Précis of *The child as hacker: Building more human-like models of learning*

Joshua S. Rule

2020, Massachusetts Institute of Technology, Department of Brain and Cognitive Sciences

Cognitive development is central to human cognition. People master an unparalleled cognitive repertoire, building on a limited set of core cognitive abilities to construct intuitive and formal theories, master natural languages, acquire complex perceptual and motor abilities, develop intrapersonal and interpersonal social skills, and cultivate capacities for creative expression. Human-like performance seems substantially beyond current artificial intelligence in all of these domains, yet people essentially acquire these abilities simultaneously and universally given appropriate experience. People also learn how to learn, developing general strategies for rapidly adapting to novel situations. The right kinds of practice can even lead them to develop world-class expertise and discover new ideas that radically alter humanity's understanding of the world. How do we get so much from so little so quickly?

Many of our best models treat learning as analogous to computer programming because symbolic programs provide the most compelling account of sophisticated mental representations, a modern formulation of Fodor's *Language of Thought* (LOT; Fodor, 1975) as something like a programming language. Learning then becomes programming, forming computational expressions that encode knowledge—for instance, composing computational primitives like CAUSE, GO, and UP to form LIFT (Siskind, 1996) or discovering simple procedures for counting (Piantadosi et al., 2012).

This dissertation extends the idea of learning as programming by hypothesizing that learning from childhood onward is analogous to the specific ways that people make programs better. I call this distinctively human style of programming *hacking*, emphasizing the way that people weave together activities, values, and goals to rapidly improve programs. For example, hackers' techniques can vary from tuning parameters in existing programs to developing entire new languages; their values can range from efficient resource management to the unabashed pursuit of novelty and fun;

and their goals can span a single subroutine or an entire codebase. This *child as hacker* hypothesis is important because it identifies a key source of inductive constraints on learning (formatting mental representations as program-like structures) and an abundant supply of concrete hypotheses about how these constraints impact learning (the programming practices of actual hackers) that can be applied to go beyond simpler models of learning and better understand cognitive development.

This dissertation also combines computational and behavioral approaches to formalize and test these ideas. It reports a large-scale empirical study of human and machine concept learning involving 250 concepts, nearly 400 people, and five classes of computational models. As part of this effort, I develop several useful tools. First, I develop a computational model of learning that embodies core features of hacking. Second, I introduce a novel behavioral tool in the form of list functions, a domain well-suited to studying inference mechanisms in learning. Third, I introduce a benchmark set of 250 list functions to focus the present investigation and support future research.

The study contrasts two accounts of learning. The first explains learning primarily in terms of simplicity measures like *description length* (Feldman, 2000; Feldman, 2003; Chater & Vitányi, 2003). It argues that learning is similar to random guessing, generating hypotheses largely independently of data. The more complex a concept is, the more difficult it is to guess, and the more difficult it will be to learn. The second explains learning as hacking: learners carefully observe data and use structured activities to revise them into better theories of themselves. For example, the internal structure of the list [1, 2, 3, 4, 5, 6] suggests that it was generated using a count routine, while [5, 2, 5, 2, 5, 2] implies copying or repetition. Like hacking, these insights rely on: 1) carefully observing data (e.g. each item in the first list is one larger than the previous item); and 2) drawing structured inferences from those observations (e.g. the list was likely generated by counting to 6). This kind of constructive thinking (Xu, 2019; Lombrozo, 2019) is largely absent in simplicity-based models but present in a model of learning as hacking. This model outperforms alternative models both in explaining human learning and in raw performance. These results suggest that people may: 1) dissociate the constructive inferences involved in learning a concept from the representation of the concept itself; and 2) search over the space of inferences rather than the space of representations. These findings are more generally notable because they show that the analogy between learning and hacking productively contributes to our understanding of conceptual development.

Logic	first-order, modal, deontic logic
Mathematics	number systems, geometry, calculus
Natural language	morphology, syntax, number grammars
Sense data	audio, images, video, haptics
Computer languages	C, Lisp, Haskell, Prolog, LaTeX
Scientific theories	relativity, game theory, natural selection
Operating procedures	Robert's Rules, bylaws, checklists
Games & sports	Go, football, 8 queens, juggling, Lego
Norms & mores	class systems, social cliques, taboos
Legal codes	constitutions, contracts, tax law
Religious systems	monastic orders, vows, rites & rituals
Kinship	genealogies, clans/moieties, family trees
Mundane chores	knotting ties, making beds, mowing lawns
Intuitive theories	physics, biology, theory of mind
Domain theories	cooking, lockpicking, architecture
Art	music, dance, origami, color spaces

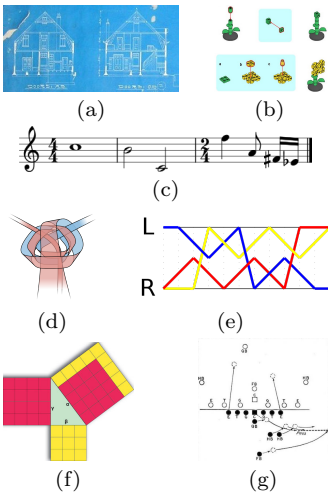


Table 1 & Figure 1: Domains requiring algorithmic knowledge: (a) construction plans; (b) assembly instructions; (c) musical notation; (d) knots (e) juggling patterns; (f) graphical proofs; and (g) football.

The child as hacker[1]

The **first chapter** in this dissertation reviews the idea of *learning as programming*, which holds that symbolic programs provide the best formal knowledge representation we have. A critical mass of work throughout cognitive science has converged on the hypothesis that human learning operates over structured (Fodor, 1975; Fodor & Pylyshyn, 1988), probabilistic (Lake et al., 2017; Goodman et al., 2015), program-like (Turing, 1936; Baum, 2004) representations. While there have been many other proposals for modeling conceptual representations, only programs arguably capture the full breadth and depth of people's algorithmic abilities (Goodman et al., 2015). Code can model both procedural and declarative information and allow them to interact seamlessly. Universal programming languages integrate all this knowledge into a single formal representation. Programs can also be communicated in many forms, including not only formal code but forms familiar in all cultures, such as natural language (e.g. *If you want to X, first you need to Y, then try to Z, but if that fails...*; Lupyan & Bergen, 2016), images, and diagrams (Table 1 & Figure 1).

If knowledge is code, learning is then program induction—discovering programs that explain how observed data were generated (Kitzelmann, 2009; Flener & Schmid, 2008; Gulwani et al., 2017). This technique draws on literature stretching back to the birth of cognitive science (Newell et al., 1958; Newell et al., 1959), across many formalizations of learning, e.g. deep learning (LeCun

[1]The research in this section appears in the literature as Rule, J. S., Piantadosi, S. T., & Tenenbaum, J. B. (2020). The child as hacker. *Trends in Cognitive Sciences*.

et al., 2015), connectionism (Rumelhart et al., 1987), reinforcement learning (Sutton & Barto, 2018), probabilistic graphical models (Koller & Friedman, 2009), and production systems (Lovett & Anderson, 2005). Learning as programming is importantly different, however, in providing learners the full expressive power of symbolic programs both theoretically (i.e. Turing completeness) and practically (i.e. freedom to adopt any formal syntax). This approach aligns with rational constructivist models of development (Xu, 2019; Gopnik & Wellman, 2012) and applies broadly to developmental phenomena (e.g. Siskind, 1996; Goodman et al., 2008; Ullman et al., 2012; Piantadosi et al., 2012; Lake et al., 2015; Amalric et al., 2017; Ullman et al., 2018).

Despite its successes, much work remains to develop learning as programming into a robust account of children's learning. Most real-world problems requiring program-like solutions are complex enough that there is no single metric of utility nor unified process of development. Modern computational approaches to learning—whether standard learning algorithms or more recent LOT models—use far fewer techniques and values than human programmers. Doing justice to learning as programming—both in what it means to learn and in what it means to program—requires going beyond current approaches (Figure 2). The **second chapter** in this dissertation enriches learning as programming with a distinctly human style of programming called *hacking*. As used here, hacking is about making code better. It adopts all the values that make code better, the many techniques people use to improve code, and a profound sense of internal motivation.

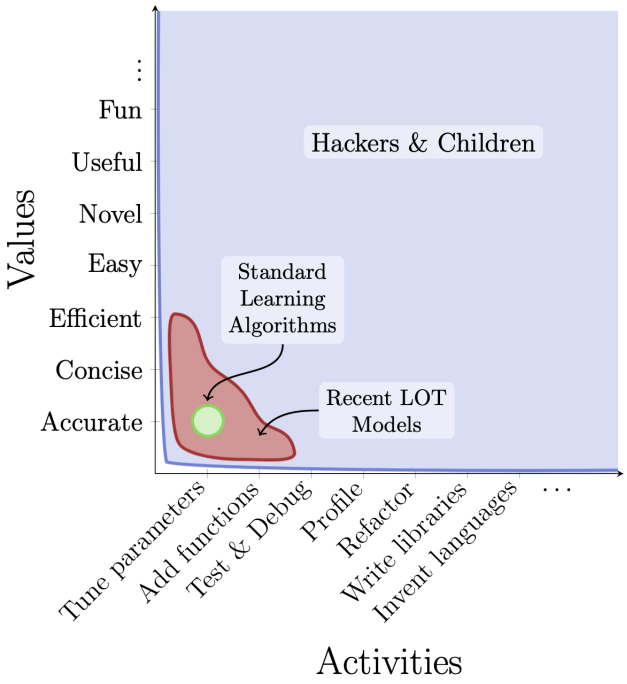


Figure 2: Overview of the child as hacker hypothesis. Code can be changed using many techniques (x-axis) and assessed according to many values (y-axis). Standard learning models in machine learning and psychology (green region) tend to focus solely on tuning the parameters of statistical models to improve accuracy. Recent LOT models (red region) expand this scope, writing functions in program-like representations and evaluating them for conciseness and sometimes efficiency. Yet, the set of values and techniques used by actual hackers (and by hypothesis, children; blue region) remains much larger.

There are many dimensions along which hackers seek to improve their code, making it not only more accurate, but perhaps faster, clearer, cleverer, more modular, more memory-efficient, and so

on. The simplest program is unlikely to be the most general; the fastest is usually not the easiest to write. Real world systems do not focus exclusively on the metrics that have come to the forefront of current LOT-learning paradigms. Alongside accuracy and simplicity, hackers also consider values related to resource use, intrinsic reward, aesthetic concerns, and complexity management.

Hackers have developed many process-level mechanisms for improving code according to these values (Fowler, 2018), including adding new functions and data structures, debugging faulty code, refactoring working code, and even inventing new languages. Hackers understand dozens, even hundreds, of these mechanisms and their potential impacts; most are specially tailored to specific kinds of problems. A hacker might care about speed and so cache the output of subcomputations in an algorithm, or she might seek modularity and so define data structures that encapsulate information and make it accessible through specific interfaces. This diversity of techniques makes hacking different from both common learning algorithms and recent LOT models. These other models typically explore a small set of mechanisms based on simple local methods like gradient descent, random sampling, or exhaustive enumeration.

Hacking is intrinsically motivated. Though a hacker may often be motivated in part by an extrinsic goal, she always generates her own goals—choosing specific dimensions she wants to improve—and pursues them at least as much for the intrinsic reward of better code as for any instrumental purpose. Whatever their origins, her choice of goals often appears spontaneous, even stochastic, constantly updated in light of recent changes to her code. Rather than randomly walking from goal to goal, however, she learns to maintain a network of goals. Even if she eventually achieves her initial goal, the path she follows may not be the most direct available. Her goals are thus primarily a means to improve her code rather than ends in themselves.

Hacking thus represents a collection of epistemic values and practices adapted to organizing knowledge using programs, and there is growing evidence that programs are a good model of mental representations. The child as hacker combines these ideas into a roadmap toward a computational account of learning and cognitive development. It is compatible with other core developmental metaphors—including the child as scientist (Piaget, 1955; Carey, 2009; Schulz, 2012b; Gopnik, 2012) and the evolutionary metaphor (Siegler, 1996; see also Siegler & Jenkins, 1989)—and, like them, makes testable claims about human learning. For example, it identifies a general class of inductive biases humans ought to have—namely those related to synthesizing, executing, and

analyzing information as programs. It also concretely identifies the representations, objectives, and processes supporting learning with those of human hackers. The rest of the dissertation tests these ideas in a large-scale study of human and machine concept learning in the domain of list functions.

New tools for investigating human learning

the list [8, 2, 7, 0, 3]

```
(λ xs [8, 2, 7, 0, 3])
```

```
[1, 4, 23, 21]          → [8, 2, 7, 0, 3]
[60, 7]                 → [8, 2, 7, 0, 3]
[8, 67, 54, 54, 97]     → [8, 2, 7, 0, 3]
[3, 3, 6, 55, 63, 7]    → [8, 2, 7, 0, 3]
[29]                    → [8, 2, 7, 0, 3]
```

add the index to every element

```
(λ xs (mapi + xs))
```

```
[2, 0, 92, 21, 33]          → [3, 2, 95, 25, 38]
[7, 51, 94, 72, 88, 19]     → [8, 53, 97, 76, 93, 25]
[75, 32, 46, 71, 49, 60]    → [76, 34, 49, 75, 54, 66]
[10, 12, 11, 8, 9, 7]       → [11, 14, 14, 12, 14, 13]
[52, 87, 27, 25]            → [53, 89, 30, 29]
```

keep only elements followed by an even number

```
(λ xs (map first
       (filter (λ y (is_even (second y)))
               (zip (droplast 1 xs)
                    (drop 1 xs)))))
```

```
[29, 88, 44, 75, 5, 17, 36, 0, 89, 31] → [29, 88, 17, 36]
[54, 4, 7, 43, 8, 97, 25, 5, 0]        → [54, 43, 5]
[24, 41, 96, 14, 93, 47]               → [41, 96]
[19, 81, 1, 53, 85, 3, 97]             → [ ]
[3, 76, 20, 11, 86, 8, 5, 94]          → [3, 76, 11, 86, 5]
```

Figure 3: Three example list functions of varying difficulty. Each example gives an English gloss, a working program in a domain-specific lambda-calculus developed in the dissertation, and several *input* → *output* examples.

The third and fourth chapters of this dissertation develop novel behavioral and computational tools for investigating the child as hacker empirically. The **third chapter** identifies the domain of list functions—computable functions over lists of natural numbers—as a prime target for studying the child as hacker hypothesis empirically (Figure 3). This domain has a long history in artificial intelligence (Green et al., 1974; Shaw et al., 1975; Biermann, 1978; Green, 1981; Smith, 1984; Feser et al., 2015; Osera & Zdancewic, 2015; Polikarpova et al., 2016; Cropper et al., 2019) but is virtually unstudied in cognitive psychology. It is psychologically interesting, however, because it combines many of the best properties of classic concept learning domains. It is familiar and engaging to learners in numerate societies, who can draw on meaningful background knowledge about numbers and sequences. List functions also support multiple tasks requiring different kinds of reasoning (e.g. learning functions from input/output pairs, generating new functions, generating input/output pairs to teach someone a function). Both lists and numbers have internal structure over which many relations can be defined. These features combine to support broad variance in problem difficulties. Many concept learning domains are either formally tractable but computationally simple, such as Boolean concepts, or computationally

sophisticated but formally challenging, such as visual analogies. List functions are both formally tractable and computationally sophisticated. The domain decomposes into a small set of meaningful primitives but is also Turing-complete and naturally supports the full range of human computational abilities. Many well-studied cognitive psychology tasks can be naturally represented as list function tasks, such as: *Give-a-Number*, *How-Many*, *2-4-6*, the *Number Game*, sequence prediction, text-based analogy, Boolean concept learning, sorting/seriation, and small number addition.

Perhaps the most exciting feature of list functions, however, is the potential they have to illuminate human learning. Suppose we were trying to learn a list function from the following data:

$$
\begin{aligned}
[88,\ 93,\ 73,\ 54,\ 79] &\rightarrow [88,\ 1,\ 93,\ 2,\ 73,\ 3,\ 54,\ 4,\ 79,\ 5] \\
[11,\ 0,\ 85,\ 98] &\rightarrow [11,\ 1,\ 0,\ 2,\ 85,\ 3,\ 98,\ 4] \\
[62,\ 53,\ 21] &\rightarrow [62,\ 1,\ 53,\ 2,\ 21,\ 3]
\end{aligned}
$$

The first thing we might notice is that the inputs and outputs both begin with the same elements. Further examination might show that all the elements of the input appear in the output and in the same order. We might then see that the input elements are not the only elements in the output; they do not even appear consecutively. They are instead mixed with many other elements. We might then reconceptualize our examples as something like this, clearly marking the new elements:

$$
\begin{aligned}
[88,\ 93,\ 73,\ 54,\ 79] &\rightarrow [88,\ 1,\ 93,\ 2,\ 73,\ 3,\ 54,\ 4,\ 79,\ 5] \\
[11,\ 0,\ 85,\ 98] &\rightarrow [11,\ 1,\ 0,\ 2,\ 85,\ 3,\ 98,\ 4] \\
[62,\ 53,\ 21] &\rightarrow [62,\ 1,\ 53,\ 2,\ 21,\ 3]
\end{aligned}
$$

This reframing shows exactly one additional element in the output for each input element. These input elements and the new elements occur in pairs, and the order of the new elements is consistent. We might then notice that the new elements are drawn in order from the count list, a familiar numerical sequence. The function appears to interleave the count list and the input list, effectively placing the index of each element in the input after its occurrence in the output. What is remarkable is how we are able to bring all these observations to bear during the process of learning. It suggests a process very unlike the unstructured curve fitting of many statistical models or the guess-and-check of many models of learning in an LOT. It is instead reminiscent of the way hacking weaves together activities and values to improve code.

The **fourth chapter** describes HL, a computational model of inductive learning designed to embody core aspects of hacking. HL is interesting because each facet of the model's architecture—including its representations, objectives, and learning mechanisms—interprets some phenomenon of human learning through the lens of hacking. This allows HL to address richer aspects of cognition generally ignored in previous learning as programming models. For example, HL models learning as the development of an entire programming language similar to the way that humans develop conceptual systems (Gopnik, 1983; Carey, 1985, 2009). It also uses complex objective functions that vary based on the task at hand. These objectives allow it to avoid premature optimization, exploring suboptimal and even known-to-be-wrong hypotheses during search while favoring the best hypotheses during later decision-making (Schulz, 2012a). They also also implement a sort of queried-guided search (Chu et al., 2019), using high-level information about the task to rule out syntactically well-formed but semantically inappropriate hypotheses.

Perhaps most importantly, however, is the way that HL restructures search. Most existing models of learning in the LOT directly compose a program-like mental representation by trying to correctly guess each constituent symbol. This approach is like observing that "This swan is white" and "That swan is white" and then proceeding through a series of hypotheses about black bears, purple elephants, and tangerine pangolins before eventually arriving at something like "All swans are white". Humans, by contrast, induce a conclusion about white swans precisely because the premises are about white swans. We are sensitive to the information in the data and use it directly. This might seem intuitive, but it is *not* how traditional models of learning in the LOT operate. It is however, similar to how HL works. Its search mechanism is based on the idea that people possess mechanisms for thinking about structures they observe in the world as mental programs and for iteratively revising those programs using specific inference rules. Each inference rule encapsulates a computational pattern for transforming mental programs. Rather than directly composing programs from primitives, learners compose inferences that produce programs as output. HL thus performs a sort of constructive learning (Xu, 2019; Lombrozo, 2019) that enables it to rapidly learn certain kinds of complex concepts.

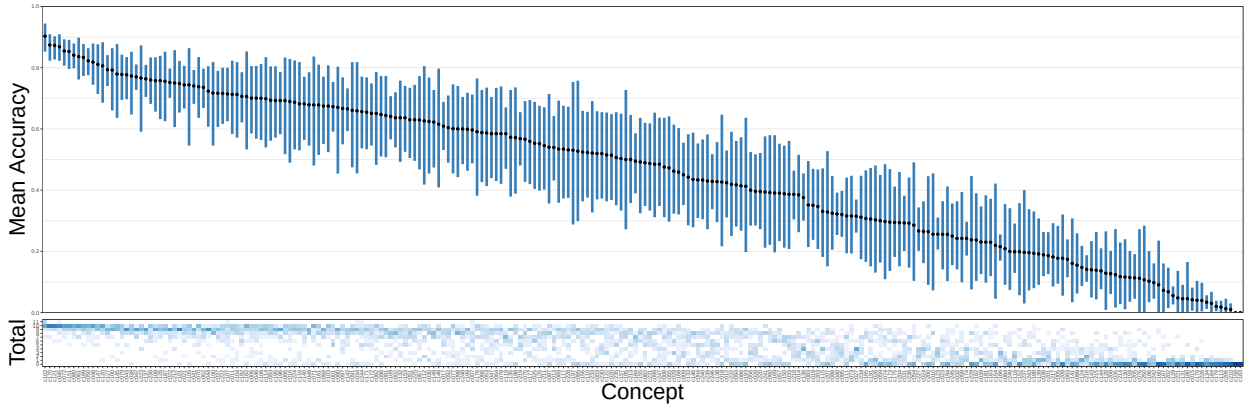


Figure 4: Human list function learning performance. (Top): Mean accuracy (y-axis) on each concept (x-axis) in descending order of mean accuracy, showing between-function variation. Error bars (blue bars) are bootstrapped 95% CIs. (Bottom): For each concept (x-axis), the percentage of participants with each possible number of correct responses (y-axis), showing within-function variation. Color varies from white (0%) to blue (100%).

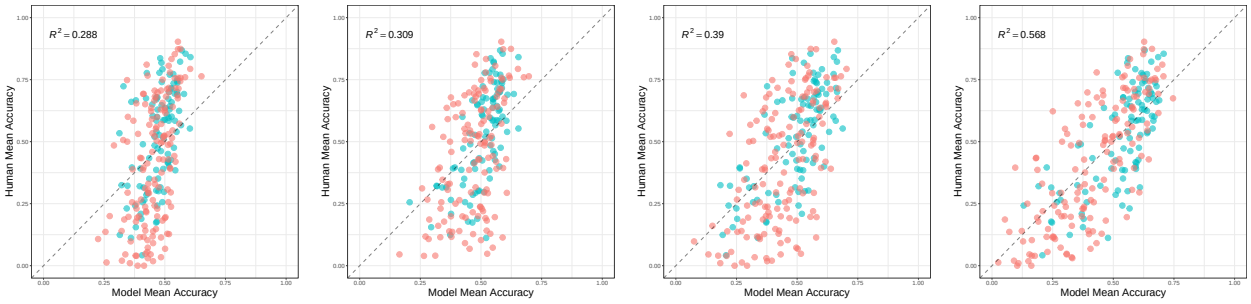
Structural sources of difficulty in human concept learning

The fifth and sixth chapters of this dissertation use list functions to test the child as hacker hypothesis empirically. Together, they describe a large-scale behavioral study of concept learning that contributes to our understanding of both human learning and artificial intelligence.

These chapters specifically explore the question of why people find some concepts harder to learn than others. Both humans and computational models find simple concepts easier to learn than complex concepts (Feldman, 2000; and later Goodman et al., 2008; Piantadosi et al., 2016; Kemp & Tenenbaum, 2008; Kemp et al., 2010; Ullman et al., 2012; cf. Lafond et al., 2007), leading to arguments that learning difficulty is strongly determined by simplicity (e.g. Feldman, 2003)[2]. The child as hacker challenges this idea and suggests that people may be able to learn much faster than might be predicted by simplicity alone. It predicts that learners rely on a diverse set of specialized learning mechanisms similar to those used in hacking (Fowler, 2018; Abelson et al., 1996). These mechanisms perform structured inferences that effectively shrink a learner's hypothesis space and thereby speed learning.

The **fifth chapter** reports a large-scale list function learning experiment testing these hypothe-

[2]There are at least two places simplicity occurs in discussions of learning. One focuses on the goal of learning and develops normative arguments about the kinds of concepts that learners should eventually acquire. These are often based on Occam's Razor and conclude that learners ought to prefer simple explanations (Chater & Vitányi, 2003; Baum, 2004). This question isn't being discussed here. The second, which is under discussion here, focuses on the process by which learning occurs. It asks the empirical question of whether simpler things are easier to learn than more complex things.



(a) (b) (c) (d)

Figure 5: Logistic model predictions (x-axis) of mean human accuracy (y-axis) by concept (circles): (a) based on description length in a rich model LOT; (b) based on the word count of correct participant verbal responses; (c) based on the word count of standardized verbal glosses; and (d) based on semantic and syntactic features, including visibility. Concepts examined in the formal model comparison are marked in blue. R^2 is variance in human performance explained by model predictions.

ses in humans. I developed a benchmark set of 250 list functions and taught them to nearly 400 people. These functions ranged from the identity function to complex operations involving conditional, numerical, and recursive reasoning. I followed prior work on Boolean concept learning and presented data in an online fashion (Piantadosi et al., 2016), asking participants to make predictions after each observation in order to capture trial-by-trial learning and generalization. I found broad variance in overall performance both across functions and within functions (Figure 4).

I also extracted a series of features based on formal and informal descriptions of each function. Some features measured a function's simplicity, including three measures of description length. Others captured aspects of a function's semantics, such as the use of conditional or recursive reasoning, the use of counting knowledge, argument structure, and visibility—a concept introduced here to measure how transparently each symbol in an LOT expression is encoded in the observed data. For example, consider the function, `f(x) = slice(x, 2, (length(x) - first(x))`. It extracts a slice, or sublist, of the input, `x`, containing the second through M^{th} elements, where M is the difference between the length of `x` and the first element of `x`. Here, `slice` and the first use of `x` are considered visible. That the output always includes a portion of the input immediately suggests `x` is used somewhere. `slice` is not the only symbol that can extract a sublist, but it is the most likely to produce non-trivial sublists of the input which excludes both endpoints. The symbol `2`, by contrast, is somewhat less visible, requiring several examples to rule out alternatives like indexing based on the first element of the list or removing elements failing some test. The term computing M is hidden. It relies on a complex relationship between the length of the list and an

element of the list; while both are directly encoded in the input, the relationship between them is obscure.

Analyzing the data in terms of these features revealed that while description length has some predictive value in explaining learning, each symbol's impact on learning is heavily modulated by visibility (Figure 5). A function composed of symbols that can be inferred easily from data is much easier to learn than a function of similar length whose symbols are not as transparently encoded by input/output data. Moreover, semantic features like the use of recursive reasoning or counting information also explain significant variation in learning performance independently of visibility.

These results show that humans are strongly sensitive to the semantic content of data and can leverage it to learn more quickly than expected based on simplicity alone. As such, they provide empirical support in favor of the child as hacker hypothesis while challenging simpler approaches to learning in the LOT. I argue that humans may use learning mechanisms which—unlike enumeration or random sampling—decouple the complexity of learning a concept from the complexity of that concept's representation in the LOT. This experiment also introduces a benchmark data set for understanding human learning that can be used to compare formal theories of concept acquisition throughout cognitive science.

More human-like models of learning

The **sixth chapter** compares HL with several alternative computational models of learning in the LOT. It specifically compares them as explanations of the behavioral data collected in the fifth chapter. I focused on 100 of the 250 functions tested in humans, selecting concepts that represent a broad spectrum of algorithmic content while remaining tractable for current learning models. While HL learns through a diversity of structure-sensitive learning mechanisms, the remaining models exemplify other common approaches to learning in the LOT. One exhaustively enumerates all possible hypotheses in order of length. Another stochastically samples from a Bayesian posterior over hypotheses, favoring those which are short and accurate. A third performs a proof-guided search, representing the problem as a logical theory whose free variables must be instantiated to solve the task. The fourth is a form of neural program synthesis, training a deep neural network on millions of programs and input/output pairs. When given novel data, the trained network

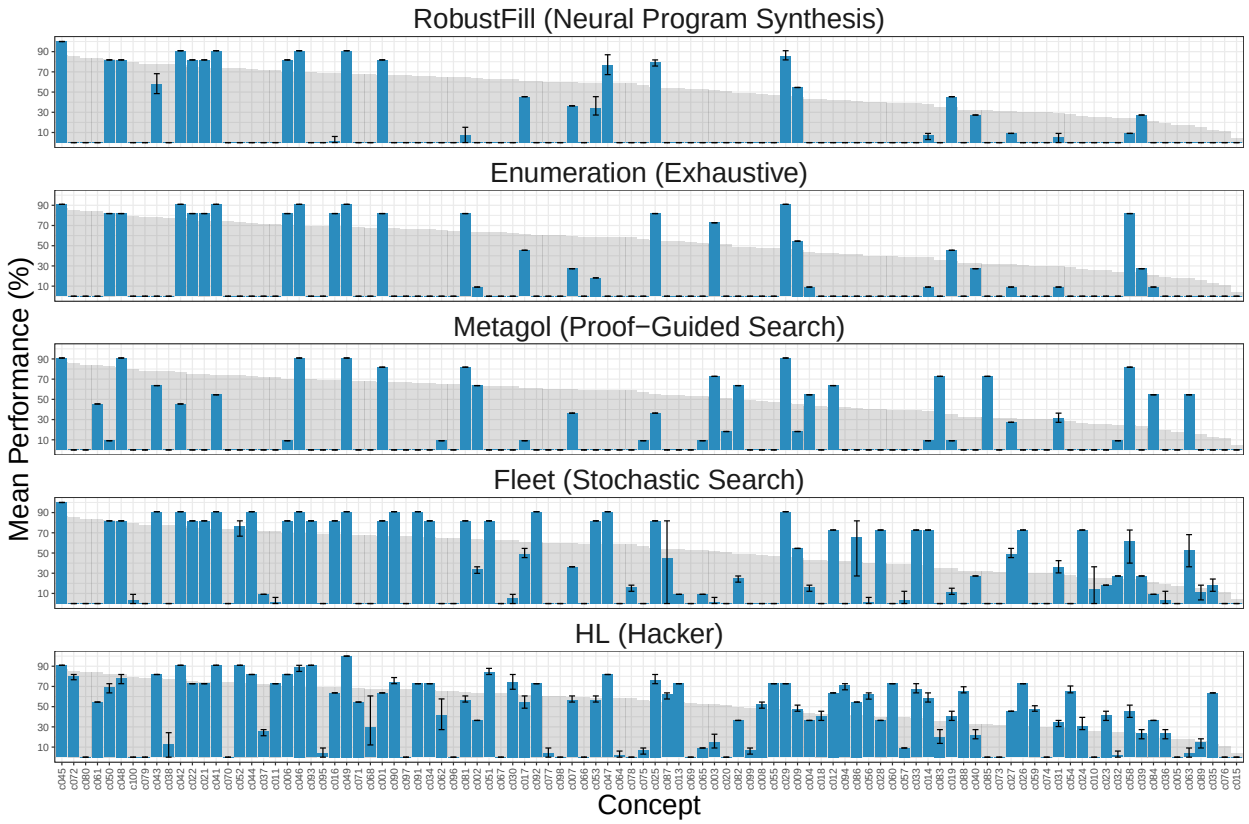


Figure 6: Mean accuracy (y-axis) on each concept (x-axis) by model (subplots). Concepts are ordered by mean human accuracy. Error bars are bootstrapped 95% CIs and gray region is human mean accuracy.

then generates a distribution over hypothesis programs that can be sampled to provide candidate solutions. Crucially, each model serves as a formal hypothesis about the psychological process learners go through in completing the list functions task. By comparing their performance, we have the potential to gain meaningful insight into the way that humans complete this task and, more generally, how they learn.

Relative to these comparison models, HL better fit human patterns of success and failure both quantitatively and qualitatively. It predicted human performance significantly more accurately than the comparison models on a sizable majority of the analyzed functions (Figure 6 & Table 2). HL was also sensitive to visibility and several other semantic features to roughly the same degree as humans, while other models strongly diverged from estimates of human sensitivity. In cases where HL significantly outperformed other models, it heavily relied on features absent in the other models, such as observing structure in the raw data and iteratively applying a diverse set of learning mechanisms. Cases where HL differed significantly from humans could be largely explained by a

single key difference between HL and humans, namely humans' significantly more sophisticated recursive reasoning (Figure 7).

This work provides a direct empirical test of the child as hacker hypothesis, showing that models embodying its core principles outperform more traditional models of learning in the LOT. These traditional models rely on symbol-based guess-and-check. Effectively, they make a long series of guesses about the exact string of symbols needed to describe some concept; each successive guess differs from the previous one by at most a few symbols. Changes are at best weakly sensitive to any information contained in observed data. By contrast, the HL model searches a space of inference procedures in such a way that successive search steps may represent radically different hypotheses. It also frequently uses observed data as the basis for its entire chain of reasoning. HL's success in explaining human learning suggests that people are sensitive to the structure of observed data and leverage that structure during learning. These results are also deeply connected to empirical work on the diversity of learning strategies in children and adults (Siegler, 1996; Siegler & Jenkins, 1989), and we show here that this diversity is a better explanation of adult concept learning than models which rely on a single inference mechanism (e.g. backprop in deep learning, enumeration, or deductive proof). Together, these findings suggest that this kind of diversity may be a

	Non-Zero	**Best ±5%**	**|Error| < 25%**
RobustFill	28	35	33
Enumeration	30	36	30
Metagol	36	36	25
Fleet	64	43	43
HL	**81**	**82**	**61**

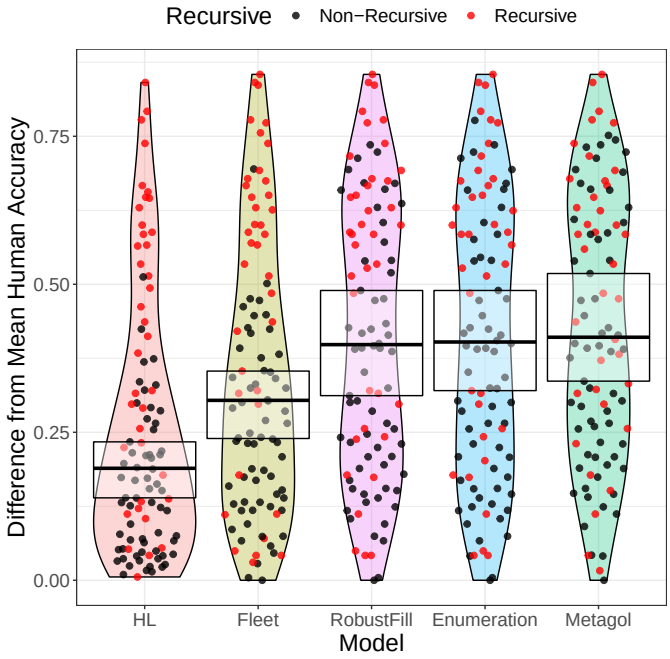


Table 2 & Figure 7: Table: A summary of model performance relative to human learners, including the number of concepts for which each model gives non-zero performance, the number for which it is within 5% of providing the closest prediction of human accuracy, and the number of concepts for which each model's absolute error in predicting human performance is greater than 25%. Figure: Differences between mean human accuracy and mean model accuracy (dots) summarized with a Gaussian kernel density estimate (colored regions). Red dots indicate recursive concepts. The crossbar plots the median with a 95% bootstrapped CI.

key factor in the success of human learning generally.

Conclusion

This dissertation proposes and tests novel hypotheses about the computational underpinnings of human development. It argues that a style of computer programming called hacking plays a critical role in explaining the flexibility and breadth of learning. It builds on prior work suggesting that mental representations function like computer programs. In contrast with this work, the dissertation specifically emphasizes the importance of diversity in representations, objectives, and mechanisms during learning and hypothesizes that these things operate similarly to the structures, values, and activities of hacking. It shows how this child as hacker hypothesis could be used to understand developmental phenomena like the acquisition of declarative theories and mathematical procedures. This dissertation also introduces list functions as a behavioral domain well suited to studying the richness of human learning. Consistent with the child as hacker, it finds that people use observed data to acquire list functions in ways that cannot be predicted from their description length alone. It also reports that a computational model of learning as hacking better accounts for human learning than alternative models of learning in the LOT. These results demonstrate that the child as hacker hypothesis productively contributes to our understanding of development theoretically, empirically, and computationally.

References

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

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*. https://doi.org/10.1371/journal.pcbi.1005273

Baum, E. B. (2004). *What is thought?* MIT Press.

Biermann, A. W. (1978). The inference of regular lisp programs from examples. *IEEE transactions on Systems, Man, and Cybernetics*, *8*(8), 585–600.

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

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

Chater, N., & Vitányi, P. (2003). Simplicity: A unifying principle in cognitive science? *Trends in Cognitive Sciences*, *7*(1), 19–22.

Chu, J., Gauthier, J., Levy, R., Tenenbaum, J., & Schulz, L. (2019). Query-guided visual search. *Proceedings of the 41st Annual Conference of the Cognitive Science Society*.

Cropper, A., Morel, R., & Muggleton, S. H. (2019). Learning higher-order logic programs. *arXiv preprint arXiv:1907.10953*.

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

Feldman, J. (2003). The simplicity principle in human concept learning. *Current Directions in Psychological Science*, *12*(6), 227–232.

Feser, J. K., Chaudhuri, S., & Dillig, I. (2015). Synthesizing data structure transformations from input-output examples. *ACM SIGPLAN Notices*, *50*(6), 229–239.

Flener, P., & Schmid, U. (2008). An introduction to inductive programming. *AI Review*, *29*(1), 45–62.

Fodor, J. (1975). *The Language of Thought*. Harvard University Press.

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

Fowler, M. (2018). *Refactoring: Improving the design of existing code*. Addison-Wesley Professional.

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

Goodman, N., Tenenbaum, J. B., & Gerstenberg, T. (2015). Concepts in a probabilistic language of thought. In E. Margolis & S. Laurence (Eds.), *The conceptual mind: New directions in the study of concepts* (pp. 623–654). MIT Press.

Gopnik, A. (1983). Conceptual and semantic change in scientists and children: Why there are no semantic universals. *Linguistics*, *21*(1), 163–180.

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

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

Green, C. C., Waldinger, R. J., Barstow, D. R., Elschlager, R., Lenat, D. B., McCune, B. R., Shaw, D. E., & Steinberg, L. I. (1974). *Progress report on program-understanding systems* (tech. rep. AIM-240). Stanford Artificial Intelligence Laboratory.

Green, C. (1981). Application of theorem proving to problem solving. *Readings in artificial intelligence* (pp. 202–222). Elsevier.

Gulwani, S., Polozov, O., & Singh, R. (2017). Program synthesis. *Foundations and Trends in Programming Languages*, *4*(1-2), 1–119.

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

Kemp, C., Tenenbaum, J. B., Niyogi, S., & Griffiths, T. L. (2010). A probabilistic model of theory formation. *Cognition*, *114*(2), 165–196.

Kitzelmann, E. (2009). Inductive programming: A survey of program synthesis techniques. *International workshop on approaches and applications of inductive programming*, 50–73.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. MIT Press.

Lafond, D., Lacouture, Y., & Mineau, G. (2007). Complexity minimization in rule-based category learning: Revising the catalog of boolean concepts and evidence for non-minimal rules. *Journal of Mathematical Psychology*, *51*(2), 57–74.

Lake, B., Ullman, T., Tenenbaum, J., & Gershman, S. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*. https://doi.org/10.1017/S0140525X16001837

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

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Lombrozo, T. (2019). "learning by thinking" in science and in everyday life. In A. Levy & P. Godfrey-Smith (Eds.), *The scientific imagination* (pp. 230–249). Oxford University Press.

Lovett, M. C., & Anderson, J. R. (2005). Thinking as a production system. In K. J. Holyoak & R. G. Morrison (Eds.), *The Oxford handbook of thinking and reasoning* (pp. 401–429). Cambridge University Press.

Lupyan, G., & Bergen, B. (2016). How language programs the mind. *Topics in cognitive science*, *8*(2), 408–424.

Newell, A., Shaw, J. C., & Simon, H. A. (1959). Report on a general problem solving program. *IFIP Congress*, *256*, 64.

Newell, A., Shaw, J. C., & Simon, H. A. (1958). Elements of a theory of human problem solving. *Psychological Review*, *65*(3), 151.

Osera, P.-M., & Zdancewic, S. (2015). Type-and-example-directed program synthesis. *ACM SIGPLAN Notices*, *50*(6), 619–630.

Piaget, J. (1955). *The child's construction of reality*. Routledge & Kegan Paul.

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

Piantadosi, S., Tenenbaum, J., & Goodman, N. (2016). The logical primitives of thought: Empirical foundations for compositional cognitive models. *Psychological Review*, *123*(4), 392–424.

Polikarpova, N., Kuraj, I., & Solar-Lezama, A. (2016). Program synthesis from polymorphic refinement types. *ACM SIGPLAN Notices*, *51*(6), 522–538.

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

Rumelhart, D. E., McClelland, J. L., & PDP Research Group. (1987). *Parallel distributed processing*. MIT Press.

Schulz, L. (2012a). Finding new facts; thinking new thoughts. *Advances in child development and behavior* (pp. 269–294). Elsevier.

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

Shaw, D. E., Swartout, W. R., & Green, C. C. (1975). Inferring lisp programs from examples. *IJCAI*, *75*, 260–267.

Siegler, R., & Jenkins, E. (1989). *How children discover new strategies*. Erlbaum.

Siegler, R. S. (1996). *Emerging minds*. Oxford Univesity Press.

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

Smith, D. R. (1984). The synthesis of lisp programs from examples: A survey. In A. W. Biermann, G. Guiho, & Y. Kodratoff (Eds.), *Automatic program construction techniques* (pp. 307–324). Macmillan.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning*. MIT Press.

Turing, A. M. (1936). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, *2*(42), 230–265.

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

Ullman, T. D., Stuhlmüller, A., Goodman, N. D., & Tenenbaum, J. B. (2018). Learning physical parameters from dynamic scenes. *Cognitive psychology*, *104*, 57–82.

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