Learning Structured Generative Concepts

Andreas Stuhlmüller, Joshua B. Tenenbaum, Noah D. Goodman
Brain and Cognitive Sciences, MIT
{*ast, jbt, ndg*}*@mit.edu*

Abstract

Many real world concepts, such as "car", "house", and "tree", are more than simply a collection of features. These objects are richly structured, defined in terms of systems of relations, subparts, and recursive embeddings. We describe an approach to concept representation and learning that attempts to capture such structured objects. This approach builds on recent probabilistic approaches, viewing concepts as generative processes, and on recent rule-based approaches, constructing concepts inductively from a language of thought. Concepts are modeled as probabilistic programs that describe generative processes; these programs are described in a compositional language. In an exploratory concept learning experiment, we investigate human learning from sets of tree-like objects generated by processes that vary in their abstract structure, from simple prototypes to complex recursions. We compare human categorization judgements to predictions of the true generative process as well as a variety of exemplar-based heuristics.

Introduction

Concept learning has traditionally been studied in the context of relatively unstructured objects that can be described as collections of features. Learning and categorization can be understood formally as problems of statistical inference, and a number of successful accounts of concept learning can be viewed in terms of probabilistic models defined over different ways to represent structure in feature sets, such as prototypes, exemplars, or logical rules (Anderson, 1990; Shi, Feldman, & Griffiths, 2008; Goodman, Tenenbaum, Feldman, & Griffiths, 2008). Yet for many real world object concepts, such as "car", "house", "tree, or "human body", instances are more than simply a collection of features. These objects are richly structured, defined in terms of features connected in systems of relations, parts and subparts at multiple scales of abstraction, and even recursive embedding (Markman, 1999). A tree has branches coming out of a trunk, with roots in the ground; branches give rise to smaller branches, and there are leaves at the end of the branches. A human body has a head on top of a torso; arms and legs come out of the torso, with arms ending in hands, made of fingers. A house is composed of walls, roofs, doors, and other parts arranged in characteristic functional and spatial relations that are harder to verbalize but still easy to recognize and reason about. Besides objects, examples of structured concepts can be found in language (e.g. the mutually recursive system of phrase types in a grammar), in the representation of events (e.g. a soccer match with its fixed subparts), and processes (e.g. the recipe for making a pancake with steps at different levels of abstraction).

Such concepts have not been the focus of research in the probabilistic modeling tradition. Here we describe an approach to representing structured concepts—more typical of the complexity of real world categories—using probabilistic generative processes. We test whether statistical inference with these generative processes can account for how people categorize novel instances of structured concepts and compare with more heuristic, exemplar-based approaches.

Because a structured concept like "house" has no single, simple perceptual prototype that is similar to all examples, learning such a concept might seem very difficult. However, each example of a structured concept itself has internal structure which makes it potentially very informative. Consider figure 1, where from only a few observations of a concept it is easy to see the underlying structural regularity that can be extended to new items. The regularities underlying structured concepts can often be expressed with instructions for *generating* the examples: "Draw a sequence of brown dots, choose a branch color, and for each brown dot draw two dots of this color branching from it."

Figure 1: Three examples of a structured concept described by a simple generative process.

We build on the work of Goodman, Tenenbaum, et al. (2008), who introduced an approach to concept learning as Bayesian inference over a grammatically structured hypothesis space—a "language of thought." Single concepts expressed in this language were simple propositional rules for classifying objects, but this approach naturally extends to richer representations, providing a concept learning theory for any representation language. Here we consider a language for generative processes based on *probabilistic programs*: instructions for constructing objects, which may include probabilistic choices, thus describing distributions on objects—in our case distributions on colored trees. Because this language describes generative processes as programs, it captures regularities as abstract as subparts and recursion.

The theory of concept representation that we describe here shares many aspects with previous approaches to concepts. Like prototype and mixture models (Anderson, 1990; Griffiths, Canini, & Sanborn, 2007), probabilistic programs describe distributions on observations. However, prototypes and mixtures generate observations as noisy copies of ideal prototypes for the concept and thus cannot capture more abstract structures such as recursion. Like rule-based models of concept learning, our approach supports compositionality: complex concepts are composed out of simple ones—but rather

than deterministic rules, our concepts denote distributions. Finally, the probabilistic program approach can be seen as a generalization of previous approaches to generative representations of concepts (Kemp, Bernstein, & Tenenbaum, 2005; Rehder & Kim, 2006; Feldman, 1997).

We investigate human learning for classes of generating processes that vary in their abstract structure, from simple prototypes to complex multiply recursive programs. We compare predictions for categorization judgments based on the true generative model to the predictions of exemplar models, which exploit the relational structure of the examples to varying degrees but cannot detect more abstract structure. We find two regimes: for concepts with simple prototype-like structures, human judgements are well described by a relational exemplar model, but humans can also easily learn more abstract regularities—such as sub-concepts and recursion—which are better captured by a model using more expressive generative descriptions based on probabilistic programs.

Formal Framework

In the following, we first explain the formal language we use to describe generative processes, then the different methods of categorization (or generalization) we compare to subjects' judgements.

Concept Representation

We analyze concepts as generative models, i.e. as formal descriptions of processes that generate observations. We do so within a simple domain where we can fully know and manipulate the actual generating processes behind complex objects. We use tree-structured graphs with colored nodes as observations in our experiments—these are a simple proxy for many real-world concepts, where the dependencies among parts are hierarchical or tree-like. Human bodies, buildings, and events all consist of parts that themselves contain parts, with each part standing in interesting relation to the others.

We represent these trees as nested lists: each list denotes a tree, with the first element in the list specifying the color of the root node and the remaining elements describing the children of this node, each child itself being a list (tree). For example, the second tree shown in figure 1 can be represented as `'(● (●) (● (●) (●)) (●))`.

We formalize the processes that generate these observations using a subset of Church, a Lisp-like stochastic programming language[1] (Goodman, Mansinghka, Roy, Bonawitz, & Tenenbaum, 2008). Programs in Church describe processes that produce values; running a program corresponds to generating a value from such a process. Because Church contains primitive functions that randomly choose from a distribution on values (e.g. the function `flip` that randomly chooses `true` or `false`), Church programs describe *stochastic* processes. The meaning of a Church program is a distribution on return values—which may be complex values such as nested lists—and any given execution results in a sample from this distribution. In what follows we describe Church programs which sample colored trees.

We group generative models into classes by the abstract constructions they use. Table 1 illustrates each of these types using a single concept program and observations drawn from this program. The simplest tree-generating processes in our language use only the stochastic function `node`, which takes as its first argument a color symbol and as its remaining arguments subtrees. With high probability, `node` returns a tree that has the given color symbol at its root and the given subtrees as its children, but with some probability ε, it switches to a noise process that can return any tree, that is, `node` introduces a random noise process into the tree construction. Under the noise process, the number of children for a node is sampled from a geometric distribution with parameter ε and the node color is sampled uniformly.

Programs like `(node ● (node ●) (node ●))` denote stochastic *prototypes*. They are most likely to generate the tree that corresponds to the given colors, in this case `'(● (●) (●))`, but they can return any tree with a certain probability. The more a tree deviates from the prototype, the less likely this process is to generate it. For example, the simple program described above could switch at the third `node` to the noise process and produce `'(● (●) (● (●)))` instead of the prototype. By introducing the noise process, `node` turns a deterministic prototype into a stochastic process.

All of the more abstract ways of formalizing generative models in our tree domain compose these basic processes. *Nested prototypes* formalize the intuition that a concept or a part of a concept can be "either this or that". Running the program `(if (flip .5) (node ●) (node ●))` will flip a fair coin and return a sample from `(node ●)` with probability .5, otherwise a sample from `(node ●)`.

One of the central reasons for analyzing concepts as represented in a language of thought is that they compose analogously to the components of natural and artificial languages—*parts* similarly allow composition through reuse in our domain. A part concept is defined first and can then be used in arbitrarily many places within other concepts. For example, the program `(define (part) (node ● (node ●)))` names a simple part consisting of only two nodes. This part can now be reused in other concepts. For example, the most likely return value for `(node ● (part) (part))` is `'(● (● (●)) (● (●)))`. When parts are defined, they are available to the noise process. This leads to some invariance to the position of parts and captures the idea that a generating process may give rise to observations that contain a part in a different place, although with lower probability compared to an observation with the part in the correct place.

Parameterized parts can capture both deterministic structure and random choices and reuse them in multiple places. When a part like `(define (part x) (node ● x x))` is used, for example in the program `(part (node ●))`, it evaluates

[1] Church uses prefix notation, i.e. function application is written with the operator first, the operands following. For example, `(node x y)` means that the function `node` is called with the arguments `x` and `y`.

the body of the part—here `(node • x x)`—with `x` assigned to its argument, here `(node •)`. Evaluating the program `(part (node •))` is therefore most likely to result in the observation `'(• (•) (•))`.

Allowing parts to call themselves introduces *recursion*, a means to capture a large amount of repetitive observed structure in a single short definition. For example, the part `(define (p) (if (flip) (node •) (node • (p))))` can generate arbitrarily deep lists of single blue nodes, with shorter ones being more likely.

The power of these program constructs is that they may be used compositionally to build more complex concepts, such as those shown in table 1.

Categorization

In order to model generalization and categorization behavior of human subjects, we need not only a way to represent concepts, but also a way to compute the probability of any given observation belonging to a known concept. We analyze our experimental results using four models that differ in how much they make use of representational structure.

On the unstructured end of the scale, we use a model that computes generalization judgements solely by comparing the fraction of nodes that have a given color. On the other end of the scale, a generative Bayesian model uses the likelihood under the true generative process to judge category membership. In between, an exemplar model makes use of tree structure in the observations, but not of the more abstract generative process that led to the observations.

Generative Model In modeling concept learning as Bayesian program induction, we follow the approach taken by Goodman, Tenenbaum, et al. (2008). Since we formalize concepts as probabilistic programs, the likelihood $P(O|C)$ of an observation O under a given concept C corresponds to the probability of the program making its random choices such that it returns the observation as its value (see Goodman, Mansinghka, et al. (2008)). The posterior probability of a concept C given observations O is proportional to this likelihood multiplied by the prior:

$$P(C|O) \propto P(O|C)P(C) \tag{1}$$

In the last section, we described a language for programs which generate trees; a prior $P(C)$ could be derived from this language, as in Goodman, Tenenbaum, et al. (2008). An ideal learner would then infer the posterior distribution $P(\mathbf{C}|O)$ over concepts $\mathbf{C}$ given the observation O and make predictions about whether a new observation t belongs to the category of the observed objects using each concept $C \in \mathbf{C}$ in proportion to its posterior probability:

$$P(t|O) \propto \sum_{C} P(t|C)P(C|O) \tag{2}$$

In order to make computational modeling tractable, we make the simplifying assumptions that (1) subjects' reasoning is dominated by the maximum a posteriori (MAP) estimate of this distribution, i.e. by the single concept that has the highest posterior probability and that (2) the true generating concept C_{true} is a good approximation to the MAP estimate. Thus, for each of the concept types we investigate, we model subjects' behavior using the program from which the training data was sampled. The likelihood of a new observation t belonging to this concept is simply $P(t|C_{true})$ which we compute using an adaptive importance sampling algorithm.

We do not claim that subjects necessarily identify the true generating concept from a few examples; this approximation is made for computational tractability. The full Bayesian model, which maintains uncertainty over generating concepts, can make different predictions in certain cases, but it is not clear whether this represents a bias for or against the approximation—to the extent that people remain uncertain of the concept after a few examples, the Bayesian model would capture human inferences better than our approximation.

Tree Exemplar Model This and the next two models are versions of the exemplar-based generalized context model (GCM) (Nosofsky, 1986). For observations $O_1, \ldots, O_n$ from category C and a new observation t for which we would like to estimate the likelihood under category C, we use $P(t \in C|O_1, \ldots, O_n \in C) \propto \frac{1}{n}\sum_{i=1}^{n} e^{-d(O_i,t)}$ where d is a distance measure that is sensitive to the tree structure of the observations. Starting from the root node, this measure matches the trees as much as possible, incrementing by 1 for each node that differs in color between the two trees and for each node that must be generated because it exists in one tree but not in the other tree. This approach is similar to the structure mapping approach used by Tomlinson and Love (2006).

Frequency-based Exemplar Models As in the tree exemplar model, we use a distance measure d to estimate the likelihood of an observation belonging to a category for which we have only positive examples. In this version of the model, $d(t_1, t_2)$ is the RMSE between the transition count vectors of t_1 and t_2. For each pair of node colors, the transition count vector contains the number of times this pair occurs adjacent (as parent-child) in the given tree. We call this model *Transition GCM*. We also investigate a simplified version that uses the distance between the color count vectors. The length of this vector corresponds to the number of possible node colors, with each entry in the vector denoting how often this node color appears in the tree of interest. We call this *Set GCM*.

Experiment

This experiment is an exploratory investigation into generalization from observations of structured objects. Since our main goal in this study is to investigate the representation of concepts and their use for categorization and generalization rather than the memory aspects of learning, we use a paradigm that minimizes memory demands. By doing so, we hope to focus on how people represent the commonalities between observed instances of a concept and how they use this knowledge to generalize to new instances. We chose a

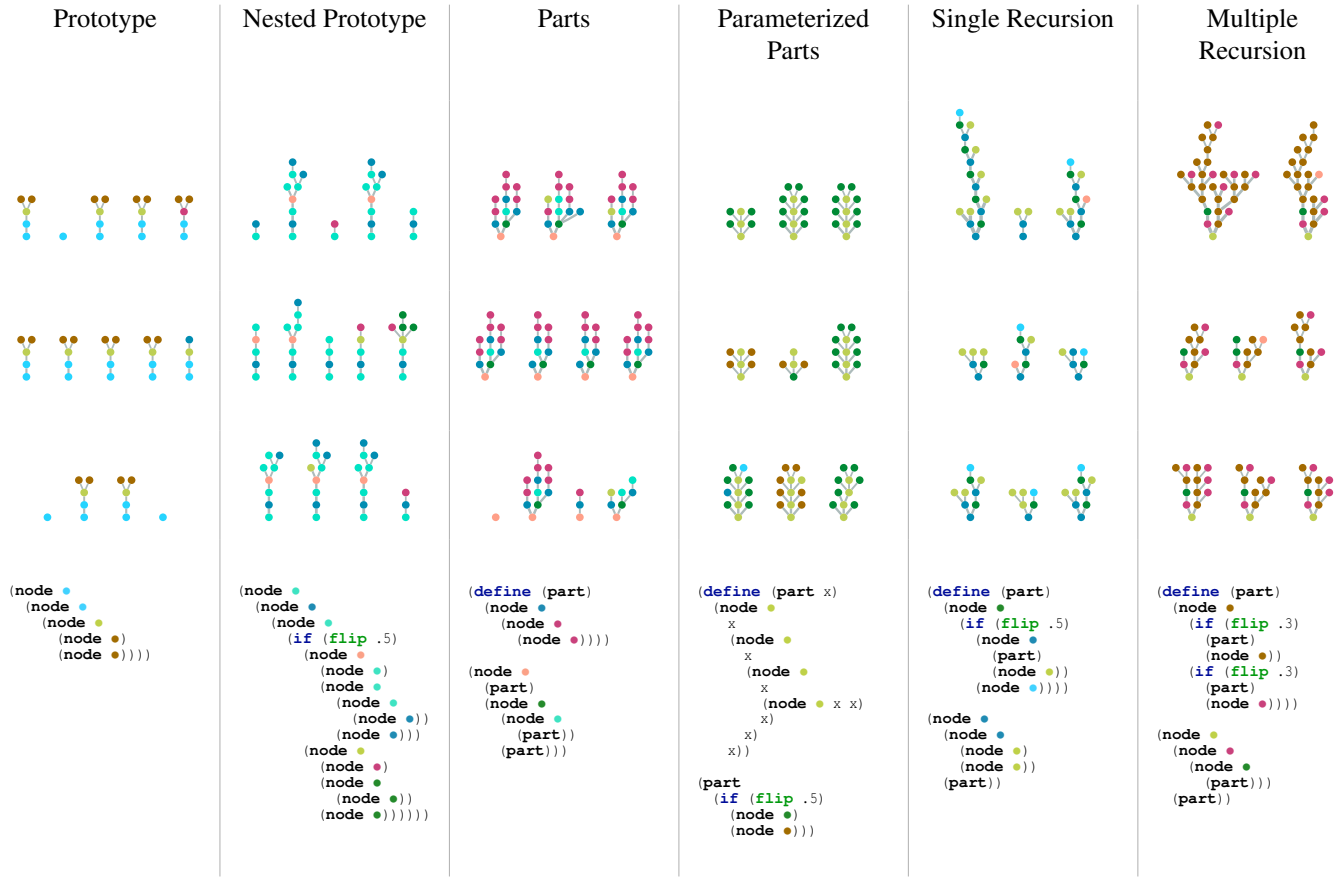

Table 1: This table illustrates the concept types that can be represented within our language for generative models. For each type, an example of a concept (a stochastic program) is shown together with observations drawn from this program. The stochastic function `node` generates a mixture of the subtrees that are passed to it as its arguments and a noise process that, with low probability, can generate any tree. The abstraction methods stochastic branching, (parameterized) parts and recursion compose these stochastic prototypes into more structured generative processes.

domain that both contains observations with simple structure and allows for interesting generative processes—the domain of colored trees generated by probabilistic programs.

Methods

Participants 250 members of Amazon's crowdsourcing service *Mechanical Turk* took part in the online experiment. Subjects were compensated for participation.

Stimuli Subjects were told that they are looking at newly discovered kinds of plants that grow in extreme environments. Each subject saw 18 pages, with each page consisting of 15 training examples, a control question, and a test example together with a classification question. Both training and test examples were images of simple trees with colored nodes drawn from tree-generating programs (see e.g. table 3). For each of the concept types shown in table 1, there were three tree-generating programs, and for each program there were 7 test examples. These test examples were chosen to cover a wide range of both intuitive and model judgements of category membership. Both training example order and stimuli colors were randomized.

Procedure In order to ensure that subjects process the training stimuli, a control question on each page asked how many of the training trees consist of more than 7 dots. 55 subjects answered less than 13 out of the 18 control questions correctly within an error margin of 2. We did not include these subjects in the analysis.

The categorization question asked: "How likely is it that the following plant is the same kind of plant as the plants above?" Subjects chose on a seven-step scale ranging from "certainly the same kind" to "certainly not the same kind". For each subject, the responses were normalized to $[0, 1]$.

Results

Table 2 summarizes the correlation results for all models. Figure 2 shows for each concept type human results and model results for both the exemplar and generative model. For each concept type, three different concepts were part of the experiment, and for each concept, seven different test observations were shown. A single point in the scatterplot contains information on the mean subject response for a single test tree and on the model prediction for this tree.

Neither of the two exemplar models based on simple statis-

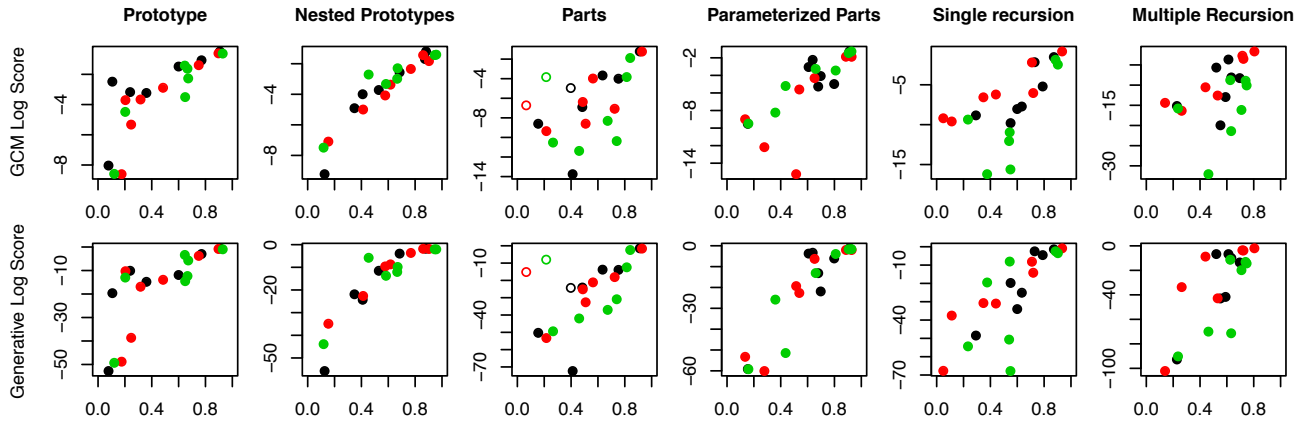

Figure 2: Comparison between human and model responses across concept types for tree exemplar and generative model. For each of the six concept types, three examples were shown; the color of the dots indicates to which example any given datapoint belongs. Empty circles denote isolated part cases that were excluded from the correlation analysis.

	Set GCM	Transition GCM	Tree GCM	Generative Model
Prototype	0.589	0.751	**0.803**	0.748
Nested Prototype	0.544	0.851	**0.937**	0.904
Parts*	0.320	0.617	0.705	**0.835**
Parameterized Parts	0.298	0.591	0.778	**0.911**
Single Recursion	0.284	0.499	0.637	**0.773**
Multiple Recursion	0.505	0.561	0.451	**0.770**

Table 2: Human-model correlations for the experiment. Each row shows how well the different models predicted subjects' performance for a particular concept type. *Correlations excluding isolated part cases (see text).

tics was the best predictor for any of the concept types, with the transition-based exemplar model performing strictly better than the set-based model. An effect that is not accounted for by the less structural exemplar models is illustrated by the nested prototype example in table 3: Subjects generalize significantly more to examples with branches they have seen before than to examples that have a mixture of two known branches. Likewise, subjects seem to generalize significantly more to trees with known branches than to trees that have new branches with similar surface statistics. Both results are expected under the two models that make use of tree structure.

If we group prototype and nested prototype as "less structured" and subconcepts with and without arguments, single recursions, and multiple recursions as "more structured", then the tree exemplar model best predicts human responses for the less structured stimuli whereas the true generative model best predicts performance for the more structured stimuli.

Our generative model makes the simplifying assumption that the learner infers a single generating concept from the examples whereas one interpretation of the tree exemplar model is that it uses each of the training examples as a hypothesis on what the true concept looks like. A fully Bayesian learner, which maintains a distribution over generative processes, may predict human behavior in ways similar to the tree exemplar model for less structured examples and similar to the true generating process model for the more structured examples.

Having seen how different models predict human judgements for different concept types, we will now look at individual response patterns in order to determine ways in which both of the two structural models can be improved.

The part example in table 3 shows how changes to the location of a part can have significantly different effects depending on whether the overall concept is preserved (resulting in high generalization) or the part is moved into a completely different environment (resulting in low generalization). By analogy, a Picasso face, with eyes in odd places, is still more of a face than an eye alone. Parts seen out of context constitute a problem for all models (except for the simplest set-based one): subjects judged these isolated parts as unlikely to come from the concept that included them as subparts whereas the models did give a high score to these examples. Since including these outliers dramatically changed the scores and made the interpretation of the model comparison difficult, we excluded these data points from the analysis in table 2. Without correction, the model-human correlations for the part concepts are: 0.403 for the set-based exemplar model, 0.505 for the transition exemplar model, 0.512 for the tree-based exemplar model, and 0.543 for the generative model (note that rank-order among the models does not change as a result of excluding these data points).

For the parameterized part example in table 3, changing the argument uniformly, i.e. in all places where it occurs, leads to consistently higher scores than changing the argument differently in different places; however, this difference is not significant. This difference is expected if subjects inferred the true generative model, since changes to the argument require only one use of the noise process, whereas nonuniform changes require many different nodes to be generated by the noise process. Future research needs to determine whether this effect is real, perhaps by manipulating the diversity of

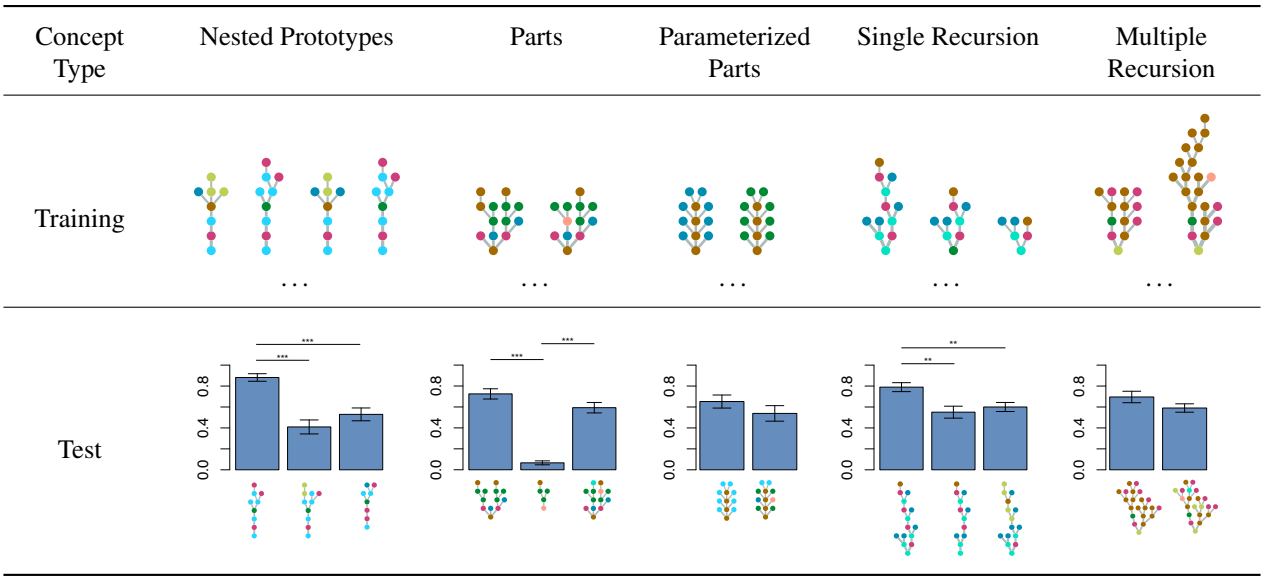

Table 3: This table illustrates a small selection of our experimental results. For five different concept types, training observations from a single concept of this type are shown together with subjects' generalizations for particularly interesting test examples. The error bars are standard errors of the mean.

parameter arguments in the observations.

For the single recursion example in table 3, changing the color of a few nodes within the recursion results in a significantly lower generalization. At the same time, a very similar manipulation does not result in a significant change in the generalization rating for the multiple recursion example. Intuitively, we sometimes see a change as destroying a very obvious pattern structure whereas at other times, the change in structure is not assumed to be relevant. Future research needs to characterize when subjects infer that such a pattern exists, and when they instead assume coincidence.

The comparison between the frequency based exemplar models and the two models that rely on tree structure in the observations makes clear that subjects do make use of the fact that the observations are structured in their generalization judgements. Furthermore, comparing the tree exemplar model to the true generative model that makes use of more abstract structure hints at the possibility that subjects are relying on recursive structure in the observations. The individual response patterns in the results of our exploratory experiment highlight ways in which both the exemplar-based model and the generative model can be improved to more closely reflect human generalization patterns.

Conclusion

Most studies of concept learning have focused on relatively unstructured objects based on simple features. We have suggested viewing concepts as probabilistic programs that describe stochastic generative processes for more structured objects. In this view concepts denote distributions over objects, and these distributions are built compositionally. We explored this idea within a domain of tree-like objects, and carried out a study of human generalization using a broad variety of concepts in this domain. Our results suggest that humans are able to extract abstract regularities, such as recursive structure, from examples, but also that there are many subtle effects to be discovered and accounted for in such domains.

Acknowledgements We thank Frank Jäkel and Brenden Lake for useful comments. This work was funded in part by grants from the ONR (N00014-09-0124) and the AFOSR (FA9550-07-1-0075).

References

Anderson, J. (1990). The adaptive character of thought.

Feldman, J. (1997). The structure of perceptual categories. *Journal of Mathematical Psychology*.

Goodman, N. D., Mansinghka, V., Roy, D. M., Bonawitz, K., & Tenenbaum, J. B. (2008). Church: a language for generative models. *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence*, 220–229.

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

Griffiths, T., Canini, K., & Sanborn, A. (2007). Unifying rational models of categorization via the hierarchical dirichlet process. *Proceedings of the 29th Annual Conference of the Cognitive Science Society*.

Kemp, C., Bernstein, A., & Tenenbaum, J. (2005). A generative theory of similarity. *Proceedings of the Twenty-Seventh Annual Conference of the Cognitive Science Society*.

Markman, A. (1999). Knowledge representation.

Nosofsky, R. (1986). Attention, similarity, and the identification-categorization relationship. *Journal of Experimental Psychology: General*, *115*(1), 39–57.

Rehder, B., & Kim, S. (2006). How causal knowledge affects classification: A generative theory of categorization. *Journal of Experimental Psychology: Learning, Memory, and Cognition*.

Shi, L., Feldman, N., & Griffiths, T. L. (2008). Performing bayesian inference with exemplar models. *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, 745–750.

Tomlinson, M., & Love, B. (2006). From pigeons to humans: Grounding relational learning in concrete examples. *Proceedings of the National Conference on Artificial Intelligence*, *21*(1), 199.