Chapter 18

Probabilistic programs as a unifying language of thought

Noah D. Goodman, Tobias Gerstenberg, and Joshua B. Tenenbaum

The previous chapters have successively opened up a landscape of probabilistic models with increasing power and generality. At each step, we introduced a new class of models and argued that it was able to capture important aspects of cognition that models in earlier chapters could not. Some of these models stricly generalized the simpler ones that preceded them, but many of the new model classes were disjoint from each other: grammars cannot be expressed as graphical models (nor vice versa), non-parametric models cannot be expressed as logical models (nor vice versa), and so on. This proliferation is worrying if our goal is to develop a simple set of principles for cognition, and it leaves us wondering if there is a formalism for probabilistic models that can unify all of these model classes?

At the same time, the model classes we have discussed are increasingly general and representationally powerful, and we may wonder how far this process can go? From the point of view of cognitive science, a natural upper limit on complexity is given by the constraint of **computability**. If cognition is computation, then the cognitive system must be limited to employing probabilistic models which are computable; that is, given some random input, the process of drawing a sample from the probability distribution can be carried out by a computer program.

In this chapter, we consider how far it is possible to develop the Bayesian approach to cognitive science given only this constraint of computability. We will develop a formalism—**universal probabilistic programs** based on the **stochastic λ-calculus**—that unifies and extends all of the model classes described in previous chapters. We introduce a particular **probabilistic programming language** known as **Church** (Goodman, Mansinghka, Roy, Bonawitz, & Tenenbaum, 2012) that allows us to express any computable probabilistic model, and to write down a simple and effective (if not generally practical) procedure for performing any computable Bayesian inference in such a model. First we outline the theoretical foundations of this approach, introducing stochastic λ-calculus, then Church, and the fact that such a language is universal. We then go on to illustrate how Church can express a range of familiar and novel modeling concepts from other chapters of the book. Indeed, we have suggested that Church (or any computationally universal probabilistic programming language) provides a framework for modeling all human concepts, as functions in a **Probabilistic Language of Thought** (PLoT). We review the PLoT hypothesis (Goodman, Tenenbaum, & Gerstenberg, 2014) and the unifying perspective it provides on concepts as the basic units of human cognition.

18.1 Probabilistic programs and the stochastic lambda calculus

We begin by defining the stochastic λ-calculus, a mathematical system that is able to represent complex generative models using only a few basic constructs. It is based on the λ-calculus, which is one of the principal models of deterministic computation. After introducing this mathematical basis we will add a number of features to make modeling easier, making this a more practical **probabilistic programming language** (PPL) for cognitive science. In particular we will adopt the PPL Church (Goodman et al., 2012), which elegantly extends the sparse mathematical system of stochastic λ-calculus based on the Scheme dialect of LISP.

How should we begin to build a compositional formal system—a language—for expressing probability distributions over complex world states? Intuitively, a first clue comes from the idea of representing distributions as **generative processes**: the series of random steps by which the world comes to be as it is. But while generative processes are a useful way to represent probabilistic knowledge, adopting such a representation only transforms our problem into one of finding a compositional language for generative processes. The solution to this version of the problem comes from a simple idea: if we have described a deterministic process compositionally in terms of the computation steps taken from start to end, but then inject noise at some point along the way, we get a stochastic process; this stochastic process unfolds in the original steps, except where a random choice is made. In this way a distribution over outputs is generated, not a single deterministic output, and this distribution inherits all the compositionality of the original deterministic process. The stochastic λ-calculus realizes this idea formally, by extending a universal computational system (λ-calculus) with points of primitive randomness.

The essence of the λ-calculus is a simple but general way to construct new functions and to apply functions to arguments. To this we will add a choice operation: a way to randomly choose which of two sub-computations to use. The resulting system, the *stochastic* λ-calculus, describes random computation.

Definition. *Let V be a countable set, called the variables. The* lambda terms *are specified inductively by:*

- $v \in V$,
- $(\lambda(x)\ M)$ *for terms* M, $x \in V$,
- $(M\ N)$ *for terms* M *and* N.
- $(M \oplus N)$ *for terms* M *and* N.

Intuitively, $(\lambda(x)\ M)$ specifies a new function who's inputs will be called x and which performs the computation specified by the *body* M; the term $(M\ N)$ specifies an application of the *operator* function M to the *operand* N; the term $(M \oplus N)$ specifies a random choice between M and N. To make this precise we will need to describe how we substitute the operand for the input variable, when such substitution can be done (and some additional bookkeeping).

Definition. *The free variables of a lambda term are specified inductively by:*

- $FV(v) = \{v\}$ *for* $v \in V$,
- $FV((\lambda(x)\ M)) = FV(M) \setminus \{x\}$,
- $FV((M\ N)) = FV(M) \cup FV(N)$,
- $FV((M \oplus N)) = FV(M) \cup FV(N)$.

Substitution replaces all free occurrences of a variable by a specified term, written $M[x := N]$ *when* x *is to be replaced by* N *in* M. *Substitution is also given inductively:*

- $x[x := N]$ *is* N,
- $y[x := N]$ *is* y *if* $x \neq y$,
- $(M1\ M2)[x := N]$ *is* $(M1[x := N]\ M2[x := N])$,
- $(M1 \oplus M2)[x := N]$ *is* $(M1[x := N] \oplus M2[x := N])$,
- $(\lambda(y)\ M)[x := N]$ *is* $(\lambda(y)\ M[x := N])$ *if* $x \neq y$ *and* $y \notin FV(N)$.

In the λ-calculus we model function application by substitution. However, notice that substitution is not always defined: there can be "name collisions" between a lambda term and a substitution, which require us to "rename" variables before trying to substitute. The stochastic λ-calculus is specified by the following **reduction rules**, which change a lambda term into a simpler one by renaming, substitution, and random choice:

- α-conversion: if $x \notin FV(N)$ and y does not appear in N, then N is replaced by $N[x := y]$ (i.e. bound variables can be renamed).

- β-reduction: $((\lambda(v)\ E)\ E')$ is replaced by $E[v := E']$.
- η-conversion: if $x \notin FV(F)$, $(\lambda(x)\ (F\ x))$ is replaced by F.
- choice-reduction: $(N \oplus M)$ is replaced uniformly at random by either N or M.

This set of reduction rules gives a model of computation. To make the connection with more typical ideas of computation, such as arithmetic, we would need to specify how standard values (such as true, false, 0, 1, 2,...) and standard operators (and, or, +, *,...) should be encoded by lambda terms. As an example, Boolean logic can be encoded by `True` = (λ(`x`) (λ(`y`) `x`)), `False` = (λ(`x`) (λ(`y`) `y`)), `and` = (λ(`x`) (λ(`y`) ((`x y`) `x`))), etc. (It is a good exercise to show that this behaves as it should: e.g., the function `and` composed with `True` and `False` reduces to `False`, and so on.) Rather than describing all of these encodings directly as lambda terms, which is possible, we will use them as "primitives" without specifying their encoding—a more convenient way to deal with them, and standard for programming languages.

Notice that for a given lambda term there are often many different sequences of reductions which could be applied. For the ordinary lambda calculus (the subset of stochastic λ-calculus lambda terms with no choice operator) the Church-Rosser theorem guarantees that the order of reductions does not matter.[1] For the stochastic λ-calculus this no longer holds. For instance, the term $((\lambda(x)(=x\ x))\ (0 \oplus 1))$ will always be true if we first reduce the operand (using choice-reduction) but it will sometimes be false if we first substitute into the operator (using β-reduction). We therefore must choose a convention; we assume that at an application the operand is always reduced as far as possible before applying β-reduction to substitute it into the operator (this is called "eager" or "call by value" order). Under this convention the term $((\lambda(x)(=x\ x))\ (0 \oplus 1))$ will always reduce to true (though the random choice may result in 0 or 1).

We can think of lambda terms as specifying generative processes: they give us instructions for the computational steps (including elementary random choices) that we should go through to compute a final value (which will be a lambda term that cannot be reduced further). We expect generative models to specify distributions, and indeed we can talk about the distribution on final values that result from reducing a lambda term until no reduction is possible. The probability of a particular sequence of reductions will be related to the number of times choice-reduction is used, and the probability of a final value is the sum over the possible reductions:

Definition. *For lambda terms M, N, let $\mu_M(N) = \sum_{r \in R_{M,N}} 2^{-C(r)}$, where $R_{M,N}$ is the set of finite reduction sequences from M resulting in N and $C(r)$ is the number of times choice-reduction is used in r. If $\sum_N \mu_M(N) = 1$ we call μ_M the distribution induced by M, and we say the distribution μ_M is represented by M.*

Not every lambda term induces a proper distribution: some have an uncountable number of infinite reduction sequences ("non-halting computations"), which means the probability of the finite reductions will be less than one.

As we describe in more detail below, stochastic λ-calculus is general enough to capture all the distributions we might plausibly want in cognitive science. It might seem natural to simply use the stochastic computations we can now describe directly to express cognitive processes – as input to output functions intended to describe the algorithmic nature of the mind. A key move in Bayesian cognitive science is to instead use distributions to represent generative knowledge about the world, which are then put to work for reasoning, decision-making and other cognitive processes via conditional inference. Using stochastic λ-calculus in this way allows us to describe a very large set of potential input-output functions as different conditional inferences for the same set of beliefs about the world. Surprisingly, we can make this move without further increasing the computational expressivity of our modeling language because stochastic

[1] More precisely, it says that any two ways to reduce a lambda term until no further reductions are possible must result in the same final term.

λ-calculus itself can already express the computations underlying conditional inference. Before we can explain this further, however, we will first make stochastic λ-calculus into a more convenient language for expressing mental representations and processes by adding some syntactic features borrowed from computer programming languages.

18.2 A probabilistic programming language: Church

We now move from an essentially mathematical object, stochastic λ-calculus, to the probabilistic programming language Church (Goodman et al., 2012), named after Alonzo Church who invented the λ-calculus. This is not a change in fundamental expressivity, but rather a set of added features and "libraries" that make it more straightforward to write useful models. Many of these additions will be treated very informally, especially when they are standard features of the Scheme programming language (see for instance Abelson, Sussman, and Sussman (1996)) which Church builds on.

We will give only a brief introduction here to the syntax and ideas of probabilistic programming using Church. Further details and many examples can be found in the *Probabilistic Models of Cognition* web book, available at `http://v1.probmods.org`. We encourage readers to explore this online resource for a complementary perspective on many of the ideas presented in the present book, and also to explore the many runnable coding examples in the browser-based **WebChurch** dialect of Church which are accessible at the `v1.probmods.org` site. We also recommend looking at the second edition web book, `http://probmods.org`, which uses a newer probabilistic programming language called **WebPPL** (`http://WebPPL.org/`), based on Javascript.

Probabilistic programming languages are similar to directed graphical models, in that they provide both a (strictly more general) mathematical formalism for describing complex probabilistic models as well as practical implementation tools for performing automated or semi-automated scalable inference in those models. But probabilistic programs are both finer-grained—including more of the model details within the specification language—and wider in coverage—capturing models with dynamic, unbounded and recursive structure. A universal language such as Church naturally expresses Bayesian inference algorithms in the same language that we write our generative models, and lets us work with classes of meta-generative models, or generative models whose outputs are any of our more familiar generative models—so that Bayesian learning for arbitrary models can also be expressed as inference in the same framework that we use to reason within those models.

Church uses a syntax inherited from the LISP family of languages (McCarthy, 1960). Thus operators precede their arguments, and are written inside grouping parentheses: e.g., `(+ 1 2)` encodes the operation "add 1 and 2". Two additions to λ-calculus that make it easier to specify functions are allowing multiple arguments and attaching values to names by `define`. For instance, we will write `(define add2 (lambda (x y) (+ x y)))` for the function `add2` that sums its two inputs. We can now reuse this function to define a new function that performs an analogous sum on four inputs: `(define add4 (lambda (w x y z) (add2 (add2 x y) (add2 w z))))`. We also allow the shortcut of dropping the lambda when we define a function: `(define (add2 x y) (+ x y))`. When we define functions in this way, we allow them to refer to their own name—that is, to be defined **recursively**. [2]

Perhaps the biggest difference between stochastic λ-calculus and Church is the addition of **elementary random procedures** (ERPs) to replace the uniform choice operation. For instance, the ERP `flip` represents the Bernoulli distribution (it "flips a coin" with "faces" `true` and `false`, with a given probability): `(flip 0.7)` results in `true` 70% of the time. Every time the function is called, a new random value is generated—the coin is flipped anew each time.

[2] It is always possible to convert a recursive function definition into pure stochastic λ-calculus, by using a **fixed point combinator**.

These random primitives can be combined just as we do for ordinary functions. For instance `(and (flip 0.5) (flip 0.5))` is the more complex process of taking the conjunction of two random Booleans. This returns `true` if and only both flips came up `true`. We can use ERPs plus recursion to define more interestingly structured generative processess. For instance, we can use `flip` and the conditional `if` function[3] to define the geometric distribution recursively: `(define (geom p) (if (flip p) (+ 1 (geom p)) 0))`.

Notice that `flip` is effectively a replacement for the choice-reduction operator, because we can define $(M \oplus N)$ as `(if (flip 0.5) M N)`; below we discuss how to write `flip` using $\oplus$.

A Church program specifies not a single computation, but a distribution over computations. This **sampling semantics** for probabilistic programs (see Goodman et al., 2012 for more details) means that composition of probabilities is achieved by ordinary composition of functions, and it allows us to specify probabilistic models using all the tools of representational abstraction in a modern programming language. We will not provide a primer on the power of function abstraction and other such tools here, but we will use them in ways that we hope are intuitive and illustrative.

Several other useful additions to stochastic λ-calculus which will be broadly useful are constructing arbitrary symbolic expressions, equality operations, and memoization. These language features in Church parallel core proposals for human conceptual representations that have long been studied and debated in cognitive science. Perhaps the most familiar (and most controversial) for cognitive modeling is the use of arbitrary **symbols**. In Church (as in LISP) a symbol is a basic value that has only the property that it is equal to itself and not to any other symbol: `(equal? 'bob 'bob)` is true, while `(equal? 'bob 'jim)` is false. (The single quote syntax simply indicates that what follows is a symbol.) Critically, symbols can be used as unique identifiers on which to hang some aspect of conceptual knowledge. For instance they can be used to refer to functions, as when we used `define` to create the `add2` function above and then reused this function by name to create `add4`. Symbols can also be used together with functions to represent knowledge about (an unbounded set of) objects. For instance, the function

```
(define eyecolor (lambda (x) (if (flip) 'blue 'brown)))
```

takes a person `x` and randomly returns an eye color (e.g. `(eyecolor 'bob)` might return `'blue`). That is, the function definition wraps up the knowledge about how eye color is generated *independently* of which person is being asked about—a person is simply represented by a symbol (`'bob`) that is associated with another symbol (`'blue`) via the `eyecolor` function.

Of course the above representation of an object's property has a flaw: if we ask about the eye color of Bob twice we may get different answers! Church includes an operator `mem` that takes a function and returns a *memoized* version: one that makes its random choices only once for each distinct value of the function's arguments, and thereafter, when called, returns the answer stored from that first evaluation. For instance, a memoized version of the `eyecolor` function,

```
(define eyecolor (mem (lambda (x) (if (flip) 'blue 'brown))))
```

could output either `'blue` or `'brown` for Bob's eye color, but only one of these possibilities, to be determined the first time the function is called. This ensures that `(equal? (eyecolor 'bob) (eyecolor 'bob))` is always true.

Thus symbols can be used as "indices" to recover random properties or as labels which allow us to recover stored information about objects and their relations. These uses are conceptually very similar, though they have different syntax, and they can be combined. For instance, we can access one function inside the definition of another by its name, passing along the current objects of interest:

[3]In Church the conditional has a traditional but possibly cryptic syntax: `(if a b c)` returns `b` if `a` is true, and `c` otherwise. Thus `(if (flip) b c)` randomly returns `b` or `c`.

```
(define eyecolor
  (mem (lambda (x)
    (if (flip 0.1)
      (if (flip) 'blue 'brown)
      (if (flip) (eyecolor (father x)) (eyecolor (mother x)))))))
```

This (false, but perhaps intuitive) model of eye color asserts that the color is sometimes simply random, but most of the time depends on the eye color of one of a person's parents—which is accessed by calling the `father` or `mother` function from inside the `eyecolor` function, and so on. Symbols and symbolic reference are thus key language constructs for forming representations of complex concepts and situations out of simpler ones.

The rest of this chapter will provide more examples and motivation on how and why we should think about Church, and probabilistic programming more generally, as a unifying computational framework for cognitive modeling. In particular we will focus on its potential for unifying the representation of diverse human concepts and conceptually driven reasoning in what has been called a "probabilistic language of thought". But first we will use Church and its capactiy for recursive definition to explain how we can think of the underlying mathematical formalism, stochastic λ-calculus, as a universal theoretical formalism for describing generative models.

18.3 Universality

We began this chapter with the goal of exploring the space of cognitive models limited only by computability, but what exactly is a computable model? One reasonable notion of computable probabilistic model is a model defined by a **computable distribution**.

Definition. *A computable probability distribution is specified by a computable function (a deterministic lambda term) $p : D \times \mathbb{N} \to Bool$: $p(x, n)$ gives the n^{th} (binary) digit of the probability of x.*

It is straightforward to see that the distribution induced by a term of stochastic λ-calculus is computable (the definition of this distribution given above, plus a strategy for enumerating reduction sequences, results in an algorithm to compute the probability to a given precision). Is it also the case that stochastic λ-calculus can represent any computable distribution? We first examine a simple case, then state the more general theorem.

Lemma. *Every computable binomial distribution is represented by some stochastic λ-calculus term.*

Proof: Let $p : \mathbb{N} \to Bool$ be the (computable) function that gives the n^{th} binary digit of the probability p of `True` (i.e. the coin weight). We will construct a stochastic λ-calculus function to sample from the binomial distribution. We start by imagining that we've already sampled a (computable) uniform random number in $[0, 1]$, call it $a : \mathbb{N} \to Bool$, and we need only compare this number to p: if $a < p$ we return `True`, otherwise we return `False`. To do this comparison we check the digits one by one, looking for the first difference between a and p (at which point we can decide which is bigger):

```
(define (L p nth)
  (if (a nth)
      (if (p nth) (L p (+1 nth)) False)
      (if (not (p nth)) (L p (+1 nth)) True)))
```

However, since a is uniform, the n^{th} bit of a is a uniform random choice between `True` and `False`, and we may make this choice within the comparison function instead of choosing all the digits of a in advance:

```
(define (L p nth)
  (if (flip)
      (if (p nth) (L p (+1 nth)) False)
      (if (not (p nth)) (L p (+1 nth)) True)))
```

(Note that `(flip)` is simply a short form for (`True` $\oplus$ `False`).)

To finish, we will show that the binomial distribution `(flip p)` is represented by `(L p 0)`. Each recursion of `L` leads to one choice, and it is possible to stop after n recursions only if all previous choices have matched p, thus there is a unique reduction sequence that halts after n choices. Further, only if the n^{th} digit of p is false can `L` result in `True` on the n^{th} recursion, so there is a reduction sequence with n choices if and only if the n^{th} digit is false. This leads to: $\mu_{(\mathtt{L\ p\ 0})}(\mathtt{True}) = \sum_{\mathtt{n}} 2^{-\mathtt{n}\delta_{\mathtt{p(n)}}}$. By using the definition of the binary representation, $p = \sum_n 2^{-n\delta_{p(n)}}$. QED.

We now state the more general theorem without proof—it is parallel in spirit to the above proof, but more complex.

Theorem. *Any computable distribution is represented by some stochastic λ-calculus expression.*

There is thus an equivalence in expressivity between stochastic λ-calculus and models which can be described as a computable distribution. Why would we choose to use one representation or the other in practice? Probabilistic programs have two related advantages: they make the generative structure of models explicit, and they are naturally compositional (whereas the laws for composing distributions are complex, requiring integration). The equivalence between these formalisms gives us confidence that we have a sufficiently broad class of models—limited only by computability—so we hazard a thesis:

Thesis (Universality)**.** *Any computable probabilistic model can be represented as an expression in stochastic λ-calculus.*

This is a thesis, rather than a theorem or hypothesis, because we mean for it to apply to any reasonable notion of computable probabilistic model, rather than specifying a particular one. (Cf. the **Church-Turing thesis** that ordinary λ-calculus exhaustively captures computable functions.)

18.4 Conditional inference

How does inference enter into this system? In Bayesian cognitive science we are concerned not only with specifying generative models, which capture an agent's beliefs about the world, but with conditional inferences over these models, to describe learning, reasoning and other proccesses of belief updating. It would be natural to think we needed to introduce a new operator into the stochastic λ-calculus to express conditioning. Remarkably, it turns out that this is not necessary. Conditioning can be directly defined within Church via a recursive function that corresponds to the notion of rejection sampling (see Chapter 6)—a stochastic recursion that embodies the naive (but always reliable) "guess-and-check" approach to inference.

Assume that we have a distribution `dist` (a stochastic function with no input arguments) that represents a generative model of interest by drawing samples from it, and a predicate `pred` (a deterministic function taking as input a value of the type output by dist, and returning `True` or `False`) that represents some piece of information on which we would like to condition belief updates. Then we can define a higher-order stochastic function `condition` that takes as input the model `dist` and the information `pred`, and returns a sample from a new distribution equal to `dist` conditioned on the predicate being `True`:

```
(define (condition dist pred)
    (define sample (dist))
    (if (pred sample) sample (condition dist pred)))
```

That is, we keep sampling from `dist` until we get a sample that satisfies `pred`, and then we return this sample. This directly implements Bayesian belief updating via the sampling semantics: `condition` takes a prior model specified by `dist` and effectively constructs a new model (specified via a process for generating a sample from it) that represents the posterior, corresponding to `dist` conditioned on the evidence that `pred` is true. In Church we usually use a more convenient syntax for conditionals in the form of the `query` function, which specifies a particular random variable of interest or "query-expression", and which can be converted automatically into the above form:

```
(query
  ...definitions...
  query-expression
  condition-expression)
```

Our initial distribution is now specified by the `query-expression` evaluated in the context of the generative world model given by...`definitions`..., and our predicate for conditioning is the `condition-expression` evaluated in the same context.

For instance, if we wanted to compute the conditional distribution of two samples from the geometric distribution, conditioned on their sum being less than four, we could write:

```
(query
  ;; ...definitions...
 (define (geom p) (if (flip p) (+ 1 (geom p)) 0))
 (define a (geom 0.5))
 (define b (geom 0.5))

 ;; query-expression
 (list a b)

 ;; condition-expression
 (< (+ a b) 4))
```

Or referring again to the eye-color example, if we wanted to ask about Bob's mother's likely eye color, given that Bob has blue eyes, we could write:

```
(query
  (define eyecolor
    (mem (lambda (x)
    (if (flip 0.1)
      (if (flip) 'blue 'brown)
      (if (flip) (eyecolor (father x)) (eyecolor (mother x)))))))
      ;; ... and other definitions

  ;; query-expression
  (eyecolor (mother 'bob))

  ;; condition-expression
  (equal? (eyecolor 'bob) 'blue))
```

In these examples, the ";; ..." syntax indicates code comments. Notice that there is a distinction here between the definitions, which represent probabilistic knowledge reusable across many queries, and the query and condition expressions, which represent the particular question of interest at the moment. In this example, the particular people need to be introduced only in the question of interest because the conceptual knowledge is defined over arbitrary symbols. But also note that to make this a well-defined model, the definitions need to include some specifications for the functions `(mother ...)` and `(father ...)` that return valid values. This could be something sophisticated like a generative process for family trees, or as simple as `(define (mother x) gensym)` which returns a randomly generated, arbitrary "generic symbol" (effecitvely a placeholder name for an individual's mother – but someone who can have an eye color, as far as the `eyecolor` function is concerned).

Though we may *define* conditional inference by using the rejection sampler, it is not the case that `query` must be *implemented* by rejection sampling. Indeed probabilistic programming languages routinely provide implementations of `query` based on other algorithms borrowed from computational Bayesian statistics (see Chapter 6), such as Metropolis-Hastings (Hastings, 1970) and Sequential Monte Carlo (Doucet, Freitas, & Gordon, 2001), and of course exact enumeration (see Chapter 3). Since the development of Church and other early probabilistic programming approaches (e.g., BLOG, Milch & Russell, 2006; Anglican (Wood, Meent, & Mansinghka, 2014)), work on the engineering front has focused on developing increasingly more powerful inference algorithms and more scalable inference frameworks, Important innovations include Hamiltonian Monte Carlo (HMC) (Carpenter et al., 2017), variational inference using neural network encoders to learn a variational approximate posterior, as in Pyro (Bingham et al., 2019b) and Turing (Ge, Xu, & Ghahramani, 2018); and meta-programming and programmable inference, for flexibly combining different modeling and inference motifs, as in Venture (Mansinghka, Selsam, & Perov, 2014) and Gen (Cusumano-Towner, Saad, Lew, & Mansinghka, 2019).

These engineering innovations—and many more that are still under development—will be crucial in realizing the promise of probabilistic programming. But it must be emphasized that even the probabilistic programming languages we already have represent remarkable artifacts and intellectual achievements, in their universality and their simplicity. With Church, for example, we have a programming language that is close enough to the grain of conscious thought that anyone can learn it, and that can be used to describe any probabilistic generative model and any probabilistic inference algorithm that any computer could possibly implement. The simple inference algorithms implemented in the web books `v1.probmods.org` and `probmods.org`, for Church and its descendant WebPPL, are far from the most efficient and not designed for practical deployment on large-scale problems. But they are available to you right now to run in your web browser, and they are *universal* inference algorithms: In principle, if you are willing to wait long enough, they can compute any conditional Bayesian inference that can be computed. As the engineering side of probabilistic programming moves forward, the field should move from these in-principle possibilities to increasingly practical tools: programming languages and platforms that provide the easiest, most robust way to implement all of the models in this book, and an effective route to scaling up the Bayesian approach to much more integrative and large-scale accounts of human cognition.

18.5 From Probabilistic Programs to a Probabilistic Language of Thought

So far in this chapter we have defined a unifying theoretical formalism, stochastic λ-calculus, spanning the different probabilistic modeling formalisms presented in this book (and indeed any computable model), embedded this formalism in an elegant and conceptually natural programming language for modeling cognition, Church, and described a very general and useful class of Bayesian inference algorithms, implementing the `query` function, which you can access in your web browser via the probmods web books. The remainder of the chapter presents a deeper dive into how probabilistic programs can be used to model human thought, with two specific goals. We want to show more concretely how a PPL like Church can express a diverse range of cognitive models, and we want to show how it provides a way to think in reverse-engineering terms about fundamental aspects of thinking that are harder to approach with the more familiar tools of Bayesian modeling presented in earlier chapters.

The core message of this book has been that robust, resource-rational probabilistic inference is at the heart of human cognition. Reverse engineering the mind is enabled by the right toolkit of theory, representations and algorithms for learning and reasoning with uncertain knowledge – including probabilistic versions of hierarchies, relational systems, logic, grammars, and many other forms of symbolic structure and abstraction. But we have yet to bring the rational probabilistic inference approach into contact with perhaps the most fundamental property of human minds: their productivity. The set of thoughts we can think, and the situations we can think about, is open-ended. It is not unconstrained, but it is unbounded. Our experience in the world, however, is finite and limited in many ways that our thinking is not. This is why generalization must also be at the heart of cognition. The ability to generalize both rationally and productively, to reason robustly under uncertainty in an endless range of novel situations, remains a central open challenge for computational cognitive science. And addressing *this* challenge is the most important reason why the Bayesian approach to cognitive modeling needs the tools of probabilistic programming.

The oldest and arguably best proposal for how productive generalization in cognition can be possible is that our thinking is compositional: that is, our thoughts are built out of **concepts**—elements of knowledge—that can be combined and recombined in something like a **language of thought** (Lot)to reason flexibly about an unbounded set of novel situations (Fodor, 1975, 1998; Carey, 2009; Spelke, 2022). Probabilistic programming gives us a way to extend this view, to think about concepts as a combinatorial system for probabilistic reasoning about an open-ended world of situations; each such situation specifies a probability distribution over what we expect to see in the world, given what we have seen. Indeed we can recast the stochastic λ-calculus and Church as formal instantiations of a *probabilistic* language of thought in which mental representations are built from language-like composition of concepts but the semantic content of those representations is not a logical specification of possible worlds as in traditional LoT proposals; rather it specifies a distribution on more or less probable worlds, conditioned on everything else we know or could imagine being true.

Goodman et al. (2014) originally introduced the **Probabilistic Language of Thought (PLoT) hypothesis** as a way to unify different views of human concepts and their role in mental life, which have traditionally been approached from distinct and incommmensurate computational paradigms. At the risk of oversimplifying we could call these the *statistical* and *symbolic* views. The statistical view emphasizes rational probabilsitic inference, and concepts as the locus of our predictive generalizations – the kinds of inferences that have been the focus of the early chapters of this book. Concepts give ways of picking out structure in the world that can be learned and generalized from small numbers of positive examples via Bayesian inference (Tenenbaum, 2000). They summarize stable regularities in the world, such as typical distributions of objects, properties and events as encoded by prototype and exemplar models (Murphy, 2002) and their generalizations in nonparametric Bayesian models (Sanborn, Griffiths, & Navarro, 2010). The symbolic view emphasizes the ways concepts provide the building blocks of compositional thought, as just discussed: how they can be flexibly combined to form an infinite array of thoughts in order to reason productively about an infinity of situations, and how they can be composed to make new concepts, which are building blocks of yet more complex thoughts. Related is the notion that concepts get much of their meaning and their function from the role that they play in explanatory structures: both larger-scale systems of abstract thought, and fine-grained mechanistic accounts of how causes give rise to effects. These explanatory aspects of concepts are traditionally addressed by the "theory theory" (Gopnik, 2003) and other accounts based on inferential or conceptual roles (Block, 1998), which have been formalized using causal graphical models, hierarchical Bayesian models and relational nonparametric models (see examples in Chapter 1, and more detail Chapters 4, 8, and 9). For our purposes, we group all these accounts under the heading of *symbolic* views, because some kind of logical, relational or other expressive symbolic representational machinery has traditionally been seen as essential to their implementation, as opposed to or in addition to statistical inference machiney.

We can phrase the PLoT hypothesis, informally, as:

> **Probabilistic language of thought hypothesis (informal version):**
> Concepts encode probabilistic generative knowledge about possible entities, relations, events, situations and types, through the roles they play in a language-like (compositional and recursive) system for world modeling and inference.

The PLoT gives us a way to think about concepts that integrates their statistical, compositional, and explanatory functions, unifying acrooss these different views and accounting for more of the richness of human reasoning than could be captured using any of the traditional computational approaches alone. It can be seen as a way of making the statistical view of concepts more flexible and systematic by enriching it with a fine-grained notion of composition coming from symbolic approaches. It can also be seen as making symbolic approaches to concepts more useful for reasoning in an uncertain world, by embedding them in a probabilistic framework for inference and decision.

Let's unpack these claims. Because concepts are mental representations that are simultanesouly probabilistic and composable with language-like means, they can support graded inferences under uncertainty and inductive learning in ways that extend productively to an unbounded range of new situations. Because they are both generative and recursive, they can represent world knowledge that spans from high levels of abstraction to highly specific causal mechanisms, and rich spaces of possibility; thus they support probabilistic inference in the service of causal reasoning, explanation, planning and imagination – not just statistical prediction in the actual world.

The stochastic λ-calculus and its embedding in Church allows us to make these ideas more formal:

> **Probabilistic language of thought hypothesis (formal version):**
> Concepts are stochastic functions in a universal probabilistic programming language.

In Church, knowledge is encoded in stochastic (potentially recursive) function definitions. These functions describe elements of stochastic processes that can be composed together to describe various situations, to pose various questions and to answer those questions with reasonable probabilistic guesses. Just as concepts are often seen as the stable and reusable units of human thought, stochastic functions are the units of knowledge encoded in a Church program. Identifying concepts with stochastic functions in Church (or any universal PPL) together with the construct of `query` immediately gives rise to all the desired features of concepts: their capacities to represent uncertainty about the world, to compose into an infinite set of more complex concepts, and to support statistical learning, generalization and probabilistic inference, as well as causal and explanatory inferences, over unbounded spaces.

The formalism also lets us explore more precise answers to classic questions about the nature and origins of concepts: What constitutes meaning? How are the meanings of different concepts related? How are they acquired and used? Our answers to these questions can be subtle; for instance, on the face of it, the meaning of a stochastic function is simply its definition and the relation between concepts is determined by constituency—in our example earlier in the chapter, the meaning of `eyecolor` is its definition and it is related to other concepts only by its use of `mother` and `father` functions. However, when we consider the inferential relationships between concepts that come from conditional inference—`query`—we see additional aspects of meaning and conceptual relation. Conditioning on parentage can influence eye color, but also vice versa; conditioning on hair color may influence judgements about eye color indirectly, and so on. In the next section we give an extended example in the domain of simple team games, illustrating these foundational issues as well as exploring the empirical adequacy of the PLoT hypothesis.

18.6 Putting the PLoT to work: Bayesian Tug-of-War and Ping Pong

As an example of how the PLoT has been developed to capture people's striking flexibility in understanding new situations and making productive probabilistic inferences, consider some of the inferences you might make watching matches in a team game such as tug-of-war. In each match, we see two teams pitting their strength against each other by pulling on opposite ends of a heavy rope. If a team containing the first author of this chapter (NG) loses to a team containing the second author (TG), that might provide weak evidence that TG is the stronger of the two. If these teams contain only two members each, we might believe more in TG's greater strength than if the teams contain eight members each. If TG beats NG in a one-on-one tug-of-war, and NG goes on to beat three other individuals in similar one-on-one contests, we might believe that TG is not only stronger than NG but strong in an absolute sense, relative to the general population, even though we have only directly observed TG participating in a single match. However, if we later found out that NG did not try very hard in his match against TG, but did try hard in his later matches, our convictions about TG's strength might subside.

This reasoning is clearly probabilistic. We may make good guesses about these propositions, but we will be far from certain; we will be more certain of TG's strength after seeing him play many games. But our reasoning here is also highly abstract. It is not limited to a particular set of tug-of-war contestants. We can reason in analogous ways about matches between teams of arbitrary sizes and composition, and are unperturbed if a new player is introduced. We can also reason about the teams as collections: if team Alpha wins its first four matches but then loses to team Bravo, whom it has not faced before, we judge team Bravo very likely to be stronger than average. The smaller team Bravo is, the more likely we are to judge a particular member of team Bravo to be stronger than the average individual. And similar patterns of reasoning apply to inferences about skill and success in other kinds of team contests: we could be talking about math teams, or teams for doubles ping-pong, and make analogous inferences for all the situations above.

Our reasoning also supports inferences from complex combinations of evidence to complex conclusions. For example, suppose that participants have been paired up into teams of two. If we learn that NG was lazy (not trying hard) whenever his team contested a match against TG's team, but NG's team nonetheless won each of these matches, it suggests both that NG is stronger than TG and that NG is often lazy. If we then learned that NG's teammate is stronger than any other individual in the population, we would probably revise the former belief (about NG's strength) but not the latter (about his laziness). If we learned that NG's team had won all of its two-on-two matches but we were told nothing about NG's teammate, it is a good bet that the teammate—whomever they are—is stronger than average; all the more so, if we also learned that NG had lost several one-on-one matches while trying hard.

Finally, our reasoning in this one domain can be modularly combined with knowledge of other domains, or manipulated based on subtle details of domain knowledge. If we observed TG lifting a number of very heavy boxes with apparent ease, we might reasonably expect his tug-of-war team to beat most others. But this would probably not raise our confidence that TG's math team (or even his ping-pong team) are likely to be unusually successful. If we know that NG is trying to ingratiate himself to TG, perhaps to receive a favor, then we might not weight his loss very heavily in estimating strength. Likewise if we knew that NG had received a distracting text message during the match.

How can we account for the wide range of flexible inferences people draw from diverse patterns of evidence such as these? What assumptions about the cognitive system are needed to explain the productivity and gradedness of these inferences? What kind of representations are abstract enough to extend flexibly to novel situations and questions, yet concrete enough to support all the detailed predictions we can make about the world? The PLoT provides a way to approach all of these questions, and its realization in a language such as Church lets us instantiate the approach in quantitatively predictive rational models of all the judgments a person can make in this (or any similarly rich) domain.

18.6.1 Formalizing tug-of-war

Let us use Church to formalize a simple intuitive theory of the tug-of-war domain, capturing some of the concepts above and illustrating how they can be used for reasoning. The theory is built around matches, teams, and people. We observe a set of matches. A match is between two teams of people. Each person has a strength which is fixed, and each person may be lazy or not on a given match. A match is won by the team that collectively pulls the hardest. Notice how we have a set of concepts that express our commonsense understanding of this domain—concepts like strength, laziness, pulling, and winning—and they can be used to reason about a wide range of situations and questions: about the likely outcomes of matches between different people, about different players' strengths or laziness, or about hypothetical outcomes of different kinds of matches between different kinds of players or teams.

In Church, we can use symbols as placeholders for unspecified individuals or entities of these types. This means that we do not need to define in advance how many people participate, what the size of the teams will be, or how many matches a tournament will have. Symbols can be constructed as needed by using a single quote: e.g., `'alice`, `'bob`, `'team-bayes`, `'match-17`. In stochastic λ-calculus symbols were used only for variables, but in Church we can use symbols as values—and even values that serve no purpose other than to stand for themselves (so we can ask whether two symbols are equal) are remarkably useful for individuating situations and objects on the fly, as needed.

The only knowledge we need to specify in advance is a lexicon of abstract concept definitions useful for reasoning in general about the entities in the domain – the people, teams, and matches – along with their properties and relations. We define an individual player's strength, `strength`, via a function that draws from a Gaussian distribution (with arbitrary mean $M = 10$ and standard deviation $SD = 3$):

```
(define strength (mem (lambda (person) (gaussian 10 3))))
```

This function is memoized to ensure that the strength value assigned to a person is persistent (as with our eye color function earlier in this chapter) and does not change between games. However, we assume that players are sometimes lazy. The chance of a person being lazy in a particular match is 10%:

```
(define lazy (mem (lambda (person match) (flip 0.1))))
```

A lazy person only plays with half of their actual strength. The overall strength of a team for a given match is the sum of the strength each person on the team contributes in that match, taking into account the possibility that any player could be lazy for that match:

```
(define teamstrength
  (mem (lambda (team match)
         (sum (map (lambda (person)
                     (if (lazy person match)
                         (/ (strength person) 2)
                         (strength person)))
                   team)))))
```

Note how this definition uses the function `map`, which applies a function to each element of a list; here the function is the lambda expression encoding how being lazy on a given match divides a person's strength by 2, and the list is the variable `team` which encodes the set of people comprising a team participating in the given match.

Finally, we specify how the winner of a game is determined. We simply say that the team wins who has the greater overall strength:

```
(define winner
  (mem (lambda (team1 team2 match)
         (if (> (teamstrength team1 match) (teamstrength team2 match))
             'team1 'team2))))
```

Taken together, this set of function definitions specifies a lexicon of concepts for reasoning about tug-of-war; it can also apply to many other team sports, as in the experiment we report below where participants reason about a "ping pong tournament".

With this set of concepts we can ask many different questions about many different scenarios, taking advantage of multiple forms of compositionality in Church. For instance, we can ask how strong Bob is, given only the information that Bob played in one match on a team with Mary, and won against a team of two other people, Sue and Bill:

```
(query
 ... CONCEPTS ...

 (strength 'bob)

 (equal? 'team1 (winner '(bob mary) '(sue bill) 'match-1)))
```

Here ...CONCEPTS... is shorthand for the definitions introduced above—the concepts that let us make inferences about a player's strength not only in this situation but in a multitude of possible situations with varying teams composed of any number of people, playing against each other with all thinkable combinations of game results in different tournament formats. With these concepts, we could just as easily ask about Bob's strength if we observed him participating on a team of three people in an asymmetric match against a team of six:

```
(query
 ... CONCEPTS ...

 (strength 'bob)

 (equal? 'team1 (winner '(bob mary alice) '(sue bill frank joe henry nancy) 'match-1)))
```

We encourage you to try these examples and others below in WebChurch[4], a Church implementation that runs online in your browser, at `http://v1.probmods.org/play-space.html`. Think about what inferences you expect to see before you run each piece of code; this is a good way both to gain intuitions about probabilistic inference, and also to appreciate how the PLoT can capture the texture of your own intuitive thinking.

Begin by typing the following (with some WebChurch-specific syntax at the start and end):

[4]See also the dedicated section on reasoning in tug-of-war, at
`http://v1.probmods.org/conditioning.html#example-reasoning-about-the-tug-of-war`.

```
(define samples
  (mh-query 1000 100

     ;; the concepts (definitions):
    (define strength (mem (lambda (person) (gaussian 10 3))))

    (define lazy (mem (lambda (person match) (flip 0.1))))

    (define teamstrength
       (mem (lambda (team match)
            (sum (map (lambda (person)
                     (if (lazy person match)
                         (/ (strength person) 2)
                         (strength person)))
                   team)))))

    (define winner
       (mem (lambda (team1 team2 match)
          (if (> (teamstrength team1 match) (teamstrength team2 match))
             'team1 'team2))))

    ;; the query:
     (strength 'bob)

    ;; conditioning on the evidence:
    (equal? 'team1 (winner '(bob mary) '(sue bill) 'match-1))
  )
)

(display (list "Expected strength: " (mean samples)))
(density samples "Bob strength" true)
```

Running this code in WebChurch generates 1000 samples from the posterior for `(strength 'bob)` conditioned on the evidence that a team of Bob and Mary having beat Sue and Bill; it will visualize the distribution of samples, and also print out its mean (expectation). You should see a distribution that is roughly Gaussian, with a mean near 11 – or slightly above the prior mean of 10.

Now try the same code but with the conditioning line `(equal? team1 ... )` removed or commented out. This will generate samples from the prior on `(strength 'bob)` – a distribution that is not conditioned on any information. You should see a distribution that is Gaussian with the same standard deviation (equal to 3) but now with the prior mean of 10.

Finally, try the same code but with the conditioning statement of the query changed to observe the very asymmetric match,

```
(equal? 'team1 (winner '(bob mary alice) '(sue bill frank joe henry nancy) 'match-1))
```

but before running the code, think about how you would expect this piece of evidence from Bob winning in a 3-on-6 match to affect your inference about Bob's strength differently from the evidence before, where Bob won in a 2-on-2 match.

Intuitively, this will provide evidence that Bob is much stronger than before. Running the code above, you should see this effect in the posterior: the mean should now be around 13 (roughly a full standard deviation above the mean).

This capacity for productive extension over different possible situations including different persons, different teams and different winners of each game, renders the Church implementation a powerful model for general-purpose human reasoning. Contrast this way of defining domain knowledge, where specific individuals are represented with new symbols (`'bob`, etc.) only as needed, with more familiar forms of knowledge representation in probabilistic generative models, such as a Bayes net, where we require a fixed finite set of random variables (strength-of-bob, and so on) specified in advance; a Church program is like a specification from which infinitely many different Bayes nets can be built.

Another sense in which Church captures productive use of concepts comes from the ability to compose functions to form complex statements—with each new concept there is a combinatorial increase in the number of queries that can be constructed. We can use this aspect of compositionality to capture complex patterns of data we might observe, or complex questions we can answer. Consider asking how strong Bob is given that he and Mary won a match against Jane and Jim, and that Jane is more than twice as strong as Mary, by keeping the WebChurch code as above but replacing the conditioning line with:

```
 (and (equal? 'team1 (winner '(bob mary) '(jane jim) 'match-1))
      (> (strength 'jane) (* 2 (strength 'mary))))
```

Or to ask how strong is Bob given that his team of three beat a team of six, and also the knowledge that he's stronger than his two teammates taken together, condition on:

```
(and (equal? 'team1 (winner '(bob mary alice) '(sue bill frank joe henry nancy) 'match-1))
     (> (strength 'bob) (+ (strength 'mary) (strength 'alice))))
```

To ask how likely it is that Bob is stronger than both Jane and Jim, given the data that Bob and Mary beat Jane and Jim twice in a row (in matches 1 and 2), and the knowledge that Jane and Jim are each individually at least half as strong as Mary, replace the query `(strength 'bob)` with

```
 (> (strength 'bob) (max (strength 'jane) (strength 'jim)))
```

and the conditioning statement with

```
 (and (equal? 'team1 (winner '(bob mary) '(jane jim) 'match-1))
      (equal? 'team1 (winner '(bob mary) '(jane jim) 'match-2))
      (> (strength 'jane) (* 1.5 (strength 'mary)))
      (> (strength 'jim) (* 1.5 (strength 'mary))))
)
```

You'll also want to replace the last two lines `(display ...)` and `(density ...)` with:

```
(hist samples "Bob stronger?")
```

Or given the same information, we can ask how likely it is that Bob's strength is more than one standard deviation above the mean of the population, by replacing the previous query with this line that estimates both mean and standard deviation from an imagined sample of 100 random individuals:

```
 (> (strength 'bob) (+ (mean (map strength (iota 100)))
                       (sqrt (var (map strength (iota 100)))) ))
```

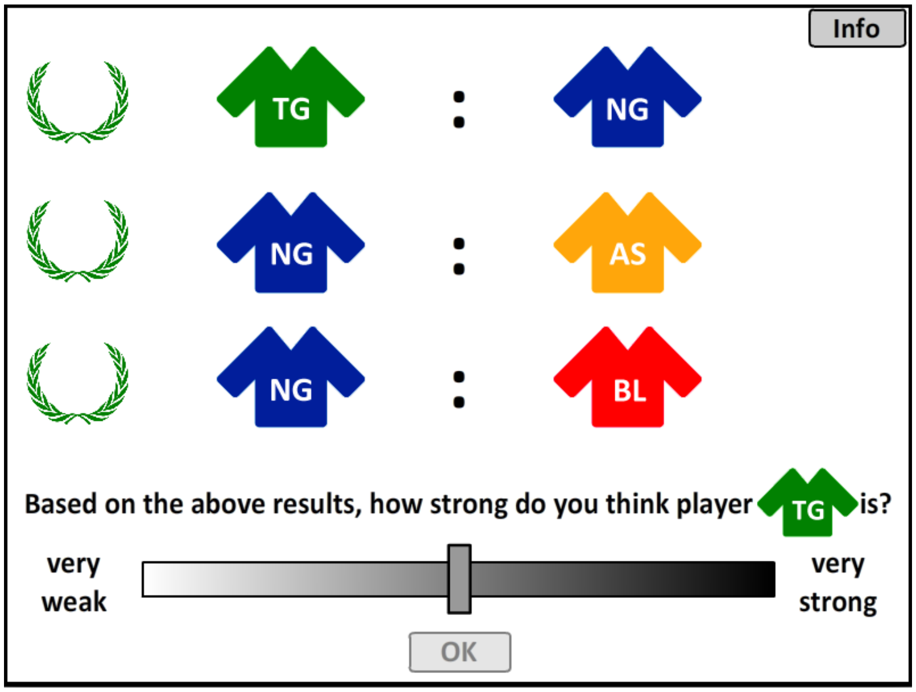

Info
TG
NG
NG
AS
NG
BL
Based on the above results, how strong do you think player TG is?
very weak
very strong
OK

Figure 18.1: An example of a single-player tournament shown to human reasoners in (Gerstenberg & Goodman, 2012). A laurel wreath marks the winner of each match.

Again, we can see that the set of church definitions lets us construct variables and relationships on the fly that would have to be pre-specified in a more familiar Bayes net. Here it is not just simple variables (like the strength of Bob) which are constructed, but quite complex conditions.

18.6.2 A quantitative experiment

The examples above illustrate qualitatively how inference in probabilistic programs can model intuitive reasoning, but this account also performs well when assesed quantitatively in behavioral experiments. In (Gerstenberg & Goodman, 2012)), participants were presented with information about a set of ping pong tournaments and asked to make judgments about the relative strengths of the players. Figure 18.1 shows an example of a single-player tournament (with all matches between teams of one player each). In this example, most people conclude that TG is relatively strong, while BL may be a little weaker than average.

This inference can be described with the following query in Church:

```
(query
  ...CONCEPTS...

  ;The query:
  (strength 'TG)

  ;The evidence:
  (and (equal? 'team1 (winner '(TG) '(NG) 'match-1))
       (equal? 'team1 (winner '(NG) '(AS) 'match-2))
```

Table 18.1: Patterns of observation for the single-player tournaments in Experiments 1 and 2.

confounded evidence (Trials 1,2)	strong indirect evidence (Trials 3,4)	weak indirect evidence (Trials 5,6)	diverse evidence (Trials 7,8)
A > B	A > B	A > B	A > B
A > B	B > C	B < C	A > C
A > B	B > D	B < D	A > D
lazy,match: B,2	B,1	B,1	C,2

Note. Each tournament features three matches, and each match is between two players. The notation "A > B" means that player A won against player B in one match. Numbers 1-8 indicate the trial types, with descriptive names for the patterns of evidence shown (e.g., "confounded evidence", "strong indirect evidence", etc.). Trials with odd numbers are shown explicitly, and an additional set of 4 patterns (even-numbered trials) were identical to these except that the outcomes of the matches were reversed. The bottom row shows the omniscient commentator's information in Experiment 2. For example, in the confounded evidence case, the commentator reported that player B was lazy in the second match.

```
    (equal? 'team1 (winner '(NG) '(BL) 'match-3))))
```

Participants in Experiment 1 of (Gerstenberg & Goodman, 2012)) saw both single-player tournaments and two-player tournaments (see Tables 18.1 and 18.2 for the specific patterns of evidence shown). Participants were always asked to judge the strength of player A. They were also given some basic information about the scenarios corresponding to our modeling assumptions: that individual players have a fixed strength which does not vary between matches, and that any player has a small chance of not playing as strongly as they can in any match, so that even a strong player can sometimes lose to a weaker player. In Experiment 2, participants saw only the single-player tournaments, but they also saw additional evidence in the form of a commentator (like a sportscaster, who always told the truth) reporting that a particular player (not player A) was lazy on a particular match. We asked for judgments about player A's strength in two stages: first after people saw only the matches, and then after they also heard the commentator's report, to assess how they updated their beliefs.

Across both Experiments 1 and 2, our Church model accurately predicts variations in mean human strength estimates (see Figures 18.2 and 18.3, respectively). Mean human participants' judgments and model estimates correlate extremely highly ($r = .98$, Experiment 1; $r = .97$ Experiment 2). We also find a high median correlation between the model and individual participants' judgments of ($r = .92$, Experiment 1; $r = .86$, Experiment 2) showing that the close fit is not merely an aggregation effect; we see it reflected in the untrained intuitions of almost every participant.

These results demonstrate quantitatively the strong form of generalization in human reasoning that we set out to capture with the PLoT. We presented people with a novel situation, bearing some similarity to familiar settings but also its own idiosyncratic features: one- and two-player teams, multiple overlapping match-ups in a tournament, the potential for laziness, and information presented in both observed outcomes and linguistic commentary. With no additional training, people rationally and productively integrate multiple diverse pieces of evidence to form a reasonable overall estimate of a player's underlying strength. Additional studies are in progress to probe how far this ability goes – and we already know there are questions where human and model predictions start to diverge. On balance, still, it is striking how well the patterns of intuitive human reasoning in a novel domain can be well explained both qualitatively and quantitatively using only the very simple but very general machinery of a probabilistic language of thought.

Table 18.2: Patterns of observation for the two-player tournaments in Experiment 1.

confounded with partner (Trials 9,10)			confounded with opponent (Trials 11,12)			strong indirect evidence (Trials 13,14)		
AB	>	CD	AB	>	EF	AB	>	EF
AB	>	EF	AC	>	EG	BC	<	EF
AB	>	GH	AD	>	EH	BD	<	EF
weak indirect evidence (Trials 15,16)			**diverse evidence (Trials 17,18)**			**round robin (Trials 19,20)**		
AB	>	EF	AB	>	EF	AB	>	CD
BC	>	EF	AC	>	GH	AC	>	BD
BD	>	EF	AD	>	IJ	AD	>	BC

Note. Trials with odd numbers are shown explicitly, and an additional set of 6 patterns (even-numbered trials) were identical to these except that the outcomes of the matches were reversed.

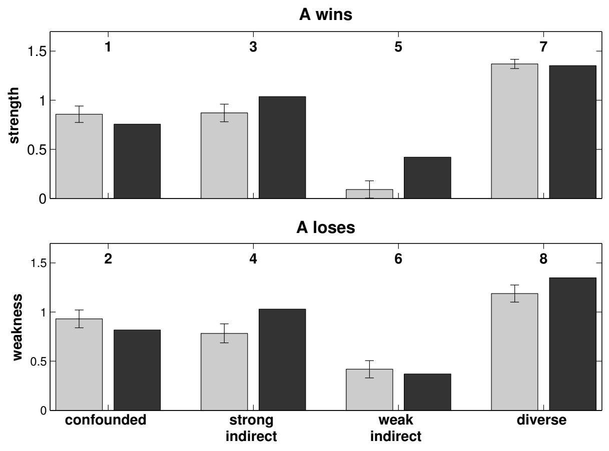
18.7 Intuitive theories

The examples above provide concrete illustrations of how to represent concepts as functions in a probabilistic language of thought, how a system of such concepts supports inferences that are both productive and probabilistic, and how these inferences can capture the outputs of human reasoning at a high level of quantitative accuracy. But these examples are also quite limited in scope relative to the conceptual systems at the heart of human commonsense reasoning. In this section we discuss how the same machinery can describe abstract concepts that are the backbone of thinking about everyday life, and which have not fit easily into more traditional computational frameworks (Bayesian or otherwise).

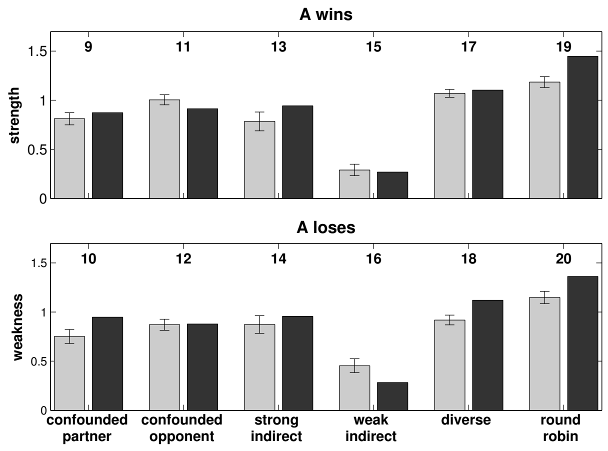
Intuitive theories (Gopnik & Wellman, 2012; Wellman & Gelman, 1992; Carey, 1985, 2009; Gerstenberg & Tenenbaum, 2017) like their more familiar scientific counterparts, comprise a system of interrelated and inter-defined concepts articulating a basic ontology of entities, the properties of and relations between those entities, and the causal laws that govern how these entities evolve over time and interact with each other. For instance intuitive physics is a system for reasoning about physical objects, and intuitive psychology for reasoning about intentional agents. These are called “theories” because, like in scientific theories, the essential constructs of intuitive theories are typically not directly observable. Yet intuitive theories also specify how unobservable states, properties and processes do impact observable experience—and thus how they support competencies such as prediction, explanation, learning and reasoning.

Intuitive theories can be found in some form in young infants, and are also to some extent shared with many other species; they are arguably the earliest and oldest abstract concepts we have (Carey, 2009). They provide the scaffolding for many of children’s conceptual achievements over the first few years as well as core building blocks for meaning in natural language. They are also transformed fundamentally as children develop their natural language abilities. Enriched, deepened and broadened through language input, they continue to serve as the basis for adults’ common-sense understanding of the world. For all these reasons intuitive theories have long been a prime target for exploration by developmental and cognitive psychologists, linguists, and philosophers, but it is only recently – with the advent of probabilistic programs – that we have the machinery needed to represent them in computational terms.

Do intuitive theories really require the full power of probabilistic programs? In earlier chapters we have discussed several approaches that were previously proposed to capture some aspects of folk theories.



(a) Single player tournament



(b) Two-player tournament

Figure 18.2: Mean strength estimates (grey bars) and model predictions (black bars) for Experiment 1, (a) single player and (b) two-player tournaments of Experiment 1. The top row shows strength judgments for cases in which the player won her game(s). The bottom row shows weakness judgments for cases in which the player lost. Numbers above the bars correspond to the trial patterns described in Tables 18.1 and 18.2. Error bars are ± 1 *SEM. Note:* For ease of comparison, we z-scored both model predictions and each individual participants' judgments, and reverse-coded judgments and predictions for the situations in which the outcomes of the games were reversed (even-numbered trials) so that both strength and "weakness" judgments go in the same direction.

Bayesian networks are one natural candidate: interpreted causally, they provide a way to formalize how people use their intuitive theories to reason backwards from observed effects to unobserved causes in the world, and to learn about causal relations from sparse patterns of observations (Gopnik et al., 2004; Rehder, 2003; Goodman et al., 2006; Griffiths & Tenenbaum, 2005, 2009). These efforts were ultimately limited by the fact that Bayesian networks, like neural networks before them, fail to capture genuine productivity in thought. An intuitive theory of physics or psychology must be able to handle an infinite range of novel situations, differing in their specifics but not their abstract character, just as we illustrated above on a much smaller scale in sketching a causal domain theory for tug-of-war or ping pong. Hierarchical Bayesian models, especially those defined over structured relational representations such as first-order logic schemas, have been proposed as one way to increase the representational power of Bayesian networks, and they have given reasonable accounts of some aspects of abstract causal reasoning and learning with intuitive theories (e.g. Tenenbaum, Griffiths, & Kemp, 2006; Tenenbaum, Kemp, Griffiths, & Goodman, 2011; Goodman, Ullman, & Tenenbaum, 2011; Gopnik & Wellman, 2012,). But these approaches still lack sufficiently fine-grained compositionality to capture our understanding of the physical and social worlds. The PLoT allows us to take a major step forward in this regard. Both Bayesian networks and hierarchical Bayesian models of intuitive theories can be naturally written as Church programs, preserving their insights into causal reasoning, abstraction, and learning, but Church programs go much further in letting us capture the essential representations of common-sense physics and psychology that have defied previous attempts at formalization within the probabilistic modeling tradition.

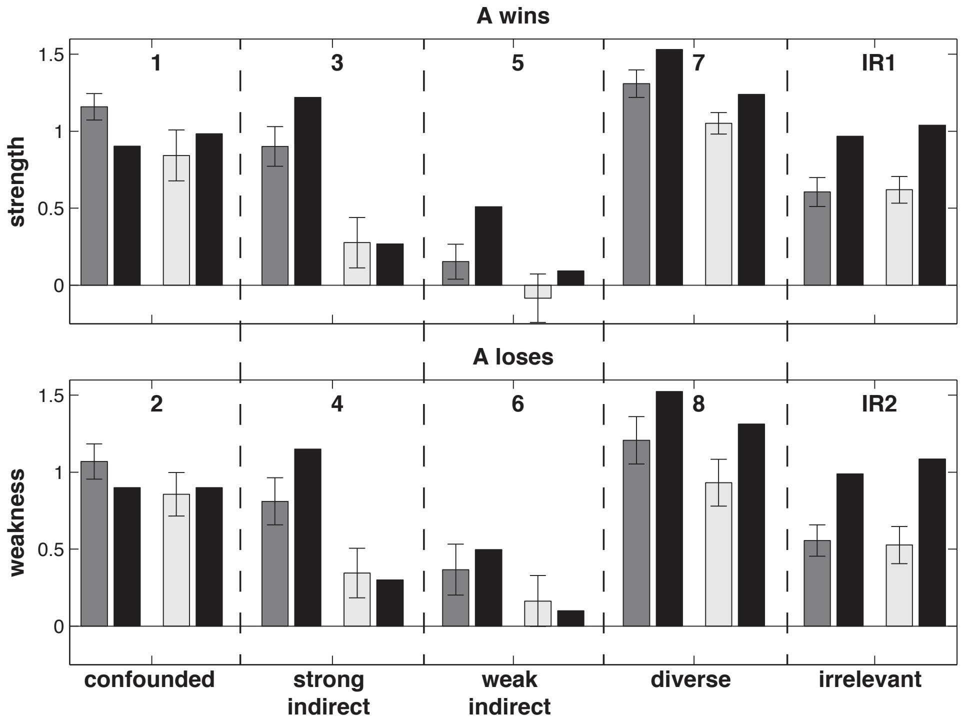


Figure 18.3: Mean strength estimates (grey bars) and model predictions (black bars) in Experiment 2, for estimates given tournament results only (dark grey) and after adding omniscient commentator information (light grey). *Note:* Situations IR1 and IR2 were cases in which the information given by the commentator was irrelevant and the model predicts no differences; see (Goodman et al., 2014) for details.

18.7.1 Mental simulation, sampling, and intuitive theories of physics

There is a central link between the sampling semantics of Church programs, mental simulation, and the causality central to many intuitive theories. A Church program naturally expresses the causal, generative aspect of people's knowledge through the function dependencies in the program. The function dependencies dictate the causal flow of the sampling process: functions whose outputs serve as an input to another function must be evaluated first. Each run of a Church program can be interpreted as the dynamic generation of a possible world that is consistent with the causal laws as specified in the program (Chater & Oaksford, 2013). Because the sampling process is stochastic, a Church program specifies a probability distribution over possible worlds, and different modes of reasoning can be seen as different forms of mental simulation on top of this basic sampling process. While the notion of mental representation and simulation of possible worlds has had many advocates (Craik, 1943; Hegarty, 2004; Johnson-Laird, 1983), the PLoT view integrates this idea naturally into a view of mental models that is also probabilistic, causal and sufficiently expressive to capture core intuitive theories.

We will illustrate the implications of this view via some concrete examples in an intuitive physics domain which was explored extensively in Chapter 15. People can use their intuitive theories of physics to make many different judgments about the trajectories of objects in motion, including predictions of where objects will go in the future, inferences about where objects' latent physical properties or initial conditions, and explanations about *why* objects moved as they did, or judgments of "actual causation": whether one specific physical event caused another specific event to happen. Consider an extension of the billiards scenarios in Chapter 15: in addition to two billiard balls A and B and some solid walls with an

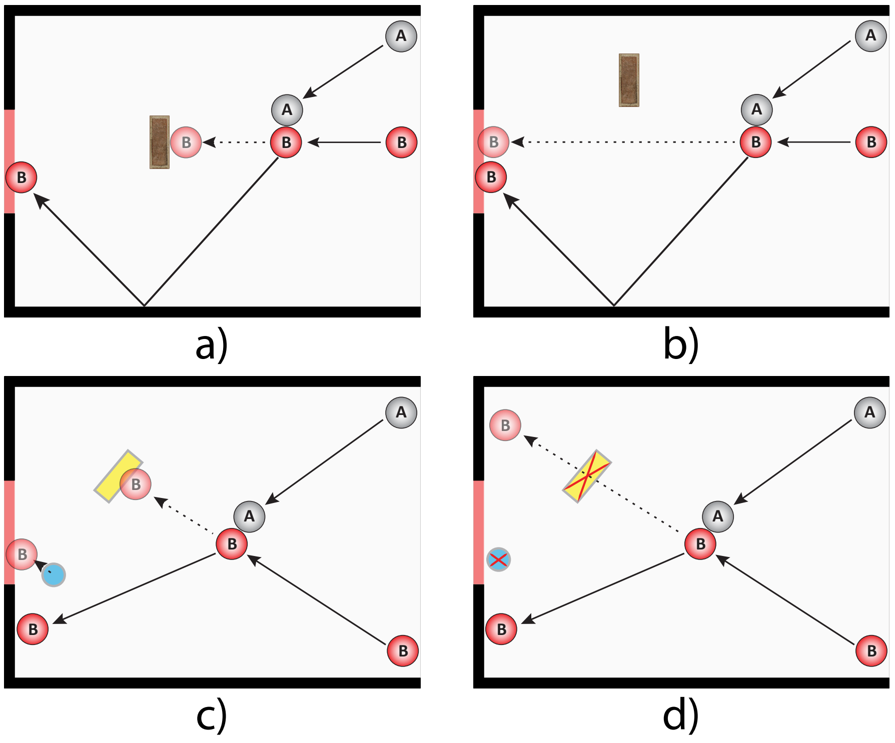


Figure 18.4: Diagrammatic illustrations of four collision events in a simple physics world (see Gerstenberg et al., 2021). *Note*: Solid arrows represent the actual trajectories of ball A before the collision and of ball B before and after the collision. Dashed arrows and faded circles represent the counterfactual trajectory of ball B. The brown rectangle, yellow rectangle and blue circle represent a brick, and the entry and exit of a teleport, respectively.

opening gate, we also have a brick and a teleport gate that can be either active or inactive. Figure 18.4 shows diagrammatic illustrations of causal interactions between balls A and B in this world, assuming simple Newtonian elastic collisions between moving bodies. We might ask someone to predict how B will move at the point A makes contact with it, or to judge where A must have come from if they see only B's motion, or having seen the entirety of any of these four scenarios, to judge whether ball A's collision with ball B caused ball B to go through the red gate on the left of the screen, prevented it from going through, or did neither. The tools of probabilistic programs and sampling-based inference allow us to give a precise formal account of these causal judgments, which also accords well with intuitions of mental simulation and gives strong quantitative fits to behavioral experiments (see Gerstenberg et al., 2021).

To explain these judgments, we first need to be able to represent the relevant physical knowledge at the right level of abstraction. Despite its simplicity, our domain already affords an infinite number of interactions between A and B and we want a model that yields a causal judgment for each possible situation. Rather than having to specify a new model for each causal interaction of interest (as we would have to do if we adopted a Bayesian network formulation Pearl, 2000), we want to represent the general laws that govern the interactions between the objects in our world. One way of representing people's knowledge of physical object motion in Church is by writing down a probabilistic and approximate version of some aspects of Newtonian mechanics. Functions in the Church program compute the inertial time-evolution and the outcome of collisions by taking as input the mass and velocity of objects as well as more general aspects of the world such as friction and gravity. So far these are standard, deterministic simulation routines (so we leave out details). Critically, as in the models in Chapter 15, we also assume that some noise in each object's momentum is inserted just after each collision, and perhaps at other times as well, resulting in trajectories that are noisy versions of their Newtonian counterparts. Research reviewed in Chapter 15 has shown that people's intuitive physical judgments across many domains such as this are well described by such noisy Newtonian simulations (Sanborn, Mansinghka, & Griffiths, 2013; Battaglia, Hamrick, & Tenenbaum, 2013; Smith, Dechter, Tenenbaum, & Vul, 2013; Smith & Vul, 2013). Once we have a Church program that captures people's intuitive physics, we can model predictions about the future (e.g., will ball B go through the gate?) as simple forward simulations, and inferences about the past (e.g., where did ball A likely come from?) by a `query` of the past given the present—simulating possible histories that could have led up to the current state.

More subtly, a Church program can also be used to evaluate probabilistic counterfactuals (e.g., would ball B have gone through the gate if the collision with A hadn't happened?) – the key ingredients in judgments of actual causation as we explained in Chapter 15. In line with the theory in (Pearl, 2000) for counterfactual reasoning in Bayesian networks, the evaluation of counterfactuals in a Church program involves three steps: First, we condition all the random choices in the program based on what actually happened to estimate the unobserved values of the actual world. Second, we realize the truth of the counterfactual antecedent (e.g. that the collision did *not* happen) by intervening in the program execution that generated the actual world. This intervention breaks the normal flow of the program by setting some function inputs to desired values. For example, to model what would have happened if there had been no collision between A and B, we could set ball A's velocity to zero or move ball A outside of the scene shortly before the time of collision. Finally, to evaluate the truth of the counterfactual, we reevaluate all the functions downstream from the point at which we intervened in the program. This process generates a sample over counterfactual world states, and repeatedly running this process allowing for different stochastic functions evaluations can be used to express people's uncertainty over what would have happened in the relevant counterfactual world. Notice that the key feature of Church that allows this process to work is that it specifies a process for sampling particular situations and makes explicit the steps of the causal history that lead up to a situation (in the form of a program execution trace). Counterfactuals are then evaluated by a series of "simulation" steps that result in imagined counterfactual worlds, with a probability distribution over nearby possible worlds induced by the inherently probabilistic nature of the mental simulations.

As we explained in Chapter 15, people's quantitative judgments of actual causation in scenarios like this are closely linked to such a probabilistic counterfactual analysis (Gerstenberg et al., 2021). When judging whether one event caused another event to happen, people appear to estimate something like the probability that the candidate cause was *necessary* to produce the outcome event: the probability that the outcome, which did in fact occur, would not have occurred in a counterfactual world where the candidate cause was absent. By implementing these computations in a probabilistic programming language such as Church, they can be evaluated just as easily on simple and familiar billiards scenarios (like those in Chapter 15) as on a wide range of novel and complex scenarios such as those shown in Figure 18.4, featuring objects that rarely or never appear on a billiards table, but that we can still reason about through mental simulation.

Consider the top pair of diagrams shown in Figure 18.4. In Figure 18.4(a), ball B would have bounced off the brick if it hadn't collided with ball A. In Figure 18.4(b), the motions of the balls are identical, but because the brick is in a slightly different position, we can infer that ball B would have gone through the gate even without the collision with ball A. As predicted, people's judgments about whether A caused B to go through the gate are significantly higher for Figure 18.4(a) compared to 18.4(b) (Gerstenberg et al., 2021). In the bottom pair of cases, the contrast in the relevant counterfactual worlds was realized by comparing situations in which the teleport was either on or off (Figure 18.4(c) and (d), respectively). Again the motions of objects in these two cases were identical, but only when the teleport was on (18.4c) did people judge that A prevented B from going through the gate – because only in this case would B have gone through the gate (via first passing through the teleport), if it had not collided with A.

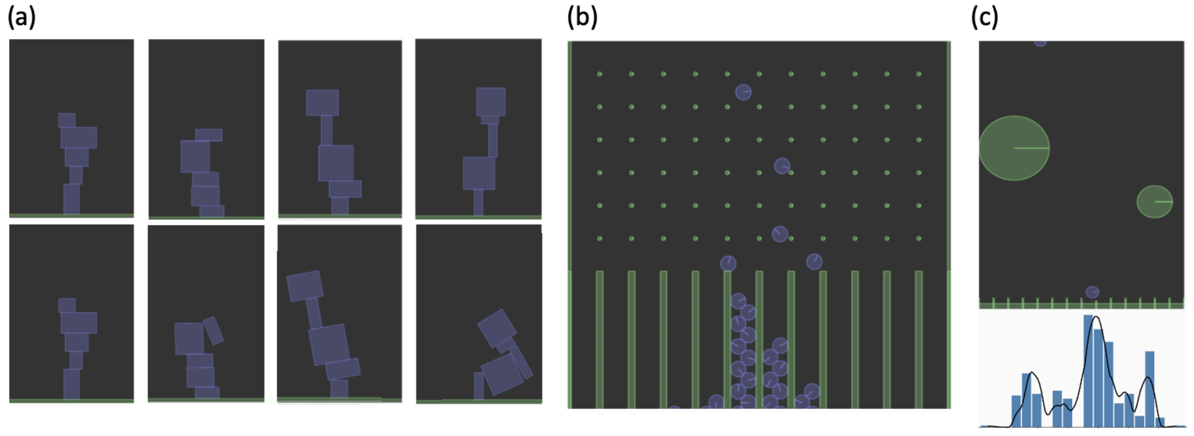


Figure 18.5: Examples of intuitive physical inferences that can be modeled using WebChurch with its built-in physics simulator, where green shapes indicate stationary or fixed objects, and blue shapes indicate freely moveable objects (with uniformly distributed mass): (a) Is a given tower stable (column 1), or slightly, moderately or highly unstable (columns 2-4)?
(`http://v1.probmods.org/generative-models.html#example-intuitive-physics`)
(b) How will a large number of little balls dropped near the middle of this box tend to distribute themselves across the slots?
(`http://v1.probmods.org/generative-models.html#models-simulation-and-degrees-of-belief`)
(c) Given that a blue ball was dropped into the scene somehwere along the top edge and comes to rest at approximxately the location shown along the bottom edge, at what horizontal position along the top edge was the ball likely to have been dropped?
(`http://v1.probmods.org/conditioning.html#example-inverse-intuitive-physics`)

These examples demonstrate the flexibility of people's intuitive theories and the critical way that embodying a theory as a probabilistic generator for mental simulations supports modular counterfactual and causal thinking in novel situations. Once people have learned how balls bounce rigidly off solid objects, whether they are walls or other balls or bricks, the same simulation mechanisms apply. Once people have learned how the teleport works, they have no trouble imagining its effects either and incorporating them into their simulations and causal judgments. And because they are expressed at a high level of abstraction, the same simulation and inference programs can be used for a much wider range of intuitive physical judgments as well: Figure 18.5 shows several very different inferences that can all be implemented as queries in WebChurch using its built-in rigid body simulator. The last example in Figure 18.5(c) follows in the spirit of (Gerstenberg, Siegel, & Tenenbaum, 2018), who present many more quantitative and systematic studies of people making these inverse physical inferences in a simulated "Plinko box". The histogram below shows an approximate posterior of inferred horizontal locations based on 200 MCMC samples, given a final ball position that is observed to be a normal distribution around the horizontal position 200 (on a range between 0 and 350), with standard deviation 10. Note how the three peaks in the posterior reflect three qualitatively distinct hypotheses for where the ball could have been dropped: most likely it was dropped in the center and fell straight down, but it could also have been dropped near either side of the box and landed in the middle after bouncing off the left or right circular obstacles – although if so, it had to have been dropped to the right side of the left-hand obstacle or the left side of the right-hand obstacle. This richly structured and fine-grained posterior belief state – which is also very intuitive to human reasoners – is easily computed in Church via a combination of general purpose sampling-based approximate inference and general purpose approximate physics simulation. Although the implementation details will surely differ, some analogously modular combination of sampling-based inference and simulation is both computationally and psychologically plausible as an account of how these inferences unfold in the human mind.

18.7.2 Recursive inference and intuitive theories of mind

This same general framework for flexible sampling-based inference and causal reasoning applies not only in the domain of intuitive physics, but in any domain for which we are able to represent people's intuitive theories in terms of a probabilistic program. For example, we could model people's judgments about whether agent A's argument convinced agent B to try harder as a function of what actually happened and people's subjective degree of belief that B would still have tried harder had A not said anything. A probabilistic program that captures people's intuitive understanding of psychology looks different from a program that captures people's intuitive understanding of physics, but we can understand people's causal judgments in terms of the same process that compares the actual outcome with the outcomes of mental simulations of the relevant counterfactual worlds.

To illustrate how we might capture the core concepts of an intuitive psychology—a probabilistic model of how agents act rationally in response to their mental states, aiming to satisfy their desires as efficiently as possible given their beliefs—consider as an initial step an extension of the tug of war model above. Now we model laziness for a given player in a given game not simply as a random event, but as a rational choice (not necessarily conscious!) on the part of the player. That is, looking at both their team and the opposition in a given match, a player might estimate that the effective difference in team strengths is so great – either because their team is so strong, or the other team so weak – that it is not worth their effort to try hard. We could express this in a church model by imagining that a player asks themselves, "how should I act, such that my team will win?"; this translates into a `query`:

```
(define lazy (mem (lambda (person game)
  (query
    (define action (flip L))
    action
    (= (teamof person) (winner (team1of game) (team2of game) game))))))
```

where we have helped ourselves to some innocuous helper functions to look up the team of a player and so on. The parameter `L` controls the *a priori* tendency to be lazy; this gives a simple way of including a principle of efficiency: a tendency to avoid undue effort. The condition statement of the query specifies the player's goal—for his team to win the game—hypothetically assuming that this goal will be achieved. The output of the query is an action (trying hard, or not) that is a reasonable guess on the player's part for how that goal may be achieved. An inference about which team will win a match now leads to a sub-inference modeling each player's choice of whether to exert their full effort, given the players on each team. We could further extend this model to take into account private evidence that each player might have about the strengths of the other players, expressing his or her process of belief-formation about the total strengths of the two teams as an additional set of nested sub-inferences.

Because `query` can be defined within the stochastic λ-calculus it is formally a well-defined probabilistic inference to nest one query within another. In typical probabilistic models of cognition we use query to model inferences of an agent (reasoning, learning, and so on), so a model with nested queries can be interpreted as *inference about inference*. The pattern of using an embedded `query` to capture the choices of another agent is a very general pattern for modeling intuitive psychology (Stuhlmüller & Goodman, 2014).

For instance, one general way to model an agent as approximately rational is to model their choice of actions as an inference about what action is likely to achieve their goal given their beliefs about the world's current state and dynamics (Goodman et al., 2012; Botvinick & Toussaint, 2012; Stuhlmüller & Goodman, 2014):

```
(define (choose-action goal? transition state)
  (query
    (define action (action-prior))
    action
    (goal? (transition state action))))
```

Here `state` is a variable encoding the (agent's belief about) the current state of the world, and `transition` is a function encoding (the agent's beliefs about) the transition model or environment dynamics: we can think of this as a causal model, either deterministic or stochastic, which outputs the next world state as a function of the previous state and (optionally) an action taken. `goal?` is a function encoding what the agent desires to see happen in the world: we can think of this as a predicate that takes as input a state and returns `True` or `False` depending on whether the state satisfies the agent's desire.

This formulation gives a natural inferential role semantics (Block, 1986) for core concepts of belief-desire psychology. What is an agent's *goal*? It is a predicate that can be applied to states of the world and which governs the agent's action choices: when the agent has that goal, they are expected to choose actions that they believe will lead to that predicate being true of the world's future state. What are the agent's *beliefs* about the world's causal structure? They are represented by a function that takes as input the agent's belief about the world's current state and any action the agent could take, and returns as output an expected future state, such that if that future state satsfies the agent's current goal, then the agent could reasonably be expected to take that action.

We can now model canonical theory-of-mind inferences by nesting the above query representing a (probabilistic) rational planning process inside a query for goal inference. Conditioning on an agent having taken some action in a given state, this outer query asks which goal (drawn from some goal prior) is most likely to have given rise to that action choice via the above inference process:

```
(define (inferred-goal? action transition state)
  (query
    (define goal? (goal-prior))
    goal?
    (equal? action (choose-action goal? transition state))))
```

This corresponds to a simple sampling-based version of the Bayesian **inverse planning** view of theory of mind, as described in Chapter 14 and Baker, Tenenbaum, and Saxe (2007).

The nested query formalism allows planning and inverse planning computations to be flexibly interleaved to reason about how multiple agents will think and act in a wide range of social settings. Many examples can be found in the "Inference about inference" section of the Church-based web book (`http://v1.probmods.org/inference-about-inference.html`) or its successor using WebPPL. Two other books dive more deeply into modeling agents using probabilistic programs (both in WebPPL): *Modeling agents with probabilistic programs* (Evans, Stuhlmüller, Salvatier, & Filan, 2017) (`https://agentmodels.org/`) develops PPL models of planning based on Markov decision processes (MDPs) and partially observable MDPs (or POMDPs) (see Chapter 7), and uses these for modeling intuitive reasoning about agent's goals, preferences, and beliefs as Bayesian inverse planning (see Chapter 14), as well as modeling the expectations about the actions of boundedly rational agents and simple multi-agent interactions (e.g., playing simple games). *Probabilistic language understanding* (Scontras, Tessler, & Franke., 2017) takes a PPL approach to Rational Speech Act (RSA) modeling of communication as a form of rational goal-directed action, and modeling language understanding as Bayesian inference about a speaker's communicative intents (see our Chapter 16). Zhi-Xuan, Mann, Silver, Tenenbaum, and Mansinghka (2020) extends probabilistic programming for Bayesian inverse planning to more abstract symbolic planning settings, using the **Gen** PPL (Cusumano-Towner et al., 2019) and the **Planning Domain Description Language** to represent an agent's world models, goals, and plans in first-order logic terms; using Gen also enables explicit probabilistic modeling of a boundedly rational agent's approximate planning algorithm, to make goal inference robust when the agent might be prone to planning mistakes of various kinds.

Of course many additional components will be needed to build fully adequate models of human intuitive psychology—agents form higher-order beliefs about each other, experience emotions, make moral judgments, and so on. Yet the ease with which researchers have already been able to express core aspects of theory of mind in probabilistic programming terms – especially those within the broad framework of inverse planning, shows the potential of the PLoT approach both here and more generally for capturing core human commonsense reasoning. Extending the inverse planning approach to model inferences about agents' emotions (Houlihan, Kleiman-Weiner, Hewitt, Tenenbaum, & Saxe, 2023), communication about social roles and relationships (Radkani, Tenenbaum, & Saxe, 2022), and other rich social-cognitive inferences is one of the most active and exciting areas of research in probabilistic programming models of human cognition.

18.8 Concept acquisition

Thus far we have sketched a notion of concepts as stochastic functions in a probabilistic programming language, and intuitive theories as systems (or libraries) of interrelated concepts. We have also described how such intuitive theories can be used to describe the complex causal knowledge that people use to reason about the world. If concepts constitute a library of useful (stochastic) functions, what then is concept learning?

Forming new concepts from examples is fundamentally a problem of induction—in our case the problem of program induction. This can be formulated as Bayesian inference of a set of concepts that best explain the experience we have in the world: conditioned on generating the examples we have seen, what is the likely new concept? Hypothesized concepts are formed in an effective language of thought based on the concepts learned so far—all the expressions that can be formed by composing the underlying PLoT and the already-defined function symbols. We can view these hypotheses as being generated by a higher-order **program-generating program**, a stochastic meta-program that generates candidate stochastic programs in our PLoT that might explain a given set of observed examples. Concept learning then reduces to a standard Church query: conditioning on the output of this process (the output of the stochastic program generated by the stochastic meta-program) being equal to the observed examples, we query for symbolic expressions in our PLoT that are both likely to be generated from the meta-program and that meet this condition.

While concept learning as probabilistic program induction is thus philosophically and mathematically well-posed, inference over such a vast combinatorial space can be extremely challenging and much more research is needed to fully realize this vision. The approach has already proven successful, however, in many settings of inductive learning that are simple enough to be studied in behavioral experiments: Boolean category learning (Goodman, Tenenbaum, Feldman, & Griffiths, 2008), relational concept learning (Kemp, Goodman, & Tenenbaum, 2008), learning action concepts such as the drawing programs underlying handwritten characters (Lake, Salakhutdinov, & Tenenbaum, 2015), learning number concepts (Piantadosi, Tenenbaum, & Goodman, 2012), learning list routines (Rule et al., under review), learning grammatical expressions (Yang & Piantadosi, 2022), and spatiotemporal sequence learning (Mills, Cheyette, & Tenenbaum, 2023). Chapter 18 presents an in-depth treatment of this work, including several detailed case studies of Bayesian program learning and concrete illustrations of the sampling-based inference algorithms that have been most successfully used to learn such expressive representations.

Of course, human concept learning is not always or merely a process of inducing a model that best accounts for a set of observed examples, and the PLoT view is well-positioned to engage richer learning dynamics and mechanisms. In particular, social and communicative factors, with or without natural language experience, can play critical roles. Shafto, Goodman, and Frank (2012) show how the social context of examples in concept learning—for instance, examples generated communicatively by a helpful teacher—can strongly and rationally impact the generalizations learners make beyond the examples. Natural language provides a strongly constraining form of evidence for concept formation especially in children's cognitive development —for instance Piantadosi et al. (2012) model how a natural language count-list provides children with a crucial input in learning concepts of natural number. The PLoT offers a powerful way to think about these and other bootstrapping phenomena at the interface of social interaction, language acquisition and concept acquisition, such as the contributions of syntactic and semantic bootstrapping in learning verbs, or the contributions of pragmatic inference in learning quantifiers. Again, much further research is needed to understand how these social and linguistic factors integrate with inductive learning from examples in a full theory of concept acquisition; this will also need to build on general-purpose probabilistic programming tools for modeling pragmatic language understanding (Goodman & Stuhlmüller, 2013; Lassiter & Goodman, 2013) and grounding natural language meaning representations compositionally and probabilistically in PLoT-based world models for reasoning (Goodman & Lassiter, 2015; Wong et al., 2023). What is important for us here is that the PLoT provides us with a theory of concepts, an approach to concept learning as probabilistic inference, and a unifying framework for modeling people's intuitive theories of social cognition and linguistic meaning that lets us explore these interactions in a productive way.

Note that this PLoT view of concept learning, although it emphasizes compositionality, is not incompatible with the view that many natural concepts begin life as little more than unanalyzed atoms or placeholder symbols in the mind, which only later come to acquire rich content. The impetus to add such a placeholder symbol may come from natural language (upon hearing a new word), from the interaction of knowledge about natural kinds and specific examples (as suggested by Margolis, 1998 and Carey, 2015), or from other explanatory pressures. Deeper causal content, knowledge about the causal relationships that instances of this concept enter into with other kinds of entities, and relationships to other concepts can be incorporated into the web of function definitions inductively and progressively, as the learner encounters additional examples and gains broader experience in a new domain.

An important implication of this richer inductive approach is that concept learning can change the effective language of thought in a domain: this happens when learned concepts are added as primitives to the language out of which new mental programs and new concepts will be composed. While the new effective language has the same mathematical expressivity as the underlying universal PLoT (which generates the hypotheses for all new concepts past and future), particular thoughts may be vastly simpler—expressed in shorter programs, and thus more computationally tractable—in the effective language. This has many cognitive implications. In particular, as the effective language of thought evolves, it gives rise to an evolving inductive bias for learning. Concepts which are initially complex and unlikely to be constructed by a naive learner may become simpler and more plausible later on in the process of elaborating the learner's conceptual library. This process may be a critical driver of children's long-term cognitive development, as well as adults' development of expertise in a novel domain (Rule, Tenenbaum, & Piantadosi, 2020).

Although not directly implemented in a probabilistic programming language, DreamCoder (Ellis et al., 2021, 2020) and its descendants (Wong, Ellis, Tenenbaum, & Andreas, 2022; Bowers et al., 2023) are recent AI approaches for Bayesian program learning inspired by Church and the PLoT hypothesis that show how such "library learning" can dramatically expand the repertoire of effectively learnable concepts given a limited amount of computational resources. These systems work by not only adding newly learned concepts to the library, but also by abstracting out program components that are implicitly shared between previously learned concepts, in order to identify the most explanatory and compact theory of a domain – effectively performing a heuristic version of hierarchical Bayesian inference to learn the prior for generating novel concepts. LAPS (Wong et al., 2022) extends DreamCoder to learn a joint prior on programs and natural language translations of those programs, and models how – consistent with studies in cognitive development (Carey, 2009) – experience with natural language in a doman can bootstrap learner's theory acquisition, and joint acquisition of word meanings and novel concepts, well beyond what could be gleaned merely from observed perceptual examples.

18.9 Future directions

The tools and ideas in this chapter inherit much from earlier Bayesian modeling approaches that work with richly structured representations, but frame them in new ways that also suggest important open directions for research in Bayesian cognitive science. This starts with the basic mechanics of reasoning under uncertainty, which gives a natural connection to inferential role notions of concept meaning (Block, 1986): It is not merely the proximal definitions, but the complex ways that information flows under inference that determine the content of our concepts—much as we saw with the phenomena of explaining away in generative perception and causal reasoning. Rather than working at the level of monolithic probability distributions, the stochastic lambda calculus and Church allow us to work from the point of view of generative, sampling systems. As a form for knowledge representation, this makes a key connection to classic ways of studying mental simulation and mental imagery (Shepard, 1994), and more generally opens up rich avenues for modeling the roles of imagination, counterfactual reasoning, and other ways human beings can use their conceptual systems to think flexibly about possible ways the world could be. It also naturally brings in the tools of sampling-based approximate inference that can operate very generally on a wide range of symbolic structures—even the traces of Turing-complete programming languages—as candidate mechanistic models for thinking. And finally, by drawing on hierarchical Bayesian frameworks for learning inductive constraints and learning to learn, the PLoT lets us think in terms of programs as hypotheses for novel concepts, and probabilistic meta-programs that generate domain-specific languages of programs as powerful and dynamically evolvable hypothesis spaces and priors for inductive concept learning.

Looking ahead, we should start by acknowledging that Church models, and the PLoT more generally, are intended to capture the knowledge people use to reason about the world and the inferences supported by this knowledge, but not in any precise way the algorithmic processes underlying inference, much less their neural instantiation. Connecting across these levels of analysis is one of the key future challenges for the PLoT hypothesis. Early implementations of Church as a programming language suggest one avenue of approach: Church's `query` works through various combinations of caching and Monte Carlo simulation, a very different view of computation than one might expect from a course on probability—not so much arithmetic tabulation as stochastic dynamical systems tuned to result in samples from the desired distributions, and not so unlike the way many neuroscientists think about computation in the brain. Long engineering practice shows that these algorithms can give efficient solutions to tough statistical inference problems, and recent work has provided initial connections between such sampling-based inference algorithms and the multiple levels at which human cognitive processes can be analyzed (see Chapter 11). But we are just at the beginning of this journey. The design and implementation of more recent probabilistic programming languages suggest fascinating new ways to connect the computational, algorithmic and neural levels of analysis that come from thinking about how to engineer probabilistic programs at scale: for example via amortized inference in deep neural networks (Bingham et al., 2019a), or programmable inference that mixes symbolic and neural motifs (Cusumano-Towner et al., 2019), or most intringuingly, generalizations of sequential Monte Carlo that can be compiled into the network dynamics of biologically realistic spiking neurons (Bolton, Matheos et al., in prep).

18.10 Conclusion

As the other chapters in this book illustrate so well, advances in Bayesian models of cognition have the potential to impact almost every area of cognitive science. These models share the basic mechanics of probabilistic inference and a core philosophical commitment to what it means to reverse-engineer the mind, but they can bring a bewildering and heterogenous array of additional representational tools and claims. The universal probabilistic programming view presented in this chapter, and the underlying mathematics of the stochastic λ-calculus, serves as a key unification showing that all of these Bayesian models can be represented in, and hence reduced to, a simple system built from little more than function abstraction and random choice. This gives hope that future advances in probabilistic modeling of targeted domains will continue to be compatible with each other and can ultimately be combined into a broader architecture for modeling human knowledge, reasoning and learning.

In a step towards that unification, we showed how embedding the stochastic λ-calculus in a probabilistic programming language such as Church offers a powerful way to model the structure and function of core concepts in higher-level cognition, along with the larger conceptual systems, or intuitive theories, that individual concepts contribute to and gain meaning from. Viewing concepts as the stable representations of a *probabilistic language of thought*—more formally, as functions in an enriched stochastic lambda calculus— builds on and unifies traditional approaches to concepts and mental representation that have often been cast as rivals. Like classical theories of concepts, the PLoT puts compositionality and symbolic scaffolding at center stage. Unlike these theories however, but very much in the spirit of prototype, exemplar, and connectionist approaches to concepts, the PLoT views human reasoning as fundamentally graded. Compositionality has long been recognized as crucial to explaining the open-endedness and productivity of thought, while probabilistic inference explains why graded reasoning is necessary to make good guesses and good bets in an uncertain world. The PLoT seeks to explain how these different aspects of complex human cognition fit together and distinctively enable each other, making each work better, in a way that previous formal theories have not. It also helps us understand many themes in everyday cognition with both new qualitative insights and quantitative accuracy.

Beyond the conceptual advances and unification that Church brings to Bayesian cognitive modeling, more recent PPLs such as Pyro and Gen focus on scaling up inference so that universal probabilistic programs can also be the basis for practical and state-of-the-art research in machine learning and AI. These tools are already letting data scientists and AI engineers work effectively with the kinds of richly structured probabilistic models that cognitive scientists have long see as central in everyday human thinking – and which should be just as central in modern computational science and autonomous systems that we want to function in the human world. As we finish writing the first edition of this book, engineering efforts to scale up probabilistic programming for AI are just beginning to hit an inflection point of speed, cost-effectiveness and generality. We look forward to following and building on the trajectories of these engineering developments, and to exploring the hypotheses they will generate for reverse-engineering probabilistic inference in the mind and brain.

References

Abelson, H., Sussman, G. J., & Sussman, J. (1996). *Structure and interpretation of computer programs.* MIT Press.

Baker, C. L., Tenenbaum, J. B., & Saxe, R. R. (2007). Goal inference as inverse planning. In *Proceedings of the 29th Annual Meeting of the Cognitive Science Society.*

Battaglia, P., Hamrick, J., & Tenenbaum, J. B. (2013). Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, *110*(45), 18327–18332.

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., & Goodman, N. D. (2019a). Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research*, *20*(28), 1–6.

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., & Goodman, N. D. (2019b). Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, *20*(1), 973–978.

Block, N. (1986). Advertisement for a semantics for psychology. In (Vol. 10, pp. 615–678).

Block, N. (1998). Conceptual role semantics. In *The Routledge Encyclopedia of Philosophy.* Routledge.

Botvinick, M., & Toussaint, M. (2012). Planning as inference. *Trends in Cognitive Sciences*, *16*(10), 485–488.

Bowers, M., Olausson, T. X., Wong, L., Grand, G., Tenenbaum, J. B., Ellis, K., & Solar-Lezama, A. (2023). Top-down synthesis for library learning. *Proceedings of the ACM on Programming Languages*, *7*(POPL), 1182–1213.

Carey, S. (1985). *Conceptual change in childhood.* MIT Press.

Carey, S. (2009). *The origin of concepts.* Oxford University Press.

Carey, S. (2015). Why theories of concepts should not ignore the problem of acquisition. In *The conceptual mind: New directions in the study of concepts.* MIT Press.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., & Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of statistical software*, *76*.

Chater, N., & Oaksford, M. (2013). Programs as causal models: Speculations on mental programs and mental representation. *Cognitive Science*, *37*(6), 1171–1191.

Craik, K. J. W. (1943). *The nature of explanation.* Cambridge University Press.

Cusumano-Towner, M. F., Saad, F. A., Lew, A. K., & Mansinghka, V. K. (2019). Gen: A general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th acm sigplan conference on programming design and implementation* (pp. 221–236). ACM.

Doucet, A., Freitas, N. de, & Gordon, N. (2001). *Sequential Monte Carlo methods in practice.* Springer.

Ellis, K., Wong, C., Nye, M., Sablé-Meyer, M., Morales, L., Hewitt, L., Cary, L., Solar-Lezama, A., & Tenenbaum, J. B. (2021). Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation* (pp. 835–850).

Ellis, K., Wong, L., Nye, M., Sable-Meyer, M., Cary, L., Morales, L., Hewitt, L., Solar-Lezama, A., & Tenenbaum, J. B. (2020). Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning.

Evans, O., Stuhlmüller, A., Salvatier, J., & Filan, D. (2017). *Modeling Agents with Probabilistic Programs.* `http://agentmodels.org`. (Accessed: 2023-9-20)

Fodor, J. A. (1975). *The language of thought.* Harvard University Press.

Fodor, J. A. (1998). *Concepts: Where cognitive science went wrong.* Oxford University Press.

Ge, H., Xu, K., & Ghahramani, Z. (2018). Turing: A language for flexible probabilistic inference. In *International Conference on Artificial Intelligence and Statistics* (pp. 1682–1690).

Gerstenberg, T., & Goodman, N. D. (2012). Ping Pong in Church: Productive use of concepts in human probabilistic inference. In *Proceedings of the 34th Annual Meeting of the Cognitive Science Society* (pp. 1590–1595).

Gerstenberg, T., Goodman, N. D., Lagnado, D. A., & Tenenbaum, J. B. (2021). A counterfactual simulation model of causal judgments for physical events. *Psychological Review.*

Gerstenberg, T., Siegel, M. H., & Tenenbaum, J. B. (2018). What happened? reconstructing the past from vision and sound. In *Proceedings of the 40th Annual Meeting of the Cognitive Science Society.*

Gerstenberg, T., & Tenenbaum, J. B. (2017). Intuitive theories. In M. Waldmann (Ed.), *Oxford handbook of causal reasoning* (pp. 515–548). Oxford University Press.

Goodman, N., Baker, C., Bonawitz, E., Mansinghka, V., Gopnik, A., Wellman, H., Schulz, L., & Tenenbaum, J. (2006). Intuitive theories of mind: A rational approach to false belief. In *Proceedings of the 28th Annual Meeting of the Cognitive Science Society.*

Goodman, N., Mansinghka, V., Roy, D. M., Bonawitz, K., & Tenenbaum, J. B. (2012). Church: a language for generative models. *arXiv preprint arXiv:1206.3255.*

Goodman, N. D., & Lassiter, D. (2015). Probabilistic semantics and pragmatics: Uncertainty in language and thought. In S. Lappin & C. Fox (Eds.), *The handbook of contemporary semantic theory* (2nd ed.). Wiley-Blackwell.

Goodman, N. D., & Stuhlmüller, A. (2013). Knowledge and implicature: Modeling language understanding as social cognition. *Topics in Cognitive Science*, *5*(1), 173–184.

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

Goodman, N. D., Tenenbaum, J. B., & Gerstenberg, T. (2014). *Concepts in a probabilistic language of thought* (Tech. Rep.). Center for Brains, Minds and Machines (CBMM).

Goodman, N. D., Ullman, T. D., & Tenenbaum, J. B. (2011). Learning a theory of causality. *118*(1), 110–119.

Gopnik, A. (2003). The theory theory as an alternative to the innateness hypothesis. In L. M. Antony & N. Hornstein (Eds.), *Chomsky and his critics* (pp. 238–254). Wiley.

Gopnik, A., Glymour, C., Sobel, D., Schulz, L., Kushnir, T., & Danks, D. (2004). A theory of causal learning in children: Causal maps and Bayes nets. *Psychological Review*, *111*, 1-31.

Gopnik, A., & Wellman, H. M. (2012). Reconstructing constructivism: causal models, bayesian learning mechanisms, and the theory theory. *Psychological Bulletin*, *138*(6), 1085-1108.

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

Griffiths, T. L., & Tenenbaum, J. B. (2009). Theory-based causal induction. *Psychological review*, *116*(4), 661.

Hastings, W. K. (1970). Monte Carlo methods using Markov chains and their applications. *Biometrika*, *57*, 97-109.

Hegarty, M. (2004). Mechanical reasoning by mental simulation. *Trends in Cognitive Sciences*, *8*(6), 280–285.

Houlihan, S. D., Kleiman-Weiner, M., Hewitt, L. B., Tenenbaum, J. B., & Saxe, R. (2023). Emotion prediction as computation over a generative theory of mind. *Philosophical Transactions of the Royal Society A*, *381*(2251), 20220047.

Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference, and consciousness.* Harvard University Press.

Kemp, C., Goodman, N. D., & Tenenbaum, J. B. (2008). Learning and using relational theories. *Advances in Neural Information Processing Systems*, *20*, 753–760.

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, *350*(6266), 1332–1338.

Lassiter, D., & Goodman, N. (2013). Context, scale structure, and statistics in the interpretation of positive-form adjectives. *Semantics and Linguistic Theory*, *23*, 587-610.

Mansinghka, V., Selsam, D., & Perov, Y. (2014). Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099.*

Margolis, E. (1998). How to acquire a concept. *Mind & Language*, *13*(3), 347–369.

McCarthy, J. (1960). Recursive functions of symbolic expressions and their computation by machine, Part i. *Communications of the ACM*, *3*(4), 184–195.

Milch, B., & Russell, S. (2006). General-purpose MCMC inference over relational structures. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI)* (pp. 349–358).

Mills, T., Cheyette, S. J., & Tenenbaum, J. B. (2023). Spatiotemporal sequence learning as probabilistic program induction. In *Thirty-seventh conference on neural information processing systems.*

Murphy, G. L. (2002). *The big book of concepts.* MIT Press.

Pearl, J. (2000). *Causality: Models, reasoning and inference.* Cambridge University Press.

Piantadosi, S. T., Tenenbaum, J. B., & Goodman, N. D. (2012). Bootstrapping in a language of thought: A formal model of numerical concept learning. *Cognition*, *123*(2), 199-217.

Radkani, S., Tenenbaum, J. B., & Saxe, R. (2022). Modeling punishment as a rational communicative social action. In *Proceedings of the Annual Meeting of the Cognitive Science Society* (Vol. 44).

Rehder, B. (2003). A causal-model theory of conceptual representation and categorization. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *29*, 1141-1159.

Rule, J. S., Piantadosi, S. T., Cropper, A., Ellis, K., Nye, M., & Tenenbaum, J. B. (under review). Human-level and human-like: an efficient model of symbolic learning.

Rule, J. S., Tenenbaum, J. B., & Piantadosi, S. T. (2020). The child as hacker. *Trends in Cognitive Sciences*, *24*(11), 900–915.

Sanborn, A. N., Griffiths, T. L., & Navarro, D. J. (2010). Rational approximations to rational models: Alternative algorithms for category learning. *Psychological Review*, *117*(4), 1144–1167.

Sanborn, A. N., Mansinghka, V. K., & Griffiths, T. L. (2013). Reconciling intuitive physics and Newtonian mechanics for colliding objects. *Psychological Review*, *120*(2), 411–437.

Scontras, G., Tessler, M. H., & Franke., M. (2017). *Probabilistic language understanding: An introduction to the Rational Speech Act framework.* `http://problang.org`. (Accessed: 2023-9-20)

Shafto, P., Goodman, N. D., & Frank, M. C. (2012). Learning from others: The consequences of psychological reasoning for human learning. *Perspectives on Psychological Science*, *7*(4), 341-351.

Shepard, R. N. (1994). Perceptual-cognitive universals as reflections of the world. *Psychonomic Bulletin & Review*, *1*, 2–28.

Smith, K. A., Dechter, E., Tenenbaum, J. B., & Vul, E. (2013). Physical predictions over time. In *Proceedings of the 35th Annual Meeting of the Cognitive Science Society.*

Smith, K. A., & Vul, E. (2013). Sources of uncertainty in intuitive physics. *Topics in Cognitive Science*, *5*(1), 185–199.

Spelke, E. S. (2022). *What babies know: Core knowledge and composition.* Oxford University Press.

Stuhlmüller, A., & Goodman, N. D. (2014). Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. *Cognitive Systems Research*, *28*, 80–99.

Tenenbaum, J. B. (2000). Rules and similarity in concept learning. In *Advances in Neural Information Processing Systems 12.*

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

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

Wellman, H. M., & Gelman, S. A. (1992). Cognitive development: Foundational theories of core domains. *Annual Review of Psychology*, *43*, 337-375.

Wong, L., Ellis, K., Tenenbaum, J. B., & Andreas, J. (2022). Leveraging language to learn program abstractions and search heuristics. *arXiv preprint arXiv:2106.11053.*

Wong, L., Grand, G., Lew, A. K., Goodman, N. D., Mansinghka, V. K., Andreas, J., & Tenenbaum, J. B. (2023). From word models to world models: Translating from natural language to the probabilistic language of thought.

Wood, F., Meent, J. W., & Mansinghka, V. (2014). A new approach to probabilistic programming inference. In *International Conference on Artificial Intelligence and Statistics* (pp. 1024–1032).

Yang, Y., & Piantadosi, S. T. (2022). One model for the learning of language. *Proceedings of the National Academy of Sciences.*

Zhi-Xuan, T., Mann, J., Silver, T., Tenenbaum, J., & Mansinghka, V. (2020). Online bayesian goal inference for boundedly rational planning agents. *Advances in neural information processing systems*, *33*, 19238–19250.