Learning from Games: Inductive Bias and Bayesian Inference

Michael H. Coen[1,2] and **Yue Gao**[2]
Department of Biostatistics and Medical Informatics[1]
Department of Computer Sciences[2]
University of Wisconsin, Madison, WI 53706
{mhcoen, gao}@cs.wisc.edu

Abstract

A classic problem in understanding human intelligence is determining how people make inductive inferences when presented with small amounts of data. We examine this question in the context of the *guess-the-next-number* game, where players are presented with short series of numbers and asked to guess the next one in the sequence. Our approach is unique in that we use a stochastic context free grammar to model the mathematical operations that generate a given sequence. The individual probabilities in this grammar are learned by observing people play this game, and thereby, they capture some of the mathematical inductive bias of our sample population. We then use this framework to solve novel sequence guessing problems computationally, mirroring human performance. Our goal is to better understand how people approach math problems by examining the space of mathematical functions they find easiest to both generate and recognize. We are also interested in tracking how this changes over time as functions of education and age. Finally, we examine how our results confirm a large body of psychological observations about how people approach mathematics problems.

Keywords: Inductive Bias; Mathematical Modeling; Stochastic Context Free Grammars; Bayesian inference.

Introduction

People regularly make inferences by induction, even when presented with very small amounts of information. The purpose of our work is to understand more about how this occurs in mathematical reasoning, especially when these inferences are remarkably consistent and have little formal justification. We explore this problem in the context of the *guess-the-next-number* game. For example, suppose we present someone with the sequence [1, 2, 4] and ask him to guess the next number in the series. If he were to suggest *8*, stating this set corresponded to "powers of 2," we would presumably find this a plausible explanation. On the other hand, were he to suggest the number is *6*, explaining the sequence seems to be "1 followed by the even numbers," we might find that answer less satisfactory.[1]

Of course, this determination is highly arbitrary. Because inductive inferences are not logically entailed, they must rest upon some set of assumptions, known as an *inductive bias* (Mitchell 1980). In the absence of such a bias, all inferences consistent with a finite dataset are equally valid, as observed by Hume (1739) and formalized by Wolpert (1996). While this is true in a theoretical sense, people surely do have preferences; in other words, it is quite reasonable to suppose that we find some answers more plausible than we do others, even in the absence of objective justifications.

Although there is no doubt that humans generalize using an inductive bias, formally characterizing it can be challenging. For example, even in domains where one believes that Occam's Razor is the principle guiding human induction (Myung and Pitt 1997), the numerous formulations of this classic notion of parsimony may lead to diametrically opposed conclusions. (For an interesting discussion of this issue in elucidating the innate mathematical abilities of infants, see Carey 2002).

In this paper, we explore a domain where we can formally model inductive bias using probabilities in a stochastic context free grammar, which appear to capture how people do math "in their heads." Because our notion of inductive bias is rigorously defined, we can also trace how it changes over time. Thus, we expect grammar school, high school, and college students to have very different inductive biases, which can of course be further characterized by their fields of study. We are also interested in tracking how these mathematical biases change as people age. More generally, this work provides a window in the types of mathematical operations with which people are most comfortable as well as the types of operations that perhaps require more careful instruction.

(a) 1, 8, 27, ? Guess is **64**.
Why? This is the sequence x^3.

(b) 2, 3, 5, 7, ? Guess is **11**.
Why? These are the prime numbers.

(c) 1, 2, 4, ?

- **6**, *n* = prime numbers – 1
  1, 2, 4, **6**, 10, 12, 16, ...
- **7**, $n = i(i + 1)/2 + 1$
  1, 2, 4, **7**, 11, 16, 22, ...
- **8**, $n = 2^i$
  1, 2, 4, **8**, 16, 32, 64, ...
- **9**, *n = partial sums of Catalan numbers*
  1, 2, 4, **9**, 23, 65, 197, ...

Figure 1. Examining the guess-the-next-number game. While the examples in (a) and (b) are fairly straightforward, the sequence in (c) is somewhat more ambiguous.

[1] We will see, however, that appeals to minimum description length (e.g., Rissanen 1978) do not capture human preferences in making these judgments.

The Next Number Game

We examine our use of the next number game and some of the ambiguities it presents to players. This game is generally familiar to people around the world and is played by both children and adults. In Figure 1, we examine several instances of this game and begin to understand why specific solutions might be subject to debate.

When the sequences correspond to familiar concepts, such as the *prime numbers* or *perfect cubes*, players already familiar with these concepts are likely to latch onto them. In that sense, this tells us far more about what predetermined sets of numbers a player is already familiar with than it does about his mathematical preferences. This was the approach pursued in Tenenbaum (1999), where he presented predetermined, unordered sets to players and asked them to rank how likely it was some other number belonged to each set. He was able to predict their rankings using a clever variant of minimum description length. However, his framework did not generalize to simple unknown sets, in other words, data it had not been trained on, such as *prime numbers – 1*. We wanted to create a more generative framework that could predict arbitrary sequences and simultaneously provide insight into the human mathematical reasoning process, in a spirit similar to (Leslie et al. 2008). We also wanted to create a platform that enables tracking how these identified mathematical biases change over time.

The problem of examining inductive bias in fitting data to closed form equations has been studied by Haverity and Koedinger (2000). While they did not computationally model the reasoning process behind their subjects' performance, our results are in strong agreement with their observations and previous psychological studies, such as Huesmann and Cheng (1973).

We note that Sloane (2008) maintains a delightful, online encyclopedic database of integer sequences. While both highly esoteric and thorough, it is simply a lookup table of series submitted by users. Many of its sequences are generated by physical and mathematical processes that have no obvious closed form generative formulae. Thus, it is not relevant to our interests here, as it provides little, if any, information about human mathematical cognition.

A Generative Framework

We constructed a generative framework for mathematical expressions using a stochastic context free grammar (SCFG). This allows us to capture the notions of "function recognition" and "function creation" in a well-defined framework. This approach for recognition has become extremely popular for parsing in the Statistical Natural Language Processing community (e.g., Collins 2003), and seemed extremely well-suited for expressing mathematical operations. However, a basic distinction is that our grammars output mathematical expressions, which are then evaluated to produce numerical values.

Our grammar is shown in Figure 2. It defines the notion of an *expression*, which is a function that generates the elements of a sequence. This function generating each element may be based upon its position in the sequence (represented by "Index"), the immediately prior numbers in the sequence (represented by "Previous_{i-k}"), or perhaps even a constant (represented by "Number"). The generating function is constructed out of both prefix and infix operations ("PrefixOp" and "InfixOp" respectively). A prefix operator takes a single argument, such as is the case with *log*, while an infix operator takes two arguments, such as when performing addition or subtraction. Constants are divided into three categories: (1) Small numbers, which we assume are easier to process cognitively; (2) Large number that are presumably more difficult to involve in mental arithmetic; and (3) Special numbers such as 10 or π, that simplify many types of operations or have some other special significance. For example, trigonometric operations on simple functions of π will be very familiar to many university students. One might in fact formulate the grammar to "confine" the use of π to trigonometric functions. However, as reasonable as this appears, it complicates the grammar and eliminates other functions of π that are not unreasonable, e.g., Index + π. Fortunately, this type of simplification is unnecessary, as we do not expect that people are able to play this game with arbitrarily complex generating functions. Thus, the assumption that people *can* play this game eliminates concerns about pathologically complex functions that few (or no) people could ever recognize. We return to the notion of function

Expression → PrefixOp (Expression) p_1
Expression → Expression InfixOp Expression p_2
Expression → Previous_{i-1} p_3 | Previous_{i-2} p_4 | Previous_{i-3} p_5
Expression → Number p_6
Expression → Index p_7
PrefixOp → exp p_8 | log p_9 | sin p_10 | cos p_11 | tan p_12
PrefixOp → floor p_13 | ceiling p_14 | mod p_15 | rem p_16 | prime p_17
InfixOp → + p_18 | – p_19 | × p_20 | ÷ p_21 | ^ p_22
Number → SmallNum | LargeNum | SpecialNum
SmallNum → [-9 p_26,…,9 p_45]
LargeNum → [-50,…, -11, 11, …, 50] p_46
SpecialNum → -100 p_47 | -10 p_48 | ¼ p_49 | ½ p_50
SpecialNum → π p_51 | 10 p_52 | 100 p_53
Index → [1,…,10] p_54

Figure 2. Our stochastic context free grammar for generating mathematical functions. Non-terminals begin with capital letters. Terminals symbols are indicated by lower-case strings or numbers. Each production rule is associated with some probability *p_i*, indicating its likelihood according to our training corpus of people playing this game. Players of the *guess the next number* game provide both their answers and the function they think generated it. Our goal is to use this corpus of games to derive the probabilities *p_i* in order to both discover and to duplicate human inductive bias playing this game.

complexity and cognitive plausibility below.

Each production rule in this SCFG is associated with some probability, represented by the superscripted *p_i* following it. This probability represents the observed likelihood our sample population has employed this productive rule while playing a series of games. To reduce the amount of training data required, we "lumped together" certain groups, such as large numbers, represented by the *LargeNum* production rule. While we assume any of these numbers may appear in the generating formula, gathering data for each individual number would require that each subject play a very large of games. Furthermore, given the natural variability in determining the function generating a given sequence, there is no way to guarantee subjects would employ a specific number. In other words, it can be difficult to "force" subjects to employ particular production rules. Ad hoc sequences, such as using a constant difference between successive elements to "lead" players to each number, e.g., for 11, we might use [1, 12, 33, 44, …], would simply generate a uniform distribution among the members of *LargeNum*, which is what we were seeking to avoid in the first place. Thus, we avoid the problem entirely and make certain classes of number probabilistically equivalent.

Finding the Probabilities

To determine the probabilities *p_i* for each production rule, we first collected data from 20 university undergraduates with nonmathematical backgrounds[2], who played up to 22 different rounds of the guess-the-next-number game over ascending and descending sequences. The students were asked both to guess the next number in a displayed sequence and to provide the formula generating it, in light of the presented sequence. There was no imposed time limit and a student was free to skip a sequence if he could not solve it. Our goal for this experiment was to learn the mathematical inductive biases of college-aged non-mathematicians. We examine three sample sequences from this experiment.

(1) Consider the sequence [1, 4, 9]. We found that all subjects predicted the next number would be *16*, but provided two syntactically different but numerically equivalent generating formulae, at least up to index 4.

- 40% guessed: $f(index) = index^2$
- 60% guessed: $f(index) = Previous_{index-1} + 2 \times index + 1$

For this example, we see that most subjects preferred conceptually simple arithmetic operations, even if the resulting functional description was longer and bordered on being convoluted. We found this type of result quite surprising, as it would not have occurred to us this sequence would be identified as anything other than *perfect squares*.

[2] Although one can debate our selection criterion, we eliminated students majoring in mathematics, computer science, or physics.

Table 1. Examining the Bayesian probabilities of some production rules in our generative mathematical grammar. The probabilities are determined for each non-terminal rule in our grammar separately.

Production Rule	Probability
Expression → PrefixOp (Expression)	0.00402
Expression → Expression InfixOp Expression	0.349
Expression → $Previous_{i-1}$	0.177
Expression → $Previous_{i-2}$	0.0321
Expression → Number	0.317
Expression → Index	0.104
InfixOp → +	0.388
InfixOp → −	0.143
InfixOp → ×	0.263
InfixOp → ÷	0.0388
InfixOp → ^	0.163
SmallNum → -1	0.04
SmallNum → 1	0.24
SmallNum → 2	0.40
SmallNum → 3	0.08
SmallNum → 4	0.04
…	…

(2) Consider the sequence [1,2,10]. That this sequence is in some sense more difficult was apparent because subjects spent more time studying it, often commenting it felt "difficult" or under constrained.

- All but one: $f(index) = Previous_{index-1} + (index - 1)^3$
  Yielding: [1, 2, 10, 37, 101, …]
- One guess: $f(index) = Previous_{index-1} + (Previous_{index-1})^3$
  Yielding: [1, 2, 10, 1010, 1.0303×10^9,…]

Note that although these guesses lead to different predictions of subsequent sequence values, they are structurally quite similar. From our Bayesian perspective, they will lead to very similar priors in the grammar. There are a multitude of other functions that were not selected by our sample population, e.g., $f(index) = Previous_{index-1} + 8^{(index-1)}$, presumably reflecting a distaste for this level of complexity.[3]

(3) Finally, we examine the sequence [0,7,26], where all the subjects agreed on the next element (*63*) and on the generating formula:

- $f(index) = index^3 - 1$

[3] This formula and similar variants were provided by several graduate students in Computer Science, who were also asked to solve this problem. It comes as little surprise they have very different inductive biases for playing this game.

Here, alternative explanations avoiding a difficult operation, such as exponentiation, are so complex that they are disregarded. We explore the measure of functional complexity in our system below.

To derive individual production probabilities from all presented sequences, we used the inside-outside algorithm (Baker 1979). Specifically, we employed a Gibbs sampler for SCFGs developed by (Johnson et al. 2007), which derived Bayesian priors for the production rules based on the formulas generated by the subjects using Markov chain Monte Carlo methods. Finally, we had to perform a renormalization of the infix operators (*InfixOp*) probabilities to account for commutative operations such as × and +. This is due to the fact that the Viterbi algorithm (see the next section) has no way of realizing, for example, that $a+b$ is equal to $b+a$ and thereby undercounts its likelihood. (It computes a single path to the answer, without realizing there are numerically equivalent ones that are syntactically different in trivial ways.)

We examine some of the more interesting results of this derivation in Table 1. Among the most significant but unsurprising findings is that people do not like performing division in their heads; it represents 3.88% of infix operations. In contrast, they are ten times more likely to prefer addition, which represents 38.8% of infix operations. Our subjects also preferred transforming non-terminal expressions into concrete numbers rather quickly, as opposed to developing complex expressions. As might be expected, 1 and 2 were clearly the most popular numbers for mental arithmetic. (Note that 0 does not appear, as it has a probability of zero; this is because no one used it to solve any sequence problems. This makes sense, as it contributes nothing given the mathematical operations presented here. In other words, there is no reason to employ it.)

How much does the selection of sequences itself bias these results? For example, were we to only present ascending series, there would be an innate bias in favor of monotone functions such as addition and multiplication, at least with the ontology presented here. Thus, we made an effort to balance the sequences to remove obvious sources of such bias. However, it should be noted that generating representative sequences of three to five numbers that are amenable to human solution is non-trivial. One cannot simply produce them randomly, e.g., by typing `round(rand(1,4)*10))` in Matlab, and expect to produce a sequence that holds any meaning or lends itself to an obvious generative formula. Thus, while there are an infinite number of "solvable" sequences, they are somewhat sparsely distributed and must be selected with some care.

We therefore generated a large list of sequences and randomly selected from among them those solvable upon inspection. However, one of the benefits of our constructed generative framework is that we can use it to automatically generate sequences for future experimentation that capture the types of operations people prefer. In this sense, we can use our grammar to produce rather than recognize sequences. This helps insure that new sequences do not violate our learned inductive biases, at least for retesting this population on larger sequence corpora or comparing their biases with that of another target population, e.g., math majors.

Processing the Sequences

After deriving the probabilities for our generative SCFG via Gibbs sampling, we encoded the grammar in Prism (Sato and Kameya 2008), a probabilistic version of Prolog that requires parameterized probability distributions over its production rules. Prism's inference engine incorporates the Viterbi algorithm (Forney 1973). Therefore, its resolution is guided by following the most likely series of rule expansions to satisfy a given query. One can view Prism as an extension of Prolog that provides the most probable solution to a given query. Queries here corresponded to the question: given an input sequence, what is the most likely next number according to our SCFG? The process of determining the next number generates the function responsible for doing so via Prolog's resolution mechanism.

Because our recursive grammar is computing mathematical functions, as opposed to parsing a sentence, it will never "run out" of input data in resolving this query. Instead, it would continue down the most likely mathematical path forever, constantly generating ever more complex expressions. Because the Viterbi algorithm needs to hit a leaf to trigger backtracking, resolution would never halt in our framework. We therefore add an explicit stopping criterion, using the probabilities in the SCFG to determine the overall probability of any expression examined in the course of resolution. By the definition of a context free grammar, the probabilities are independent, so we can simply multiply all non-terminal probabilities to determine the value for a given expression.[4] If this probability falls below a predefined threshold, our system automatically triggers backtracking, essentially ruling the current line of investigation as too complex to be plausible. This threshold can be computed directly from the gathered corpus of human responses.

We view this as a probabilistic version of working memory, as defined in (Miller 1956). We believe that preferred operations are easier to cognitively track, whereas less likely (or more difficult) operations have a greater impact in limiting the size of the overall expression. Thus, we are not explicitly modeling the expression size. Rather, the probability threshold implicitly limits the complexity of the internal mathematical computation. This seems cognitively reasonable and agrees with our observed results.

Results

We now examine some sample outputs of our system. They demonstrate how its behavior changed after acquiring the inductive bias of the observed population and show how it

[4] For the sake of efficiency, the overall probability is adjusted dynamically during rule expansions and backtracking.

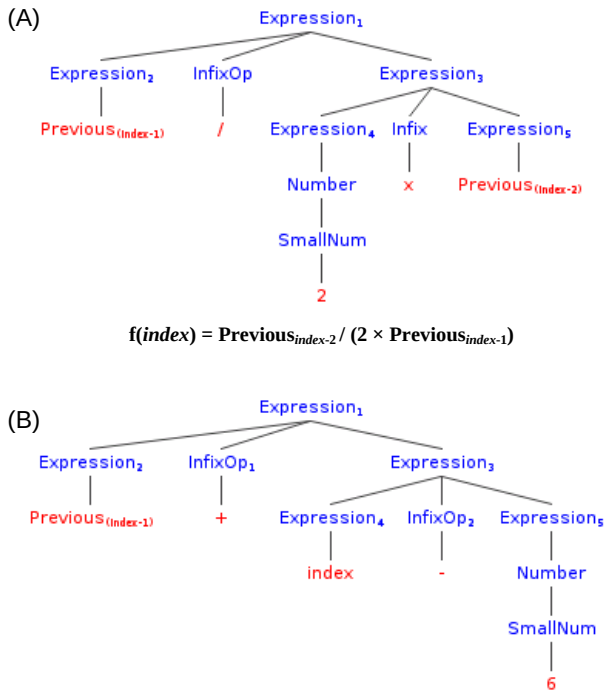

Figure 3. Our system's output on the sequence [8,4,1], *before* training (A) and *after* training (B). Without any inductive bias, the system predicts the explanation in (A), as we assume a uniform distribution over each production rule in the absence of any bias. After deriving the production probabilities via Gibbs sampling using the inside-outside algorithm, our system acquires the priors representing the inductive biases of our sample population. It then changes its answer to (B), agreeing with 88% of human subjects, even though it has never seen this sequence before. It now assigns the answer in (B) more than ten times the probability as the answer in (A).

generalizes to handle out-of-set examples, namely, sequences it has never seen before. We then discuss implications of this work, particularly what it reveals about the capabilities for performing mental arithmetic in people.

The Effects of Learning

We presented subjects with the sequence [8, 4, 1]. The test subjects overwhelming (88%) guessed the generating function was f(*index*) = Previous$_{index\text{-}1}$ + *index* – 6. Before training, our sequence guesser, using uniform distributions on its production rules, predicted the function was:

$$f(index) = Previous_{index\text{-}2} / (2 \times Previous_{index\text{-}1})$$

However, after training on examples that *did not* include this sequence, our system changed its answer to agree with the solution provided by the vast majority of human subjects on this problem. The expansion of these formulae in terms of our grammar is displayed in Figure 3. We note the acquisition of human inductive bias now leads it to predict the human answer is more than an order of magnitude more likely than the system's original, untrained solution to this problem. This change is due to the system having learned not only that division is less likely but also to its preference for using small numbers over more distant terms (Previous$_{index\text{–}2}$) in the sequence's generating formula.

Modifying Familiar Sets

We now look at an example that demonstrates the benefits of not restricting our approach to a predetermined set of sequences. Put somewhat differently, we can see the power of a generative inductive framework in examining how it copes with functional transformations to familiar sequences, such as the prime numbers. We presented our system with the sequence [1,2,4,6], which was not part of its training data. It predicted the generating function was f(*index*)=Prime(*index*) – 1 with a confidence level of 91% based on its acquired inductive probabilities, which agrees with human subjects. We see the second most likely candidate for this sequence in Figure 4, which has a probability of approximately 1%.[5] Note that the most likely

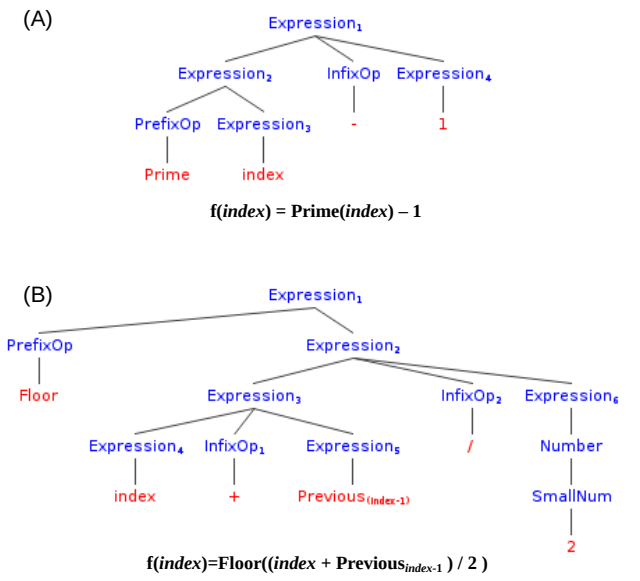

Figure 4. Generated solutions to the sequence [1, 2, 4, 6]. The solution in (A) has a probability of 91%. The second most likely solution, show in (B), is determined to be only 1% likely. The example demonstrates the need for a generative mathematical framework for playing the guess-the-next-number game, as opposed to enumerating huge numbers of predetermined training sets. Many familiar sets are easily recognizable under various simple functional transformations, e.g., subtracting one from them.

[5] Final probabilities for expressions are determined by generating *all* possible explanatory functions within our threshold, calculating their probabilities according to the SCFG, and then normalizing these into a probability distribution. In the event we simply want the most likely expression, it is unnecessary to enumerate every generating function.

solution is simply a straightforward modification to a familiar set. There is any number of modifications to such sets that are recognizable, where the likelihood of recognition depends upon the complexity of the applied operations. The advantage of working within a generative framework is clear from this example, as opposed to exhaustively listing sets that might be encountered and then defining a metric that attempts to compute human preferences among them.

Conclusions

This paper has presented a framework for solving the guess-the-next-number game that is based upon acquiring a realistic model of human inductive bias. As these brief sequence problems are highly unconstrained and yet different people often arrive at identical results, we find the hypothesis that there are innate cognitive preferences guiding mathematical reasoning extremely reasonable. While these may vary by particular age groups and educational background, our preliminary results agree with previous psychological studies of induction in mathematical problem solving, notably including the work of (Huesmann and Cheng, 1973, Gerwin and Newsted 1977, Qin and Simon 1990). Specifically, we verified that people have clear preferences among operators and their formulation of generative functions is very much driven by the underlying data. Our results also agreed with people having a clear preference for linear functions, in cases where the data make them possible.

As part of this work, we constructed a system that acquires mathematical inductive biases observed in our sample population. In doing so, it is able to imitate their problem solving, even in cases where it must ignore more compact functions because they are mathematically complex according to the acquired bias. It thus employs a different notion of simplicity than would be described by formalizations of generative brevity.

Our future plans are to test different age groups to track the temporal development of their mathematical inductive biases. We are particularly interested in bias invariants that persist over time and in educational strategies that may be suggested by elucidating limitations in how people approach mathematical problem solving.

We also believe the framework in this paper is quite general and can acquire inductive biases in a wide variety of areas that have similar probabilistic generative structure. We intend to employ it for modeling and predicting human behavior in these realms. Here, the primary challenge will be modeling actions or decisions via the SCFG formalism, so we may derive their probabilities through observation using the Bayesian framework presented here.

Acknowledgments

This work was supported by the School of Medicine and Public Health, the Wisconsin Alumni Research Foundation, the Department of Biostatistics and Medical Informatics, and the Department of Computer Sciences at the University of Wisconsin-Madison. Thanks to W. Richards, C. Dyer, and M.H. Ansari for helpful comments.

References

Baxter, J. A. (2000). Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research*. 149(12).

Carey, S. (2002). Evidence for numerical abilities in young infants: a fatal flaw? *Developmental Science*, 5(2), pp202-205.

Collins, M. (2003). Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics* 29:4, pp589-637.

Forney, G. D. (1973). The Viterbi algorithm. *Proceedings of the IEEE*. 61(3), pp268-278, March.

Gerwin, D and Newsted, E. (1977). A comparison of some inductive inference models. *Behavioral Science*. 22:1-11.

Haverity, L.A., and Koedinger, K.R. (2000). Solving inductive reasoning problems in Mathematics: Not-so-Trivial Pursuits. *Cognitive Science: A multidisciplinary Journal*. 24(2), pp249-298.

Huesmann, L.R. and Cheng, C. (1973). A model for the induction of mathematical functions. *Psychological Review*. 80, pp126-138.

Hume, D. (1739). A Treatise of Human Nature. (eds.) Norton, D.F., and Norton, M.J. Oxford University Press. New York. 2000.

Johnson, M., Griffiths, T.L., and Goldwater, S. (2007) Bayesian Inference for PCFGs via Markov Chain Monte Carlo. *Proceedings of the 2007 Conference of the North American Chapter of the Association for Computational Linguistics*. pp139-146.

Leslie, A.M., Gelman, R., and Gallistel, C.R. (2008). The generative basis of natural number concepts. *Trends in cognitive sciences*. 12(6), pp213-218.

Miller, G.A. (1956). The Magical number seven, plus or minus two: some limits on our capacity of processing information. *The Psychological Review*. 63(2).

Mitchell, T. (1980). The need for biases in learning generalizations. Technical Report CBM-TR-117, Department of Computer Science, Rutgers University.

Myung, I. and Pitt, M. (1997). Applying Occam's Razor in Modeling Cognition: A Bayesian approach. *Psychonomic Bulletin & review*. 4(1), pp79-95.

Qin, Y., and Simon, H.A. (1990). Imagery and problem solving. *Proceedings of the 12th Annual Conference of the Cognitive Science Society*. pp646-65.

Rissanen, J. (1978) Modeling by the shortest data description. *Automatica 14*, pp465-471.

Sato, T. and Kameya, Y. (2008). New advances in logic-based probabilistic modeling by PRISM. In Probabilistic Inductive Logic Programming, LNCS 4911, Springer, pp118-155.

Sloane, N. J. A. (2008). The On-Line Encyclopedia of Integer Sequences. Electronically published at: www.research.att.com/~njas/sequences/.

Tenenbaum, J.B. (1999) A Bayesian Framework for Concept Learning. Ph.D. Thesis, Massachusetts Institute of Technology.

Wolpert, D. H. (1996). The lack of *a priori* distinctions between learning algorithms. *Neural Computation*, 8(7), pp1341-1390.