interval. The likelihood that the n sampled values of D_j match the historical record is

$$p_j^k \cdot (1-p_j)^{n-k}, \tag{12}$$

where k stands for the number of records where disease j is present. By Bayes' theorem, in the special case of a uniform prior distribution on p_j, the density of the conditional distribution of p_j given the historical evidence is proportional to the likelihood (12). This implies that, conditionally on the historical record, p_j has a so-called Beta(α_1, α_0) distribution with mean

$$\frac{\alpha_1}{\alpha_1+\alpha_0} = \frac{k+1}{n+2}$$

and concentration parameter $\alpha_1+\alpha_0 = n+2$. Figure 3 illustrates beta distributions under varying parameterizations, highlighting the fact that, as the concentration grows, the distribution begins to concentrate rapidly around its mean. As n grows, predictions made by $\mathsf{QUERY}(\mathsf{DS}', \mathsf{OS}')$ will likely be those of runs where each disease marginals p_j falls near the observed frequency of the jth disease. In effect, the historical record data *determines* the values of the marginals p_j.

A similar analysis can be made of the dynamics of the posterior distribution of the latent parameters ℓ_m and $c_{n,m}$, although this will take us too far beyond the

scope of the present article. Abstractly speaking, in finite dimensional Bayesian models like this one satisfying certain regularity conditions, it is possible to show that the predictions of the model converge to those made by the best possible approximation within the model to the distribution of the data. (For a discussion of these issues, see, e.g., [Bar98].)

While the original DS program makes the same inferences in each case, DS′ learns to behave from experience. The key to this power was the introduction of the latent table of probabilities, modeled as random variables. This type of model is referred to as a *hierarchical Bayesian model*. The term "Bayesian" refers to the fact that we have modeled our uncertainty about the unknown probabilities by making them random and specifying a distribution that reflects our subjective uncertainty, rather than a frequency in a large random sample of patients. The term "hierarchy" refers to the fact that in the graphical model representing the program, there is yet another level of random variables (the table of probabilities) sitting above the rest of the original graphical model. More complicated models may have many more layers, or potentially even an infinite number of layers.

An interesting observation is that DS′ is even more compact than DS, as the specification of the distribution of the random table is logarithmic in the size of the table. On the other hand, DS′ relies on data to help it reduce its substantial *a priori* uncertainty regarding these values. This tradeoff—between, on the one hand, the flexibility and complexity of a model and, on the other, the amount of data required in order to make sensible predictions—is seen throughout statistical modeling. We will return to this point in Section 6.3.

Here we have seen how the parameters in prior programs can be modeled as random, and thereby learned from data by conditioning on historical diagnoses. In the next section, we consider the problem of learning not only the parameterization but the structure of the model's conditional independence itself.

6. Random structure

Irrespective of how much historical data we have, DS′ cannot go beyond the conditional independence assumptions implicit in the structure of the prior program. Just as we framed the problem of learning the table of probabilities as a probabilistic inference over a random table, we can frame the problem of identifying the correct structure of the dependence between symptoms and disease as one of probabilistic inference over random conditional independence *structure* among the model variables.

In Section 4.4, we saw that conditional independence relationships among a collection of random variables can be captured by a directed acyclic graph. The approach we will discuss involves treating this graph as a random variable, whose distribution reflects our uncertainty about the statistical dependence among the diseases and symptoms before seeing data, and whose posterior distribution reflects our updated uncertainty about these relationships once the graph is forced to explain any evidence of dependence or independence in the historical data.

The model that we describe in this section introduces several additional layers and many more latent variables. Outside of the Bayesian framework, these latent variables would typically be additional parameters that one would tune to fit the data. Typically, when one adds more parameters to a model, this improves the fit to the data at hand, but introduces a risk of "overfitting", which leads to poor

predictive performance on unseen data. However, as we will see in Section 6.3, the problem of overfitting is mitigated in this Bayesian approach, because the latent variables are not optimized, but rather sampled conditionally.

6.1. **Learning structure as probabilistic inference.** Within AI and machine learning, the problem of learning a probabilistic model from data is a quintessential example of *unsupervised learning*, and the approach of identifying a graph capturing conditional independence relationships among model variables is known as *structure learning*.

In Section 4.4 we saw that every distribution on n binary random variables $X_1, \ldots, X_n$ can be expressed in the form

$$p(X_1, \ldots, X_k) = \prod_{j=1}^{k} p_j(X_j \mid \mathrm{Pa}(X_j)). \tag{13}$$

where G is a directed acyclic graph over the set $\{X_1, \ldots, X_k\}$ of model variables; $\mathrm{Pa}(X_j)$ denotes the parent vertices of X_j; and the $p_j(\cdot \mid \cdot)$ are conditional probability mass functions specifying the distribution of each variable in terms of its parents' values.

From the perspective of this factorization, the tack we took in Section 5.1 was to assume that we knew the graphical structure G (given by DS) and learn (the parameterization of) the conditional mass functions by modeling them as random variables. We will now consider learning both ingredients simultaneously, and later pause to critique this strategy.

6.2. **A random probability distribution.** Let us return to the setting of medical diagnosis, and in particular the problem of modeling the presence/absence of the 11 diseases and 7 symptoms, represented by the variables $\{D_1, \ldots, D_{11}; S_1, \ldots, S_7\}$.

Towards this end, and with the factorization (13) in mind, consider a prior program, which we will call RPD (for *Random Probability Distribution*), that takes as input two positive integers n and D and produces as output n independent samples from a random probability distribution on $\{0,1\}^D$.

Intuitively, RPD works in the following way: First, RPD generates a random directed acyclic graph G with D vertices. Next, it generates a *random* probability mass function p, which will specify a distribution over D random variables, $X_1, \ldots, X_D$. The probability mass function will be generated so that it satisfies the conditional independencies implied by the graph G when it is viewed as a directed graphical model. The probability mass function p is generated by choosing random conditional probability mass functions $p(X_j|\mathrm{Pa}(X_j))$, one for each variable X_j as in the factorization (13). Specifically, if a variable X_j has k parents $\mathrm{Pa}(X_j)$ (which collectively can take on 2^k possible $\{0,1\}$-assignments), then we must generate 2^k probabilities, one for each $\{0,1\}$-assignment v of the parents, indicating the probability $p_{j|v}$ that $X_j = 1$ given that $\mathrm{Pa}(X_j) = v$. In particular, $p_j(1|v) = p_{j|v}$. This fully determines p. RPD then proceeds to generate n samples from p, each a list of D binary values with the same distributions as $X_1, \ldots, X_D$.

More formally, RPD begins by sampling a directed acyclic graph G uniformly at random from the set $\mathcal{G}_D$ of all directed acyclic graphs over the vertex set $\{X_1, \ldots, X_D\}$. For every vertex j and every $\{0,1\}$-assignment v to X_i's parents $\mathrm{Pa}(X_j)$, we sample a probability value $p_{j|v}$ uniformly at random from $[0,1]$. Let

$j_1, \ldots, j_D$ be a topological ordering of the vertices of G. We then repeat the following procedure n times: First, sample $X_{j_1} \in \{0, 1\}$ with mean $p_{j_1|()}$, and then for $i = 2, \ldots, D$, sample $X_{j_i} \in \{0, 1\}$ with mean $p_{j_i|v}$ where $v = (X_p : p \in \mathrm{Pa}(j_i))$ is the $\{0, 1\}$-assignment of X_j's parents. We then output the variables in order $X_1, \ldots, X_D$, and repeat until we have produced n such samples as output.

With RPD fully specified, let us now consider the output of

$$\mathsf{QUERY}(\mathsf{RPD}(n + 1, 18), \mathsf{OS}') \tag{14}$$

where OS' is defined as in Section 5.1, accepting $n + 1$ diagnoses if and only if the first n agree with historical records, and the symptoms associated with the $n+1$'st agree with the current patient's symptoms. (Note that we are identifying each output $(X_1, \ldots, X_{11}, X_{12}, \ldots, X_{18})$ with a diagnosis $(D_1, \ldots, D_{11}, S_1, \ldots, S_7)$, and have done so in order to highlight the generality of RPD.)

As a first step in understanding RPD, one can show that, conditioned on the graph G, the conditional independence structure of each of its n outputs $(X_1, \ldots, X_D)$ is precisely captured by G, when viewed as a Bayesian network (i.e., the distribution of the X's satisfies the factorization (13)). It is then not hard to see that the probabilities $p_{j|v}$ parameterize the conditional probability mass functions, in the sense that $p(X_j = 1 \mid \mathrm{Pa}(X_j) = v) = p_{j|v}$. Our goal over the remainder of the section will be to elucidate the posterior distribution on the graph and its parameterization, in light of historical data.

To begin, we assume that we know the graph G for a particular output from (14), and then study the likely values of the probabilities $p_{j|v}$ conditioned on the graph G. Given the simple uniform prior distributions, we can in fact derive analytical expressions for the posterior distribution of the probabilities $p_{j|v}$ directly, conditioned on historical data and the particular graph structure G. In much the same was as our analysis in Section 5.1, it is easy to show that the expected value of $p_{j|v}$ on those runs accepted by QUERY is

$$\frac{k_{j|v} + 1}{n_{j|v} + 2}$$

where $n_{j|v}$ is the number of times in the historical data where the pattern $\mathrm{Pa}(X_j) = v$ arises; and $k_{j|v}$ is the number of times when, moreover, $X_j = 1$. This is simply the "smoothed" empirical frequency of the event $X_j = 1$ given $\mathrm{Pa}(X_j) = v$. In fact, the $p_{j|v}$ are conditionally Beta distributed with concentration $n_{j|v} + 2$. Under an assumption that the historical data are conditionally independent and identically distributed according to a measure P, it follows by a law of large numbers argument that these probabilities converge almost surely to the underlying conditional probability $P(X_j = 1|\mathrm{Pa}(X_j) = v)$ as $n \to \infty$.

The variance of these probabilities is one characterization of our uncertainty, and for each probability $p_{j|v}$, the variance is easily shown to scale as $n^{-1}_{j|v}$, i.e., the number of times in the historical data when $\mathrm{Pa}(X_j) = v$. Informally, this suggests that, the smaller the parental sets (a property of G), the more certain we are likely to be regarding the correct parameterization, and, in terms of QUERY, the smaller the range of values of $p_{j|v}$ we will expect to see on accepted runs. This is our first glimpse at a subtle balance between the simplicity of the graph G and how well it captures hidden structure in the data.

6.3. **Aspects of the posterior distribution of the graphical structure.** The space of directed acyclic graphs on 18 variables is enormous, and computational hardness results [Coo90, DL93, CSH08] imply there will be no simple way to summarize the structure of the posterior distribution, at least not one that suggests an efficient method in general for choosing structures with high posterior probability. It also goes without saying that one should not expect the PTM defined by (14) to halt within a reasonable time for any appreciable value of n because the probability of generating the structure that fits the data is astronomically small. However it is still instructive to understand the conceptual structure of the posterior distribution of the graph G. On the one hand, there are algorithms that operate quite differently from the naive mechanism of QUERY and work reasonably well in practice at approximating the task defined here, despite hardness results. There are also more restricted, but still interesting, versions of this task for which there exist algorithms that work remarkably well in practice and sometimes provably so [BJ03].

On the other hand, this example is worth studying because it reveals an important aspect of some hierarchical Bayesian models with regard to their ability to avoid "overfitting", and gives some insight into why we might expect "simpler" explanations/theories to win out in the short term over more complex ones.

Consider the set of probability distributions of the form (13) for a particular graph G. We will refer to these simply as the *models in* G when there is no risk of confusion. The first observation to make is that if a graph G is a strict subgraph of another graph G' on the same vertex set, then the set of models in G is a strict subset of those in G'. It follows that, no matter the data set, the best-fitting probability distribution corresponding with G' will be no worse than the best-fitting model in G. Given this observation, one might guess that samples from (14) would be more likely to come from models whose graphs have more edges, as such graphs always contain a model that fits the historical data better.

However, the truth is more subtle. Another key observation is that the posterior probability of a particular graph G does not reflect the best-fitting model in G, but rather reflects the *average* ability of models in G to explain the historical data. In particular, this average is over the random parameterizations $p_{j|v}$ of the conditional probability mass functions. Informally speaking, if a spurious edge exists in a graph G', a typical distribution from G' is less likely to explain the data than a typical distribution from the graph with that edge removed.

In order to characterize the posterior distribution of the graph, we can study the likelihood that a sample from the prior program is accepted, assuming that it begins by sampling a particular graph G. We begin by focusing on the use of each particular probability $p_{j|v}$, and note that every time the pattern $\mathrm{Pa}(X_j) = v$ arises in the historical data, the generative process produces the historical value X_j with probability $p_{j|v}$ if $X_j = 1$ and $1 - p_{j|v}$ if $X_j = 0$. It follows that the probability that the generative process, having chosen graph G and parameters $\{p_{j|v}\}$, proceeds to produce the historical data is

$$\prod_{j=1}^{D} \prod_{v} p_{j|v}^{k_{j|v}} (1 - p_{j|v})^{n_{j|v} - k_{j|v}}, \tag{15}$$

where v ranges over the possible $\{0,1\}$ assignments to $\mathrm{Pa}(X_j)$ and $k_{j|v}$ and $n_{j|v}$ are defined as above. In order to determine the probability that the generative process produces the historical data (and thus is accepted), assuming only that it

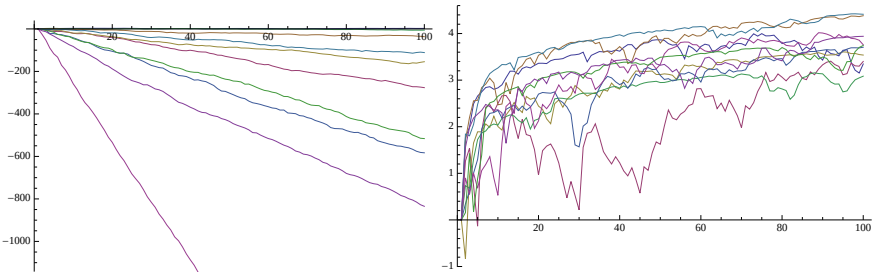

FIGURE 4. Weight of evidence for independence versus dependence (positive values support independence) of a sequence of pairs of random variables sampled from $\mathsf{RPD}(n, 2)$. (left) When presented with data from a distribution where (X, Y) are indeed dependent, the weight of evidence rapidly accumulates for the dependent model, at an asymptotically linear rate in the amount of data. (right) When presented with data from a distribution where (X, Y) are independent, the weight of evidence slowly accumulates for the independent model, at an asymptotic rate that is logarithmic in the amount of data. Note that the dependent model can imitate the independent model, but, on average over random parameterizations of the conditional probability mass functions, the dependent model is worse at modeling independent data.

has chosen graph G, we must take the expected value of (15) with respect to the uniform probability measure on the parameters, i.e., we must calculate the marginal probability of the historical data conditioned the graph G. Given the independence of the parameters, it is straightforward to show that this expectation is

$$\text{score}(G) := \prod_{j=1}^{D} \prod_{v} (n_{j|v} + 1)^{-1} \binom{n_{j|v}}{k_{j|v}}^{-1} \tag{16}$$

Because the graph G was chosen uniformly at random, it follows that the posterior probability of a particular graph G is proportional to $\text{score}(G)$.

We can study the preference for one graph G over another G' by studying the ratio of their scores:

$$\frac{\text{score}(G)}{\text{score}(G')} .$$

This score ratio is known as the *Bayes factor*, which Good termed the *Bayes–Jeffreys–Turing factor* [Goo68, Goo75], and which Turing himself called the *factor in favor of a hypothesis* (see [Goo68], [Zab12, §1.4], and [Tur12]). Its logarithm is sometimes known as the *weight of evidence* [Goo68]. The form of (16), a product over the local structure of the graph, reveals that the Bayes factor will depend only on those parts of the graphs G and G' that differ from each other.

Consider the following simplified scenario, which captures several features of learning structure from data: Fix two graphs, G and G', over the same collection of random variables, but assume that in G, two of these random variables, X and Y, have no parents and are thus independent, and in G' there is an edge from X to Y, and so they are almost surely dependent. From (16), we may deduce that the

score ratio is

$$\frac{(n_1+1)(n_0+1)}{(n+1)}\frac{\binom{n_1}{k_1}\binom{n_0}{k_0}}{\binom{n}{k}}, \tag{17}$$

where n counts the total number of observations; k counts $Y=1$; n_1 counts $X=1$; k_1 counts $X=1$ and $Y=1$; n_0 counts $X=0$; and k_0 counts $X=0$ and $Y=1$. In order to understand how the Bayes factor (17) for graph G behaves, let us first consider the case where G' is the true underlying graph, i.e., when Y is indeed dependent on X. Using the law of large numbers, and Stirling's approximation, we can reason that the evidence for G' accumulates rapidly, satisfying

$$\log\frac{\text{score}(G)}{\text{score}(G')} \sim -C\cdot n, \quad \text{a.s.},$$

for some constant $C>0$ that depends only on the joint distribution of X and Y. As a concrete example, when X and Y have mean $\frac{1}{2}$, the constant is given by

$$\log\frac{(1-d)^{d-\frac{1}{2}}}{(1+d)^{d+\frac{1}{2}}},$$

where $d=\mathbb{P}\{Y=1|X=1\}=1-\mathbb{P}\{Y=1|X=0\}$. For example, $C\to 0$ as $d\downarrow 0$; $C\approx 0.13$ when $d=1/2$; and C achieves its maximum, $\log 2$, as $d\uparrow 1$. The first plot in Figure 4 shows the progression of the weight of evidence when data is drawn from distributions generated uniformly at random to satisfy the conditional independencies captured by G'. As predicted, the evidence rapidly accumulates at a linear rate in favor of G'.

On the other hand, when G is the true underlying graph and Y is independent and X, one can show using similar techniques to above that

$$\log\frac{\text{score}(G)}{\text{score}(G')} \sim \frac{1}{2}\log n, \quad \text{a.s.}$$

The second plot in Figure 4 shows the progression of the weight of evidence when data is drawn from distributions generated uniformly at random to satisfy the conditional independencies captured by G. As predicted, the evidence accumulates, but at a much slower logarithmic rate.

In both cases, evidence accumulates for the correct model. In fact, it can be shown that the expected weight of evidence is always non-negative for the true hypothesis, a result due to Turing himself [Goo91, p. 93]. Because the prior probabilities for each graph are fixed and do not vary with the amount of data, the weight of evidence will eventually eclipse any prior information and determine the posterior probability. On the other hand, as we have seen, the evidence accumulates rapidly for dependence and much more slowly for independence and so we might choose our prior distribution to reflect this imbalance, preferring graphs with fewer edges *a priori*.[2]

6.4. **Bayes' Occam's razor.** In the example above when X and Y are independent, we see that evidence accumulates for the simpler graph over the more complex graph, despite the fact that there is almost always a parameterization of the more complex graph that assigns a higher likelihood to the data than any parametrization of the simpler graph. This phenomenon is known as Bayes' Occam's razor

[2]This analysis in terms of Bayes factors also aligns well with experimental findings on human judgments of evidence for causal structure (see, e.g., [GT05]).

[Mac03, Ch. 28], and it represents a natural way in which hierarchical models like RPD with several layers—a random graph, random conditional probability mass functions generated given the graph, and finally, the random data generated given the graph and the conditional probability mass functions—end up choosing models with intermediate complexity.

One way to understand this phenomenon is to note that, if a model has many degrees of freedom, then each configuration must be assigned, on average, less probability than it would under a simpler model with fewer degrees of freedom. Here, a graph with additional edges has more degrees of freedom, and while it can represent a strictly larger set of distributions than a simpler graph, a distribution with simple graphical structure G is assigned greater probability density under G than under a more complex graph. Thus, if fewer degrees of freedom suffice, the simpler model is preferred.

We can apply this same perspective to DS, DS′ and RPD: The RPD model has many more degrees of freedom than both DS and DS′. In particular, given enough data, RPD can fit any distribution on a finite collection of binary variables, as opposed to DS′, which cannot because it makes strong and immutable assumptions. On the other hand, with only a small amount of training data, one would expect the RPD model to have high posterior uncertainty. Indeed, one would expect much better predictions from DS′ versus RPD, if both were fed data generated by DS, especially in the low-data regime.

An important research problem is bridging the gap between RPD and DS′. Whereas DS′ makes an immutable choice for one particular structure, RPD assumes *a priori* that every graphical structure is equally likely to explain the data. If, instead, one were uncertain about the structure but expected to find some particular regular pattern in the graphical structure, one could define an alternative model RPD′ that placed a non-uniform distribution on the graph, favoring such patterns, and one could then expect better predictions when that pattern was indeed present in the data. However, one often does not know exactly which pattern might arise. But in this case, we can take the same step we took when defining RPD and consider a *random* pattern, drawn from some space of possible patterns. This would constitute an additional level to the hierarchical model. Examples of this idea are described by Mansinghka et al. [MKTG06] and Kemp et al. [KSBT07], and this technique constitutes one aspect of the general approach of "theory-based Bayesian models" [GT06, TGK06, GKT08, KT08, GT09].

Up until this point, we have considered the problem of reasoning and representing our own uncertainty in light of evidence. However, in practice, representations of uncertainty are often useful because they support decision making under uncertainty. In the next section, we show how the QUERY framework can be used to turn models of our uncertainty, including models of the effects of our own actions, into decisions.

7. Making decisions under uncertainty

Until now, we have discussed how computational processes can represent uncertain knowledge, and how these processes can be transformed using QUERY to reflect our uncertainty after incorporating new evidence. In this section, we consider the problem of making decisions under uncertainty, which will require us to reason not only about the immediate effects of the actions we take, but also about

future decisions and the uncertainty we are likely to face when making them. In the end, we will give a recursive characterization of an approximately optimal action, and show how this relates to simple feedback rules that Turing himself proposed.

The solution we describe models decisions as random variables and decision making as sampling. Using PTMs and QUERY, we construct distributions over actions a decision maker might take after reasoning about the effects of those actions and their likely success in achieving some goal or objective. The particular distributions we will construct are based in part on the exponentiated choice rule introduced by Luce [Luc59, Luc77] in the context of modeling human choices.

Our presentation extends that for the "fully observable" case given by Goodman, Mansinghka, Roy, Bonawitz, and Tenenbaum [GMR+08] and Mansinghka [Man09, §2.2.3]. In particular, recursion plays a fundamental role in our solution, and thus pushes us beyond the representational capacity of many formalisms for expressing complex probabilistic models. Even more so than earlier sections, the computational processes we define with QUERY will not be serious proposals for *algorithms*, although they will define distributions for which we might seek to implement approximate inference. However, those familiar with traditional presentations may be surprised by the ease with which we move between problems often tackled by distinct formalisms and indeed, this is a common feature of the QUERY perspective.

7.1. **Diagnosis and Treatment.** Returning to our medical diagnosis theme, consider a doctor faced with choosing between one or more treatment plans. What recommendation should they give to the patient and how might we build a system to make similar choices?

In particular, imagine a patient in need of an organ transplant and the question of whether to wait for a human donor or use a synthetic organ. There are a number of sources of uncertainty to contend with: While waiting for a human donor, the patient may become too ill for surgery, risking death. On the other hand, the time before a human organ becomes available would itself be subject to uncertainty. There is also uncertainty involved post-operation: will the organ be accepted by the patient's immune system without complication? How long should the patient expect to live in both conditions, taking into consideration the deterioration that one should expect if the patient waits quite a while before undergoing a transplant?

This situation is quite a bit more complicated. The decision as to whether to wait changes daily as new evidence accumulates, and how one should act today depends implicitly on the possible states of uncertainty one might face in the future and the decisions one would take in those situations. As we will see, we can use QUERY, along with models of our uncertainty, to make decisions in such complex situations.

7.2. **A single decision.** We begin with the problem of making a single decision between two alternative treatments. What we have at our disposal are two simulations SIM_x and SIM_y capturing our uncertainty as to the effects of those treatments. These could be arbitrarily complicated computational processes, but in the end we will expect them to produce an output 1 indicating that the resulting simulation

of treatment was successful/acceptable and a 0 otherwise.[3] In this way, both simulations act like predicates, and so we will say that a simulation accepts when it outputs 1, and rejects otherwise. In this section we demonstrate the use of PTMs and QUERY to define distributions over *actions*—here, treatments—that are likely to lead to successful outcomes.

Let RT (for *Random Treatment*) be the program that chooses a treatment $Z \in \{x, y\}$ uniformly at random and consider the output of the program

$$\mathsf{CT}^* = \mathsf{QUERY}(\mathsf{RT}, \mathsf{SIM}_Z),$$

where SIM_Z is interpreted as the program that simulates SIM_z when RT outputs $Z = z \in \{x, y\}$. The output of CT^* is a treatment, x or y, and we will proceed by interpreting this output as the treatment chosen by some decision maker.

With a view towards analyzing CT^* (named for *Choose Treatment*), let p_z be the probability that SIM_z accepts (i.e., p_z is the probability that treatment z succeeds), and assume that $p_x > p_y$. Then CT^* "chooses" treatment x with probability

$$\frac{p_x}{p_x + p_y} = \frac{\rho}{\rho + 1}$$

where $\rho := p_x/p_y$ and $\rho > 1$ by assumption. It follows that CT^* is more likely to choose treatment x and that the strength of this preference is controlled by the multiplicative ratio ρ (hence the multiplication symbol $*$ in CT^*). If $\rho \gg 1$, then treatment x is chosen essentially every time.

On the other hand, even if treatment x is twice as likely to succeed, CT^* still chooses treatment y with probability $1/3$. One might think that a person making this decision should always choose treatment x. However, it should not be surprising that CT^* does not represent this behavior because it accepts a proposed action solely on the basis of a single successful simulation of its outcome. With that in mind, let $k \geq 0$ be an integer, and consider the program

$$\mathsf{CT}^*_k = \mathsf{QUERY}(\mathsf{RT}, \mathsf{REPEAT}(k, \mathsf{SIM}_Z)),$$

where the machine REPEAT on input k and SIM_Z accepts if k independent simulations of SIM_Z all accept. The probability that CT^*_k chooses treatment x is

$$\frac{\rho^k}{\rho^k + 1}$$

and so a small multiplicative difference between p_x and p_y is exponentiated, and CT^*_k chooses the more successful treatment with all but vanishing probability as $k \to \infty$. (See the left plot in Figure 5.) Indeed, in the limit, CT^*_∞ would always choose treatment x as we assumed $p_x > p_y$. (For every k, this is the well-known exponentiated Luce choice rule [Luc59, Luc77].)

The behavior of CT^*_∞ agrees with that of a classical decision-theoretic approach, where, roughly speaking, one fixes a loss function over possible outcomes and seeks the action minimizing the expected loss. (See [DeG05] for an excellent resource on statistical decision theory.) On the other hand, classical decision theory, at least when applied naively, often fails to explain human performance: It is well-documented that human behavior does not agree with mathematically "optimal"

[3]Our discussion is couched in terms of successful/unsuccessful outcomes, rather than in terms of a real-valued loss (as is standard in classical decision theory). However, it is possible to move between these two formalisms with additional hypotheses, e.g., boundedness and continuity of the loss. See [THS06] for one such approach.

behavior with respect to straightforward formalizations of decision problems that humans face. (See Camerer [Cam11] for a discussion of models of actual human performance in strategic situations.)

While we are not addressing the question of designing efficient algorithms, there is also evidence that seeking optimal solutions leads to computational intractability. (Indeed, PSPACE-hardness [PT87] and even undecidability [MHC03] can arise in the general case of certain standard formulations.)

Some have argued that human behavior is better understood in terms of a large collection of heuristics. For example, Goldstein and Gigerenzer [GG02] propose the "recognition heuristic", which says that when two objects are presented to a human subject and only one is recognized, that the recognized object is deemed to be of greater intrinsic value to the task at hand. The problem with such explanations of human behavior (and approaches to algorithms for AI) is that they often do not explain how these heuristic arise. Indeed, a theory for how such heuristics are learned would be a more concise and explanatory description of the heuristics than the heuristics themselves. A fruitful approach, and one that meshes with our presentation here, is to explain heuristic behavior as arising from approximations to rational behavior, perhaps necessitated by intrinsic computational hardness. Human behavior would then give us clues as to which approximations are often successful in practice and likely to be useful for algorithms.

In a sense, CT^*_k could be such a model for approximately optimal behavior. However, since CT^*_k chooses an action on the basis of the ratio $\rho = p_x/p_y$, one problematic feature is its sensitivity to small differences $|p_x - p_y|$ in the absolute probabilities of success when $p_x, p_y \ll 1$. Clearly, for most decisions, a $1/10\,000$ likelihood of success is not appreciably better than a $1/20\,000$ likelihood. A result by Dagum, Karp, Luby and Ross [DKLR00] on estimating small probabilities to high relative accuracy suggests that this sensitivity in CT^*_k might be a potential source of computational hardness in efforts to design algorithms, not to mention a point of disagreement with human behavior. It stands to reason that it may be worthwhile to seek models that do not exhibit this sensitivity.

To this end, let SIM_x and SIM_y be independent simulations and consider the predicate MAJ_x (named for *Majority*) that accepts if SIM_x succeeds and SIM_y fails, rejects in the opposite situation, and chooses to accept or reject uniformly at random otherwise. We then define

$$\mathsf{CT}^+ = \mathsf{QUERY}(\mathsf{RT}, \mathsf{MAJ}_Z),$$

It is straightforward to show that treatment x is chosen with probability

$$\frac{1 + (p_x - p_y)}{2} = \frac{1 + \alpha}{2}$$

and so the output of CT^+ is sensitive to only the additive difference $\alpha = p_x - p_y$ (hence the addition symbol $+$ in CT^+). In particular, when $p_x \approx p_y$, CT^+ chooses an action nearly uniformly at random. Unlike CT^*, it is the case that CT^+ is insensitive to the ratio p_x/p_y when $p_x, p_y \approx 0$.

Similarly to CT^*_k, we may define CT^+_k by

$$\mathsf{CT}^+_k = \mathsf{QUERY}(\mathsf{RT}, \mathsf{REPEAT}(k, \mathsf{MAJ}_Z)),$$

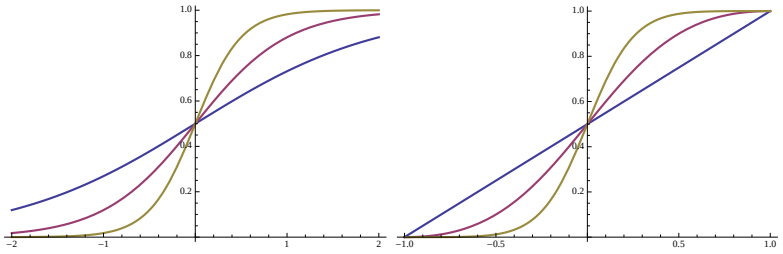

FIGURE 5. Plots of the sigmoidal curves arising from the probability of treatment x under CT_k^* (left) and CT_k^+ (right) as a function of the log probability difference $\log \rho = \log p_x - \log p_y$ and probability difference $\alpha = p_x - p_y$, respectively, between the treatments. Here $k \in \{1, 2, 4\}$. The straight line on the right corresponds to $k = 1$, and curvature increases with k.

in which case it follows that CT_k^+ accepts treatment x with probability

$$\frac{1}{1 + \left(\frac{1-\alpha}{1+\alpha}\right)^k}.$$

Figure 5 shows how this varies as a function of α for several values of k. Again, as $k \to \infty$, the decision concentrates on the treatment with the greatest probability of success, although there is always a region around $\alpha = 0$ where each treatment is almost equally likely to be chosen.

In this section, we have shown how a model of one's uncertainty about the likely success of a single action can be used to produce a distribution over actions that concentrates on actions likely to achieve success. In the next section, we will see how the situation changes when we face multiple decisions. There, the likely success of an action depends on future actions, which in turn depend on the likely success of yet-more-distant actions. Using recursion, we can extend strategies for single decisions to multiple decisions, defining distributions over sequences of actions that, under appropriate limits, agree with notions from classical decision theory, but also suggest notions of approximately optimal behavior.

7.3. **Sequences of decisions.** How can we make a sequence of good decisions over time, given a model of their effects? Naively, we might proceed along the same lines as we did in the previous section, sampling now a *sequence* of actions conditioned on, e.g., a simulation of these actions leading to a successful outcome. Unfortunately, this does not lead to a sensible notion of approximately optimal behavior, as the first action is chosen on the basis of a fixed sequence of subsequent actions that do not adapt to new observations. Certainly one should react differently when a door leads not to the next room but to a closet!

In order to recover classical notions of optimality under an appropriate limit, we need to evaluate exhaustive plans—called *policies* in the planning literature—that specify how to act in every conceivable situation that could arise. Indeed, the optimal action to take at any moment is that which, when followed by optimal behavior thereafter, maximizes the probability of a successful outcome. As one might expect, the self-referential nature of optimal behavior will lead us to recursive definitions.

Returning to our transplant scenario, each day we are faced with several possible options: waiting another day for a human donor match; running further diagnostic tests; choosing to go with the synthetic organ; etc. As time passes, observations affect our uncertainty. Observations might include the result of a diagnostic test, the appearance of the patient, how they feel, news about potential donor matches, etc. Underlying these observations (indeed, *generating* these observations) are a network of processes: the dynamics of the patient's organ systems, the biophysical mechanisms underlying the diagnostic tests included in our observations, the sociodynamics of the national donor list, etc.

Observations are the channel through which we can make inferences about the state of the underlying processes and, by reducing our uncertainty, make better decisions. On the other hand, our actions (or inaction) will influence the evolution of the underlying latent processes, and so, in order to choose good actions, we must reason not only about future observations (including eventual success or failure) but our own future actions.

While their may be many details in any particular sequential decision task, we can abstract away nearly all of them. In particular, at any point, the sequence of observations and actions that have transpired constitutes our *belief state*, and our model of the underlying latent processes and the effects of our actions boils down to a description of how our belief state evolves as we take actions and make observations. More concretely, a model for a sequential decision task is captured by a PTM $\mathsf{NEXTSTATE}$, which takes a belief state and an action as input and returns the new, random belief state arising from making an additional observation. Certain belief states are *terminal* and correspond either with a successful or unsuccessful outcome.

The internal details of $\mathsf{NEXTSTATE}$ can be arbitrarily complex, potentially representing faithful attempts at simulating the types of processes listed above, e.g., employing detailed models of physiology, diagnostic techniques, the typical progression of the national donor list, success rates of organ transplants, life expectancy under various conditions, etc. Our goal is to transform the computational process $\mathsf{NEXTSTATE}$ characterizing the sequential decision task into a computational process representing a *stochastic* policy that maps belief states to *distributions on* actions that have a high probability of success.

To begin, we describe a PTM $\mathsf{OUTCOME}$, which uniformly in a belief state b and index π for a stochastic policy, simulates an outcome resulting from following the policy π, starting from a belief state b. We may describe the behavior of $\mathsf{OUTCOME}$ inductively as follows: First, a check is made as to whether b is a terminal belief state. If so, the machine halts and returns 1 (accept) or 0 (reject) depending on whether b represents a successful or unsuccessful outcome, respectively. Otherwise, $\mathsf{OUTCOME}$ evaluates $\pi(b)$, producing a random action a, and then performs an independent simulation of $\mathsf{NEXTSTATE}(b, a)$ to produce a new belief state b', at which point the process repeats anew. In order to simplify the analysis, we will make the following assumptions: First, we will assume that there is some positive integer M such that, for every belief state b and policy π, we have that $\mathsf{OUTCOME}(b, \pi)$ halts within M iterations. Second, for every non-terminal belief state b and policy π, we will assume that $\mathsf{OUTCOME}(b, \pi)$ accepts with positive probability.

We can now cast the problem of choosing the first of a sequence of actions into the single decision framework as follows: Let $\mathsf{SIM}_{b,\pi,z}$ be the PTM that, uniformly in a belief state b, index π for a stochastic policy, and action z, simulates $\mathsf{NEXTSTATE}(b, z)$, producing an updated belief state b', and then simulates $\mathsf{OUTCOME}(b', \pi)$, randomly accepting or rejecting depending on whether the simulation of the policy π starting from b' resulted in a successful outcome or not. Let RA be a PTM that samples an action uniformly at random. Then

$$\mathsf{ACT}(b, \pi) := \mathsf{QUERY}(\mathsf{RA}, \mathsf{SIM}_{b,\pi,Z}) \tag{18}$$

represents a distribution on actions that concentrates more mass on the action leading to a higher probability of success under the policy π. Here $\mathsf{SIM}_{b,\pi,z}$ plays a role similar to that played by SIM_z in the single decision framework described earlier. The two additional inputs are needed because we must assign an action (or more accurately, a distribution over actions) to every belief state and policy. As before, we can amplify our preference for the action having the higher probability of success by asking for k simulations to succeed.

In order to determine a complete policy, we must specify the policy π that governs future behavior. The key idea is to choose actions in the future according to (18) as well, and we can implement this idea using recursion. In particular, by Kleene's recursion theorem, there is a PTM POLICY satisfying

$$\mathsf{POLICY}(b) = \mathsf{ACT}(b, \mathsf{POLICY}). \tag{19}$$

The simplifying assumptions we made above are enough to guarantee that POLICY halts with probability one on every belief state. Those familiar with Bellman's "principle of optimality" [Bel57] will notice that (18) and (19) are related to the value iteration algorithm for Markov decision processes [How60].

In order to understand the behavior of POLICY, consider the following simple scenario: a patient may or may not have a particularly serious disease, but if they do, an special injection will save them. On the other hand, if a patient is given the same injection but does not have the condition, there is a good chance of dying from the injection itself. Luckily, there is a diagnostic test that reliably detects the condition. How would POLICY behave?

More concretely, we will assume that the patient is sick with the disease with probability 0.5, and that, if the patient is sick, the injection will succeed in curing the patient with probability 0.95, but will kill them with probability 0.8 if they are not sick with the disease. If the patient is sick, waiting things out is likely to succeed with probability 0.1. Finally, the diagnostic is accurate with probability 0.75. In Figure 6, we present the corresponding belief state transition diagram capturing the behavior of $\mathsf{NEXTSTATE}$.

In order to understand the behavior of POLICY at the initial belief state ε, we begin by studying its behavior after receiving test results. In particular, having received a negative test result, the WAIT action succeeds with probability 0.91 and the INJECT action succeeds with probability 0.275, and so $\mathsf{ACT}(\text{Negative Test}, \mathsf{POLICY})$ chooses to WAIT with probability ≈ 0.77. (As $k \to \infty$, WAIT is asymptotically almost always chosen.) On the other hand, having received a positive test result, $\mathsf{ACT}(\text{Positive Test}, \mathsf{POLICY})$ chooses to WAIT with probability ≈ 0.18. (Likewise, as $k \to \infty$, INJECT is asymptotically almost always chosen.) These values imply that $\mathsf{SIM}_{\varepsilon,\mathsf{POLICY},\mathrm{TEST}}$ accepts with probability ≈ 0.76, and so $\mathsf{ACT}(\varepsilon, \mathsf{POLICY})$ assigns probability 0.40, 0.29, and 0.31 to TEST, WAIT,

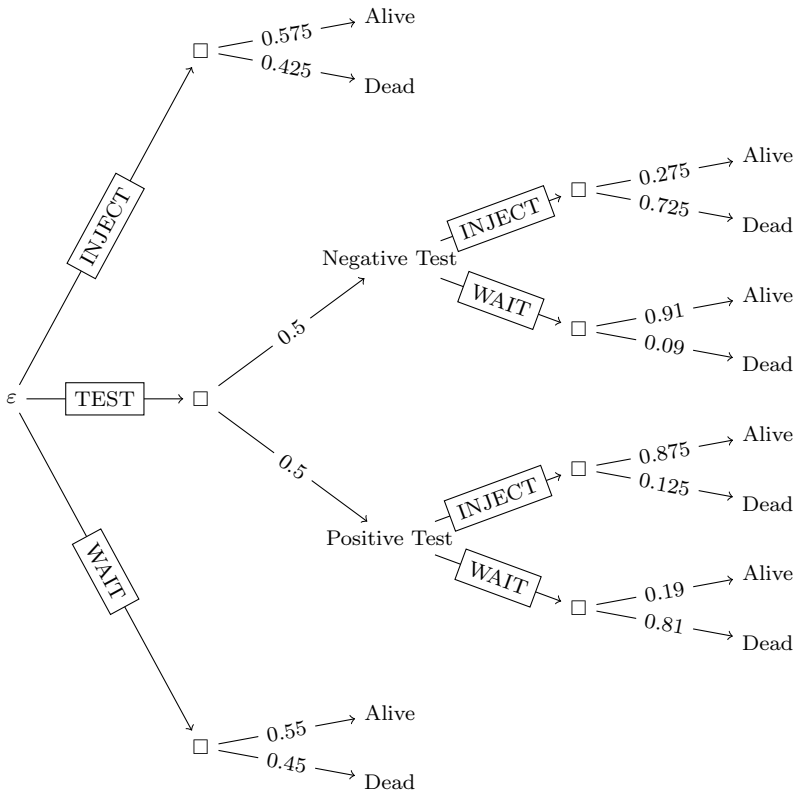

FIGURE 6. A visualization of NEXTSTATE for a simple diagnostic scenario. The initial belief state, ε, corresponds with the distribution assigning equal probability to the patient having the disease and not having the disease. Edges corresponding with actions are labeled with the name of the action in a box. For example, three actions are initially available: *waiting* it out, running a diagnostic *test*, and deciding to administer the *injection*. Square nodes (□) represent the random belief state transitions that follow an action. Edges leaving these nodes are labeled with the probabilities of the transitions.

and INJECT, respectively. As $k \to \infty$, we see that we would asymptotically almost always choose to run a diagnostic test initially, as it significantly reduces our uncertainty. This simple example only hints at the range of behaviors that can arise from POLICY in complicated sequential decision tasks.

In the definition of ACT, we choose a random action conditioned on future success. Although we can find policies that optimize the probability of success by taking $k \to \infty$, there are good reasons to not use a random action and instead use an initial policy that incorporates one's prior knowledge about the optimal action, much like it is advantageous to use a so-called admissible heuristic in search algorithms like A^*. It is interesting to note that if the initial policy succeeds with very high probability then the QUERY expression above, viewed as algorithms,

is even efficient. Returning to a quote in Section 1.1, Turing himself noted that knowing which actions to try first would "make the difference between a brilliant and a footling reasoner" and that this knowledge might be "produced by the machine itself, *e.g.* by scientific induction" [Tur50, p. 458]. Indeed, recent work in Bayesian reinforcement learning has demonstrated the utility of more informative prior distributions on policies [DWRT10, WGR+11].

7.4. **Turing's insights.** There are a wide range of algorithms that have been developed to make decisions under uncertainty. Those familiar with economics and AI may see the connection between our analysis and the Markov Decision Process (MDP) formalism [How60], where the decision maker has full observability, and the Partially Observable MDP (POMDP) formalism, where the agent, like above, has access to only part of the state (see [Mon82] for a classic survey and [KLC98] for an early AI paper introducing the formalism). In AI, these subjects are studied in an area known as *reinforcement learning*, which is in general the study of algorithms that can learn without receiving immediate feedback on their performance. (For a classic survey on reinforcement learning, see [KLM96].)

A popular class of reinforcement learning algorithms are collectively called Q-learning, and were first introduced by Watkins [Wat89]. The simplest variants work by estimating the probability that an action a leads eventually to success, starting from a belief state b and assuming that all subsequent actions are chosen optimally. This function, known as the Q- or action-value function is related to

$$\mathsf{SIM}_{b,\mathsf{POLICY},a}$$

when viewed as a function of belief state b and action a. In particular, the latter is an approximation to the former. In Q-learning, estimates of this function are produced on the basis of experience interacting with the environment. Under certain conditions, the estimate of the Q-function can be shown to converge to its true value [WD92].

It is instructive to compare the Q-learning algorithm to proposals that Turing himself made. In the course of a few pages in his 1950 article, Turing suggests mechanisms that learn by feedback in ways similar to methods in *supervised learning* (immediate feedback) and reinforcement learning (delayed feedback):

> We normally associate punishments and rewards with the teaching process. Some simple child machines can be constructed or programmed on this sort of principle. The machine has to be so constructed that events which shortly preceded the occurrence of a punishment signal are unlikely to be repeated, whereas a reward signal increased the probability of repetition of the events which led up to it. [Tur50, p. 457]

This quote describes behavior that, in broad strokes, lines up well with how a reinforcement learning algorithm such as Q-learning chooses which actions to take. Yet such simple approaches to learning from trial and error, or teaching by reward and punishment, do not capture the most powerful ways humans learn and teach each other. The last several decades of research in cognitive development [Car09, Gop12, Sch12] have emphasized the many ways in which children's learning is more driven by intrinsic curiosity and motivation than by immediate external rewards and punishments, more like the discovery and refinement of scientific theories than the mechanisms of Q-learning or the process Turing describes above. Indeed, a

close analysis of the recursively defined POLICY would show that its behavior is far more sophisticated than that of Q-learning. In particular, Q-learning does not take advantage of the model NEXTSTATE, essentially assuming that there is no predictable structure in the evolution of the belief state. On the other hand, POLICY will perform information-gathering tasks, which could be construed as curiosity-driven, but are also rational acts that may themselves improve the agent's future ability to act.

Turing himself also realized that more sophisticated learning mechanisms would be required, that a machine would have to learn from "unemotional" channels of communication, which, in the language of this section, would correspond with patterns in the observations themselves not directly linked to eventual success or failure. This type of *unsupervised* learning would be useful if the goals or criteria for success changed, but the environment stayed the same. Turing envisioned that the memory store of a child-machine

> [...] would be largely occupied with definitions and propositions. The propositions would have various kinds of status, *e.g.* well-established facts, conjectures, mathematically proved theorems, statements given by an authority, expressions having the logical form of proposition but not belief-value. [Tur50, p. 457]

The idea of using a logical language as an underlying representation of knowledge has been studied since the early days of AI, and was even proposed as a means to achieve common-sense reasoning by contemporaries of Turing, such as McCarthy [McC68]. The problem of learning logical formulae from data, especially in domains with complex, discrete structure, is actively pursued today by researchers in Statistical Relational Learning [GT07] and Inductive Logic Programming [Mug91].

Turing imagined that the same collection of logical formulae would also pertain to decision-making:

> Certain propositions may be described as 'imperatives'. The machine should be so constructed that as soon as an imperative is classed as 'well-established' the appropriate action automatically takes place. [Tur50, p. 457]

A similar mechanism would later be used in *expert systems*, which first appeared in the 1960s and rose to popularity in the 1980s as they demonstrated their usefulness and commercial viability. (See [LBFL93] for a retrospective on one of the first successful expert system.)

Having seen how QUERY can be used to make decisions under uncertainty, we now conclude with some general thoughts about the use of QUERY in common-sense reasoning.

8. Towards common-sense reasoning

As we have seen, the QUERY framework can be used to model many common-sense reasoning tasks, and the underlying formalism owes much to Turing, as do several details of the approach. In many of the applications we have considered, the key step is providing QUERY with an appropriate model—a generative description of the relevant aspects of nature.

In modeling, too, Turing was a pioneer. As evidenced by his diverse body of work across computation, statistics and even morphogenesis [Tur52], Turing excelled in

building simple models of complex natural phenomena. In morphogenesis, in particular, his reaction-diffusion model proposed a particular sort of simplified chemical interactions as a way to understand visible patterns on animals, but also potentially leaf arrangements and even aspects of embryo formation [Cop04, p. 509]. Turing hoped to make progress in understanding these biological phenomena by carefully analyzing simplified mathematical models of these natural systems.

The medical diagnosis model DS that we examined in Section 2 is a crude attempt at the same sort of mechanical/computational description of the natural patterns of co-occurrence of diseases and symptoms. As we have seen, using a generative model of these patterns as an input to QUERY, we can reason about unseen processes, like diseases, from observed ones, like symptoms. We expect the inferences produced by QUERY to be diagnostically more useful when the generative model reflects a deep scientific understanding of the mechanisms underlying the pattern of diseases and symptoms that we find in the human population. But we also expect the inferences produced by QUERY to reflect natural patterns of common-sense reasoning among lay people when fed a model that, like DS, represents a cruder state of uncertainty.

These explanations typify *computational theories* in the sense of Marr [Mar82], and especially the Bayesian accounts developed in Anderson's rational analyses [And90], Shepard's investigations of universal laws [She87], ideal observer models [Gei84, KY03], and the work of Tenenbaum, Griffiths, and colleagues on concept learning and generalization [TG01]; see also Oaksford and Chater's notion of Bayesian rationality [OC98, OC07].

A recent body of literature demonstrates that many human inferences and decisions in *natural* situations are well predicted by probabilistic inference in models defined by simple generative descriptions of the underlying causal structure. One set of examples concerns "inverse planning" in social cognition [BGT07, BST09, GBT09, UBM+09, Bak12] and in language [GS12]. Using the approximate planning/decision-making framework discussed in Section 7 as part of a generative model for human behavior, this research considers situations in which a human reasons about the goals of other agents having only observed their actions—hence the term *inverse* planning—by assuming that the other agents are acting nearly optimally in attaining their goals. These approaches can lead to models that are good at making quantitative predictions of human judgements about the intentions of others. As another example, the "intuitive physics" research by Hamrick and Battaglia [HBT11, Ham12] aims to explain human reasoning about the physical world by positing that we use a crude ability to simulate simple physical models in our minds. Other examples include pragmatics in language [FG12, SG] and counterfactual reasoning [GGLT12, MUS+12]. With all of these examples, there is a rather large gap between defining the given problem in that way and being able to computationally solve it. But still there is substantial clarity brought about by the view of using QUERY along with a generative description of underlying causal structure.

Of course, this raises the questions: how do we obtain such models? In particular, how can or should we build them when they are not handed to us? And is there any hope of automating the process by which we, as scientists, invent such models upon mental reflection? These are hard scientific problems, and we have addressed them in only a very narrow sense. In Section 5, we showed how the parameters to the DS could be learned from data by constructing a larger generative process, DS′, wherein these parameters are also expressed as being uncertain. We also showed,

in Section 6, how the conditional independence structure implicit in the DS model could itself be learned, via inference in the model RPD.

These examples suggest that one possible approach to learning models is via a more abstract version of the sort of inference we have been describing, and this approach is roughly that taken in "theory-based Bayesian models" approach of Griffiths, Kemp, and Tenenbaum [GT06, GKT08, KT08, GT09]. Some examples include attempts to learn structural forms [TGK06], and to learn a theory of causality [GUT11]. There are of course many other proposed approaches to learning models, some with flavors very different from those considered in this paper. Finally, the question of how to approach common-sense reasoning remains. Perhaps common-sense involves knowing how to build one's own models, in a general enough setting to encompass all of experience. It is clearly far too early to tell whether this, or any current approach, will succeed.

Although Turing did not frame his AI work in terms of conditioning, his generative models for morphogenesis did capture one of the key ideas presented here—that of explaining a natural phenomenon via a detailed stochastic model of the underlying causal process. More generally, given the wide range of Turing's ideas that appear together in the approach to AI we have described, it is fascinating to speculate on what sort of synthesis Turing might have made, if he had had the opportunity.

A tantalizing clue is offered by his wartime colleague I. J. Good: Near the end of his life, Turing was a member, along with several prominent neurologists, statisticians, and physicists, of a small exclusive discussion group known as the *Ratio Club*, named in part because of "the dependence of perception on the judging of ratios" [Goo91, p. 101].

One can only guess at how Turing might have combined his computational insight, statistical brilliance, and passion for modeling natural phenomena into still further pursuits in AI.

Acknowledgements

The authors would like to thank Nate Ackerman, Chris Baker, Owain Evans, Leslie Kaelbling, Jonathan Malmaud, Vikash Mansinghka, and Timothy O'Donnell for very helpful discussions and critical feedback on drafts, and Noah Goodman, Susan Holmes, Max Siegel, Andreas Stuhlmüller, and Sandy Zabell for useful conversations. This publication was made possible through the support of grants from the John Templeton Foundation and Google. The opinions expressed in this publication are those of the authors and do not necessarily reflect the views of the John Templeton Foundation. This paper was partially written while C.E.F. and D.M.R. were participants in the program *Semantics and Syntax: A Legacy of Alan Turing* at the Isaac Newton Institute for the Mathematical Sciences. D.M.R. is supported by a Newton International Fellowship and Emmanuel College.

References

[AB] J. Avigad and V. Brattka. Computability and analysis: The legacy of Alan Turing. In R. Downey, editor, *Turing's Legacy*. Cambridge University Press, Cambridge, UK.

[AFR11] N. L. Ackerman, C. E. Freer, and D. M. Roy. Noncomputable conditional distributions. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science (LICS 2011)*, pages 107–116. IEEE Computer Society, 2011.

[And90] J. R. Anderson. *The Adaptive Character of Thought*. Erlbaum, Hillsdale, NJ, 1990.

[Bak12] C. L. Baker. *Bayesian Theory of Mind: Modeling Human Reasoning about Beliefs, Desires, Goals, and Social Relations*. PhD thesis, Massachusetts Institute of Technology, 2012.

[Bar98] A. R. Barron. Information-theoretic characterization of Bayes performance and the choice of priors in parametric and nonparametric problems. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics 6: Proceedings of the Sixth Valencia International Meeting*, pages 27–52, 1998.

[Bel57] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[BGT07] C. L. Baker, N. D. Goodman, and J. B. Tenenbaum. Theory-based social goal inference. In *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, pages 1447–1452, 2007.

[BJ03] F. R. Bach and M. I. Jordan. Learning graphical models with Mercer kernels. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, pages 1009–1016. The MIT Press, Cambridge, MA, 2003.

[Bla97] J. Blanck. Domain representability of metric spaces. *Annals of Pure and Applied Logic*, 83(3):225 – 247, 1997.

[BST09] C. L. Baker, R. Saxe, and J. B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[Cam11] C. F. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction*. The Roundtable Series in Behavioral Economics. Princeton University Press, 2011.

[Car09] S. Carey. *The Origin of Concepts*. Oxford University Press, New York, 2009.

[Coo90] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

[Cop04] B. J. Copeland, editor. *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life: Plus the Secrets of Enigma*. Oxford University Press, Oxford, 2004.

[CP96] B. J. Copeland and D. Proudfoot. On Alan Turing's anticipation of connectionism. *Synthese*, 108(3):pp. 361–377, 1996.

[CSH08] V. Chandrasekaran, N. Srebro, and P. Harsha. Complexity of inference in graphical models. In *Proceedings of the Twenty Fourth Conference on Uncertainty in Artificial Intelligence (UAI 2008)*, pages 70–78, Corvalis, Oregon, 2008. AUAI Press.

[DeG05] M. H. DeGroot. *Optimal Statistical Decisions*. Wiley Classics Library. Wiley, 2005.

[DKLR00] P. Dagum, R. Karp, M. Luby, and S. Ross. An optimal algorithm for Monte Carlo estimation. *SIAM Journal on Computing*, 29(5):1484–1496, 2000.

[DL93] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, 1993.

[dMSS56] K. de Leeuw, E. F. Moore, C. E. Shannon, and N. Shapiro. Computability by probabilistic machines. In *Automata Studies*, Annals of Mathematical Studies, no. 34, pages 183–212. Princeton University Press, Princeton, N. J., 1956.

[DWRT10] F. Doshi-Velez, D. Wingate, N. Roy, and J. Tenenbaum. Nonparametric Bayesian policy priors for reinforcement learning. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23 (NIPS 2010)*, pages 532–540. 2010.

[Eda96] A. Edalat. The Scott topology induces the weak topology. In *11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996)*, pages 372–381. IEEE Computer Society Press, Los Alamitos, CA, 1996.

[EH98] A. Edalat and R. Heckmann. A computational model for metric spaces. *Theoretical Computer Science*, 193(1-2):53–73, 1998.

[FG12] M. C. Frank and N. D. Goodman. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998, 2012.

[Gác05] P. Gács. Uniform test of algorithmic randomness over a general space. *Theoretical Computer Science*, 341(1-3):91–137, 2005.

[GBT09] N. D. Goodman, C. L. Baker, and J. B. Tenenbaum. Cause and intent: Social reasoning in causal learning. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, pages 2759–2764, 2009.

[Gei84] W. S. Geisler. Physical limits of acuity and hyperacuity. *Journal of the Optical Society of America A*, 1(7):775–782, 1984.

[GG02] D. G. Goldstein and G. Gigerenzer. Models of ecological rationality: The recognition heuristic. *Psychological Review*, 109(1):75–90, January 2002.

[GG12] T. Gerstenberg and N. D. Goodman. Ping pong in Church: Productive use of concepts in human probabilistic inference. In N. Miyake, D. Peebles, and R. P. Cooper, editors, *Proceedings of the Thirty-Fourth Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society, 2012.

[GGLT12] T. Gerstenberg, N. D. Goodman, D. A. Lagnado, and J. B. Tenenbaum. Noisy Newtons: Unifying process and dependency accounts of causal attribution. In N. Miyake, D. Peebles, and R. P. Cooper, editors, *Proceedings of the Thirty-Fourth Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society, 2012.

[GHR10] S. Galatolo, M. Hoyrup, and C. Rojas. Effective symbolic dynamics, random points, statistical behavior, complexity and entropy. *Information and Computation*, 208(1):23–41, 2010.

[GKT08] T. L. Griffiths, C. Kemp, and J. B. Tenenbaum. Bayesian models of cognition. In *Cambridge Handbook of Computational Cognitive Modeling*. Cambridge University Press, 2008.

[GMR+08] N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum. Church: A language for generative models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI 2008)*, pages 220–229, Corvalis, Oregon, 2008. AUAI Press.

[Goo61] I. J. Good. A causal calculus. I. *The British Journal for the Philosophy of Science*, 11:305–318, 1961.

[Goo68] I. J. Good. Corroboration, explanation, evolving probability, simplicity and a sharpened razor. *The British Journal for the Philosophy of Science*, 19(2):123–143, 1968.

[Goo75] I. J. Good. Explicativity, corroboration, and the relative odds of hypotheses. *Synthese*, 30(1):39–73, 1975.

[Goo79] I. J. Good. A. M. Turing's statistical work in World War II. *Biometrika*, 66(2):393–396, 1979.

[Goo91] I. J. Good. Weight of evidence and the Bayesian likelihood ratio. In C. G. G. Aitken and D. A. Stoney, editors, *The Use Of Statistics In Forensic Science*. Ellis Horwood, Chichester, 1991.

[Goo00] I. J. Good. Turing's anticipation of empirical Bayes in connection with the cryptanalysis of the naval Enigma. *Journal of Statistical Computation and Simulation*, 66(2):101–111, 2000.

[Gop12] A. Gopnik. Scientific thinking in young children: Theoretical advances, empirical research, and policy implications. *Science*, 337(6102):1623–1627, 2012.

[GS12] N. D. Goodman and A. Stuhlmüller. Knowledge and implicature: Modeling language understanding as social cognition. In N. Miyake, D. Peebles, and R. P. Cooper, editors, *Proceedings of the Thirty-Fourth Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society, 2012.

[GSW07] T. Grubba, M. Schröder, and K. Weihrauch. Computable metrization. *Mathematical Logic Quarterly*, 53(4-5):381–395, 2007.

[GT05] T. L. Griffiths and J. B. Tenenbaum. Structure and strength in causal induction. *Cognitive Psychology*, 51(4):334–384, 2005.

[GT06] T. L. Griffiths and J. B. Tenenbaum. Optimal predictions in everyday cognition. *Psychological Science*, 17(9):767–773, 2006.

[GT07] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

[GT09] T. L. Griffiths and J. B. Tenenbaum. Theory-based causal induction. *Psychological Review*, 116(4):661–716, 2009.

[GT12] N. D. Goodman and J. B. Tenenbaum. The probabilistic language of thought. In preparation, 2012.

[GTFG08] N. D. Goodman, J. B. Tenenbaum, J. Feldman, and T. L. Griffiths. A rational analysis of rule-based concept learning. *Cognitive Science*, 32(1):108–154, 2008.

[GTO11] N. D. Goodman, J. B. Tenenbaum, and T. J. O'Donnell. Probabilistic models of cognition. *Church wiki*, 2011. http://projects.csail.mit.edu/church/wiki/Probabilistic_Models_of_Cognition.

[GUT11] N. D. Goodman, T. D. Ullman, and J. B. Tenenbaum. Learning a theory of causality. *Psychological Review*, 118(1):110–119, 2011.

[Ham12] J. Hamrick. Physical Reasoning in Complex Scenes is Sensitive to Mass. Master's thesis, M. Eng., Massachusetts Institute of Technology, Cambridge, MA, 2012.

[HBT11] J. Hamrick, P. W. Battaglia, and J. B. Tenenbaum. Internal physics models guide probabilistic judgments about object dynamics. In C. Carlson, C. Hölscher, and T. Shipley, editors, *Proceedings of the Thirty-Third Annual Conference of the Cognitive Science Society*, pages 1545–1550. Austin, TX: Cognitive Science Society, 2011.

[Hem02] A. Hemmerling. Effective metric spaces and representations of the reals. *Theoretical Computer Science*, 284(2):347–372, 2002.

[Hod97] A. Hodges. *Turing: A Natural Philosopher*. Phoenix, London, 1997.

[How60] R. A. Howard. *Dynamic Programming and Markov Processes*. The MIT Press, Cambridge, MA, 1960.

[Kal02] O. Kallenberg. *Foundations of modern probability*. Probability and its Applications. Springer, New York, 2nd edition, 2002.

[KGT08] C. Kemp, N. D. Goodman, and J. B. Tenenbaum. Learning and using relational theories. In *Advances in Neural Information Processing Systems 20 (NIPS 2007)* , 2008.

[KLC98] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[KLM96] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[KSBT07] C. Kemp, P. Shafto, A. Berke, and J. B. Tenenbaum. Combining causal and similarity-based reasoning. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, pages 681–688. The MIT Press, Cambridge, MA, 2007.

[KT08] C. Kemp and J. B. Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31):10687–10692, 2008.

[KY03] D. Kersten and A. Yuille. Bayesian models of object perception. *Current Opinion in Neurobiology*, 13(2):150–158, 2003.

[LBFL93] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg. DENDRAL: A case study of the first expert system for scientific hypothesis formation. *Artificial Intelligence*, 61(2):209–261, 1993.

[Luc59] R. D. Luce. *Individual Choice Behavior*. John Wiley, New York, 1959.

[Luc77] R. D. Luce. The choice axiom after twenty years. *Journal of Mathematical Psychology*, 15(3):215–233, 1977.

[Mac03] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, UK, 2003.

[Man09] V. K. Mansinghka. *Natively Probabilistic Computation*. PhD thesis, Massachusetts Institute of Technology, 2009.

[Man11] V. K. Mansinghka. Beyond calculation: Probabilistic computing machines and universal stochastic inference. *NIPS Philosophy and Machine Learning Workshop*, 2011.

[Mar82] D. Marr. *Vision*. Freeman, San Francisco, 1982.

[McC68] J. McCarthy. Programs with common sense. In *Semantic Information Processing*, pages 403–418. The MIT Press, 1968.

[MHC03] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1–2):5–34, 2003.

[MJT08] V. K. Mansinghka, E. Jonas, and J. B. Tenenbaum. Stochastic digital circuits for probabilistic inference. Technical Report MIT-CSAIL-TR-2008-069, Massachusetts Institute of Technology, 2008.

[MKTG06] V. K. Mansinghka, C. Kemp, J. B. Tenenbaum, and T. L. Griffiths. Structured priors for structure learning. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI 2006)*, pages 324–331, Arlington, Virginia, 2006. AUAI Press.

[Mon82] G. E. Monahan. A survey of Partially Observable Markov Decision Processes: Theory, models, and algorithms. *Management Science*, 28(1):pp. 1–16, 1982.

[MP43] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, 1943.

[MR] V. K. Mansinghka and D. M. Roy. Stochastic inference machines. In preparation.

[Mug91] S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[MUS+12] J. McCoy, T. D. Ullman, A. Stuhlmüller, T. Gerstenberg, and J. B. Tenenbaum. Why blame Bob? Probabilistic generative models, counterfactual reasoning, and blame attribution. In N. Miyake, D. Peebles, and R. P. Cooper, editors, *Proceedings of the Thirty-Fourth Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society, 2012.

[OC98] M. Oaksford and N. Chater, editors. *Rational Models of Cognition*. Oxford University Press, Oxford, 1998.

[OC07] M. Oaksford and N. Chater. *Bayesian Rationality: The Probabilistic Approach to Human Reasoning*. Oxford University Press, New York, 2007.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, 1988.

[Pea04] J. Pearl. Graphical models for probabilistic and causal reasoning. In A. B. Tucker, editor, *Computer Science Handbook*. CRC Press, 2nd edition, 2004.

[Pfa79] J. Pfanzagl. Conditional distributions as derivatives. *The Annals of Probability*, 7(6):1046–1050, 1979.

[PT87] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov Decision Processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[Rao88] M. M. Rao. Paradoxes in conditional probability. *Journal of Multivariate Analysis*, 27(2):434–446, 1988.

[Rao05] M. M. Rao. *Conditional measures and applications*, volume 271 of *Pure and Applied Mathematics*. Chapman & Hall/CRC, Boca Raton, FL, 2nd edition, 2005.

[RH11] S. Rathmanner and M. Hutter. A philosophical treatise of universal induction. *Entropy*, 13(6):1076–1136, 2011.

[Roy11] D. M. Roy. *Computability, Inference and Modeling in Probabilistic Programming*. PhD thesis, Massachusetts Institute of Technology, 2011.

[Sch07] M. Schröder. Admissible representations for probability measures. *Mathematical Logic Quarterly*, 53(4-5):431–445, 2007.

[Sch12] L. Schulz. The origins of inquiry: Inductive inference and exploration in early childhood. *Trends in Cognitive Sciences*, 16(7):382–389, 2012.

[SG] A. Stuhlmüller and N. D. Goodman. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. Submitted.

[SG92] A. F. M. Smith and A. E. Gelfand. Bayesian statistics without tears: A sampling-resampling perspective. *The American Statistician*, 46(2):pp. 84–88, 1992.

[SG12] A. Stuhlmüller and N. D. Goodman. A dynamic programming algorithm for inference in recursive probabilistic programs. *Second Statistical Relational AI workshop at UAI 2012 (StaRAI-12)*, 2012.

[She87] R. N. Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323, 1987.

[SMH+91] M. A. Shwe, B. Middleton, D. E. Heckerman, M. Henrion, E. J. Horvitz, H. P. Lehmann, and G. F. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. *Methods of Information in Medicine*, 30:241–255, 1991.

[Sol64] R. J. Solomonoff. A formal theory of inductive inference: Parts I and II. *Information and Control*, 7(1):1–22 and 224–254, 1964.

[Teu02] C. Teuscher. *Turing's Connectionism: An Investigation of Neural Network Architectures*. Springer-Verlag, London, 2002.

[TG01] J. B. Tenenbaum and T. L. Griffiths. Generalization, similarity, and Bayesian inference. *Behavioral and Brain Sciences*, 24(4):629–640, 2001.

[TGK06] J. B. Tenenbaum, T. L. Griffiths, and C. Kemp. Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, 10:309–318, 2006.

[THS06] M. Toussaint, S. Harmeling, and A. Storkey. Probabilistic inference for solving (PO)MDPs. Technical Report EDI-INF-RR-0934, University of Edinburgh, School of Informatics, 2006.

[Tju74] T. Tjur. *Conditional Probability Distributions*. Lecture Notes, no. 2. Institute of Mathematical Statistics, University of Copenhagen, Copenhagen, 1974.

[Tju75] T. Tjur. *A Constructive Definition of Conditional Distributions*. Preprint 13. Institute of Mathematical Statistics, University of Copenhagen, Copenhagen, 1975.

[Tju80] T. Tjur. *Probability Based on Radon Measures*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons Ltd., Chichester, 1980.

[TKGG11] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285, 2011.

[Tur36] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(1):230–265, 1936.

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

[Tur48] A. M. Turing. *Intelligent Machinery*. National Physical Laboratory Report. 1948.

[Tur50] A. M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[Tur52] A. M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72, 1952.

[Tur96] A. M. Turing. Intelligent machinery, a heretical theory. *Philosophia Mathematica. Philosophy of Mathematics, its Learning, and its Applications. Series III*, 4(3):256–260, 1996. Originally a radio presentation, 1951.

[Tur12] A. M. Turing. *The Applications of Probability to Cryptography, c. 1941*. UK National Archives, HW 25/37. 2012.

[UBM+09] T. D. Ullman, C. L. Baker, O. Macindoe, O. Evans, N. D. Goodman, and J. B. Tenenbaum. Help or hinder: Bayesian models of social goal inference. In *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pages 1874–1882, 2009.

[Wat89] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, University of Cambridge, 1989.

[WD92] C. J. C. H. Watkins and P. Dayan. Q-Learning. *Machine Learning*, 8:279–292, 1992.

[Wei93] K. Weihrauch. Computability on computable metric spaces. *Theoretical Computer Science*, 113(2):191–210, 1993.

[Wei99] K. Weihrauch. Computability on the probability measures on the Borel sets of the unit interval. *Theoretical Computer Science*, 219(1-2):421–437, 1999.

[Wei00] K. Weihrauch. *Computable Analysis: An Introduction*. Texts in Theoretical Computer Science, An EATCS Series. Springer-Verlag, Berlin, 2000.

[WGR+11] D. Wingate, N. D. Goodman, D. M. Roy, L. P. Kaelbling, and J. B. Tenenbaum. Bayesian policy search with policy priors. In T. Walsh, editor, *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, Menlo Park, CA, 2011. AAAI Press.

[WGSS11] D. Wingate, N. D. Goodman, A. Stuhlmüller, and J. M. Siskind. Nonstandard interpretations of probabilistic programs for efficient inference. In *Advances in Neural Information Processing Systems 24 (NIPS 2011)* , 2011.

[WSG11] D. Wingate, A. Stuhlmüller, and N. D. Goodman. Lightweight implementations of probabilistic programming languages via transformational compilation. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Journal of Machine Learning Research: Workshop and Conference Proceedings*, pages 770–778, 2011.

[Yam99] T. Yamakami. Polynomial time samplable distributions. *Journal of Complexity*, 15(4):557–574, 1999.

[Zab95] S. L. Zabell. Alan Turing and the central limit theorem. *American Mathematical Monthly*, 102(6):483–494, 1995.

[Zab12] S. L. Zabell. Commentary on Alan M. Turing: The applications of probability to cryptography. *Cryptologia*, 36(3):191–214, 2012.

Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory

E-mail address: freer@math.mit.edu

University of Cambridge, Department of Engineering

E-mail address: d.roy@eng.cam.ac.uk

Massachusetts Institute of Technology, Department of Brain and Cognitive Sciences

E-mail address: jbt@mit.edu