Rule Learning Through Active Inductive Inference

Tore Erdmann[1(✉)] and Christoph Mathys[2,3]

[1] SISSA, Via Bonomea 265, 34356 Trieste, Italy
terdmann@sissa.it
[2] Interacting Minds Center, Aarhus University, Jens Chr. Skous Vej 4, 8000 Aarhus, Denmark
[3] Translational Neuromodeling Unit (TNU), Institute for Biomedical Engineering, University of Zurich and ETH Zurich, Wilfriedstrasse 6, 8032 Zurich, Switzerland

Abstract. We propose a grammar-based approach to active inference based on hypothesis-driven rule learning where new hypotheses are generated on the fly. This contrasts with traditional approaches based on fixed hypothesis spaces and Bayesian model reduction. We apply these two contrasting approaches to an established active inference task and show that grammar-based agents' performance benefits from the explicit rule representation underpinning hypothesis generation. Our proposal is a synthesis of the active inference framework with language-of-thought models, which paves the way for computational-level descriptions of false inference based on an aberrant hypothesis-generating process.

Keywords: Active inference · Rule induction · Context free grammars · Structure learning · Sampling-based inference · Reasoning

1 Introduction

Structure learning is a fundamental problem for an active inference agent. Logically structured concepts can be found in domains such as mathematics, social systems or causal processes [13]. The likelihood mapping of a POMDP with discrete state space can be represented as a matrix with elements indicating the likelihood of an observation given a state. Current approaches for learning this mapping rely on separately estimating the individual elements of the matrix [4,12]. Here, we propose an approach for structure learning that uses a prior based on context-free grammars (CFG; [2]), which were invented in linguistics to describe the structure of sentences in natural language and are used to define programming languages in computer science. From such a grammar, the agent can, through recursive composition and substitution of terms, generate an infinite number of expressions, which represent the underlying structure of (parts of) its environment. As a proof of concept, we will illustrate our approach by applying it to a rule learning problem inspired by the task in [4].

This approach has previously been used in cognitive science, psychology [6, 13,14] and, in particular, in "language of thought" models [10]. Previous work

M. Kamp et al. (Eds.): ECML PKDD 2021 Workshops, CCIS 1524, pp. 715–725, 2021.
https://doi.org/10.1007/978-3-030-93736-2_51

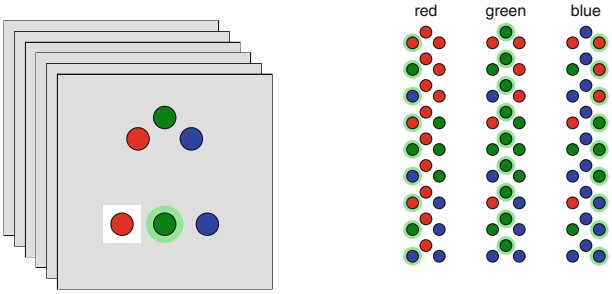

Fig. 1. Left: shows the display during a trial. The agent sees context variables (the three circles in the upper half), makes a response (indicated by the white box around the red circle) and, having made a choice, the correct choice (highlighted in green). Right: The possible contexts arranged according to the value of the middle circle, which implies where to look for the correct choice (highlighted in green). The correct response is equal to the color of the circle on the left, in the center or on the right when the color of the central circle is "red", "green" or "blue", respectively. (Color figure online)

has shown that these models can account for various features of human concept learning ([5]). Furthermore, this approach has been used to explain surprise signals in the striatum [1].

We will work through a simplified version of the task of [4]. For ease of presentation and to place our focus on structure learning, we remove state uncertainty and all intra-trial actions except for the final choices. All remaining uncertainty is thus about the hidden rule. However, our proposal can be straightforwardly applied to the case including state uncertainty and observations corrupted by noise. In this task, see Fig. 1, the agent has to infer a rule, that is a deterministic mapping from three context variables to the correct choice.

2 Active Inference

Solving this task consists of finding a policy $p(a_t|c_t, \theta)$, that gives the probability of a choice $a_t \in \{1, 2, 3\}$ given the context variables c_t and some parameters θ. The generative model the agent holds of the task is

$$c_t^{(j)} \sim U(\{1, 2, 3\}),\ j = 1, 2, 3 \tag{1}$$

$$c_t = (c_t^{(1)} c_t^{(2)}, c_t^{(3)}) \tag{2}$$

$$o_t \sim f(c_t, \cdot) \tag{3}$$

$$p(r_t = 1|a_t, o_t) = \exp(\ell(a_t, o_t)) \tag{4}$$

$$\ell(a_t, o_t) = \begin{cases} 0 & \text{if } a_t = o_t \\ -4 & \text{else} \end{cases} \tag{5}$$

where $f(c, o)$ is a function representing the hidden rule. That is, it returns the probability of observing the outcome $o \in \{1, 2, 3\}$ in context c. The prior about reward observations $p(r_t|a_t, o_t)$ represents an optimistic bias, so that the agent's beliefs are biased by desirable states and not the actual task dynamics, which is $r_t = \mathbb{I}(o_t = a_t)$. This model implies a distribution over trial sequences, which we denote $\tau = (c_{1:T}, a_{1:T}, o_{1:T}, r_{1:T})$, that factorizes as

$$p(\tau|f) = \prod_{t=1}^{T} p(r_t|a_t, o_t)p(o_t|c_t, f)p(a_t)p(c_t). \tag{6}$$

Given the biased prior over rewards we obtain the following posterior over actions when conditioning on $r_t = 1, \ \forall t = 1, \ldots, T$ and summing out o_t, which is unknown at the time of the action,

$$p(c_{1:T}, a_{1:T}, o_{1:T}|r_{1:t} = 1, f) \propto \prod_{t=1}^{T} p(r_t = 1|a_t, o) \tag{7}$$

In keeping with the active inference framework, the expected log model evidence is minimized by computation of the posterior over action, which can be done at each trial t by choosing

$$p(a_t|r_t = 1, c_{1:t}, o_{1:t-1}) = \sigma(-G_{a_t}) \tag{8}$$

$$G_{a_t} = \sum_{o} l(a_t, o) \cdot \mathbb{E}_{p(f|c_{1:t-1}, o_{1:t-1})}\Big[p(O_t = o|c_t, f)\Big] \tag{9}$$

For the implementation, this means we need to be able to evaluate the agent's posterior predictive about the belief about the outcome o_t. The above constructions leads to the maximization of the following objective (see [7])

$$D_{KL}(p^*(\tau)||p(\tau)) = \mathbb{E}_{\tau \sim p^*(\tau)}\Big[\log p(\tau) - \log p^*(\tau)\Big], \tag{10}$$

which is the Kullback-Leibler divergence of the agent's beliefs about its future states and a desired distribution over these p^*, and which is equivalent to the free energy of the expected future, which is a lower bound on the expected log model evidence [8].

2.1 Evidence Accumulating Agent

A straightforward solution for learning the rule is available if we represent it as a stochastic vector consisting of independent Dirichlet variables, $f(c, o) = \theta_{c,o}$, with $\theta_{j,\cdot} \sim Dir(\alpha_0)$, $j = 1, \ldots, 27$, for which the posterior can be computed by accumulation of concentration parameters:

$$p(\theta_{j,o}|c_{1:t}, o_{1:t}) = Dir(n_{c,o} + \alpha_0) \tag{11}$$

where $n_{c,o}$ is the number times (up until time t) the agent has observed outcome o for context c. If we define a matrix α with entries $\alpha_{c,o} = n_{c,o} + \alpha_0$, the expectation

in Eq. 8 is a that of a categorical-Dirichlet distribution and the action is chosen via

$$G_{a_t} = \sum_o \ell(a_t, o)) \cdot \frac{\alpha_{c_t,o}}{\sum_j \alpha_{c_t,j}}. \tag{12}$$

2.2 Bayesian Model Reduction

If the agent knows that there must be a deterministic rule, it can quickly recognize the rule by comparing the evidence for each potential model in a set of hypothetical models and accept a model if its evidence exceeds a certain threshold.

The model space can be considered the set of deterministic, one-to-one mappings from each color to each response (of which there 6) which are combined with the 6 possible mappings between the central color and which location the color-to-response mapping should be applied to (see [4]). There are thus 36 hypotheses, for which the evidence is computed on each trial. This allows us to represent the priors through sets of prior concentration parameters as derived in [4]. A condition for this agent is that the space of hypotheses is specified for the agent beforehand, which is a strong assumption in general. We will now introduce a way to model acquisition of new models. This has the advantage of being based on weaker assumptions about (and a different conception of) prior knowledge.

3 Grammar-Based Rule Induction

Here, we describe how rule learning can be supported through a structured prior over an auxiliary space of symbolic rule expressions. Each such rule expression is defined by a syntax tree, consisting of logical connectives (and, or), and references to the observations in a trial. The "leaf nodes" of the tree are predicates of some part of the observation c_t, for example $color(c_t^{(1)}) = red$, which is either true or false (see appendix for an example). An agent can learn a rule expression that accurately predicts the outcome of the unknown rule f by searching the space of rule expressions for hypotheses which are then evaluated against the available evidence. Hypotheses are represented by expressions that can be generated by iterating the following set of re-write (or production) rules:

$$\begin{aligned}
\text{(Start)}\quad & S \rightarrow f(c,o) \iff (D) \\
\text{(Disjunction)}\quad & D \rightarrow C \vee D \;\mid\; P \;\mid\; false \\
\text{(Conjunction)}\quad & C \rightarrow P \wedge C \;\mid\; P \;\mid\; true \\
\text{(Predicate)}\quad & P \rightarrow color(Loc) = Col \\
\text{(Location)}\quad & Loc \rightarrow c_1 \;\mid\; c_2 \;\mid\; c_3 \\
\text{(Color)}\quad & Col \rightarrow \text{“}red\text{”} \;\mid\; \text{“}green\text{”} \;\mid\; \text{“}blue\text{”}
\end{aligned}$$

These rules indicate how symbols on the left hand side of the $\rightarrow$ can be replaced by one of the options on the right hand side (options are separated by |). From this grammar, given certain production probabilities (which give the probability of each possible production for each line in the grammar; can be assumed uniform), we can generate rule expressions (we refer the interested reader to Wikipedia, for examples, or [11] for a comprehensive treatment). Note that we omit the trial index t in the formulas (since the rules only refers to variables in the current trial) and instead use the subscript to denote the location (1, 2 or 3) of the context variable.

Each generated expression describes some arrangement of context observations. Say, we wanted to describe the rule for when the correct color is red (as given in the caption of 1). This can be expressed as $color(c_2) =$ "red" $\wedge$ $color(c_1) =$ "red") $\vee$ ($color(c_2) =$ "$blue$" $\wedge$ $color(c_3) =$ "red"), which can be generated through step-wise replacement of the above rules. The prior probability of a formula (i.e. a sequence of substitutions from the grammar) is equal to the product of the probabilities of the individual substitutions. This prior naturally places higher probability on shorter and less complex expressions since they include fewer terms in the product.

For the rule learning task described above, we want to model the contexts that correspond to the three outcomes (and actions), so we will make the procedure to be learned a function of both the observed context c and the outcome o, changing the rule in the topmost line above to be a context-sensitive expression of the form

$$S \rightarrow f(c, o) \iff ((o = \text{“}red\text{”}) \wedge D) \vee ((o = \text{“}green\text{”}) \wedge D) \vee ((o = \text{“}blue\text{”}) \wedge D),$$

wherein the D terms will come to represent the parts of the rule that imply the corresponding outcome. We can then evaluate expressions with regard to each possible outcome to determine if the context c matches the outcome o. Starting from the above expression and generating sub-expressions according to the above grammar, we can represent the true hidden rule described in Fig. 1 as follows:

$$\begin{aligned} f(c, o) \iff & ((o = \text{“}red\text{”}) \wedge \\ & ((color(c_2) = \text{”}red\text{”} \wedge color(c_1) = \text{”}red\text{”}) \vee \\ & (color(c_2) = \text{”}blue\text{”} \wedge color(c_3) = \text{”}red\text{”}))) \\ & \vee ((o = \text{“}green\text{”}) \wedge \\ & ((color(c_2) = \text{”}green\text{”}) \vee (color(c_2) = \text{”}red\text{”} \wedge color(c_1) = \text{”}green\text{”}) \vee \\ & (color(c_2) = \text{”}blue\text{”} \wedge color(c_3) = \text{”}green\text{”}))) \\ & \vee ((o = \text{“}blue\text{”}) \wedge \\ & ((color(c_2) = \text{”}red\text{”} \wedge color(c_1) = \text{”}blue\text{”}) \vee \\ & (color(c_2) = \text{”}blue\text{”} \wedge color(c_3) = \text{”}blue\text{”}))) \end{aligned}$$

However, we can represent this rule more succinctly by adding more abstract terms to the grammar. For example, by adding two new production rules to the grammar above:

$$P \rightarrow color(Loc) = COL \quad | \quad o = color(Loc)$$
$$Loc \rightarrow c_1 \quad | \quad c_2 \quad | \quad c_3 \quad | \quad c_{Loc}$$

The last production will lead to a "subsetting", such as c_{c_2}, which means that the value of c_2 indexes the context variables (with the colors mapped to the numbers $\{1, 2, 3\}$). The expression $o = color(Loc)$ evaluates to true if the outcome matches the variable *Loc*. With these additions, we can now represent the true rule as a much shorter expression

$$f(c, o) \iff (o = color(c_{c_2})). \tag{13}$$

This shorter representation of the rule helps the agent to discover it much more quickly. This is because shorter rules have higher prior probabilities of being produced.

The above rule expression defines a function that evaluates to `true` if the action a is correct given the observation o and `false` otherwise. The likelihood of this expression is given by its match with the observed data, that is, the number of examples for which the rule f evaluates to true,

$$p(f|o_{1:t}, a_{1:t}, c_{1:t}) \propto \bigwedge_{c,o} f(c, o) \tag{14}$$

or, if assuming that some observations might be outliers to the rule, we have

$$p(f|o_{1:t}, a_{1:t}, c_{1:t}) \propto e^{-\gamma Q(f)} \tag{15}$$

where $Q(f) = |\{(c, o) \in (c_{1:t}, o_{1:t}) : f(c, o) = false\}|$ (the count of examples for which the rule expression evaluates to false) and γ is a parameter denoting the probability that a given example is an outlier. Here, the probabilities need not be normalized, since any normalization constants cancel in the MCMC acceptance probability. The truth value of the procedure $f(a, o)$ follows from the evaluation approach in mathematical logic [3] and is defined recursively:

1. $f(a, o)$ is a node.
2. If a node is a predicate, it can be evaluated directly
3. If it is a logical connective then it is evaluated by first evaluating the sub-expressions separately and then applying the logical function to the result. For example, $a \wedge b$ is true only if both sub-expressions a and b are true.

In our implementation, we represent the agent's belief about the correct rule expression as a set of samples that are approximately distributed according to the posterior distribution implied by the above likelihood and prior. This posterior is updated on each trial by running a Markov Chain Monte Carlo (MCMC) chain for a fixed number of iterations. The set of expressions that was visited during the

walk is taken to represent the posterior belief. This construction leads to the posterior predictive distribution, given a set H_t of hypotheses. Formally, if we denote the chain representing the belief update in trial t by $H^{(t)} = \left(h_1^{(t)}, \ldots, h_n^{(t)}\right)$, we can evaluate the posterior expectation in action selection in Eq. 8 approximately as follows

$$p(a_t = a | r_t = 1, c_{1:t}, o_{1:t-1}) = \sigma(-G_{a_t}) \tag{16}$$

$$G_{a_t} = \sum_o l(a_t, o) \cdot \mathbb{E}_{p(f|c_{1:t-1}, o_{1:t-1})}\Big[p(O_t = o | c_t, f)\Big] \tag{17}$$

$$\approx \sum_o \ell(a_t, o)) \cdot \frac{\sum_i f_{h_i^{(t)}}(c_t, a)}{\sum_{j \in \{1,2,3\}} \sum_i f_{h_i^{(t)}}(c_t, j)} \tag{18}$$

which can be seen as a model average of all hypotheses that were visited by the Markov chain during the computation of the posterior.

The iterations of the MCMC procedure propose changes to the expression by randomly selecting a sub-expression and replacing it with a newly generated sub-expression. The Metropolis-Hastings acceptance probability for a proposal balances the probability of the proposal and the reverse proposal, the prior probabilities and the likelihood (see Eq. 14) of the current and proposed expressions (tree-substitution MCMC; see [5] for details). The belief update can thus be performed by running n MCMC iterations, starting from the current state of the chain. For the current task, once the true rule has been found, proposal for moves away from it will have very low probability. In general, when the rule cannot be known with certainty, the chain will move between alternatives and thereby lead to a representation of the remaining uncertainty in the posterior belief about the rule.

4 Experiments

We simulated learning in four agents who completed 20 trial sequences each. These sequences contained 27 trials and were generated by randomly shuffling the 27 unique combinations of context variables. The four agents differed in substantial ways and could be characterized as concentration parameter accumulating agents (Agents 1 and 2, described in Sect. 2.1) with (Agent 2) and without (Agent 1) model-selection (by Bayesian model reduction, Sect. 2.2) after each trial; and the grammar-based agents (Agents 3 and 4) with the simple grammar described in Sect. 3 (Agent 3) and an extended grammar described below (Agent 4).

A comparison of the performance of the different agents are shown in Fig. 2, where the average proportion of correct responses is shown over trials. As can be seen, the grammar-based agents show higher proportions of correct responses already during early trials. This is due to the nature of the rule expressions, which can be extrapolated from rapidly.

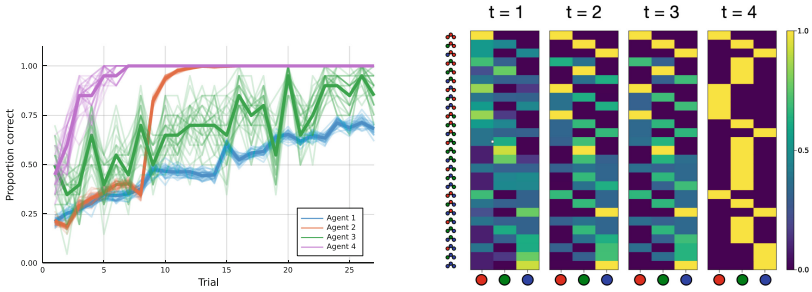

Fig. 2. Left: Proportion of correct choices (averaged over simulations) for the four agents with uncertainty indicated via bootstrapped estimates (thin lines). Right: Belief of (purple) grammar-based agent for the first 4 examples of a particular trial sequence. Each heatmap shows the probability of an action (x-axis) to be correct in a given context (y-axis). (Color figure online)

The best-performing agent (purple in Fig. 2) is Agent 4 with the extended grammar, that has two additional production rules contained in its grammar (see sec. 3). Figure 2 (right) shows the Agent 4's belief about the rule during the early trials of a particular trial sequence. The examples presented to the agent were ((●, ●, ●), (●, ●, ●), (●, ●, ●), (●, ●, ●)), for which the correct responses were (●, ●, ●, ●). We can inspect the set of hypotheses held by the agent. At $t = 3$, the hypotheses with the highest weights (about $n/3$ occurrences) are:

1. $a = color(o_1)$, "answer equal to the left circle"
2. $a = color(o_{o_2})$ "answer equal to the color at location indicated by o_2"
3. $a = color(o_{o_3})$ "answer equal to the color at location indicated by o_3"

The agent cannot tell between these explanations until observing the outcome in the 4th trial, when the predictions of hypotheses 1 and 3 are disproved and the agent correctly infers the rule.

These results show how learning speed relates to underlying assumptions. As opposed to Agents 1 and 2, who need to be equipped with a fixed hypothesis set, the grammar-based agents can learn arbitrary rules, including such for which maintaining a fixed hypothesis set would be infeasible, as long as they can be represented within the language spanned by their grammar. For example, if we had just told the agent: "In this game, there is a deterministic mapping between the three colored circles and the correct response", the hypothesis space would have to cover a space of mappings containing 2^{81} elements. Comparing each of these candidates at the end of a trial would be infeasible (at a rate of 10^9 evaluations per second (1 evaluation per nanosecond), it would take about 77 million years to evaluate all candidates). The code for all experiments reported here is available at https://github.com/ilabcode/IWAI2021.

5 Discussion

We have shown a novel way to perform structure learning in active inference agents. In particular, we demonstrate how an agent can use grammar-based structure learning to develop a model in a bottom-up fashion. This is different from the traditional approach of Bayesian model reduction, which can be considered a top-down approach. The assumption of a grammar that spans a hypothesis space is weaker and hence more generalizable than pre-defining a finite set of hypotheses. Other ways of searching for rule expressions are possible, such as genetic algorithms, but these do not represent uncertainty and are therefore not well suited as a basis for adaptive prediction and decision-making.

Our results showed differences between the two grammar-based agents that were apparent in the speed by which they learn the rule. For the task presented here, both agents converge to the same behavior, but their underlying rule representations are different. This highlights how higher-order inferences can depend on the base of concepts and abstractions they are built upon. In terms of the behavior, the agents will look the same, however, their representational vocabulary differs and so they will find separate explanations for the rule (which do describe the same contingencies), which also have different complexity (as clearly visibly in the number of terms). Given a way to update their own grammars through experience, two agents starting with different grammars but in similar environments might develop a similar conceptual toolbox. One way to enable this would be to add special "lambda expression" terms to the grammar. Such an encoding of the lambda calculus within the hypothesis language leads to the ability to define new terms and apply or re-combine them (see [9]).

An interesting aspect of your hypothesis-generating grammar-based approach is the ways in which the assumptions underlying the generation of hypotheses of can influence what the agent finally takes to be the most promising course of action. This can become a useful tool for understanding aberrations in world modeling such as those apparent in psychiatric illnesses, which might have to do with a deficient hypothesis-generating process. For example, hypotheses generated from a grammar that is poorly attuned to a domain can seem bizarre to outside observers. Such misattunement may be the result of aberrant learning processes that update the production probabilities of a grammar, or the addition or removal of terms.

The agent described in [4] did not include model-selection considerations in its actions since they were outside of its generative model (and, in any case, the actions in the task were uninformative in that regard). By contrast, with a grammar-based approach, the structure is part of the agent's prior. Therefore its actions can subserve the testing of freshly generated hypotheses about the hidden structure of a task, which corresponds to active learning. Crucially, this could be made relevant in a version of the rule learning task where the agent can choose its next set of context variables. This would require planning, where the agent finds the optimal plan for testing its currently most promising hypotheses—an interesting avenue for future research based on the approach introduced here.

Appendix

Context-free grammars

A context-free grammar is defined by a 4-tuple (V, Σ, R, S), with V a finite set of *variables*, Σ a finite set of *terminals*, R a set of *rules*, each of which consist of a variable and a string of variables and terminals, and $S \in V$ is the *start variable*. One can use a grammar to describe a language by generating strings of that language in the following manner.

1. Write down the start symbol.
2. Find a variable that is written down and a rule that starts with that variable. Replace the variable with the right-hand side of the rule.
3. Repeat step 2 until no variables remain.

$$V := \{S, C, D, P, Loc, Col, \wedge, \vee\},$$

Σ is $\{c_1, c_2, c_3, red, green, blue\}$ and rules are as given in Sect. 3. This grammar describes a basic programming language for expressions containing logical connectives and predicates. For example, the string $color(c_1) = red \wedge color(c_2) = green$ can be generated from the grammar. The sequences of substitutions to obtain a string is called a *derivation*. The derivation of the above example is shown in Fig. 3.

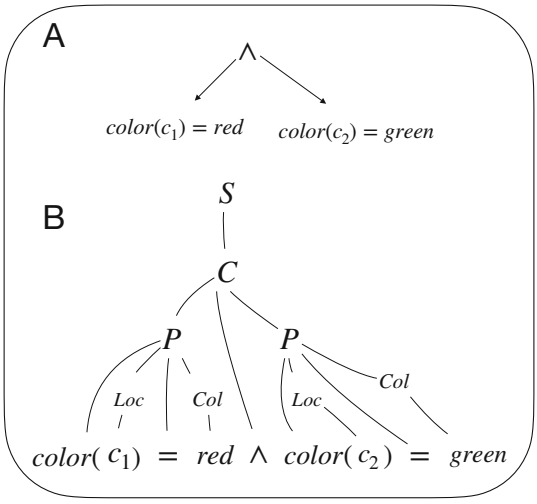

Fig. 3. A: Example of an expression that evaluates to true in the example from 1. B: Corresponding derivation from the language defined by the grammar in Sect. 3.

References

1. Ballard, I., Miller, E.M., Piantadosi, S.T., Goodman, N.D., McClure, S.M.: Beyond reward prediction errors: human striatum updates rule values during learning. Cereb. Cortex **28**(11), 3965–3975 (2018). https://doi.org/10.1093/cercor/bhx259
2. Chomsky, N.: Three models for the description of language. IRE Trans. Inf. Theory **2**(3), 113–124 (1956). https://doi.org/10.1109/TIT.1956.1056813
3. Enderton, H.B.: A Mathematical Introduction to Logic, 2nd edn. Harcourt/Academic Press, San Diego (2001)
4. Friston, K.J., Lin, M., Frith, C.D., Pezzulo, G., Hobson, J.A., Ondobaka, S.: Active inference, curiosity and insight. Neural Comput. **29**(10), 2633–2683 (2017). https://doi.org/10.1162/neco_a_00999
5. Goodman, N., Tenenbaum, J., Feldman, J., Griffiths, T.: A rational analysis of rule-based concept learning. Cogn. Sci. Multidisc. J. **32**(1), 108–154 (2008). https://doi.org/10.1080/03640210701802071
6. Kemp, C., Tenenbaum, J.B., Niyogi, S., Griffiths, T.L.: A probabilistic model of theory formation. Cognition **114**(2), 165–196 (2010). https://doi.org/10.1016/j.cognition.2009.09.003
7. Levine, S.: Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review. arXiv:1805.00909 [cs, stat] (2018)
8. Millidge, B., Tschantz, A., Buckley, C.L.: Whence the expected free energy? Neural Comput. **33**(2), 447–482 (2021). https://doi.org/10.1162/neco_a_01354
9. Piantadosi, S.T., Tenenbaum, J.B., Goodman, N.D.: Bootstrapping in a language of thought: a formal model of numerical concept learning. Cognition **123**(2), 199–217 (2012). https://doi.org/10.1016/j.cognition.2011.11.005
10. Piantadosi, S.T., Tenenbaum, J.B., Goodman, N.D.: The logical primitives of thought: empirical foundations for compositional cognitive models. Psychol. Rev. **123**(4), 392–424 (2016). https://doi.org/10.1037/a0039980
11. Sipser, M.: Introduction to the Theory of Computation. PWS Pub. Co., Boston (1997)
12. Smith, R., Schwartenbeck, P., Parr, T., Friston, K.J.: An active inference approach to modeling structure learning: concept learning as an example case. Front. Comput. Neurosci. **14** (2020). https://doi.org/10.3389/fncom.2020.00041
13. Tenenbaum, J.B., Kemp, C., Griffiths, T.L., Goodman, N.D.: How to grow a mind: statistics, structure, and abstraction. Science **331**(6022), 1279–1285 (2011). https://doi.org/10.1126/science.1192788
14. Ullman, T.D., Goodman, N.D., Tenenbaum, J.B.: Theory Acquisition as Stochastic Search, p. 6 (2010)