# Grounding Compositional Hypothesis Generation in Specific Instances

**Neil R. Bramley**[1] (neil.bramley@nyu.edu), **Anselm Rothe**[1] (anselm@nyu.edu)
**Joshua B. Tenenbaum**[2] (jbt@mit.edu), **Fei Xu**[3] (fei_xu@berkeley.edu), **Todd M. Gureckis**[1] (gureckis@nyu.edu)
[1]Department of Psychology, NYU, New York, NY, [2]Brain & Cognitive Sciences, MIT, Cambridge, MA, [3]Department of Psychology, UC Berkeley, CA

## Abstract

A number of recent computational models treat concept learning as a form of probabilistic rule induction in a space of language-like, compositional concepts. Inference in such models frequently requires repeatedly sampling from a (infinite) distribution over possible concept rules and comparing their relative likelihood in light of current data or evidence. However, we argue that most existing algorithms for top-down sampling are inefficient and cognitively implausible accounts of human hypothesis generation. As a result, we propose an alternative, Instance Driven Generator (IDG), that constructs bottom-up hypotheses directly out of encountered positive instances of a concept. Using a novel rule induction task based on the children's game Zendo, we compare these "bottom-up" and "top-down" approaches to inference. We find that the bottom-up IDG model accounts better for human inferences and results in a computationally more tractable inference mechanism for concept learning models based on a probabilistic language of thought.
**Keywords:** discovery; program induction; probabilistic language of thought, active learning; hypothesis generation

A number of recent papers argue that people come to learn abstract hypotheses about the world through inference in a "probabilistic language of thought" (Goodman, Tenenbaum, & Gerstenberg, 2015). That is, a system of conceptual primitives and stochastic rules for how they can be combined. Such models capture the fact that human thinking exhibits language-like compositionality and systematicity (Fodor, 1987; Lake, Salakhutdinov, & Tenenbaum, 2015), allowing us to combine and repurpose simple concepts to construct richer ones (Goodman, Tenenbaum, Feldman, & Griffiths, 2008; Piantadosi, Tenenbaum, & Goodman, 2016). In order to model inference in such models it is usually assumed that people approximate a posterior distribution over possible abstract expressions using stochastic search algorithms (Hastings, 1970). However, such approaches usually require the top-down generation of very large numbers of samples to stand a good chance of including the true hypothesis making them implausible algorithmic level accounts of human learning and hypothesis generation (Lewis, Perez, & Tenenbaum, 2014; Schulz, 2012).

However, assuming humans are endowed with some type of language-like representation for forming abstract hypotheses, one would assume that this language can be of use not only for inferring abstract hypotheses or ideas but also to describe concrete states of affairs in the world and specific instances. For example, looking at a visual scene one can describe various facts that hold (e.g., "the coffee cup is on top of the table", "the window is closed"). Such observations are essentially mental statements about the "data" one is perceiving. Given this ability, it seems it would be useful for a symbolic inferential hypothesis system to use these descriptions to help "seed" the hypothesis generation and search process. So for instance instead of starting from scratch and sampling repeatedly from a prior distribution, a learner might first start by considering hypotheses that match some of all of the factual statements considered about the data/world. By exploring potential generalizations and restrictions on those abstract hypotheses, one can more efficiently arrive at some deeper abstract causal knowledge (e.g., did the person leave their office in a hurry?) because each hypothesis is, at minimum, consistent with known data.

In this paper, we explore a middle ground between top-down sample-driven and bottom-up instance-driven concept learning. We propose a instance driven grammatical hypothesis generator, and demonstrate that it provides a closer characterization of human generalizations than a fully top-down approach, as well as being more computationally efficient.

## Discovery learning in Zendo

To explore this idea, we developed a new task environment that, while formally specified, is open ended enough to provide a challenging and naturalistic test bed for concept learning. The task is inspired by the tabletop scientific learning game Zendo$^{TM}$. In it, learners both observe and create their own *scenes*, which are arrangements of 2D triangular objects called *cones* on a flat surface (as depicted in Figure 1). The goal is to identify a hidden rule that divides scenes into *rule-following* and *not rule-following* scenes. Scenes could contain a varied number of cones, each of which had two immutable properties: size$\in$ {small, medium, large} and color$\in$ {red, green, blue} as well as having continuous scene-specific x$\in$(0,8), y$\in$(0,6) positions and orientations$\in(0,2\pi)$[1]. The scenes are subject to gravity meaning there are familiar physical constraints on how they might be arranged. In addition to cones' immutable and positional properties, scenes also admit many complex properties arising from the relative features and arrangement of different cones. For instance, subsets of cones might share a feature value (i.e., be the same color, or have the same orientation) or be ordered on another (i.e., be larger than, or above) and pairs of cones might have properties like pointing at one another or touching. This results in a rich space of potential concepts that require an expressive conceptual language. Following Piantadosi et al. (2016), we assume the true latent space of possible concepts in our task are those expressible in first order logic combined with lambda abstraction and full knowledge of the potentially relevant features of the scene. Lambda abstraction provides a simple general formalism for binding entities to variables (Church, 1932). For the current context, binding sets of cones

[1]We round these to one decimal place in evaluating rules to allow for perceptual uncertainty.

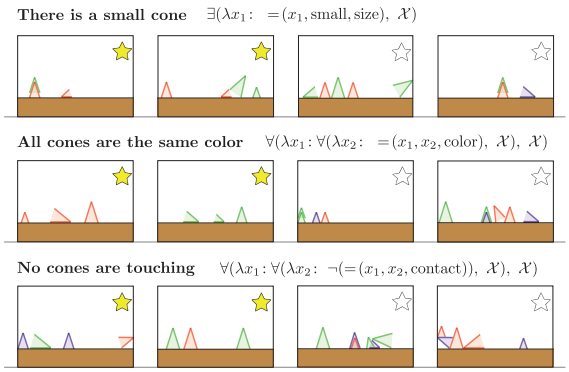


Figure 1: Three examples of rules in our task in words and lambda calculus with example scenes below. Yellow stars indicate scenes that follow the rule, white stars indicate scenes that do not.

Table 1: λ-abstraction Based Probabilistic Grammar.

| Productions | | | | |
|---|---|---|---|---|
| Start | $S \rightarrow$ | $\exists(\lambda x_i: \ A; X)$ | $\forall(\lambda x_i: \ A; X)$ | $N_I(\lambda x_i: \ A; J; X)$ |
| Bind additional | $A \rightarrow$ | $B$ | $S$ | |
| Expand | $B \rightarrow$ | C | $H(B; B)$ | $\neg(B)$ |
| Function | $C \rightarrow$ | $=(x_i; D1)$ | $I(x_i; D2)$ | $=(x_i; x_j; E1)$[a] |
| | | $I(x_i; x_j; E2)$[a] | $\Gamma(x_i; x_j; E3)$[a] | |
| Feature/value | $D1 \rightarrow$ | value[b] , | feature | |
| (numeric only) | $D2 \rightarrow$ | value[b] , | feature | |
| Feature | $E1 \rightarrow$ | feature | | |
| (numeric only) | $E2 \rightarrow$ | feature | | |
| (relational) | $E3 \rightarrow$ | feature | | |
| Boolean | $H \rightarrow$ | $\wedge$ | $\vee$ | ... |
| Inequality | $I \rightarrow$ | $\leq$ | $\geq$ | $>$ |
| | | $<$ | | |
| Number | $J \rightarrow$ | $n \in Z_5$ | | |

Note: Each production rule maps a capital letter (column 2) to several possible replacements (right hand columns). Context-sensitive aspects of the grammar: [a]Bound variable(s) sampled uniformly without replacement from set; expressions requiring multiple variables censored if only one. [b] The value is always sampled uniformly from the support of the feature selected in D.

to different variables allows our grammar to assert relations between distinct subsets of the cones in a scene.

## Top-down, "guess and check" inference mechanisms

As mentioned above, one solution to the problem of learning within an infinite hypothesis space is to sample hypotheses by composing them stochastically from an underlying grammar of sufficient expressivity. Here, we consider a grammar (specifically a *probabilistic context free grammar* or PCFG; Ginsburg, 1966) that can be used to produce any rule that can be expressed with first-order logic and lambda abstraction. When set up correctly, "simpler concepts" (i.e., those composed of fewer parts, cf. Feldman, 2000), will have a naturally higher prior probability of being produced and so will be favored over more complex ones equally able to explain the data. This is necessary since the setting ensures there will always still be an infinite number of potential rules consistent with any data (Gold et al., 1967).

With a PCFG, each hypothesis begins life as a string containing a single *non-terminal symbol* (here, $S$) that is replaced using rewrite rules, or *productions*. These productions are repeatedly applied to the string, replacing non-terminal symbols with a mixture of other non-terminal symbols and terminal fragments of first order logic, until no non-terminal symbols remain. The productions are so designed that the resulting string is guaranteed to be a valid grammatical expression. In addition, by having the productions tie the expression to bound variables and truth statements, our PCFG serves as an automatic concept generator. Table 1 details the PCFG we consider in the current paper and Figure 2a gives two example PCFG samples. We use capital letters as non-terminal symbols and each rewrite is sampled from the available productions for a given symbol.[2] Because some of the productions involve branching (e.g., $B \rightarrow H(B; B)$, see Table 1), the resultant string can become arbitrarily long and complex, involving multiple boolean functions and complex relationships between bound variables. The probabilities for each production in a PCFG can be fit to human judgments, and different PCFGs containing different primitives and expansions can be compared. In this way, recent work has attempted to infer the logical primitives of thought (Goodman et al., 2008; Piantadosi et al., 2016).

What these PCFG approaches have in common is that they provide a generative mechanism for sampling from a prior over all possible concepts. However, these sampled "guesses" must then be tested against data. Unfortunately, many samples, even from a well tuned PCFG are likely to be tautological (i.e., "All cones are red or not red"), contradictory (i.e., "There is a cone that is red and not red"), physically impossible ("There are two distinct objects have the same position and orientation") or simply inconsistent with whatever data a learner has already encountered. Indeed, around 20% of the hypotheses generated by the grammar in Table 1 are tautologies, and 15% are contradictions. For these reasons, the procedure is inherently inefficient, and typically requires a very large numbers of samples in order to reliably provide non-trivial rules.

## Instance driven hypothesis generation

Our Instance Driven Generation (IDG) proposal is related to the PCFG idea but with one major difference. Rather than generating guesses entirely stochastically, before checking them against the data, we propose that people generate guesses *inspired* by an encountered positive example (cf. Michalski, 1969). Concretely, we propose that learners start by observing the features of objects in a rule-following scene and use these to back out a true logical statement about the scene in a stochastic but truth preserving way. In this way the learner does not generate uniformly from all possible logical statements, but directly from the restricted space of statements true of the current observation. Figure 2b motivates this approach. Here a learner begins their hypothesis generation with an observation of a scene that follows the hidden rule. To generate hypotheses as candidates for the hidden rule, we assume the learner uses the following procedure:

[2]The grammar is not strictly context free because the bound variables ($x_1$; $x_2$, etc.) are shared across contexts (e.g. $x_1$ is evoked twice in both expressions generated in Figure 2a).

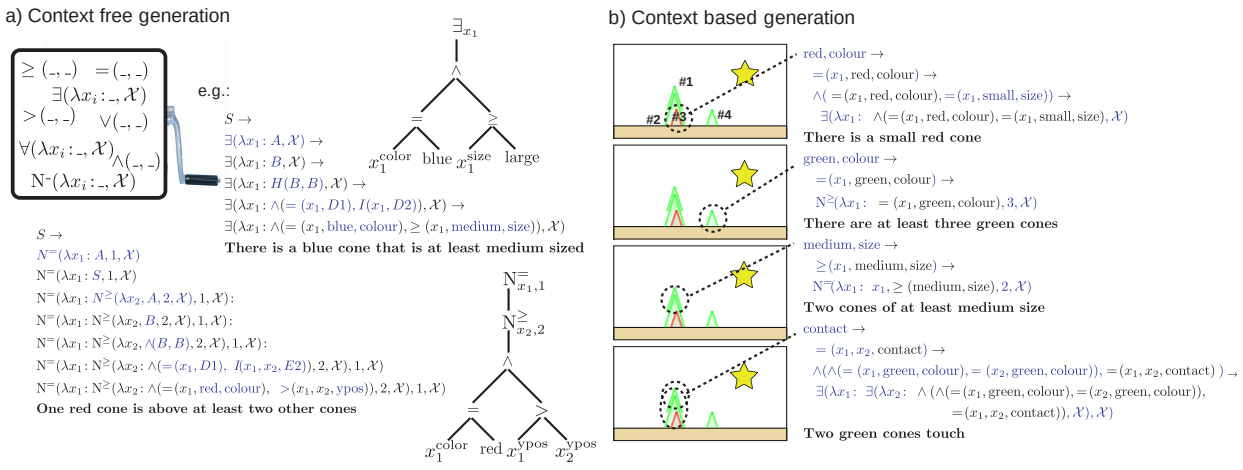


Figure 2: a) Example generation of hypotheses using PCFG in Table 1. b) Examples of IDG hypothesis generation based on an observation of a scene that follows the rule. New additions on each line are marked in blue.

1. **Observe.** With uniform probability, either:

   (a) Sample a cone from the observation, then sample one of its features — e.g., #1:[3] "medium, size" or {#3}: "red, color".

   (b) Sample two cones uniformly without replacement from the observation, and samples any shared or pairwise feature — e.g., {#1,#2}: "size", or "contact"

2. **Functionize.** Bind a variable for each sampled cone in Step 1 and sample a true (in)equality statement relating the variable(s) and feature:

   (a) For a statement involving an unordered feature there is only one possibility — e.g, {#3}: "$=(x_1;\text{red};\text{color})$", or for {#1,#2}: "$=(x_1;x_2;\text{color})$"

   (b) For a single cone and an ordered feature this could also be a nonstrict inequality ($\geq$ or $\leq$). We assume a learner only samples an inequality if it expands the number of cones picked out from the scene relative to an equality — e.g., in Figure 2b, there is also a large cone {#1} so either $\geq(x_1;\text{medium};\text{size})$ or $=(x_1;\text{medium};\text{size})$ might be selected.

   (c) For two cones and an ordered feature, either strict or non-strict inequalities could be sampled if the cones differ on the sampled feature, equivalently either equality or non-strict inequality could be selected if the cones do not differ on that dimension — e.g., {#1,#2} $>(x_1;x_2;\text{size})$, or {#3,#4} $\geq(x_1;x_2;\text{size})$.

3. **Extend.** With probability $\frac{1}{2}$ go to Step 4, otherwise sample uniformly one of the following expansions and repeat. For statements with two bound variables Step 3 is performed for $x_1$, then again for $x_2$:

   (a) **Conjunction.** A cone is sampled from the subset picked out by the statement thus far and one of its features sampled — e.g., {#1} $\wedge(=(x_1;\text{green};\text{color}), \geq(x_1;\text{medium};\text{size}))$. Again, inequalities are sampled only if they increase the true set size relative to equality — e.g., "$\wedge(\leq(x_1;3;\text{xposition}), \geq(x_1;\text{medium};\text{size}))$", which picks out more objects than "$\wedge(=(x_1;3;\text{xposition});\geq(x_1;\text{medium};\text{size}))$".

   (b) **Disjunction.** An additional feature-value pair is selected uniformly from *either* unselected values of the current feature, *or* from a different feature — e.g., $\vee(=(x_1;\text{color};\text{red});=(x_1;\text{color};\text{blue}))$ or $\vee(=(x_1;\text{color};\text{blue});\geq(x_1;\text{size};2))$. This step is skipped if the statement is already true of all the cones in the scene.

4. **Quantify.** Given the contained statement, select true quantifier(s):

   (a) For statements involving a single bound variable (i.e., those inspired by a single cone in Step 1) the possible quantifiers simply depend on the number of the cones in the scene for which the statement holds. The quantifier is selected uniformly from the existential $\exists(_)$ or numerical "exactly $N^{=}(_;J)$", "at least $N^{\geq}(_;J)$", or "at most $N^{\leq}(_;J)$" and $J$ is set to match number of cones for which the statement is true. If it it is true for all cones, the universal quantifier $\forall(_)$ may also be selected.

   (b) Statements involving two bound variables in lambda calculus have two nested quantifier statements each selected as in (a). The inner statement quantifying $x_2$ is selected first based on truth value of the expression while taking $x_1$ to refer to the cone observed in '1.'. The truth of the selected inner quantified statement is then assessed for all cones to select the outer quantifier — e.g., {#3,#4} "$\wedge(=(x_2;\text{green};\text{color});\leq(x_1;x_2;\text{size}))$" might become "$\forall(\lambda x_1:\exists(\lambda x_2:\wedge(=(x_2;\text{green};\text{color});\leq(x_1;x_2;\text{size}));X);X)$". The inner quantifier $\exists$ is selected because three of the four cones are green {#1, #2, #4}, and the outer statement is selected because all cones are less than or equal in size to a green cone.

Note that a procedure like the one laid out above is, in principle, capable of generating any rule generated by the PCFG in Table 1, but will only do so when exposed to a positive

[3] Numbers prepended with # refer to the labels on the cones in the example observation in Figure 2b.