DOI: 10.1111/cdev.14031

SPECIAL SECTION

CHILD DEVELOPMENT

The algorithmic origins of counting

Steven T. Piantadosi

Department of Psychology, UC Berkeley, Berkeley, California, USA

Correspondence
Steven T. Piantadosi, Department of Psychology, UC Berkeley, Berkeley, CA, USA.
Email: stp@berkeley.edu

Funding information
NSF, Grant/Award Number: 1760874 and 2000759

Abstract
The study of how children learn numbers has yielded one of the most productive research programs in cognitive development, spanning empirical and computational methods, as well as nativist and empiricist philosophies. This paper provides a tutorial on how to think computationally about learning models in a domain like number, where learners take finite data and go far beyond what they directly observe or perceive. To illustrate, this paper then outlines a model which acquires a counting procedure using observations of sets and words, extending the proposal of Piantadosi et al. (2012). This new version of the model responds to several critiques of the original work and outlines an approach which is likely appropriate for acquiring further aspects of mathematics.

Children's acquisition of natural number has taken center stage in recent debates about the nature of human learning. In the United States, children typically learn to correctly use the words "one", "two", "three", etc. by about age 3 or 4 (Carey, 2009; Wynn, 1992), but the underlying cognitive capacities that enable them to do this have been long debated. Many prominent theories have held that the concepts these words map to are innate, perhaps based on simple rules that are present in people without learning. For example, if both a concept of *ONE* and "plus one" were innate, then combining these operations would allow one to define all integers. Often the "plus one" operation is written as a *successor* function S defined by $S(x) = x + 1$. Thus, the number concepts would be defined through repeated applications of S,

$$ONE, \quad S(ONE), \quad S(S(ONE)), \quad S(S(S(ONE))), \quad S(S(S(S(ONE)))), \ldots.$$

This approach is *generative* in the sense that what is innately given is a set of operations which, when applied appropriately, can generate a list of numbers, building a number n out of the representation of $n-1$ (Leslie et al., 2008). This view is also motivated by mathematical logic, where integers can be defined in mathematical theories through this type of successor rule, dating back to Peano (1889). Similarly, it has also been hypothesized that number relies on innate (not learned) principles of counting (Gallistel & Gelman, 1992; Gelman & Gallistel, 1978; Gelman & Meck, 1983)—for instance, that counting words should be aligned one-to-one with objects being counted, or knowledge that the order in which a set is enumerated does not affect the final answer. Such theories do not necessarily predict that children know everything about number from birth, but historically they have been imprecise about the mechanisms of change, including the extent to which maturation and data play a role in explaining children's improvements.

Children's exact number representations have also been theorized to be closely tied to our ability to *approximately* represent and compare different quantities (Cantlon, 2012; Dehaene, 2011; Halberda et al., 2008; Shusterman et al., 2016; van Marle et al., 2018; Whalen et al., 1999). For example, Shusterman et al. (2016) show that learning a principle behind number meaning is associated with improvement in estimation. Importantly, accounts which hold that approximation plays a key role in number representations have not explained the origins of exactness in number (Carey & Barner, 2019), nor the poor performance of experimental interventions targeting approximation in improving exact mathematics (Szkudlarek et al., 2021; Szűcs & Myers, 2017).

Abbreviation: MCMC Markov-Chain Monte Carlo

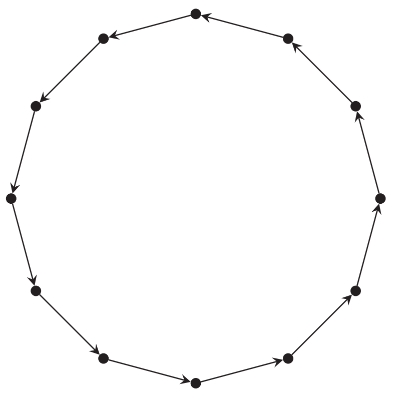
The title for this Special Section is **Formalizing Theories of Child Development**, edited by Willem Frankenhuis.

An alternative to proposals that key principles of number are innate is that number is *constructed* (e.g., Xu, 2019). Proposing a theory of how number might be constructed, Carey (2009) argues that children reason analogically about the relationship between the first few number words ("one", "two", "three", etc.) and their corresponding cardinal meanings (1, 2, 3, etc.). When they discover that one more in the verbal list corresponds to one more object in the corresponding set, they discover an abstract understanding of how verbal counting relates to quantity. This analogy is the insight that allows children to understand the early number system and extend word meaning beyond the small number range. A qualitative shift is required to get to large numbers because children have resource-limited working memory representations of sets that can only represent sets exactly up to a cardinality of three or four (Carey, 2009; Feigenson & Carey, 2003, 2005). The limitations of memory mean that children must build something new—a specific analogical or structural relationship to the list of words—in order to go beyond their memory systems. As this account would predict, people appear unable to use number to exactly match cardinality in the absence of direct visual cues unless they have the numerical labels (Pitt et al., 2022), challenging the notion that learners possess concepts before learning the words. Constructive theories have the advantage of explaining many of the facts about the diversity of number across cultures (Carey & Barner, 2019; O'Shaughnessy et al., 2021), including that some cultures have no number words at all, that number appears to be difficult for both cultures and children to create, and that the forms that number ends up taking are diverse, often reliant on local cultural or economic pressures. Indeed, members of one indigenous Amazonian group have been shown to handle even multiplication by fives without robustly understanding addition by one, indicating that learning mechanisms are capable of a striking breadth of different kinds of number-like systems (O'Shaughnessy et al., 2023). Which structures and analogies children build will be dependent on their input, although inherent limits in cognitive systems like memory play an important role in determining how acquisition proceeds.

However, Carey's account has been critiqued for failing to resolve *how* children solve the problem of discovering that numbers have an infinite, discrete structure like the following (Rips et al., 2006; Rips, Asmuth, & Bloomfield, 2008; Rips, Bloomfield, & Asmuth, 2008):

$$\bullet \rightarrow \bullet \rightarrow \bullet \rightarrow \bullet \rightarrow \bullet \rightarrow \ldots$$

as opposed to any other logical possibility, like a circular system (e.g., how we label time on a clock):



One way to phrase this critique is that Carey's account does not specify how children pick the right analogy, as opposed to any of the infinitely many possibilities. Specifically, children often learn other forms of labeling in other domains: how we label time, for example, starts over again at "one" when we reach "twelve," so why do not children think that's a good hypothesis for number (Rips et al., 2006)? Similarly, other lists children learn are finite (e.g., the presidents or the alphabet) so children must be capable of distinguishing finite and infinite lists. What leads them to believe that numbers are infinite? It seems that children must *already* know the right structure in order to craft the correct analogy, and so perhaps numbers must be built-in, in some sense, for any theory.

These issues arise in part from stating theories in intuitive language, as opposed to formal computational implementations that make explicit the hypothesis space as well as the inference mechanisms that choose between hypotheses. While models are always motivated by intuition, formalization seems critical to make progress generally in the field (Devezer et al., 2021; van Rooij & Baggio, 2021). Formalization may also help debates and theory move beyond the too-simplistic binary distinction between nativism and empiricism, and toward an integrated approach (Elman, 1997; Spelke & Kinzler, 2007), placing nativism and empiricism, in the words of Spelke and Kinzler (2009), "in dialog, not debate" (p. 96). This is because any learning model has parts that are innate (e.g., the learning mechanism and system for specifying hypotheses) and parts which are learned (e.g., the particular hypothesis chosen).

The goal of the present paper is to provide a tutorial overview of how one may construct a formalized learning model in this key case study of natural number learning. This learning model draws on a number of key theories, including Carey's, as well as inductive ideas more generally about how to learn complex, structured representations (Rule et al., 2020; Tenenbaum et al., 2011).

A PROGRAM-LEARNING MODEL FOR NUMBER

The basic idea of modeling number acquisition as algorithm learning was outlined by Piantadosi et al. (2012) (henceforth PTG). They presented a Bayesian model which learns a mental representation analogous to a computer program to map sets (e.g., $\{o, o\}$) to words (e.g., "two"). In this view, learners would observe how parents use number words—what words tend to co-occur with what sets—and construct a mental procedure to best explain the data they see. The space of possible programs such a learner considers is broad, motivated in part by the range of algorithmic representations that children develop across childhood, everything from learning to tie shoelaces, to playing games, to rules and manners, to the ability to do later mathematics like arithmetic and fractions (Rule et al., 2020). Such breadth of ability suggests that children have a fairly unconstrained space of possible representations.

One way to formalize this is to follow the general setup of both programming languages and mathematical logic, where we can assume an initial set of *primitive* operations that are used to build more complex hypotheses. A primitive is an operation which can be treated as a discrete symbol and combined with other symbols. Primitives perform *functions*, for example, in mathematics we might think of division as a primitive in arithmetic. Division takes two numbers and returns a third (e.g., $28 \div 7 = 4$). Primitives are often assumed to be not learned, perhaps innate or the result of maturation. However, number learning takes place once children are a few years old, and they may, therefore, draw on operations or representations which they have learned already. It is convenient to call all of these operations primitives as well since the role they play in the hypothesis space—at least in current models—is the same as primitives that are innate.

Primitives combine into more complex mental representations through *composition*. A composition just takes the output of one operation and sends it to the input of another. For example, in mathematics, $\exp(\sin(x))$ sends the output of $\sin(x)$ as the input of exp. Similarly, standard algebra allows us to build complex equations like $f = g \cdot m_1 \cdot m_2 / r^2$ by composing simple mathematical operations. We could, in fact, write this in a way that the functions and compositions were more explicit: $\texttt{divide}\big(\texttt{times}\big(g, \texttt{times}\big(m_1, m_2\big)\big), \texttt{square}(r)\big)$. Here, the primitive pieces are times, divide, and square, and they form the language to express more complex hypotheses.

In the context of number learning, we might assume that learners have access to an ability to check one-to-one correspondence of sets through a function like `match` (Izard et al., 2014), which takes two sets and determines if they can be lined up one-to-one. For example:

```
match({x}, {y,z})
```

would evaluate to `false` but

```
match({x,w}, {y,z})
```

would evaluate to `true`. Such a function is meant to be a *mental* operation that users can evaluate ("run") cognitively. While `match` has some of the content required for natural numbers, note that it does not provide a counting algorithm itself—it only outputs a true or false (not a mapping from one set to one word). A child learning to count might *use* it to figure out what word to say, but doing so requires a bit more.

Learners should be able to use `match` as a component of a more complex hypothesis that includes other operations too. For example, we might use it in the definition of a function (`F`), combining it with an `if` statement and some representations of linguistic words like "one" and "many":

```
def F(s):
    if match({o}, s):
        return "one"
    else:
        return "many"
```

Here, `F` is the right kind of thing for counting: it takes a set `s` and returns a word. This particular `F` uses `match` to check if `s` can be put in one-to-one correspondence with a single element ($\{o\}$), returning "one" if it can (e.g., if the `if` evaluates to `true`) and "many" otherwise. Thus, this function `F` acts like a "one-many" kind of numerical labeling system.

The hope of PTG was that learners might be able to consider essentially *all* of the different ways that a collection of primitives could be combined into programs like this. For example, using just these same operations we could consider a program like

```
def F(s):
    if match({o}, s):
        return "many"
    else:
        return "one"
```

which labels in the opposite (incorrect) way compared to a one-many system, or

```
def F(s):
    if match(s, s):
        return "one"
    else:
        return "two"
```

which always says "one", or, if we are willing to allow for logical operations like disjunction (`or`),

```
def F(s):
    ifor(match({o},s),match({o,o,o},s)):
       return "one"
    else:
        return "many"
```

which labels sets of size one and three "one", and everything else "many."

The philosophy of this approach is that a small number of built-in operations—perhaps at most a few dozen—expresses a huge number of possible concepts for learners to consider. Such compositional systems provide a close fit to empirical learning curves (Amalric et al., 2017; Feldman, 2000; Goodman et al., 2008; Lake et al., 2015; Piantadosi et al., 2016) and have plausible brain bases (Frankland & Greene, 2020). The general idea of learning over compositional systems is now used across a huge variety of domains. It is a fascinating fact about computation that there are small collections of primitives that are able to express *any* computation when they are combined via function composition as in these examples. Such combinatorial expressiveness is exactly the same as found in language, where a small number of words are combined to express sentences, paragraphs, and entire novels; or computer programming where a handful of primitive operations can be combined to express arbitrarily complex computations. That is why these models are sometimes called "language of thought" models, following Fodor (1975). In such models, we need not posit specialized innate knowledge for tying shoelaces, fractions, or flying the space shuttle, if those abilities can be expressed computationally, and learners can work over this computationally rich space (Rule et al., 2020).

Types constrain composition

Each primitive operation like `match` has a *type*, which is just an annotation of what kinds of arguments it requires and what kind it returns. `match`, for example, takes two sets and returns a Boolean (true/false) value. We might denote this type as `Set×Set→Boolean`, where the stuff on the left-hand side of "→" says what arguments the function wants (two sets), and the stuff on the right says what value it gives back (a `Boolean`). The logical function `or`, for example, might be `Boolean × Boolean → Boolean`. Types are useful because they provide a constraint on what compositions are semantically coherent or interpretable, and implementations use them to constrain what compositions of primitives are permitted. For example, we could not form a composition like `match(or,or)` because the arguments to `match` are not sets (the `or`s are functions) and moreover those bare `or`s themselves have no arguments. Programming languages also have types in this sense, and it helps to catch bugs and ensure that the output can be computed from the input. Types also play a prominent role in theories of linguistic composition in semantics where they help specify what kinds of utterances are semantically meaningful (Heim & Kratzer, 1998; Steedman, 2000) (e.g., "the" is not a verb, so "Mia the" is not a sentence).

Priors, likelihoods, and Bayes' rule

It might seem that an expressive hypothesis space is actually bad news: if there are lots of options for what representation we could build, how could learners even arrive at the right one? Learners will always be given a finite dataset, and these data will always be consistent with infinitely many generalizations. This problem has been emphasized in philosophy as the *problem of induction*. It occurs even in the simplest settings: for example, if we knew that a function f yielded $f(1)=2$, $f(2)=4$, $f(3)=6$, it would be tempting to infer that the function was $f(x)=2\cdot x$. This is an inference or a generalization because $f(x)=2\cdot x$ goes beyond the observed data to generate $f(x)$ for *any* x value, not just those we actually saw. But, in fact, even this simple dataset is consistent with infinitely many possible computations—it could be that $f(x)=-6+13\cdot x-6\cdot x^2+x^3$, for example. There is no way around the fact that learners who see such a dataset must have some way of determining which of the infinitely many possible solutions is "best."

One response to this is to throw up your hands and say that learning is impossible. This tack is sometimes taken, either in spirit or explicitly, by claims that because the data do not uniquely determine a representation, it must be that the hypothesis space is constrained innately. But innateness does not explain the breadth of things that people are actually able to learn. At the same time, the problem of induction is often waved off by empiricist approaches which sometimes claim not to build in any biases one way or the other. But this is impossible—any initialization scheme, for example, neural networks, including random or zero initialization, will bias them toward certain types of representations.

The approach taken by PTG and other Bayesian program learning models is to just be explicit about the built-in biases. Bayesian biases are present, but often *weak*, meaning that they can be overcome with sufficient data. For example, learners might prefer $f(x)=2\cdot x$ to another alternative if they had a bias for simplicity (using fewer operations). With this bias, both can be present in the hypothesis space without ruining the possibility of learning. Weak, overcomeable, biases yield the best of both worlds: we can solve the fundamental problem of induction while remaining flexible enough to acquire complex representations when needed.

In Bayesian program learning models, the weak biases take on two forms. Typically, we assume that learners prefer hypotheses which can be expressed with a *small* number of primitives. Hypotheses with more primitives are possible, they just take considerable data to justify. This prior means that there is a second way in which these Bayesian program models are biased: the set of primitives we choose will impact the representations the model arrives at because the set of primitives determines what is short or easy to express. This means that if two modelers pick different sets of primitives, they will measure simplicity in different ways, and will consequently arrive at different representations and generalizations. This fact is a strength of the approach—not a weakness—because it means that we can empirically discriminate proposed sets of primitives by testing which ones lead to more human-like generalizations. That kind of comparison was undertaken in the context of rule learning, for example, by Piantadosi et al. (2016), who showed that the primitives that best fit human generalizations included quantification (see also Kemp, 2009) as well as non-minimal sets of Boolean operations.

The preference for representations with fewer operations is typically formalized with a function $P(h)$ that assigns a "prior probability" to each hypothesis (composition of primitives) h. This prior, for example, might be exponential in the number of primitives in h (e.g., $P(h) = \exp(-l(h))$ where $l(h)$ is the number of primitives), or it might use a probabilistic context-free grammar which penalizes each use of each primitive via a generative probabilistic model (Goodman et al., 2008), which is also useful for ensuring that compositions follow the allowed types. In either case, the key point is that $P(h)$ provides a quantification of how preferred any hypothesis is. We can prefer simple hypotheses, and preferentially choose them as representations, while still admitting complex hypotheses as *possible* (i.e., $P(h) > 0$).

The second component of a Bayesian model is a likelihood function, written $P(d|h)$, which says for any hypothesis h, how likely are the data d. In most versions of Bayesian program models, the data are typically (noisy) pairs of inputs and outputs. For instance, in the $f(x)$ example above, the data might be the pairs $\{(1,2),(2,4),(3,6)\}$. Formally specifying the data for counting requires us to think about what the right input and outputs are for the child's target representation. If the child is learning counting, it is reasonable to think that the input of the counting algorithm is a Set and the output is a Word: thus, learning counting is learning to do something to a set that yields the correct word (i.e., one representing its cardinality). Learners would not be told explicitly about cardinality, but only given pairs of sets and words and asked what function relates to them. Notably, following PTG, children may not correctly identify the appropriate set when there are multiple items around—they might hear "two trees" in a forest (as in, "…those two trees are falling over…"). We, therefore, assume that the pairing of Sets and Words is noisy. A sample dataset might be, $d = \{(\{x,x,x\}, \text{"three"}), (\{x,x\}, \text{"one"}), (\{x\}, \text{"one"})\}$.

Bayes' rule is perhaps the least interesting part of these models. The rule just says that if we want to score how "good" a hypothesis h is we should *multiply* the prior $P(h)$ with the likelihood $P(d|h)$. This may be counterintuitive because these are different kinds of things—the prior is the probability of a hypothesis, and the likelihood is the probability of a dataset—but there are good theoretical reasons to follow Bayes rule (e.g., McNamara et al., 2006; Stamps & Frankenhuis, 2016). While there are many logically possible ways of combining two such terms, Bayes rule is the only way to update which yields a coherent interpretation in terms of *beliefs*. If $P(h)$ is our belief that h is the correct one before we see data, the *posterior* $P(h|d) \propto P(h) \cdot P(d|h)$ is the belief that h is the correct one *after* we see d. Intuitively, multiplying the prior with the likelihood has the result that we only strongly believe h is correct if it has high prior *and* likelihood, relative to other hypotheses. Multiplication forces us to incorporate both of these parts into our new belief, meaning that we should prefer hypotheses which, as much as possible, are simple (high $P(h)$) and which do a good job of explaining the data (high $P(d|h)$).

Learning in PTG

PTG examined the case of number learning with a simple set of basic primitive operations shown in Table 1. This spans several simple classes of operations: representing

TABLE 1 Table of primitive operations used in PTG. The counting words ("one", "two", "three", etc.) and the argument to the function are also allowed to be used in hypotheses.

Primitive	Types	Gloss
`singleton`	`Set→Boolean`	Check if a set has one element
`doubleton`	`Set→Boolean`	Check if a set has two elements
`tripleton`	`Set→Boolean`	Check if a set has three elements
`select`	`Set→Set`	Choose an element from a set
`union`	`Set×Set→Set`	Union of two sets
`intersection`	`Set×Set→Set`	Intersection of two sets
`difference`	`Set×Set→Set`	Set-difference
`complement`	`Set→Set`	Complement of a set
`and`	`Boolean×Boolean→Boolean`	Logical conjunction
`or`	`Boolean×Boolean→Boolean`	Logical disjunction
`not`	`Boolean→Boolean`	Logical negation
`if`	`Boolean×Set×Set→Set`	Conditional
`if`	`Boolean×Word×Word→Word`	Conditional
`recurse`	`Set→Word`	Evaluation of current hypothesis
`next`	`Word→Word`	Return the next word in the count list
`prev`	`Word→Word`	Return the preceding word in the count list

and manipulating sets (Carey, 2009), Boolean logic (e.g., *if*), producing the verbal counting routine (Fuson, 1988), and recursive calls (Corballis, 2014; Ferrigno et al., 2020; Hauser et al., 2002). At present, any set of primitives (here or for the next model) are primarily a conjecture about what kinds of operations are plausible for children to possess before knowing number, and could potentially support constructing the kinds of knowledge that they arrive at. It is important to note that the choice of primitives here is a real *empirical* claim in the sense that this model predicts that children should be able to compute these operations antecedently to knowing how counting works. Other work has examined empirical comparisons between logically equivalent sets of operations (Piantadosi et al., 2016), but that work has yet to be done for these kinds of operations.

As described, the hypothesis space consisted of all the possible ways of combining these operations into functions that mapped sets to words. The basic finding of PTG was that even out of a relatively unrestricted collection of options, learners could narrow in on the right hypothesis, and moreover, in doing so they went through developmentally attested "knower-level" stages (Sarnecka & Lee, 2009; Wynn, 1992). A "two-knower" child, for example, is one who is correct on the first two numbers but not higher ones. Learners might construct a "two-knower" hypothesis as:

```
def F(s):
    if singleton(s):
        return "one"
    else:
        if doubleton(s):
            return "two"
         else:
            return "many"
```

One-knowers and three-knowers are constructed similarly from `if` statements. But the correct system counted recursively: it removes one element from the set of `s` of things to be counted, and for each removed element, it goes forward one position in the counting routine using `next`, and uses recursion (calling itself via `recurse`):

```
def F(s):
    if singleton(s):
        return "one"
    else:
         return next(recurse(difference
(s,select(s))))
```

Note that this program itself is consistent with several evaluation strategies to actually "run" it. For example, children are more proficient at computing `next` starting from "one" (rather than in the middle of the list) (Fuson, 1988), and they could do so with this hypothesis by greedily evaluating `next` every time they recurse. Thus, while we can view this as a procedure, we can also view it as a formalization of knowledge or theory, which is subject to different kinds of evaluation, depending on the context. The same thing happens in your computer: when you write a `for` loop, for example, sometimes it describes the actual steps the computer takes, but sometimes—particularly in compiled languages—the computer can recognize a more efficient, but mathematically equivalent, way to evaluate it that may involve no loops at all. Thus, these programs are best thought of as formalizations of knowledge on an abstract level.

Second, note also that this hypothesis correctly generates the word for both small and large sets, and uses completely different operations than the two-knower hypothesis: this shift reflects the required conceptual change, or insight.

PTG showed that as learners accumulate data, they shift hypotheses between knower-levels, before eventually arriving at a counting process like that shown above. The shift intuitively results from the joint forces of the prior and likelihood: counting algorithms that were very simple would only get the first few numbers right, and more data were required to justify moving to complex ones, assuming that recursive processes (like the correct counting system) were especially complex. PTG showed that the cost of recursion—intuitively a difficult operation—had a strong influence on when the model predicted children would learn a full counting procedure.

This kind of program learning approach addresses Rips et al.'s challenge of saying how learners acquire the right system by demonstrating a concrete, implemented model in which the correct system followed from Bayesian inference, namely, trying to find a system that explains the observed data while being as simple as possible. While PTG showed that the model could learn other structures (like clock-like systems, or singular/plural systems), these hypotheses were dispreferred because they took more primitives to express. This approach nicely sits at a Bayesian middle ground in the debate between nativism and empiricism: many representational systems were possible, but some, like circular systems, are more strongly dispreferred due to complexity. Rips et al. are right that there is a sense in which learners must somehow prefer the right system, but there is also an important sense in which this preference is soft since the model can acquire other structures when necessary (see Piantadosi et al., 2012).

AN UPDATED PTG

The discussion in the literature about PTG raised a number of important issues. One was whether PTG examined learning in too simplistic of a setting. In particular, the model had "built in" small set representations but not approximate number operations. A basic prediction of program learning is that enriching the possible hypothesis space should not impede the model's success because the whole philosophy was to construct a learner that did not need to be restricted to start with. The question of whether the same model succeeds in a more complex setting is easily tested with an implementation. Here, we consider three primary ways of making the learning problem harder and more realistic, motivated by prior discussion: allowing objects of multiple kinds, requiring learners to construct representations for small sets, and including approximate hypotheses.

Object kinds

First, we consider a setting in which learners are in multi-object environments and must learn that number combines with a specific object *kind*. The data the learner receives are, therefore, a number word with a head noun (e.g., "two accordions") in the context of a variety of objects (e.g., two accordions, five music stands, two stools). This brings the learning setup one step closer to naturalistic types of data children observe including multiple kinds of information, some of which is not relevant.

In turn, the learning model faces a more difficult challenge of having to encode in the program how to select the appropriate subset of nouns from their current scene. Learners in this setup might consider then that "two" means "two accordions", or even "two apples" or "two non-accordions."

Working memory model with one-to-one matching

Second, instead of innate functions for small cardinalities like `singleton` in PTG, the new version of the model uses one-to-one matching with a system that represents objects with a capacity limit (e.g., up to three or four), closer to the proposal of Carey (2009). For example, the model would write a function that checks if a set `x` has a single element as `match({o}, x)`. Note that in the representations below, this matching function only needs to apply to small sets (there is evidence against matching for large sets without the number system, at least when there are no clear visual cues available, Pitt et al., 2022). The resulting model is equivalent to PTG's model under certain priors for sets and matching. Thus, the question addressed by using this matching operation is whether making these subset-knower levels more compositional (e.g., built out of `match` and sets) introduces a level of complexity that renders the learning problem impossibly hard. If learning can succeed in this setup, it is more clear that the basic primitives in this model really do not have numerical content—they are just operations on sets.

Approximation primitives

Third, responding to points from Carey (2014), the model includes approximate hypotheses in its representational space. This means that it could potentially learn that "four" meant "approximately four."

Despite interest in connections between approximation and counting (e.g., Cantlon, 2012; Dehaene, 2011; Halberda et al., 2008; Shusterman et al., 2016; van Marle et al., 2018; Whalen et al., 1999), prior theories have not been precise about how approximation may be used by learners trying to acquire the exact counting system. Here, we make a simple assumption that there are two approximate primitives, one to approximately check equality between outcomes and one to approximately compare set sizes. Approximate number estimation is often formalized as following a psychophysical law where a number n is represented as a Gaussian variable with standard deviation $w \cdot n$ (Dehaene, 2011). Thus, larger numbers are represented with a standard deviation that increases *linearly* with numerosity. The parameter w is known as a *Weber fraction* (Halberda & Odic, 2015) and here we assume $w = 0.2$. This value is toward the higher end of performance for children (Odic et al., 2013), but higher precision makes the approximate operations closer to the exact ones, and therefore provides a stronger test of whether learners could rule out approximate systems. The first operation we assume computes the probability that, under a standard Weber model, the difference between two terms is approximately zero (e.g., between $-\frac{1}{2}$ and $\frac{1}{2}$), while the latter uses the same model to compute the probability that one quantity represented with Weber noise is greater than another. Notably, because the comparison returns probabilities, these functions are used in conditionals, much like the matching function, which will stochastically return each outcome.

Example hypotheses

A nice feature of this modeling approach is that to test a model with these kinds of operations, we can simply add them as primitives and follow the same logic as PTG. Table 2 shows a summary of the primitives included in the new model. These changes are noteworthy in that each should make the learning problem substantially harder: instead of learners who only consider functions on abstracted sets, the learning model now must sort out that number is abstractly a property of objects (e.g., not specific nouns), that number is exact, and learn how to compose mental representations of small sets with matching in order to express small number concepts.

To illustrate these primitives, a two-knower might look like,

TABLE 2 Table of primitive operations used in PTG. The counting words (“one”, “two”, “three”, etc.), memorized small sets, approximate magnitudes, object types, and the argument to the function are also allowed to be used in hypotheses.

Primitive	Types	Gloss
`match`	`Set×Set→Boolean`	Check if two sets can be put in 1-1 correspondence
`select`	`Set→Set`	Choose an element from a set
`selectO`	`Set×Kind→Set`	Choose an element from a set of a specific kind
`filter`	`Set×Kind→Set`	Select only objects of a given kind from a set
`ANS-equal`	`Set×Set→Boolean`	Are two sets approximately equal
`ANS-less`	`Set×Set→Boolean`	Is the first set approximately smaller than the second
`flip`	`∅ →Boolean`	Random coin flip
`union`	`Set×Set→Set`	Union of two sets
`intersection`	`Set×Set→Set`	Intersection of two sets
`difference`	`Set×Set→Set`	Set-difference
`complement`	`Set→Set`	Complement of a set
`and`	`Boolean×Boolean→Boolean`	Logical conjunction
`or`	`Boolean×Boolean→Boolean`	Logical disjunction
`not`	`Boolean→Boolean`	Logical negation
`if`	`Boolean×T×T→T`	Conditional (one for each type T)
`recurse`	`Set→Word`	Evaluation of current hypothesis
`next`	`Word→Word`	Return the next word in the count list
`prev`	`Word→Word`	Return the preceding word in the count list
`{}`	`Set`	The empty set
$\{o\}$	`Set`	A set with one object
$\{o, o\}$	`Set`	A set with two objects
$\{o, o, o\}$	`Set`	A set with three objects

```
def F(s,k):
    if match({o}, filter(s,k)):
        return "one"
    else if match({o,o}, filter(s,k)):
        return "two"
    else:
        return undefined
```

where “one” and “two” are checked by evaluating one-to-one correspondence (`match`) with memorized sets (e.g., `{o,o}`), after filtering (`filter`) the observed set of objects *s* in the environment for the appropriate type *t*. This filtering operation allows the model to take a set and a kind (e.g., “two accordions” with a set *s* of things in the environment and *t* = accordions) and only pay attention to the accordions in the current context *s*.

To compare, another possible hypothesis for this model is one where “one” could refer to any kind of object, and “two” refers only to accordions:

```
def F(s,k):
    if match({o}, s):
        return "one"
    else if match({o,o}, filter(s,ACCORDION)):
        return "two"
    else:
        return undefined
```

An approximate two-knower might look identical except for using an approximate equality. We will write approximate primitives with “ANS” for *approximate number system*. For example,

```
def F(s,k):
    if    ANS-approximately-equal({o},
filter(s,k)):
        return "one"
    else if ANS-approximately-equal({o,o},
filter(s,k)):
        return "two"
    else:
        return undefined
```

where `ANS-approximately-equal` returns `true` with probabilities determined by approximation psychophysics described above. This particular hypothesis is approximate for “one” and “two,” but because the model considers free composition of primitives, note that learners could consider one or the other to be exact, or even more complex structures and compositions of exact and approximate operations.

When the model learns to count, it constructs a recursive algorithm very much like PTGs but involving kinds:

```
def F(s,k):
    if match({o}, filter(s,k)):
        return "one"
    else:
        return next(F(difference(s,select(s,k)),k))
```

This works by checking if there is one object of the appropriate kind in the set *s* (computed via `match({o},filter(s,k))`). If so, it says "one"; otherwise, it returns the word (next) after removing an element of kind *k* (`difference(s,select(s,k))`) and recursing (*F*). Note the free composition of primitives allows many hypotheses to be formed—for example, a learner who only calls everything approximately greater than two "three":

```
def F(s,k):
    if ANS-approximately-greater(2, filter(s,k)):
        return "three"
    else:
        return undefined
```

All approximate functions are stochastic, meaning they return a distribution over outcomes, which is handled in the likelihood of the model.

Model implementation

This model is implemented in Fleet (Yang & Piantadosi, 2022) which is a free and open source library for creating "language of thought" models. Code for this implementation can be found at https://github.com/piantado/Fleet. In Fleet, users specify the assumed primitive operations. From these primitive functions, Fleet automatically implements a Markov-Chain Monte Carlo (MCMC) sampler (Hastings, 1970) over tree expressions of compositions of these functions. This uses a technique proposed in concept learning by Goodman et al. (2008). Briefly, MCMC is a numerical technique that takes a random walk around the space of hypotheses, biased to find hypotheses which score well on the product $P(h) \cdot P(d|h)$. The algorithm works by starting from some hypothesis and proposing a change to it. Usually, this change is replacing part of its program with something random from the grammar. For example, at one time step we might represent

```
def F(s,k):
    if match({o}, filter(s,k)):
        return "one"
    else:
        return next(F(select(s),k))
```

and at the next we might propose

```
def F(s,k):
    if match({o}, filter(s,k)):
        return "one"
    else:
        return next(F(complement(select(s)),k))
```

replacing the `select` in the recursive call on the last line with a different composition of primitives. This proposal is entirely random (though constrained to satisfy the types) and therefore it requires no insight or reasoning about the hypothesis. If the change improved the score $P(h) \cdot P(d|h)$, it would be accepted, and the new hypothesis would be adopted as the starting point for future proposals. If the proposal made the score worse, it would be accepted with a probability proportional to the relative scores, following a standard MCMC algorithm.

Note that in general, most changes will be unlikely to be accepted because they often will make hypotheses much worse. For example, we might propose changing something nonsensical,

```
def F(s,k):
    if match({o}, {o,o}):
        return "one"
    else:
        return next(F(select(s),k))
```

Just as a random edit to a program, a novel, or a genome probably will not improve it, a random change to a program probably will not help. But just as in evolution, sometimes a change *will* work, and because we tend not to accept worse proposals, these changes accumulate to yield effective hypotheses.

This should give the intuition that as the algorithm works, it is stochastically optimizing subtrees of the hypothesis, endlessly proposing changes to each part of a hypothesis to try to find something that works better. However, MCMC will sometimes accept changes to hypotheses that are worse, and this is desirable to stop it from getting stuck in a local maximum. In fact, the above algorithm is a *sampling* algorithm (see MacKay, 2003), meaning that if it is run for long enough, the (relative) amount of time it spends on each hypothesis will approximate the posterior probability, which is proportional to $P(h) \cdot P(d|h)$. This means not only that MCMC searches the hypothesis space for us, but it also does proper statistical inference of computing how strongly we should believe that any hypothesis is the correct one. Importantly, MCMC does this while only representing a *single* hypothesis and proposal at any one time: we can learn over an infinite hypothesis space without needing to actively represent all of the hypotheses.

The Fleet implementation is extremely fast, drawing around 200*k* samples per second for this model. We emphasize, however, that MCMC is not intended to necessarily be how learners solve the search problem, although it may be a compelling possibility (Ullman et al., 2012). Rather, our model is meant to provide a computational level analysis (Marr, 1982) which maintains that people pick a hypothesis according to the same criteria that matter for the model, namely, approximately trading off between complexity $P(h)$ and fit to data $P(d \mid H)$ by looking for hypotheses which score well on the product of these terms. This might occur, for instance, by making smarter proposals and reasoning about the content of the representations themselves, processes that developmental theorists have not yet formalized.

RESULTS

Figure 1 shows learning curves in the model, with the *x*-axis showing an amount of data and the *y*-axis showing the model's belief (posterior probability) given to different kinds of hypotheses. These plots are made by simulating a given amount of data from a correct target hypothesis, and then running the model's search process to find hypotheses which score highly (in terms of $P(h) \cdot P(d \mid h)$). The hypotheses that are found are then grouped into "knower-level" stages (Wynn, 1992) by running them on data and seeing what responses they provide. For example, the "two-knower" line shows all hypotheses where the model responds correctly to "one" and "two" but not "three" and higher. The data for this figure are generated by sampling sets of objects, with object kinds, from a distribution that roughly matches the empirical distribution of numbers, and noisily pairing them with number words.

In Figure 1, the *y*-axis values are posterior probabilities, given how strongly a learner would believe that any particular kind of hypothesis is correct, given the noisy training data. For example, at 250 data points, the model strongly believes that two-knowers are the "best" hypothesis, but the three-knowers are starting to rise. In this figure, approximate hypotheses are grouped by the most likely outcome, averaging across input sets. The line type in Figure 1 shows whether or not the model incorporates approximate primitives.

The figure makes it clear that even with the changes outlined above, the model is able to explain the developmental trajectory of transitioning between subset-knower stages before eventually arriving at a fully correct counting algorithm, just as in PTG. This kind of learning occurs in the present model in spite of the increased difficulty of sorting out possible approximate meanings, kind of objects, and the noun kind. Interestingly, theories using approximate numbers are not high probability at any stage. This is because, for any such theory, there is a nearby exact theory which does a better job of explaining the data (since the data are exact); the model shows, however, that it does not take much data to sort out these alternatives. This model, therefore, provides an implemented learning theory that aligns with the data analyses of Lee & Sarnecka (Lee & Sarnecka, 2010, 2011; Sarnecka & Lee, 2009) showing that the ANS does not support early word meanings.

The next section discusses several points related to PTG and the present model which have been made in the prior literature. These points are informative not just in the case of number, but because they highlight several typical kinds of critiques and responses to compositional models.

Examination of examples of common critiques

One common question is about the size of the hypothesis space, which is typically infinite since it corresponds to all ways of combining primitives. Carey (2014) argues that the hypothesis space for PTG's model is too large, covering thousands of possible algorithms; the current space is even larger. Both PTG and the present model were meant to address the simpler challenge of showing how idealized learners might solve Rips et al. (2006)'s inductive puzzle about how children could discover the right structure of numbers. Showing that the right

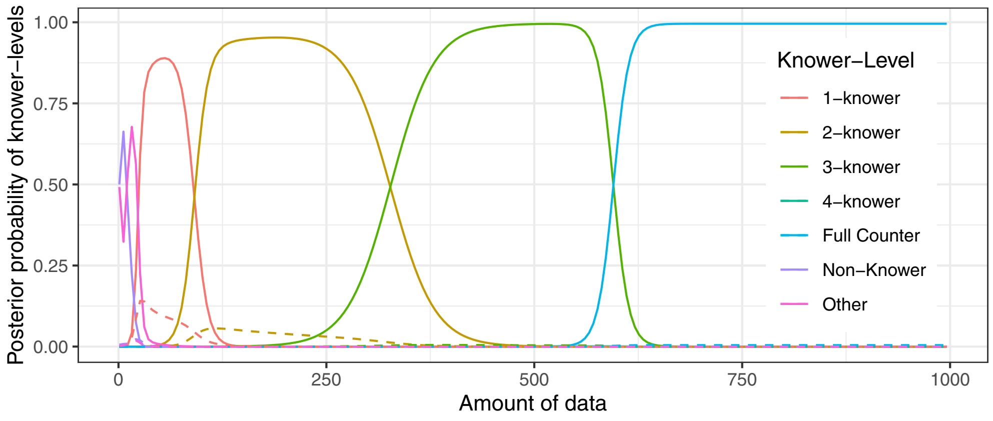


FIGURE 1 Learning curves for the new PTG model, showing knower-levels as well as whether the program uses any approximate primitives (dashed) or not (solid).

hypotheses are selected out of the infinitely many that are possible is *the necessary step* in constructing a computational-level answer to Rips et al.'s challenge.

In my own view, Carey's critique arises from a misunderstanding that PTG were proposing an algorithmic account, where learners must actively *represent* all of these hypotheses. But because the hypothesis space is generative—it lets you propose a new hypothesis at random by putting together primitives—MCMC can search the space without actively representing all hypotheses. For both PTG's and the present model, the learner only needs to represent a single hypothesis and alternative at a given point in time, and in fact, the sampling algorithm used in Fleet does just that. This is the surprising and powerful nature of Markov-Chain Monte-Carlo, that it is able to sample a hypothesis space without needing to enumerate or represent all of the possible options. This algorithm is the basis for much of modern statistical inference and has been suggested as a bona fide developmental process (Bonawitz, Denison, Gopnik, et al., 2014; Bonawitz, Denison, Griffiths, et al., 2014; Ullman et al., 2012).

Moreover, it is simply a fact that children must be able to navigate large spaces of hypotheses—we know this because there are many structures and algorithms children can acquire (Rule et al., 2020). So, the critique that there are too many possibilities for this model is a bit like claiming that no human could be the author of *In The Night Kitchen* because there are too many options for what an author *could* have written on the blank page. How could Maurice Sendak have selected the right text out of the essentially infinite space of possibilities if there are more than ten thousand choices for him to navigate? The answer is the same for Sendak as our model: the space of choices may be large, but the space of options which any learner—or writer—needs to consider at any moment in time can be quite small.

One way to visualize what a learner needs to do is shown in Figure 2, which plots a single point for each hypothesis computed in one run of the model. The x-location of each point is given by its log prior ($\log P(h)$); the y-location is given by its log likelihood ($\log P(d \mid h)$) on a number dataset. Learners in these models trade off complexity and fit to data, and so will naturally be pulled to hypotheses close to the origin (lower left). How much they care about being vertically far away from the origin as opposed to horizontally far away from the origin will depend on the amount of data they have seen: as amount of data increases, they will move along the set of possible hypotheses from the upper left to the lower right—thus, with lots of data, they will not care about being far away from the origin on the x-axis, but will care about being far away on the y-axis. The left curved edge is an optimal Pareto frontier, meaning learners along this frontier cannot improve their fit to data without increasing complexity, nor reduce complexity without sacrificing fit to the data.

An efficient learner could "walk" along this frontier from top left to bottom right as the amount of data increases, considering only hypotheses that are simple and explain the data. This occurs, at least approximately, because the frontier will contain the highest probability hypotheses for a fixed amount of data, and the inference algorithm is a sampler which means that it will tend to generate the high probability hypotheses. Less efficient learners—maybe those who make random proposals—might need to consider more hypotheses over time. Thus, the number of hypotheses a learner must traverse depends on how efficiently they can propose hypotheses. If they are efficient at proposing new hypotheses, they might hop along $10-25$ different hypotheses along this frontier. If they are inefficient, they may take a random walk along the frontier and consider more. But in either case, *most* of the hypotheses in this space need not be actively represented at all because they are not on the frontier, meaning they are either too complex or poorly fit the data. One way to think about this is that the data are exponentially informative about the underlying hypothesis. For example, we might expect that on average

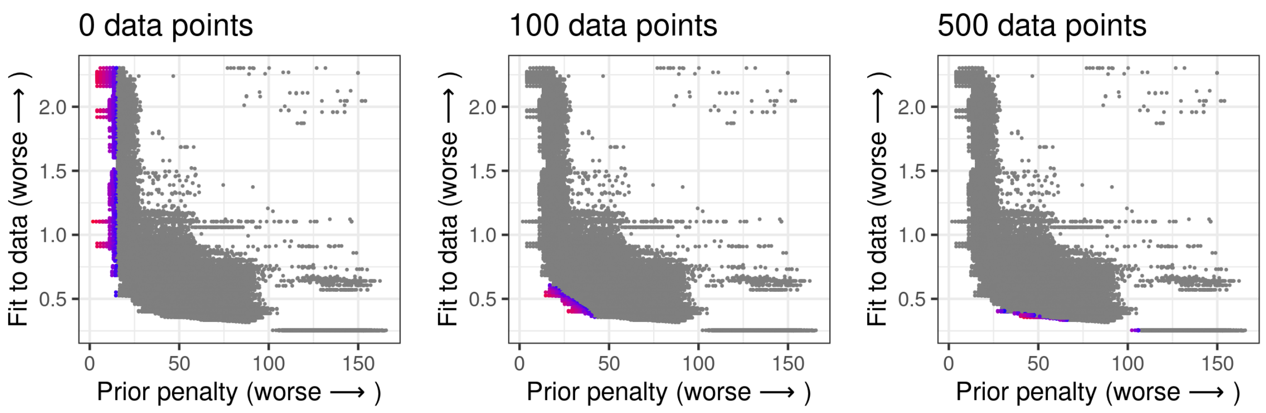


FIGURE 2 Visualization of the learner's hypothesis space. Each point is a hypothesis (program), plotted at how well it fits the data versus how complex it is. As learners acquire data, they shift beliefs in hypotheses. Most hypotheses in the space have low probability: red has highest probability, blue less probability, and gray hypotheses have probability less than about 1 in 10 million. Efficient learners might only consider a handful of hypotheses (e.g., red points) along the high probability edge.

only $1/10^N$ hypotheses get the first N number words right, meaning that a few number words provide a drastic reduction on the space of hypotheses learners would seriously or actively consider.

A second common form of criticism concerns the relationship between these models and prior theories. Often developmental theories are *informal*, meaning that they are stated in language rather than mathematics or with computational implementations. These informal theories are absolutely critical to computational modeling because they tell us *what* to implement computationally. However, the informality of most theories often leaves ambiguity about how mechanically the pieces are put together. This is one of the most important reasons to implement theories: doing so functions as a kind of consistency check, showing where you make unstated assumptions or where the informal theory is imprecise, or sometimes even inconsistent. Often implementations allow us to derive quantitative predictions, some of which surprise us: One example is that we first implemented a closely related model of quantifier learning—expressing word meanings like "most" as compositions of set-based primitives—by assuming that learners pick meanings which are most often true. It sounded intuitively, or informally, like a plausible approach. However, this model learned trivial meanings like "most" is always true. Implementation showed that assumption was not a good one. Instead, children would need to think about a proper *likelihood* specifying how likely each word would be in each context (see Piantadosi, 2011).

PTG's model has also been critiqued largely with a focus on whether or not it implements Carey (2009)'s *bootstrapping* theory (Beck, 2017; Carey, 2014; Rey, 2014; Rips et al., 2013). This debate is largely terminological, but whether PTG qualifies as "bootstrapping" in Carey's sense may be orthogonal to the question of whether it is the right general approach. Carey (2014) also questioned some of the underlying formal assumptions, for example, saying that PTG "merely assume—without evidence—that full general resources of lambda calculus and logic are available for the generation of hypotheses" (p. 151) (While it is true that PTG described the model with lambda calculus, the "full" capability was not used. In particular, PTG's lambda calculus was strongly typed, a fundamental limitation on computational power). Lambda calculus is a logical notation for specifying how functions compose and ensuring that composition is done correctly (Hindley & Seldin, 2008; Pierce, 2002). Because compositionality is often thought to be central to human-like cognition (Fodor, 1975; Fodor & Pylyshyn, 1988), lambda calculus is often considered a fundamental resource in other domains, for example, natural language semantics where it is used to derive the meaning of sentences from the component words (Heim & Kratzer, 1998; Steedman, 2000). This use in other settings makes it a plausibly available resource for number. But in either case, it is important to recognize that lambda calculus has very little content—it specifies nothing about, for example, objects, sets, or Boolean operations. It only ensures that whatever operations are present can be combined and evaluated in a coherent way. The use of this formalism actually motivated direct empirical work on compositionality, demonstrating compositional abilities in preschoolers (Piantadosi & Aslin, 2016) though perhaps not infants (Piantadosi et al., 2018).

Specifying a formalism for compositionality in such a model is a strength, not a weakness. To compare, Carey's theory drew on structural analogy, and many formal systems for implementing analogy are themselves compositional. In fact, if Carey's account was fully formalized, it would likely depend on composing *some* mental operations. Part of the problem is that Carey's account has ambiguities about some details (Fodor, 2010), although some explications have been written subsequently (e.g., Beck, 2017). Specifically, Rips et al.'s question of how children build *the right* analogy for number is closely related to this question of what resources the analogy may be built from, if not compositionality.

Another advantage of formalized models is that they allow for quantitative predictions, and can prevent us from making incorrect informal predictions. O'Rear and McNeil (2019) studied a family of experimental interventions in which children received extra information about number. They reasoned that if PTG's account based on composing primitives were correct, that extra labeling should *slow down* acquisition because labeling would prevent the system from needing to construct a new representation ("The greater the number of set-size primitives, the more input necessary to become a CP-knower … Thus, children with two set size primitives will reach the capacity of their primitive system with less input than children with four set size primitives." (p. 3)). In contrast, analogical accounts would predict *faster* learning with labeling small sets because labeling would highlight the relationships between early words. Their prediction for PTG essentially assumes that set labeling would train additional primitives (e.g., beyond `singleton`, `doubleton`, and `tripleton`) or reinforce use of the existing ones (see also Paliwal & Baroody, 2020).

There are two aspects to this prediction. First, it assumes that giving children the ability to recognize more small sets through training should slow down PTG or any program learning model. However, in PTG the primary determinant of when children become CP-knowers is the parameter penalizing recursion (e.g., PTG, section 4.1, figure 4). This was motivated by the general difficulty of recursion, in the sense that a recursive function call requires storing additional information (about the current function call) and that should be penalized because it uses extra resources. Similarly, in the present model, Figure 1 shows that children never become four-knowers, despite this being an option with the current working memory operations. If we had included working memory models up to five or six, they *would not yield different learning* because the model becomes a CP-knower before reaching these levels. Looking at the knower-level hypotheses above, a five-knower and a six-knower are both more complex than a three- or four-knower because they include the programs for these others already. Thus, even if there were memory resources to support five- or six-knower hypotheses, these would still be more complex than three- or four-knowers, and thus not learned before the CP-knower hypothesis.

Interestingly, the program learning model can explain the improvements O'Rear and McNeil (2019) found. Their intervention included set labels for sizes 1 – 6, and such additional data on numbers higher than 3 actually *does* speed acquisition of counting according to the model. Figure 3 (bottom left) shows the model with their training intervention which, for simplicity, counts each of their training data points as one additional model data point. This shows that their intervention speeds learning of the CP-transition relative to the control. Intuitively, this happens because any data not explained by a subset-knower hypothesis—for example, anything over four—will encourage the model to construct counting. O'Rear and McNeil (2019) have argued that additional data below four would slow learning because it would train additional set-size primitives. Under the PTG model, this argument is not valid: numbers below four are equally well explained by both subset- and full-counter hypotheses and so additional data points on them should not be expected to differentiate these hypotheses. It is worth emphasizing here again that modeling code is freely available. Because the model is implemented, it is unambiguous, and researchers can verify what the model predicts about specific datasets or interventions.

There is one prediction which is closely related to O'Rear and McNeil (2019)'s that would differentiate algorithm-learning accounts from analogical ones. In analogical theories, additional training on numbers 1 – 3 would predict faster acquisition because it is on these words that children recognize the analogical relationship between words and set sizes. As just discussed, for PTG or the present model, more data on 1 – 3 would not speed acquisition of counting because that data are well explained by subset-knower hypotheses. Data on 5 – 7 would speed acquisition because it would be critically *un*explained by subset-knowers. This is illustrated in Figure 3 (right top, right bottom), which varies the dataset to include additional data on low versus high numbers, and shows speeded acquisition when more data about higher number words is provided. Interestingly, this version of the model also will then initially learn some approximate meanings, presumably because they can approximately capture some of the higher words. But generally, by comparing groups that receive extra training on low versus high number words, one could empirically distinguish program induction from analogy. Indeed, Gunderson and Levine (2011) report that contrary to the predictions of the analogical theory, the frequency of high number words in children's input is more predictive of their learning than the frequency of low number words. More recently, Gibson et al. (2020) found the opposite in an intervention experiment which provided additional training on either low (1 – 3) or high (4 – 6) number words: additional small number words improved learning more. However, this effect (their Figure 3) was primarily driven by one- and two-knowers, for whom the low sets would still contain data outside their known range. Three- and four-knowers showed roughly equal improvement between the two conditions.

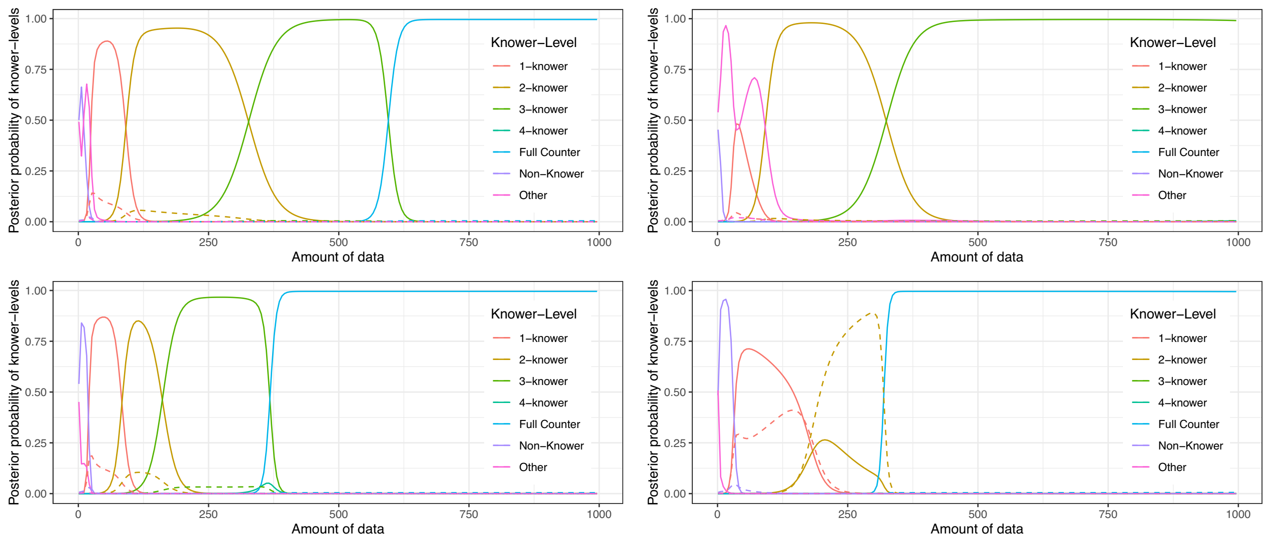


FIGURE 3 Variants of the learning model comparing the original form (top left) to O'Rear et al.'s intervention (bottom left). The right column shows extra training data on low sets (top) compared to high sets (bottom).

Similar interventions that manipulate type of training as a function of children's knowledge state have the potential to not only compare existing theories but point toward better algorithmic accounts of how children revise and improve their hypotheses.

Formal models are also often evaluated based on informal criteria—for instance, testing general assumptions or properties of their representations. This is often tricky because our informal terminology is often not perfectly calibrated to how the model actually works. Davidson et al. (2012) present empirical data showing that even children who have just learned counting fail to understand some of the semantics of number words. Specifically, such children often do not know which word is greater, or the successor relationship that words differ by quantity one—facts that one would have thought formed the basis of understanding the meaning of numbers. Indeed, related lack of knowledge about number principles beyond counting has been found in both industrialized (Izard et al., 2014) and non-industrialized communities (Jara-Ettinger et al., 2016). Though their empirical data compellingly characterize what children know and do not know, Davidson et al. (2012) err in one aspect of what PTG says. They say that PTG "explicitly equates children's implementation of the successor principle with becoming a CP-knower" (p. 171) and argue that their data argue against the model because children do not know the semantics of number. However, in both PTG and the present model, *all* that the models learn is an algorithm for counting, or a procedure that maps sets to words. It is true that this algorithm happens to obey a successor function relationship between words and sets, but also learners who discover this algorithm should not be expected to explicitly know anything beyond the algorithm. Any additional questions or principles—in essence, descriptions of how the algorithm works, or how it relates to semantics—must be learned separately. Thus, Davidson et al. (2012)'s empirical results are not only consistent with the model, they are directly predicted by it. An interesting and important direction for future work will be in understanding how children come to know not just the kinds of algorithms studied here, but the general semantic principles that characterize those algorithms.

CHALLENGES FOR ALGORITHM LEARNING AND FUTURE EXPERIMENTAL WORK

The program learning approach to mathematics is not free from challenges. Perhaps the most substantive claim from program learning models is that children have access to a collection of operations that are more primitive than number. This claim could motivate a body of work attempting to characterize what operations children know, as well as what capacities they have combining these operations via composition. The claim that number might be compositions could be surprising from the point of view of ordinary programming languages, where number is elementary enough that it is basically always built-in. But, in fact, there are plenty of logical systems where number is built from more basic elements, such as von Neumann's definition of integers in terms of sets $\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset, \}\}, \ldots$. Piantadosi (2021) describes one such system that is capable of learning many structures needed for cognition, including counting, logical operations, and quantifiers, from a primitive basis, where the operations are only defined in terms of their dynamics. The primitives in any theory present an important psychological claim, since the assumed operations determine how learners measure simplicity. Some work in logical domains has, in fact, attempted to reverse-engineer the best measure of simplicity from empirical data (Piantadosi et al., 2016; Planton et al., 2021). However, discovering the most empirically justified foundation for number or any other domain remains an important open question that could be tackled with large-scale learning experiments in children.

A second primary challenge is to discover more efficient learning algorithms which can navigate structured, algorithmic spaces. One hope is that different forms of representation may enable more effective inference—for instance, recent "neural Turing machines" embed program representations in neural network weights, allowing algorithms exactly like those required for mathematics to be learned using gradient methods (Graves et al., 2014). Such work highlights that it is likely premature to conclude much about what kind of search over hypotheses is plausible or implausible for even 4-year-olds, since sometimes a search over an infinite space of algorithms might be little more than gradient ascent. Recent advances have also combined symbolic program learning approaches with neural networks (Ellis et al., 2020). This raises a key empirical challenge for working with children, which is that of investigating ways in which they revise and refine their algorithmic representations. Seminal prior work has investigated this question of algorithm revision in arithmetic learning (e.g., Jones & Van Lehn, 1994; Shrager & Siegler, 1998; Siegler & Shrager, 1984; Svenson, 1975), but the question for most program-learning models is how children can learn algorithms in arbitrary, novel domains.

A third key challenge is in how children develop a real *theory* of mathematics, meaning one in which learners come to know about the multitude of objects, structures, and processes that mathematicians learn. The model here learns only a counting procedure, but as highlighted by Davidson et al. (2012) and others, children also come to know much more. Children learn successor principles, notions of even and odd numbers, addition and multiplication, and specific shortcuts relating to different areas of knowledge. Such richness in number should not be surprising given the reach of other human technology, including even, for instance, our ability to create physical devices—from sundials to mechanical calculators (Aspray, 2000)—for representing and manipulating quantities. However, there are no theories of how all of the pieces of number fit together. Indeed, studies of any single domain like counting pale in comparison with the full challenge of learning complex, interrelated theories. Computational work has only just begun to tackle theories with multiple interacting rules, principles, and concepts. The closest work might be Ullman et al. (2012), who developed a theory of learning magnetism based on compositions of logical primitives, or Rule (2020) who developed a system for learning arbitrary knowledge and computation. Even compared to these theory-learning models, what a U.S. adult knows about number is of a different order of magnitude. Scaling up to large networks of interconnected concepts is a challenge not just for the program-learning approach but also for any theory of number.

EMPIRICAL PREDICTIONS

Finally, it should be noted that the model here, as well as PTG, makes several clear predictions which can inform future work:

(i) As discussed above, the model predicts that data about higher number words (above 3) should be more influential in shaping children's transition to CP-knowing. This contrasts with analogical accounts in which the relationships between the first three words are what drives the CP-transition.

(ii) Children should show a capacity to learn a variety of different kinds of algorithms from these primitives, including those that express other kinds of systems. This can be seen already in the fact that children learn, for example, singular/plural systems in language, and circular systems for telling time (see PTG), but more broadly this class of model predicts a general ability to learn a variety of different types of computations, including perhaps other recursive systems of rules based on sets (see Rule et al., 2020). This ability is left unexplained by theories that number learning is based on one particular innate structure, rather than a general capacity.

(iii) Learners should prefer *simpler* processes or algorithms, when given ambiguous data. The simplicity preference is key to the model eventually arriving at a CP-knower hypothesis. Predicted simplicity biases can be seen in similar concept-learning models (Feldman, 2000; Goodman et al., 2008; Piantadosi et al., 2016).

(iv) Recursive processes should be dispreferred relative to other primitives by child learners. This means that if one gives a learner data which is ambiguous between a recursive and non-recursive process, learners should tend to generalize according to the non-recursive one. We note, however, that in general recursion is a difficult issue since all recursive processes can be rephrased as iterative processes, but in general a representation that is defined in terms of itself (like the recursive function here) should incur a penalty.

(v) Children should be able to compute the primitives required, at least by the age they start learning number, including operations on sets. They should succeed on recursive functions by the time they are CP-knowers. We note that this does not necessarily include all of the primitives in Tables 1 and 2, but certainly, the operations used in the learned representations should be easier than number, and antecedently available.

(vi) Following Carey (2009), the model predicts that the counting words themselves should play a critical role in children's learning. The counting procedure that is learned is one that critically hinges on the verbal system, and it is, therefore, likely that the concepts cannot be used without being given verbal labels (see Pitt et al., 2022).

(vii) Learning to count should be a separable piece from all of the other aspects of number, meaning that children who learn counting should not necessarily show other forms of knowledge about numbers (Davidson et al., 2012; Izard et al., 2014; Jara-Ettinger et al., 2016). The model only learns a counting procedure from naturalistic data, so other knowledge (e.g., infinity, addition, properties of sets, reasoning about the underlying algorithm, etc.) would need a separate learning setup. These types of knowledge are likely formalizable in similar terms—this is part of the strength of this kind of general "language of thought" approach.

We note also that any proposed intervention that, for example, provides extra data to the learner of a certain type can be first tested with the model. The model then makes quantitative predictions about how data of different kinds should influence acquisition. However, the model is currently only of a simple form where the data are pairs of sets and words, and data are all remembered with high fidelity. Such models lay groundwork for similar kinds of formal accounts that might take pedagogical evidence or include cognitive constraints. In particular, instruction on the process of counting itself seems to be important in children's learning (Mix et al., 2012; Paliwal & Baroody, 2018, 2020). Such richer kinds of input are not yet integrated into these types of formal models, but future work could incorporate instruction and different forms of input, as informative about the steps of the underlying algorithm. Such modeling work will be important for moving this framework toward naturalistic data and using it to help find plausible interventions.

CONCLUSION

This paper has attempted to provide an overview of how algorithm learning models could inform research on early number while addressing several critiques of PTG. Learning models that operate over spaces of computations do essentially the same thing as scientists: they look at data and craft a theory to explain it. Scientists' space of possible theories is relatively unrestricted—we can draw on probability in genetics, differential equations in understanding pattern formation, or posit novel categories and features of objects like quarks in atomic physics. Similarly, the space of possible hypotheses for children must be large in order to explain all it is that they are able to acquire.

A large hypothesis space is actually intended to "build in" as little as possible since the set of *all* hypotheses can be described very concisely (e.g., compositions of the primitives in Table 2). Yang and Piantadosi (2022) compare the situation to Borges (1941/1970)'s "Library of Babel"—a library which contains every logically possible book (every sequence of characters). The library consequently contains essentially no information at all, since its entire contents can be described with a single phrase: "all sequences of characters." What matters for the simplicity or parsimony of a hypothesis space is not its size, but rather how simply it can be expressed. Both the original PTG and the new version permit very concise descriptions of their hypothesis, and allow for learning of a vast variety of outcomes, showing that there is an important sense in which learners must construct the right representation for number and counting.

The advantage of computationally rich "language of thought" models is that they deploy similar machinery across diverse domains, including language (Piantadosi, 2011; Piantadosi et al., 2008; Rothe et al., 2017; Siskind, 1996; Yang & Piantadosi, 2022), rule-based concepts (Goodman et al., 2008; Piantadosi et al., 2016), handwriting (Lake et al., 2015), binary sequences (Planton et al., 2021), kinship systems (Katz et al., 2008; Kemp & Regier, 2012; Mollica & Piantadosi, 2022), visual concept learning and generalization (Depeweg et al., 2018; Lake & Piantadosi, 2020; Overlan et al., 2017), phonological theories (Ellis et al., 2022), geometry (Amalric et al., 2017; Romano et al., 2018), magnetism (Ullman et al., 2012), quantifiers (Kemp, 2009; Piantadosi, 2011; Piantadosi et al., 2016). Similar models have also been used in computer science to create compositional semantic learning models (Kwiatkowski et al., 2010; Liang et al., 2009, 2010; Zettlemoyer & Collins, 2005). This type of model also forms the basis of optimal inferential theories in AI which hold essentially that an intelligent agent should understand its world by building program-like theories of how the world operates (Hutter, 2005; Solomonoff, 1964).

Most generally, a hope for the study of early mathematics is that it will illuminate more about human nature than number. In some ways, it already has: consider what US children eventually learn about numbers—including prime numbers, long division, or procedures for manipulating fractions. Eventually, they learn algebra, geometry, and calculus; some learn how to solve arbitrary puzzles with mathematical structure, like a Rubik's cube or Sudoku. Children in some cultures use a body counting system (Saxe, 2012) where they acquire algorithms for arithmetic which are fundamentally unlike our culture's algorithms. Some even learn abacus as a *visual* system where they mentally picture images of beads and manipulate them with corresponding algorithms in order to solve arithmetic problems (Frank & Barner, 2012; Hatano et al., 1977, 1987; Hishitani, 1990; Miller & Stigler, 1991; Stigler, 1984; Stigler et al., 1986). Such diversity of cognition suggests that children simply *must* possess sophisticated abilities for internalizing essentially arbitrary procedures and algorithms. Accounts of learning that are grounded in constructing algorithms are the only theories that can even aspire to explain how it is we are able to do so much.

FUNDING INFORMATION

This work was supported by Grants 1760874 and 2000759 from the National Science Foundation (NSF), Division of Research on Learning. I am grateful to Sam Cheyette for helpful comments.

DATA AVAILABILITY STATEMENT

All code used in this project is available in the *Fleet* library at https://github.com/piantado/Fleet. Sociocultural policy statement: This work did not collect new experimental data. Confirmatory/exploratory data analysis statement: This work did not analyze experimental data.

ORCID

Steven T. Piantadosi https://orcid.org/0000-0001-5499-4168

REFERENCES

Amalric, M., Wang, L., Pica, P., Figueira, S., Sigman, M., & Dehaene, S. (2017). The language of geometry: Fast comprehension of geometrical primitives and rules in human adults and preschoolers. *PLoS Computational Biology*, *13*(1), e1005273.

Aspray, W. (2000). *Computing before computers*. Iowa State University Press.

Beck, J. (2017). Can bootstrapping explain concept learning? *Cognition*, *158*, 110–121.

Bonawitz, E., Denison, S., Gopnik, A., & Griffiths, T. L. (2014). Win-stay, lose-sample: A simple sequential algorithm for approximating Bayesian inference. *Cognitive Psychology*, *74*, 35–65.

Bonawitz, E., Denison, S., Griffiths, T. L., & Gopnik, A. (2014). Probabilistic models, learning algorithms, and response variability: Sampling in cognitive development. *Trends in Cognitive Sciences*, *18*(10), 497–500.

Borges, J. (1941/1970). The library of babel. In *Collected Fictions*. Trans. Andrew Hurley (pp. 79–89). Penguin.

Cantlon, J. F. (2012). Math, monkeys, and the developing brain. *Proceedings of the National Academy of Sciences of the United States of America*, *109*(Suppl. 1), 10725–10732.

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

Carey, S. (2014). On learning new primitives in the language of thought: Reply to Rey. *Mind & Language*, *29*(2), 133–166.

Carey, S., & Barner, D. (2019). Ontogenetic origins of human integer representations. *Trends in Cognitive Sciences*, *23*(10), 823–835.

Corballis, M. C. (2014). *The recursive mind: The origins of human language, thought, and civilization*. Princeton University Press.

Davidson, K., Eng, K., & Barner, D. (2012). Does learning to count involve a semantic induction? *Cognition*, *123*, 162–173.

Dehaene, S. (2011). *The number sense: How the mind creates mathematics*. Oxford University Press.

Depeweg, S., Rothkopf, C. A., & Jäkel, F. (2018). Solving bongard problems with a visual language and pragmatic reasoning. *arXiv preprint arXiv:1804.044 52*. https://doi.org/10.48550/arXiv.1804.04452

Devezer, B., Navarro, D. J., Vandekerckhove, J., & Ozge Buzbas, E. (2021). The case for formal methodology in scientific reform. *Royal Society Open Science*, *8*(3), 200805.

Ellis, K., Albright, A., Solar-Lezama, A., Tenenbaum, J. B., & O'Donnell, T. J. (2022). Synthesizing theories of human language with Bayesian program induction. *Nature Communications*, *13*(1), 5024.

Ellis, K., Wong, C., 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. *arXiv preprint arXiv:2006.08381*. https://doi.org/10.48550/arXiv.2006.08381

Elman, J. (1997). *Rethinking innateness: A connectionist perspective on development*. MIT Press.

Feigenson, L., & Carey, S. (2003). Tracking individuals via object-files: Evidence from infants' manual search. *Developmental Science*, *6*(5), 568–584.

Feigenson, L., & Carey, S. (2005). On the limits of infants' quantification of small object arrays. *Cognition*, *97*(3), 295–313.

Feldman, J. (2000). Minimization of Boolean complexity in human concept learning. *Nature*, *407*(6804), 630–633.

Ferrigno, S., Cheyette, S. J., Piantadosi, S. T., & Cantlon, J. F. (2020). Recursive sequence generation in monkeys, children, us adults, and native Amazonians. *Science Advances*, *6*(26), eaaz1002.

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

Fodor, J. (2010). Woof, woof. *Times Literary Supplement*, 7–8.

Fodor, J., & Pylyshyn, Z. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, *28*, 3–71.

Frank, M. C., & Barner, D. (2012). Representing exact number visually using mental abacus. *Journal of Experimental Psychology-General*, *141*(1), 134–149.

Frankland, S. M., & Greene, J. D. (2020). Concepts and compositionality: In search of the brain's language of thought. *Annual Review of Psychology*, *71*, 273–303.

Fuson, K. (1988). *Children's counting and concepts of number*. Springer-Verlag.

Gallistel, C., & Gelman, R. (1992). Preverbal and verbal counting and computation. *Cognition*, *44*, 43–74.

Gelman, R., & Gallistel, C. (1978). *The child's understanding of number*. Harvard University Press.

Gelman, R., & Meck, E. (1983). Preschoolers' counting: Principles before skill. *Cognition*, *13*(3), 343–359.

Gibson, D. J., Gunderson, E. A., & Levine, S. C. (2020). Causal effects of parent number talk on preschoolers' number knowledge. *Child Development*, *91*(6), e1162–e1177.

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

Graves, A., Wayne, G., & Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*. https://doi.org/10.48550/arXiv.1410.5401

Gunderson, E. A., & Levine, S. C. (2011). Some types of parent number talk count more than others: Relations between parents' input and children's cardinal-number knowledge. *Developmental Science*, *14*(5), 1021–1032.

Halberda, J., & Odic, D. (2015). The precision and internal confidence of our approximate number thoughts. In D. B. Berch, K. Mann Koepke, & D. C. Geary (Eds.), *Mathematical cognition and learning* (Vol. *1*, pp. 305–333). Elsevier.

Halberda, J., Mazzocco, M., & Feigenson, L. (2008). Individual differences in non-verbal number acuity correlate with maths achievement. *Nature*, *455*(7213), 665–668.

Hastings, W. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, *57*(1), 97.

Hatano, G., Amaiwa, S., & Shimizu, K. (1987). Formation of a mental abacus for computation and its use as a memory device for digits: A developmental study. *Developmental Psychology*, *23*(6), 832–838.

Hatano, G., Miyake, Y., & Binks, M. G. (1977). Performance of expert abacus operators. *Cognition*, *5*(1), 47–55.

Hauser, M. D., Chomsky, N., & Fitch, W. T. (2002). The faculty of language: What is it, who has it, and how did it evolve? *Science*, *298*(5598), 1569–1579.

Heim, I., & Kratzer, A. (1998). *Semantics in generative grammar*. Wiley-Blackwell.

Hindley, J. R., & Seldin, J. P. (2008). *Lambda-calculus and combinators, an introduction* (Vol. *13*). Cambridge University Press.

Hishitani, S. (1990). Imagery experts: How do expert abacus operators process imagery? *Applied Cognitive Psychology*, *4*(1), 33–46.

Hutter, M. (2005). *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Springer Science & Business Media.

Izard, V., Streri, A., & Spelke, E. S. (2014). Toward exact number: Young children use one-to-one correspondence to measure set identity but not numerical equality. *Cognitive Psychology*, *72*, 27–53.

Jara-Ettinger, J., Piantadosi, S. T., Spelke, E., Levy, R., & Gibson, E. (2016). Mastery of the logic of natural numbers is not the result of mastery of counting: Evidence from late counters. *Developmental Science*, *20*, e12459.

Jones, R. M., & Van Lehn, K. (1994). Acquisition of children's addition strategies: A model of impasse-free, knowledge-level learning. *Machine Learning*, *16*(1–2), 11–36.

Katz, Y., Goodman, N., Kersting, K., Kemp, C., & Tenenbaum, J. (2008). Modeling semantic cognition as logical dimensionality reduction. In *Proceedings of thirtieth annual meeting of the Cognitive Science Society*.

Kemp, C. (2009). Quantification and the language of thought. In *Advances in neural information processing systems* (Vol. *22*).

Kemp, C., & Regier, T. (2012). Kinship categories across languages reflect general communicative principles. *Science*, *336*(6084), 1049–1054.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., & Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing* (pp. 1223–1233).

Lake, B. M., & Piantadosi, S. T. (2020). People infer recursive visual concepts from just a few examples. *Computational Brain & Behavior*, *3*(1), 54–65.

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

Lee, M., & Sarnecka, B. (2010). A model of knower-level behavior in number concept development. *Cognitive Science*, *34*(1), 51–67.

Lee, M., & Sarnecka, B. W. (2011). Number-knower levels in young children: Insights from bayesian modeling. *Cognition*, *120*(3), 391–402.

Leslie, A. M., Gelman, R., & Gallistel, C. (2008). The generative basis of natural number concepts. *Trends in Cognitive Sciences*, *12*, 213–218.

Liang, P., Jordan, M., & Klein, D. (2009). Learning semantic correspondences with less supervision. In *Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing of the AFNLP* (Vol. 1, pp. 91–99).

Liang, P., Jordan, M., & Klein, D. (2010). Learning programs: A hierarchical Bayesian approach. In *Proceedings of the 27th international conference on machine learning*.

MacKay, D. (2003). *Information theory, inference, and learning algorithms*. Cambridge University Press.

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. W.H. Freeman & Company.

McNamara, J. M., Green, R. F., & Olsson, O. (2006). Bayes' theorem and its applications in animal behaviour. *Oikos*, *112*(2), 243–251.

Miller, K. F., & Stigler, J. W. (1991). Meanings of skill: Effects of abacus expertise on number representation. *Cognition and Instruction*, *8*(1), 29–67.

Mix, K. S., Sandhofer, C. M., Moore, J. A., & Russell, C. (2012). Acquisition of the cardinal word principle: The role of input. *Early Childhood Research Quarterly*, *27*(2), 274–283.

Mollica, F., & Piantadosi, S. T. (2022). Logical word learning: The case of kinship. *Psychonomic Bulletin & Review*, *29*, 766–799.

Odic, D., Libertus, M. E., Feigenson, L., & Halberda, J. (2013). Developmental change in the acuity of approximate number and area representations. *Developmental Psychology*, *49*(6), 1103–1112.

O'Rear, C. D., & McNeil, N. M. (2019). Improved set-size labeling mediates the effect of a counting intervention on children's understanding of cardinality. *Developmental Science*, *22*(6), e12819.

O'Shaughnessy, D. M., Cruz, T., Mollica, F., Boni, I., Jara-Ettinger, J., Gibson, E., & Piantadosi, S. T. (2023). Diverse mathematical knowledge among indigenous Amazonians. *Proceedings of the National Academy of Sciences of the United States of America*, *120*(35).

O'Shaughnessy, D. M., Gibson, E., & Piantadosi, S. T. (2021). The cultural origins of symbolic number. *Psychological Review*, *129*(6), 1442–1456.

Overlan, M., Jacobs, R., & Piantadosi, S. (2017). Learning abstract visual concepts via probabilistic program induction in a language of thought. *Cognition*, *168*, 320–334.

Paliwal, V., & Baroody, A. J. (2018). How best to teach the cardinality principle? *Early Childhood Research Quarterly*, *44*, 152–160.

Paliwal, V., & Baroody, A. J. (2020). Cardinality principle understanding: The role of focusing on the subitizing ability. *ZDM: The International Journal on Mathematics Education*, *52*(4), 649–661.

Peano, G. (1889). *Arithmetices principia: Nova methodo*. Fratres Bocca.

Piantadosi, S. T. (2011). *Learning and the language of thought* [Unpublished doctoral dissertation]. Massachusetts Institute of Technology.

Piantadosi, S. T. (2021). The computational origin of representation. *Minds and Machines*, *31*, 1–58.

Piantadosi, S. T., & Aslin, R. (2016). Compositional reasoning in early childhood. *PLoS One*.

Piantadosi, S. T., Goodman, N., Ellis, B., & Tenenbaum, J. B. (2008). A Bayesian model of the acquisition of compositional semantics. In *Proceedings of the Cognitive Science Society*.

Piantadosi, S. T., Palmeri, H., & Aslin, R. (2018). Limits on composition of conceptual operations in 9-month-olds. *Infancy*, *23*, 310–324.

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

Piantadosi, S. T., Tenenbaum, J., & Goodman, N. (2016). The logical primitives of thought: Empirical foundations for compositional cognitive models. *Psychological Review*, *123*, 392–424.

Pierce, B. C. (2002). *Types and programming languages*. MIT Press.

Pitt, B., Gibson, E., & Piantadosi, S. T. (2022). Exact number concepts are limited to the verbal count range. *Psychological Science*, *33*(3), 371–381.

Planton, S., van Kerkoerle, T., Abbih, L., Maheu, M., Meyniel, F., Sigman, M., Wang, L., Figueira, S., Romano, S., & Dehaene, S. (2021). A theory of memory for binary sequences: Evidence for a mental compression algorithm in humans. *PLoS Computational Biology*, *17*(1), e1008598.

Rey, G. (2014). Innate and learned: Carey, mad dog nativism, and the poverty of stimuli and analogies (yet again). *Mind & Language*, *29*(2), 109–132.

Rips, L., Asmuth, J., & Bloomfield, A. (2006). Giving the boot to the bootstrap: How not to learn the natural numbers. *Cognition*, *101*, 51–60.

Rips, L., Asmuth, J., & Bloomfield, A. (2008). Do children learn the integers by induction? *Cognition*, *106*, 940–951.

Rips, L., Asmuth, J., & Bloomfield, A. (2013). Can statistical learning bootstrap the integers? *Cognition*, *128*(3), 320–330.

Rips, L., Bloomfield, A., & Asmuth, J. (2008). From numerical concepts to concepts of number. *Behavioral and Brain Sciences*, *31*, 623–642.

Romano, S., Salles, A., Amalric, M., Dehaene, S., Sigman, M., & Figueira, S. (2018). Bayesian validation of grammar productions for the language of thought. *PLoS One*, *13*, e0200420. https://doi.org/10.1371/journal.pone.0200420

Rothe, A., Lake, B. M., & Gureckis, T. (2017). Question asking as program generation. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 1046–1055).

Rule, J. (2020). *The child as hacker: Building more human-like models of learning* [Unpublished doctoral dissertation]. Massachusetts Institute of Technology.

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

Sarnecka, B., & Lee, M. (2009). Levels of number knowledge during early childhood. *Journal of Experimental Child Psychology*, *103*, 325–337.

Saxe, G. B. (2012). *Cultural development of mathematical ideas: Papua New Guinea studies*. Cambridge University Press.

Shrager, J., & Siegler, R. (1998). Scads: A model of children's strategy choices and strategy discoveries. *Psychological Science*, *9*(5), 405–410.

Shusterman, A., Slusser, E., Halberda, J., & Odic, D. (2016). Acquisition of the cardinal principle coincides with improvement in approximate number system acuity in preschoolers. *PLoS One*, *11*(4), e0153072.

Siegler, R., & Shrager, J. (1984). Strategy choices in addition and subtraction: How do children know what to do? In C. Sophian (Ed.), *Origins of cognitive skills* (pp. 229–293). Lawrence Erlbaum Associates.

Siskind, J. (1996). A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, *61*, 31–91.

Solomonoff, R. J. (1964). A formal theory of inductive inference. Part I. *Information and Control*, *7*(1), 1–22.

Spelke, E. S., & Kinzler, K. (2007). Core knowledge. *Developmental Science*, *10*(1), 89–96.

Spelke, E. S., & Kinzler, K. D. (2009). Innateness, learning, and rationality. *Child Development Perspectives*, *3*(2), 96–98.

Stamps, J. A., & Frankenhuis, W. E. (2016). Bayesian models of development. *Trends in Ecology & Evolution*, *31*(4), 260–268.

Steedman, M. (2000). *The syntactic process* (Vol. *131*). MIT Press.

Stigler, J. W. (1984). "Mental abacus": The effect of abacus training on Chinese children's mental calculation. *Cognitive Psychology*, *16*(2), 145–176.

Stigler, J. W., Chalip, L., & Miller, K. F. (1986). Consequences of skill: The case of abacus training in Taiwan. *American Journal of Education*, *94*(4), 447–479.

Svenson, O. (1975). Analysis of time required by children for simple additions. *Acta Psychologica*, *39*(4), 289–301.

Szkudlarek, E., Park, J., & Brannon, E. M. (2021). Failure to replicate the benefit of approximate arithmetic training for symbolic arithmetic fluency in adults. *Cognition*, *207*, 104521.

Szűcs, D., & Myers, T. (2017). A critical analysis of design, facts, bias and inference in the approximate number system training literature: A systematic review. *Trends in Neuroscience and Education*, *6*, 187–203.

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

Ullman, T., Goodman, N., & Tenenbaum, J. (2012). Theory learning as stochastic search in the language of thought. *Cognitive Development*, *27*, 455–480.

van Marle, K., Chu, F. W., Mou, Y., Seok, J. H., Rouder, J., & Geary, D. C. (2018). Attaching meaning to the number words: Contributions of the object tracking and approximate number systems. *Developmental Science*, *21*(1), e12495.

van Rooij, I., & Baggio, G. (2021). Theory before the test: How to build high-verisimilitude explanatory theories in psychological science. *Perspectives on Psychological Science*, *16*(4), 682–697.

Whalen, J., Gallistel, C., & Gelman, R. (1999). Nonverbal counting in humans: The psychophysics of number representation. *Psychological Science*, *10*(2), 130–137.

Wynn, K. (1992). Children's acquisition of the number words and the counting system. *Cognitive Psychology*, *24*(2), 220–251.

Xu, F. (2019). Towards a rational constructivist theory of cognitive development. *Psychological Review*, *126*(6), 841–864.

Yang, Y., & Piantadosi, S. T. (2022). One model for the learning of language. *Proceedings of the National Academy of Sciences of the United States of America*, *119*(5).

Zettlemoyer, L. S., & Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence* (pp. 658–666).

How to cite this article: Piantadosi, S. T. (2023). The algorithmic origins of counting. *Child Development*, *00*, 1–19. https://doi.org/10.1111/cdev.14031